Project No: A3053-231

MYMEDTRACE: A SCALABLE PATIENT-CENTRIC BLOCKCHAIN-BASED ELECTRONIC HEALTH RECORDS MANAGEMENT PLATFORM

Submitted by: Vanessa Christy Chandra (U2020341K)

Supervisor: Assoc Prof Chua Hock Chuan

School of Electrical & Electronic Engineering

A final year project report presented to Nanyang Technological University
in partial fulfilment of the requirements for the
Degree of Bachelor of Engineering

2024

Table of Contents

Abstract

The Electronic Health Record (EHR) system plays a crucial role in modern healthcare by facilitating the storage and exchange of patient medical records among healthcare providers. However, challenges such as data security, scalability, and interoperability hinder the efficiency of EHR management. This thesis explores the integration of blockchain technology to address these issues within the EHR domain. Leveraging blockchain's decentralized and immutable nature enhances the security of the EHR storage system. Additionally, the adoption of standardized medical data types like Fast Healthcare Interoperability Resources (FHIR) and off-chain database storage promotes interoperability and scalability in managing medical records. A user-friendly web application, MyMedtrace, is developed as a final solution, incorporating blockchain technology to tackle these challenges. Through a comprehensive examination of existing literature, case studies, and empirical research, this study assesses the feasibility and effectiveness of blockchain in transforming EHR systems. Furthermore, practical implementations and outcomes are discussed, paving the way for future advancements in healthcare data management.

Acknowledgements

I extend my heartfelt gratitude to the Almighty for granting me the strength, wisdom, and perseverance throughout this journey. My deepest appreciation goes to my family for their continuous love, support, encouragement, and understanding during the highs and lows of this journey. Special thanks to my loving boyfriend, whose patience, love, and belief in me have been my pillars of strength.

I am immensely grateful to my friends for their constant encouragement which have made this journey memorable and enjoyable. Their supports have been invaluable, providing solace during challenging times and celebrating achievements together.

A special note of appreciation to Assoc Prof Chua Hock Chuan for his guidance, expertise, and unwavering support throughout this project. His mentorship has been instrumental in shaping my ideas and refining my research.

I would also like to express my sincere gratitude to all the authors of the literature out there whose work has served as a foundation for this research. Your contributions have been invaluable in shaping my understanding and informing my study.

Lastly, I extend my gratitude to all those who have contributed directly or indirectly to the completion of this thesis. Your support, encouragement, and belief in my abilities have been truly inspiring and motivating.

Vanessa Christy Chandra,

April 2024

List of Figures

List of Tables

Chapter 1 Introduction

This chapter provides an overview of the project's context, including the background of the Electronic Health Record management system and the identified problem statement. Additionally, it explores the potential of blockchain technology in addressing these challenges. Furthermore, it outlines the project's objectives and delves into its scope.

1.1. Background and Motivation

Over the last few decades, digitalization has served as a significant catalyst, ensuring the relevance and adaptability of numerous industries, including healthcare. Years ago, all the processes of recording patient data and healthcare services were completely paper-based which lacked efficiency and sustainability. The digitalization of the healthcare industry leads to the adoption of Electronic Health Records (EHR). EHR stores patient health information digitally, enhancing treatment efficiency and coordination among healthcare providers while facilitating cost-effective tracking of patient health data [1].

However, despite its advantages, there remain a few challenges such as scalability, interoperability, and privacy issues. For instance, patients may struggle to recall their complete medical history and be required for the submission of extensive medical test reports when switching healthcare providers. Failure to provide these records may result in redundant and costly tests [2]. Additionally, the existence of digital versions of patient records creates higher opportunities for cybercriminals. Data breaches in electronic health systems occur when there are security problems like vulnerabilities, unauthorized access, and not enough cybersecurity protections [3]. One of the most relevant examples is the largest medical data breach experienced by Singapore’s biggest healthcare provider, SingHealth, which happened in 2018 affecting 1.5 million patient’s data [4].

An EHR storage system should be designed to be patient-centric, optimize interoperability including consistency, compatibility, and timeliness of data, and maximize the security at the same time [5]. Therefore, with the existence of blockchain as a peer-to-peer network and decentralized ledger system, it can provide tamper-proof solutions to tackle these EHR

challenges. Blockchain-based EHR provides patients with a secure and efficient capability of Health Information Exchange (HIE) and requesting service with various healthcare providers and parties, including but not limited to doctors, laboratories, and medical insurance [2].

1.2. Project Objective and Scope

The objective of this project is to explore the usage of blockchain-based technology to improve the interoperability, scalability, and privacy of the patient-centric EHR system and develop a web application as the deliverable solution. According to [6], the established blockchain-based EHR solutions have few limitations such as common standards, scalable data storage, and user-friendliness. A recent study of the integration of EHR data segregation using on-chain and off-chain storage has the potential to address the scalability issue [7].

To address the existing solutions gap, this project aims to propose a blockchain-based EHR management system that offers scalable data storage in a user-friendly, efficient, and secure manner. The main contributions of this project are to utilize the blockchain as on-chain storage and an external database as off-chain storage with extra authorization to ensure tamper-proof off-chain storage.

1.3. Report Organization

This report shows the application of Blockchain technology to solve several key challenges of Electronic Health Record (EHR) systems. This report is structured into 5 parts:

1. Introduction provides the background information of EHR and its challenges, overview of the solutions using blockchain and the project objectives
2. Literature review highlights the blockchain overview including its key technologies, advantages and challenges
3. Design and implementation provides the detailed design and technical aspects, key features of the deliverable application, and evaluation of the application
4. Results and discussions summarizes the project’s key aspects, contributions and implications, as well as its limitation and future work

Chapter 2 Literature Review

2.1. Introduction

In this section, previous studies on blockchain technology and its implications for the Electronic Health Record (EHR) system will be reviewed. Furthermore, the review aims to identify gaps and establish the project fits in the context of the reviewed literature. The review will be structured into four areas, which are (1) blockchain overview; (2) key issues of EHR system; (3) advantages of blockchain in EHR system; and (4) challenges of blockchain in EHR system.

2.2. Blockchain Overview

Blockchain technology is a decentralized peer-to-peer network that was first introduced in 2008 by Satoshi Nakamoto, with the initial ideas of eliminating the needs of central authority on electronic transactions [10]. Blockchain comprises blocks that consist of transactions (data) that sequentially grows becoming a chain and shared among the network participants. The block also contains the hash pointer of the previous block, to ensure that the chain is tamper-proof. Any changes to the block data will result in a different hash value in the next block [8].

The two widely used consensus mechanisms of blockchain are Proof of Work (PoW) and Proof of Stake (PoS). In the PoW, in order to validate a block of transaction and add a new block to the chain, a miner needs to compete to solve a complex mathematical computation that involves altering a nonce, a value used to change the hash of data, to meet certain output criteria [9]. In the PoS, there are validators that are chosen by the amount they are willing to stake, responsible for creating new blocks and validating the transaction. The process of block creation and transaction confirmation are much faster, which makes the PoS mechanism become more popular recently [8].

The existence of smart contracts led blockchain to be widely used in different ranges of applications such as supply chain, healthcare industry, online voting, insurance, management, and IoT devices. Smart contract is a self-executing code that is stored in blockchain and immediately executed when certain conditions are met. Smart contract eliminates the needs of

physical presence, middlemen, and paper works by making contracts automatically on the blockchain network upon achieving certain conditions [10]. One of the most widely used smart contract protocols is Ethereum. Ethereum is a decentralized platform that enables developers to build and deploy decentralized applications (dApps) with wide ranges of use-cases [11].

2.3. Key Issues of Electronic Health Record System

The adoption of Electronic Health Record (EHR) gives various prominent advantages that help to enhance the healthcare workflows. Some of its benefits are time and resource efficiency, availability, accessibility, and quick communication process. However, there are also some drawbacks that must be taken into account such as security, interoperability, and scalability issues [2][3][5].

Potential threats to information security increase as the data becomes digitally accessible. As mentioned in [13], one of the motivations is to maintain patient and healthcare providers relationship, health data confidentiality and privacy are essential to build a good EHR management system. Moreover, [13] also defines a few important health data security aspects that must be considered, which are the scope of information stakeholders, sensitivity of EHR content, sharing level, and access control.

Blockchain offers robust security measures for handling healthcare data, as mentioned in [15]. Firstly, it establishes reliable digital ledgers, ensuring that transactions remain unaltered by any entities. Additionally, transparent access control is achievable, with any unauthorized access or transactions being recorded on the blockchain and disseminated to all network participants. Furthermore, blockchain-based smart contracts can efficiently enforce strict user authentication and access control to the data storage in a distributed manner.

Moreover, sharing of EHR data between different healthcare providers has its own drawbacks in terms of interoperability. The inability to share previous medical history, or different formats are provided will lead patients to redo redundant and costly tests that reduce the efficiency of the

healthcare process. Numerous technical barriers such as security of data exchange, standardized format that must be agreed upon, must be taken into account.

Recent research efforts [17][18] have focused on enhancing the interoperability of medical records and ensuring patient access to their health information. The utilization of standardized clinical data formats like Fast Healthcare Interoperability Resources (FHIR) proves effective in addressing this challenge. Additionally, the decentralized nature of blockchain improves accessibility to Electronic Health Record (EHR) data, enabling its availability across multiple systems within blockchain networks. Moreover, blockchain plays a pivotal role in advancing towards a more patient-centric and interoperable EHR system [18]. The incorporation of public key infrastructure (PKI) in blockchain serves as a centralized patient identification method applicable across healthcare providers, seamlessly linked to the respective provider's patient user account.

2.4. Challenges of Blockchain in Electronic Health Record System

Despite its numerous advantages, the blockchain framework presents significant challenges in areas like computing capacity, storage, and scalability [14]. The medical industry generates huge amounts of various medical data that require regular storage. Previous research on distributed file storage models [5][15] proposes a potential solution involving the integration of a singleton blockchain storage with a distributed file storage system capable of handling Electronic Health Record (EHR) data storage. In this approach, all metadata and hash values of the actual data are stored in the blockchain, while the complete EHR data is stored in a distributed file storage system such as the Interplanetary File System (IPFS).

Chapter 3 Design

3.1. Application Requirements

The overall goal of the deliverable MyMedtrace platform is to provide an easy to use and patient-centric Electronic Health Record (EHR) management platform that utilizes blockchain to store patient's data in scalable, interoperable, and secure manner. In order to achieve that, few technologies stacks are utilized namely Ethereum and Solidity as the blockchain smart contract builder, React as the frontend framework, Express.js and Flask as the backend framework, and MongoDB as the off-chain database. The platform stores all the collections of patient's medical records from different healthcare in a standardized format to promote an interoperable way of data exchange. Figure 3.1 shows the use case diagram of the application.

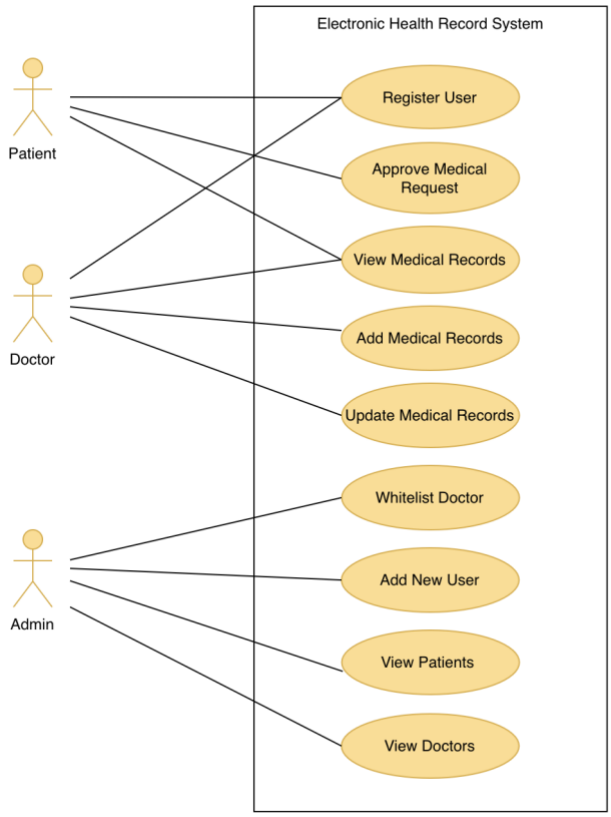

Figure 0.1 User Use Case Diagram of MyMedtrace Platform

The user types of the platform are differentiated into three types which are admin, patient, and doctor. Each user has a different access level of using the platform based on their roles. Each user will receive a unique blockchain address from the health administration for account creation purposes and it is attached to each user account.

3.1.1. Medical Data

The electronic health record is stored using the Fast Healthcare Interoperability Resources (FHIR) which is a standard healthcare data exchange format from Health Level 7 International (HL7). It is often represented in a format of XML and JSON which extends the usability of this format. The FHIR has various resource types that can support different healthcare processes such as administrative, clinical reports, and many more. In this project, we will limit our scope in using the resources types of Observations, Allergy, Conditions, and Medications.

3.1.2. Access Control

To ensure there is strict access control in updating or inserting new patient's record data, there is a whitelisting function that requires admin users to whitelist specific doctor to patient. Only whitelisted doctors can add a new health record to a patient or modify the existing record. Upon creating a new record or updating an existing record, a request is sent to the patient as the owner of their record to approve the request.

3.1.3. Natural Language Processing Integration

To support a faster insertion process of the patient record, an Artificial Intelligence (AI) using the Natural Language Processing (NLP) model is being utilized. The NLP model is able to extract key information from the free-text clinical notes using the Named Entity Recognition (NER) method and parse it to the respective input fields.

3.2. System Overview

In this project, a modular approach is utilized in the development process. The project development is divided into different components such as frontend, backend, and blockchain

smart contract. Overall, there are four repositories developed: one for the frontend application, one for the record server, one for a Natural Language Processing (NLP) server and one for the blockchain smart contract. All of the repositories are stored in the GitHub.

3.3. System Diagram

The web application, or the client, directly interacts with the server that is in charge of the record query, and the Blockchain. Every blockchain transaction is signed using the wallet address that is attached to every user account that requests the transaction. Furthermore, an off-chain database is also utilized to store the full data, and the blockchain as the on-chain storage stores the hash references and metadatas to ensure both instances are immutable and matched. Figure 3.2 shows the high-level communication scheme between the client, the record server, the smart contract, and the off-chain database. Furthermore, the client also directly communicates with the server that is in charge of the NLP model, and the server interacts with the model to obtain the parsed results and returned it back to the client l. Figure 3.3 shows the high-level communication scheme between the client, the server, and the NLP model.

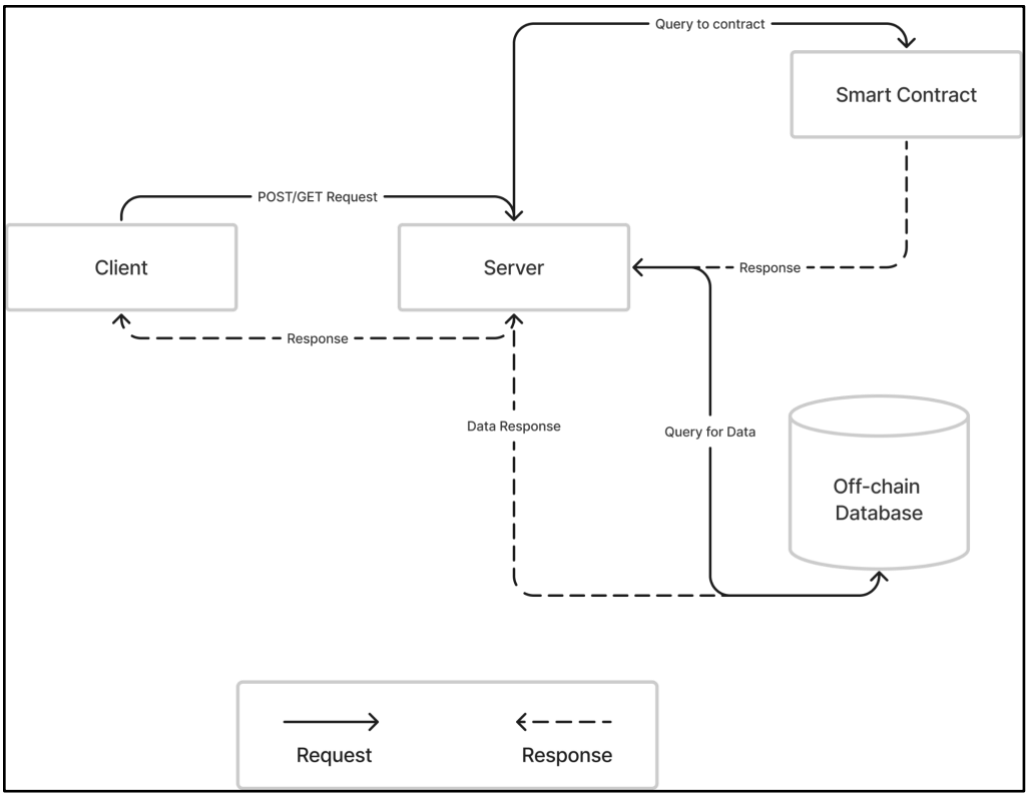

Figure 0.2 Communication Diagram between Client, Server, Smart Contract, and Off-Chain Database

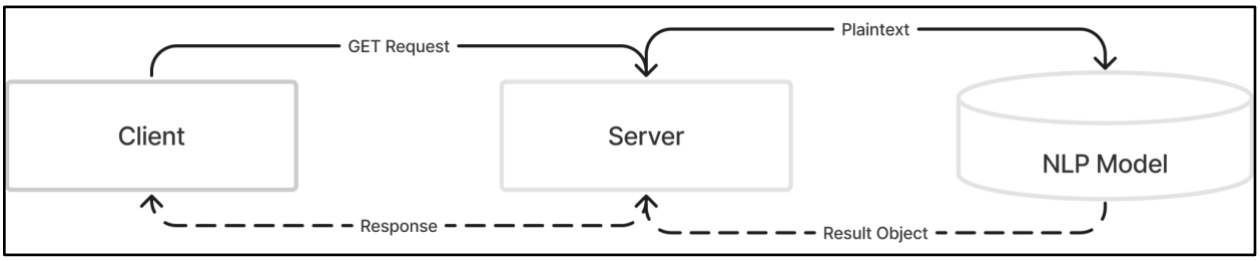

Figure 0.3 Communication Diagram between Client, Server, and NLP Model

3.3.1. Login and Sign Up

The user needs to log in using their registered user email and password. Further, the server will validate the user's existence and credentials by querying the off-chain database. If there are any mismatched credentials found during the validation on the server side, it will return an unauthorized response status, otherwise, it will return the user email and blockchain address for the client side to identify. Figure 3.4 shows the sequence diagram of the login process between the client, server, blockchain, and the off-chain database.

If the user is not registered yet, the user needs to go through the signup process and the types of data needed depending on the role (i.e. patient, doctor, admin). Each user is also allocated a blockchain address as their blockchain identifier, and additionally, an admin is allocated with an admin key. Figure 3.5 shows the sequence diagram of the signup process.

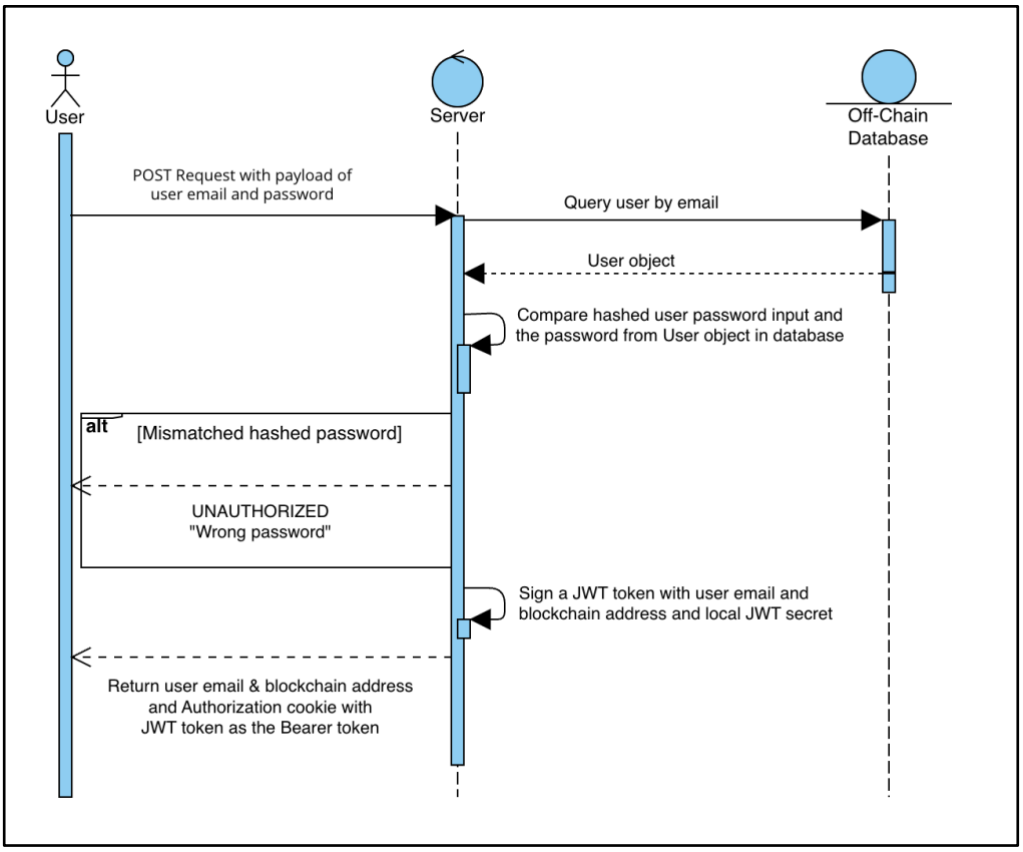

Figure 0.4 Sequence Diagram of Login Module

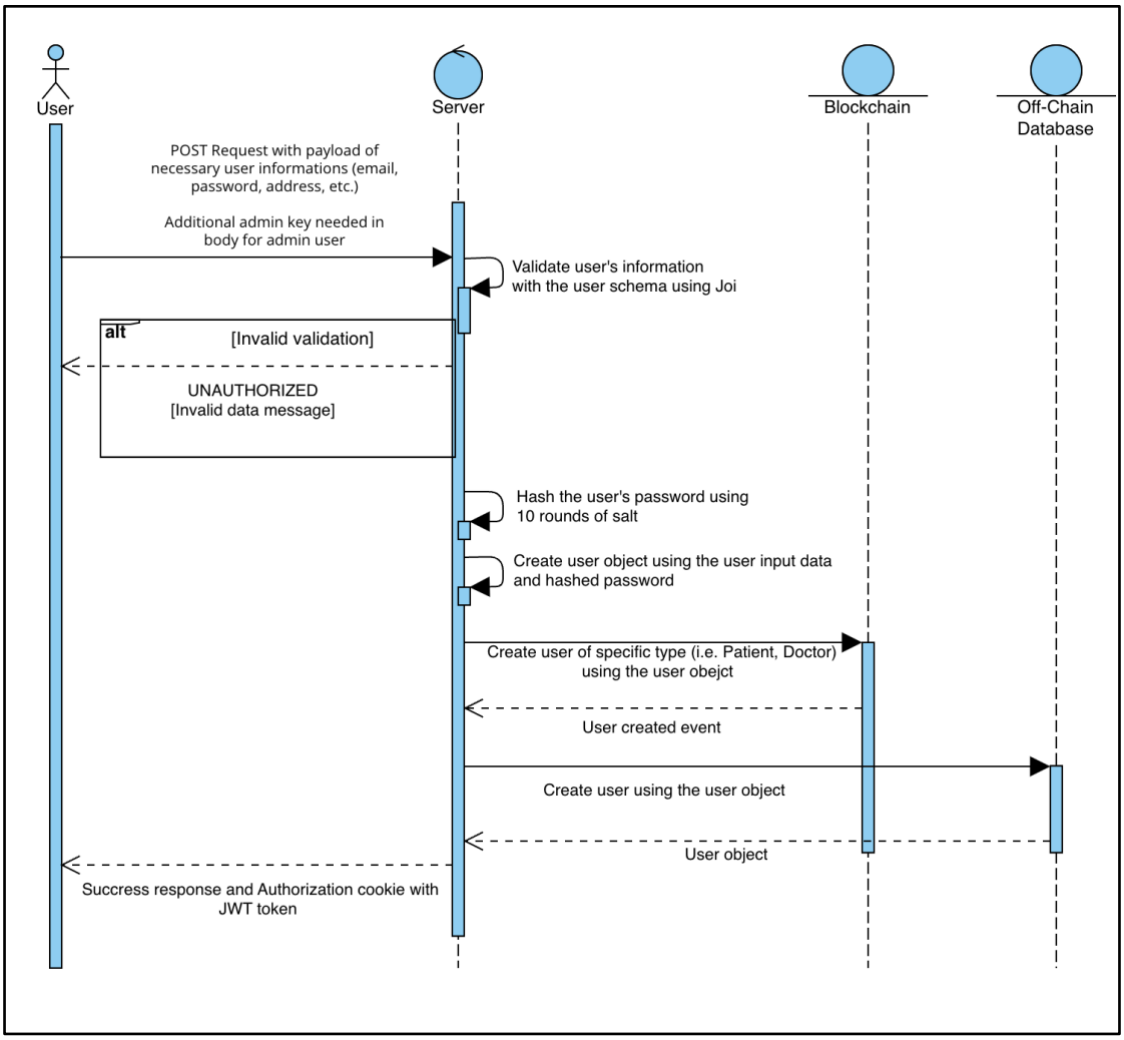

Figure 0.5 Sequence Diagram of Sign Up Module

3.3.2. Whitelist Function

Firstly, the administrator needs to whitelist the doctor to the patient for the doctor to be able to add or update the medical record of the patient. Figure 3.6 shows the sequence diagram of the whitelisting module that is done by the administrator. The admins need to sign the blockchain transaction using their wallet address to send the transaction.

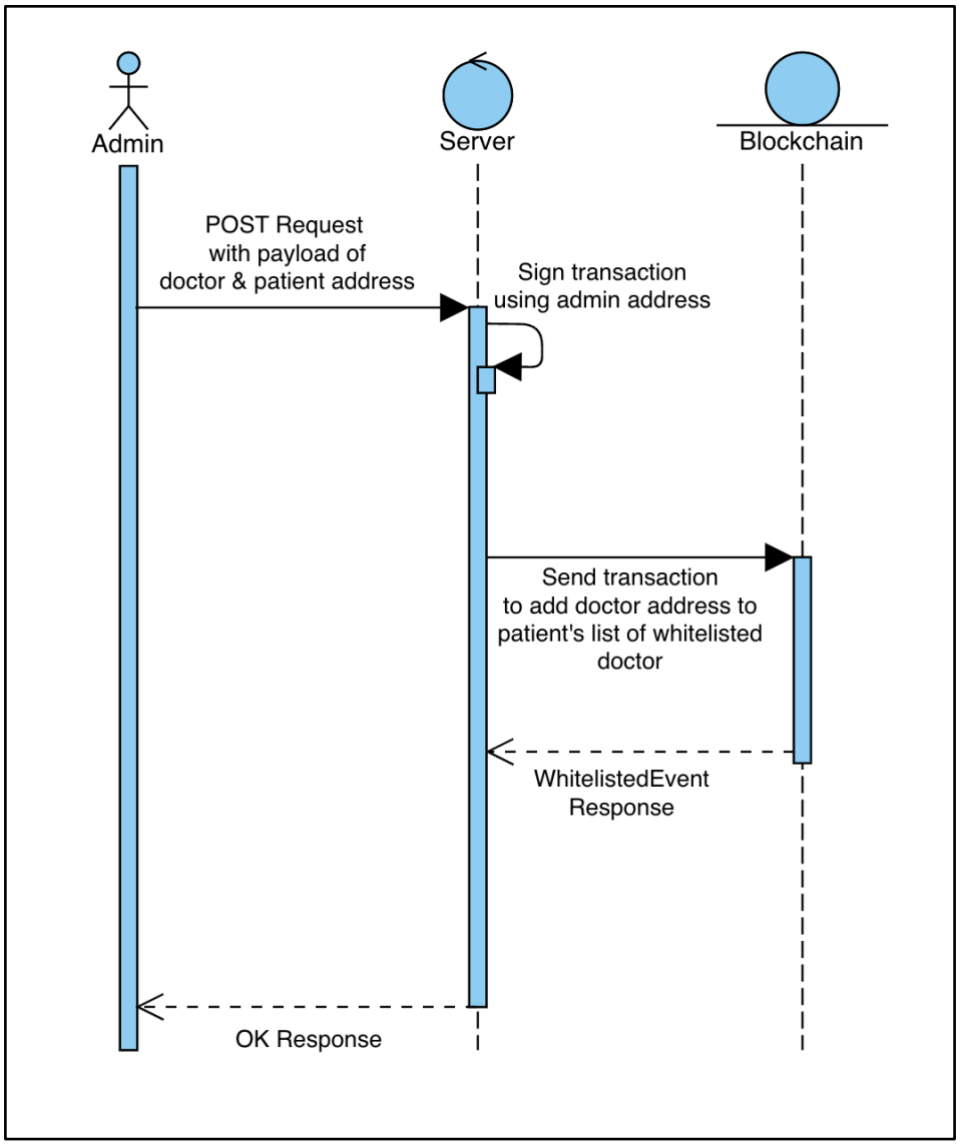

Figure 0.6 Sequence Diagram of Whitelist Module

3.3.3. Add Record Module

After the doctor is whitelisted, the doctor can perform the creation of a new medical record of the patient. Figure 3.7 shows the sequence diagram of the doctor creating a new medical record of the patient. The doctors need to sign the transaction with their assigned blockchain address to send the created record transaction to the blockchain. A unique ID will be generated on the server side. On the server side, it will also perform an encryption of the medical record using a symmetric encryption technique and store the encrypted value in the blockchain along with the unique ID as the reference. It will then also store the full medical data in the off-chain database.

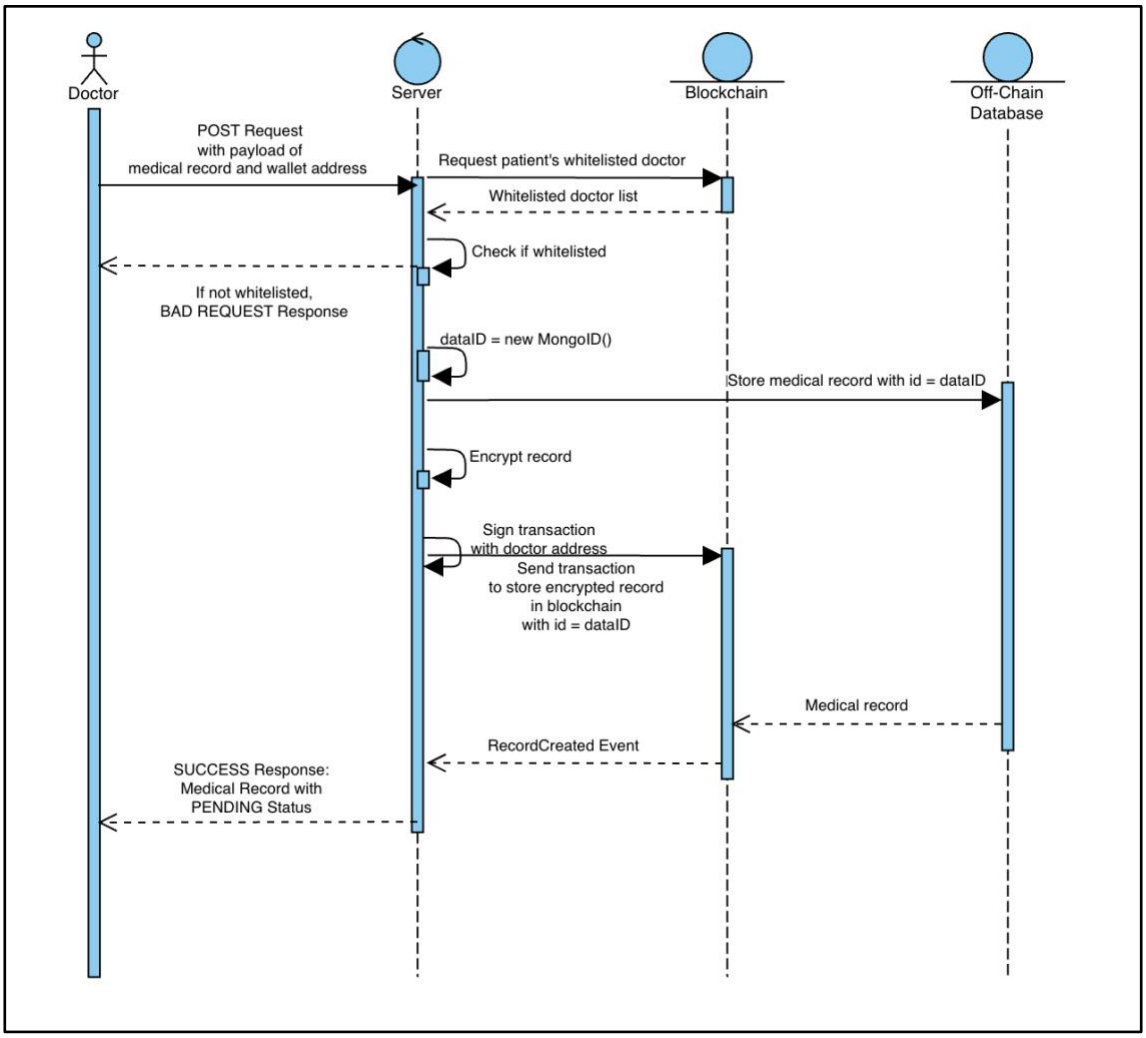

Figure 0.7 Sequence Diagram of Add Medical Record

3.3.4. View Medical Record

A medical record is visible to the related patient as the record owner, the issuer doctor, and the administrator. Figure 3.8 shows the sequence diagram of the viewing medical record module. The server side queries for both the blockchain and database references, and if the decrypted value and database value are mismatched, it will return a bad response indicating the mismatched record.

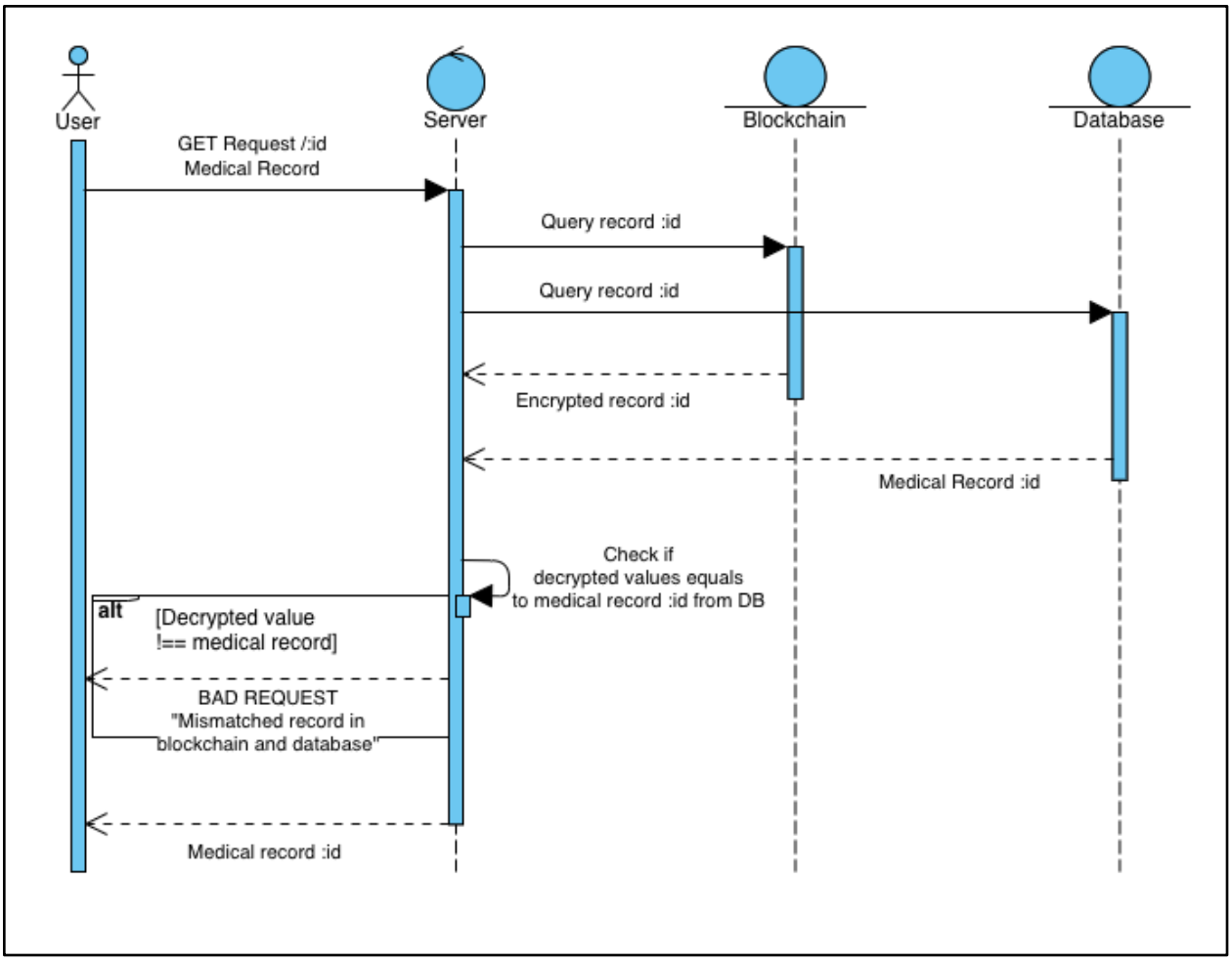

Figure 0.8 Sequence Diagram of Viewing Medical Record Module

3.3.5. Approval or Decline of Medical Record Request

Upon issuing the medical record of a patient, the status of the medical record remains pending until the patient approves the record. Therefore, the related patients can approve the record added by the doctor on their side. Figure 3.9 shows the sequence diagram of the approval process.

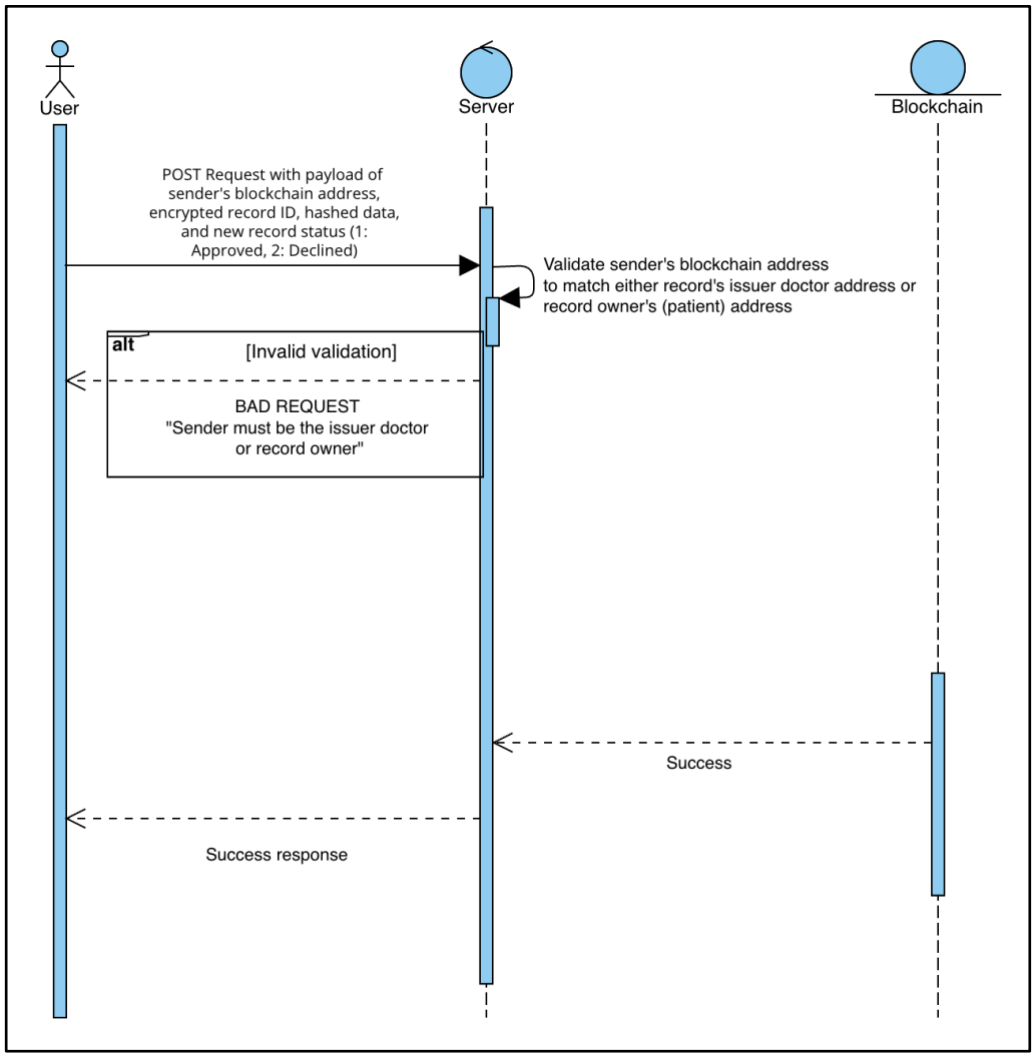

Figure 0.9 Sequence Diagram of Approval or Decline of Medical Record Module

Chapter 4 Implementation

4.1. Technology Stacks

Stack	Software
Frontend	React Typescript UI Library: ChakraUI, AntDesign, React Apex Chart
Backend	Express.Js, Flask
Web3	Solidity, Ganache, Truffle
Off-Chain Database	MongoDB

Table 4.1 Technology Stacks Used

4.2. Frontend

The frontend or client side is responsible for directly interacting with the user and providing useful information and interaction for the user. To fulfil the request of the user, the client communicates with the server to request the data needed and utilizes the response. The client comprises login and signup modules as the gatekeeper to allow for authenticated users only, the main dashboard, and the other modules that are displayed based on the user types.

In this project, React is utilized as the building framework of the frontend side, as well as Typescript to ease the process of development and enhance the code reliability. Additionally, user interface (UI) libraries such as ChakraUI [21], Ant Design [22], and React Apex Chart [23] are utilized to support the development of the UI components. Due to the limitation of the supporting UI library for the display of the FHIR resources card, a clean and intuitive UI component is built by utilizing the existing UI library to support the data display.

4.2.1. Landing Page

The landing page is built as a first point to call to action by providing the user with some information regarding the advantages of MyMedtrace as an EHR system. Figure 4.1 shows the interface of the landing page.

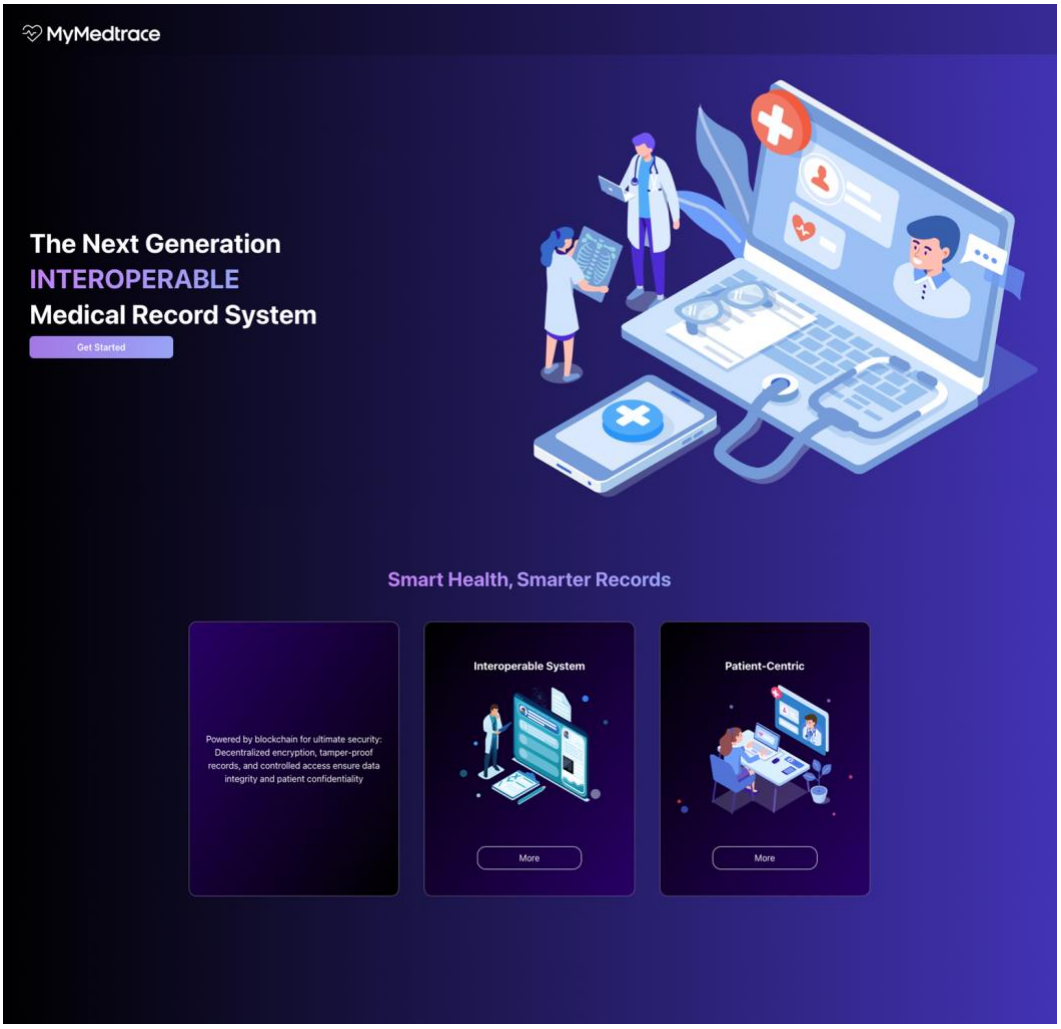

Figure 0.1 Landing Page Interface

4.2.2. Login

The login feature is built to ensure that the platform users are only authenticated users with attached blockchain wallet addresses. This is to ensure that every single transaction is signed to promote security and integrity. The user needs to key in their user email and password to be authenticated in the login process. Figure 4.2 shows the interface of the login system.

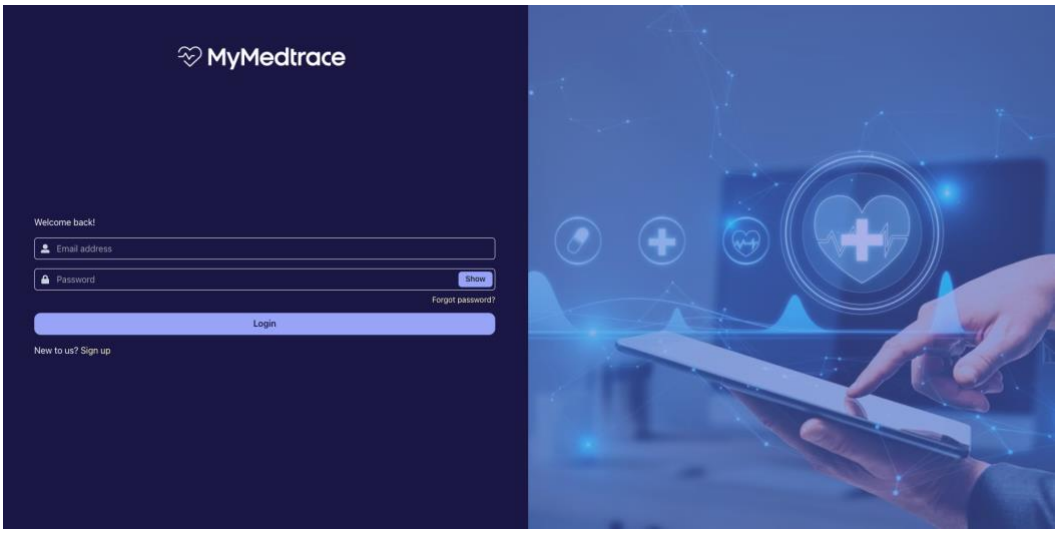

Figure 0.2 Login Interface

4.2.3. Sign Up

In the sign-up feature, the user is required to key in the primary personal information such as name, email, password, wallet address, identity card (IC) number, gender, phone number, date of birth, home address, and user type. Furthermore, based on their user type, they are required to key in additional information. Patients need to key in their emergency contact and number, blood type, height, and weight, doctors need to key in their qualifications and major, and the admin needs to key in the admin key to prevent misuse of the admin role. The admin key is retrieved upon the medical organization's request for the platform usage. Figure 4.3 to 4.6 shows the sign up module interface for different user types.

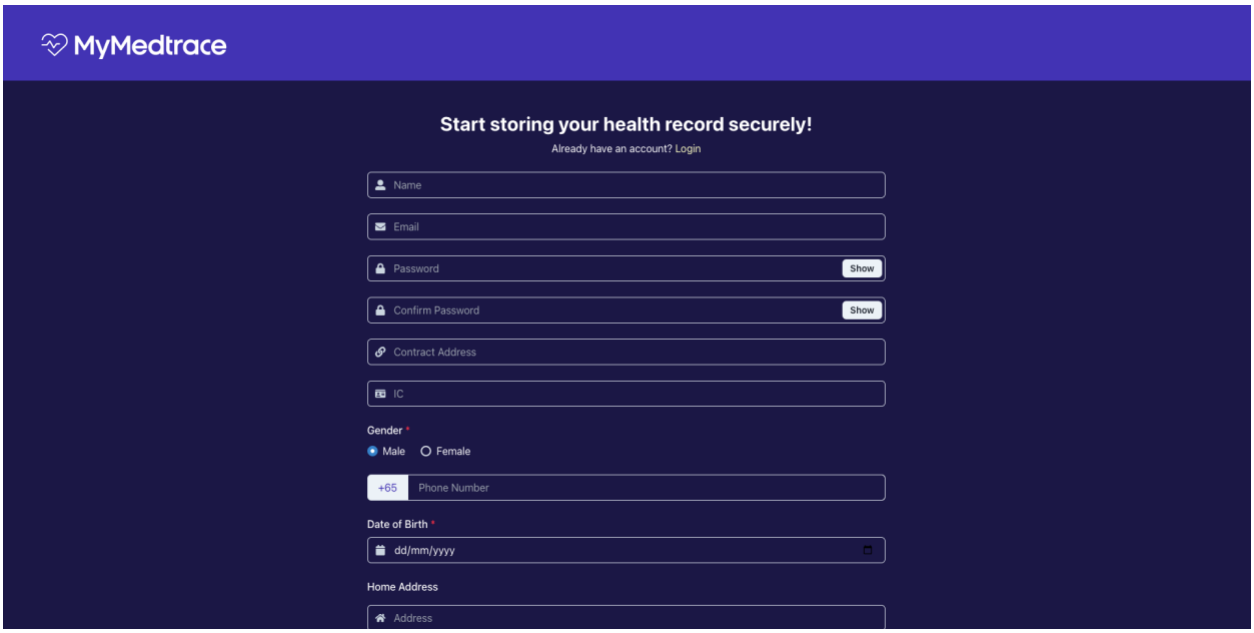

Figure 0.3 Sign Up Interface

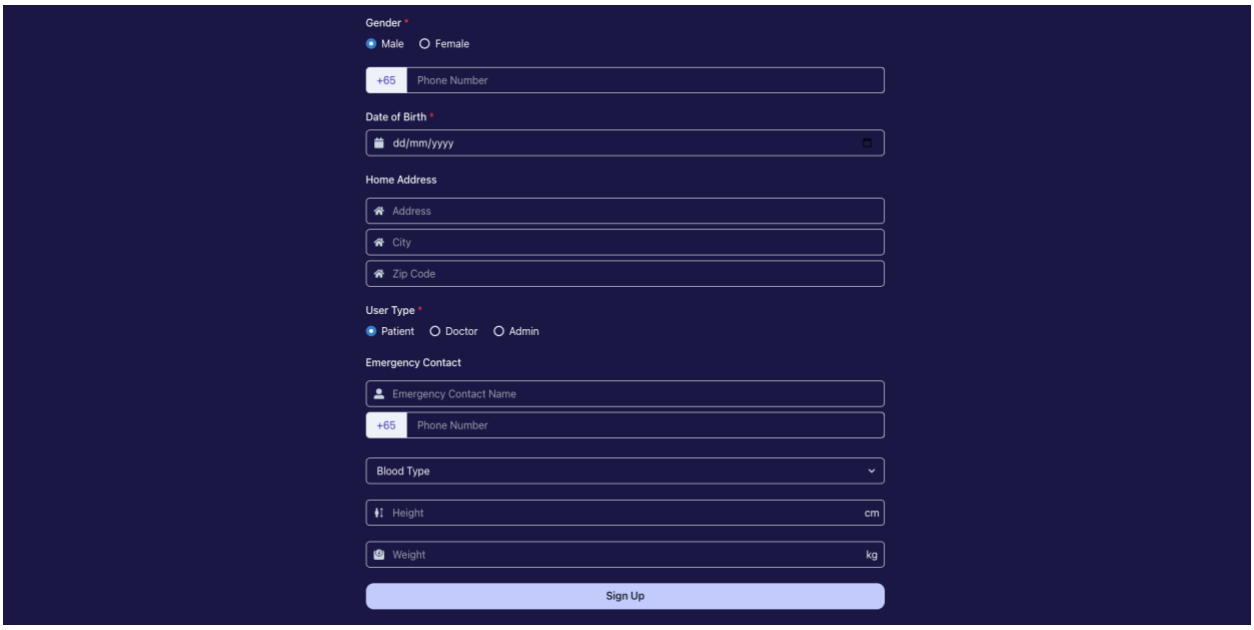

Figure 0.4 Sign Up Interface (Patient User Type)

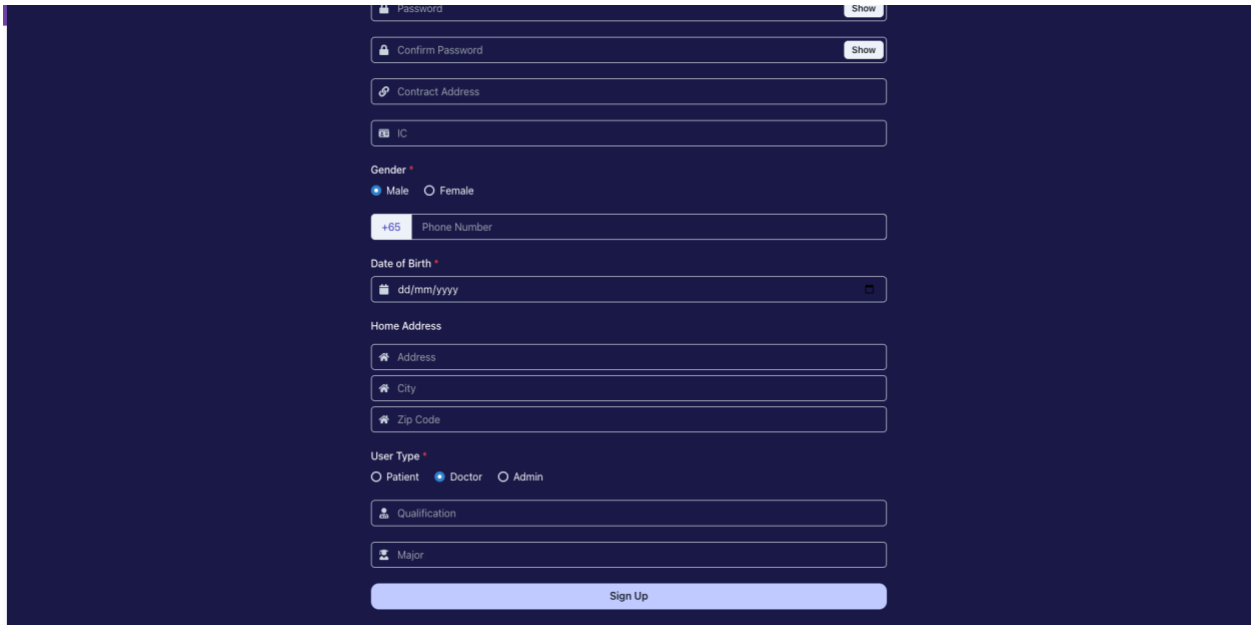

Figure 0.5 Sign Up Interface (Doctor User Type)

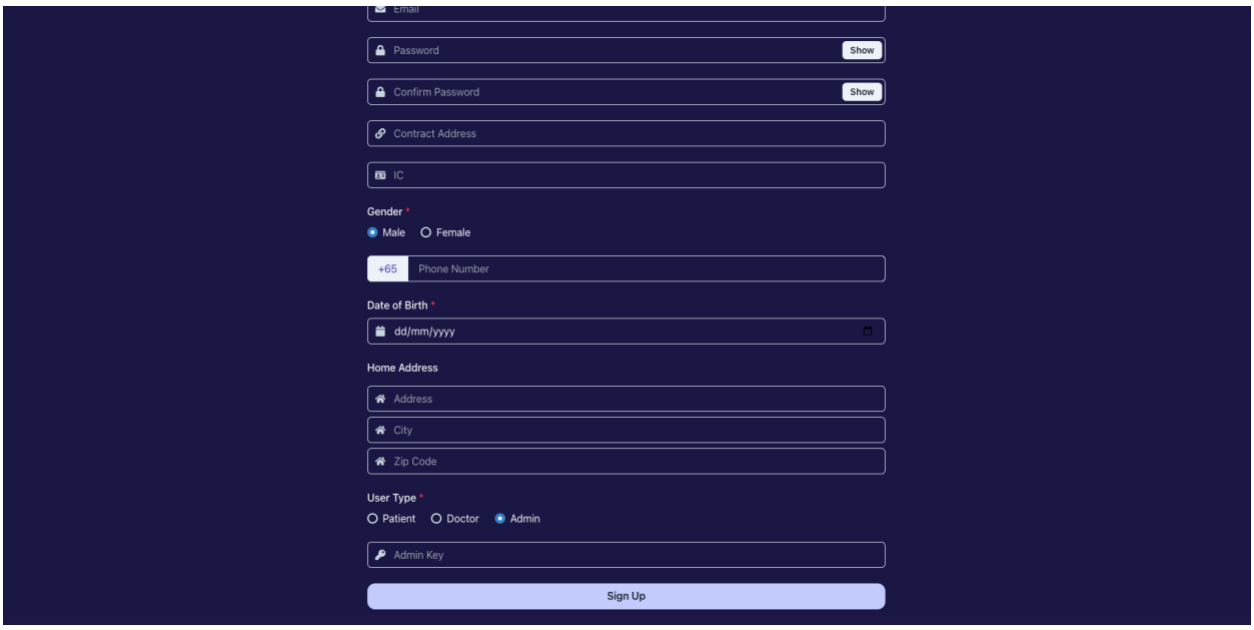

Figure 0.6 Sign Up Interface (Admin User Type)

4.2.4. Admin Pages

4.2.4.1. Dashboard

The admin dashboard comprises useful data insights related to administrative data, such as the number of patients, doctors, and records. Furthermore, the admin is also able to view the statistics of the patients and records, and the preview of the patients list. Figure 4.7 shows the interface of the admin dashboard.

Figure 0.7 Admin Dashboard Interface

4.2.4.2. Patients and Doctors List

Admin can browse all the patient and doctors lists that are registered in the platform through this page. The list comprises their administrative details such as name, blockchain address, IC number, gender, email, and registered date. There is also a search query for the admin to browse patient or doctor based on the name, address, email, or IC number. Figure 4.8 and 4.9 shows the interface of the patients and doctors list respectively.

Figure 0.8 Patients List Interface (Admin View)

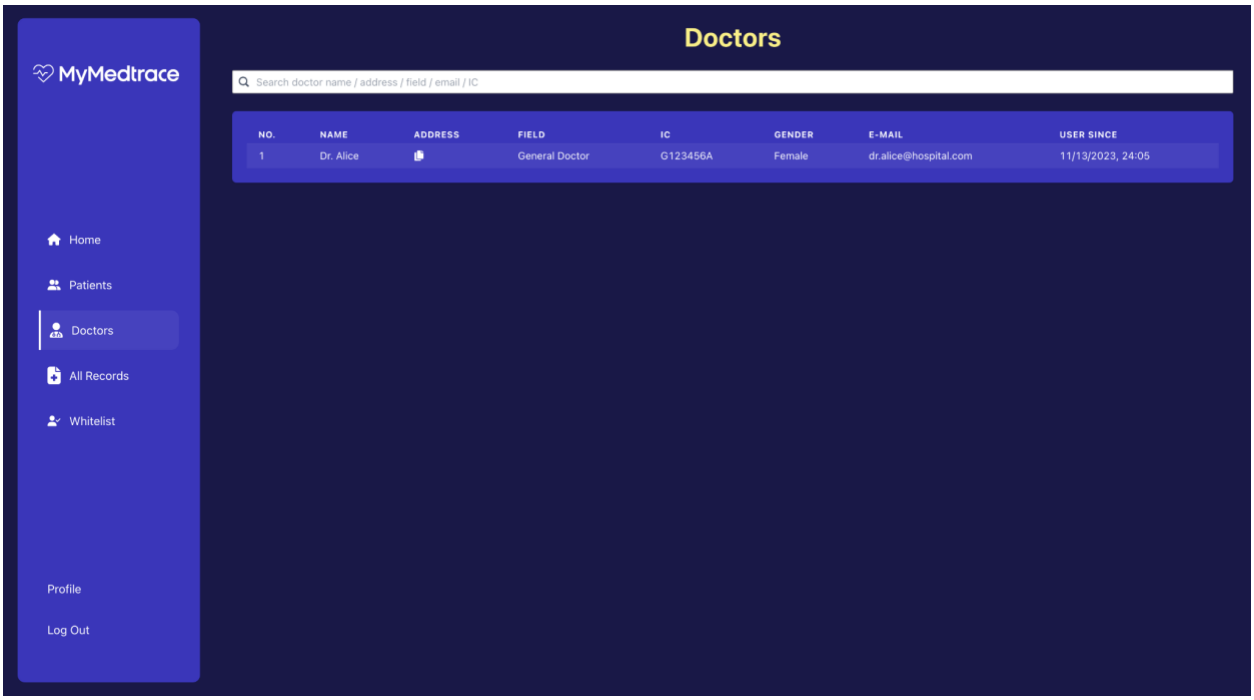

Figure 0.9 Doctors List Interface

4.2.4.3. Records List

Admin can view patients' records through this page. The list comprises the record's date, related patient name and blockchain address, issuer doctor, record type, and the main display name of the record. There is also a search query for the admin to browse record based on the patients address or name, doctors name, or record type. Figure 4.10 shows the interface of the list of patients records.

Figure 0.10 All Patients Records List Interface

4.2.4.4. Whitelist Module

The whitelist module is restricted to admin users only. The whitelist module is to enable a doctor to perform an action on the patient's record data once the doctor's address is whitelisted to the patient's blockchain address. Figure 4.11 shows the interface of the whitelist module.

Figure 0.11 Whitelist Doctor Interface

4.2.5. Doctor Pages

4.2.5.1. Dashboard

The doctor dashboard comprises useful data visualization such as patients and records diagrams. Furthermore, there are also shortcuts to create a record of different record types. Figure 4.12 shows the interface of the doctor dashboard.

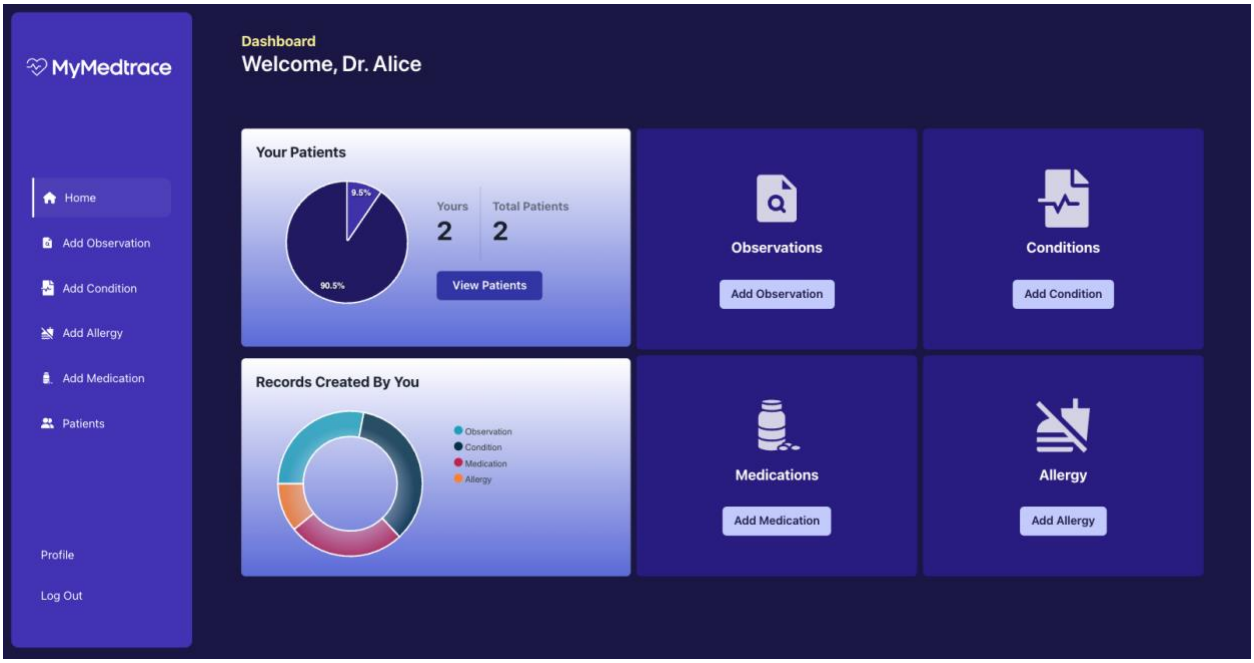

Figure 0.12 Doctor Dashboard Interface

4.2.5.2. Add Observation Module

The add observation module comprises input fields related to each attribute in the FHIR Observation data type. It includes the patient address, observation status, performer, value quantity, and value reference ranges, and other related information. Figure 4.13 shows the add observation module interface.

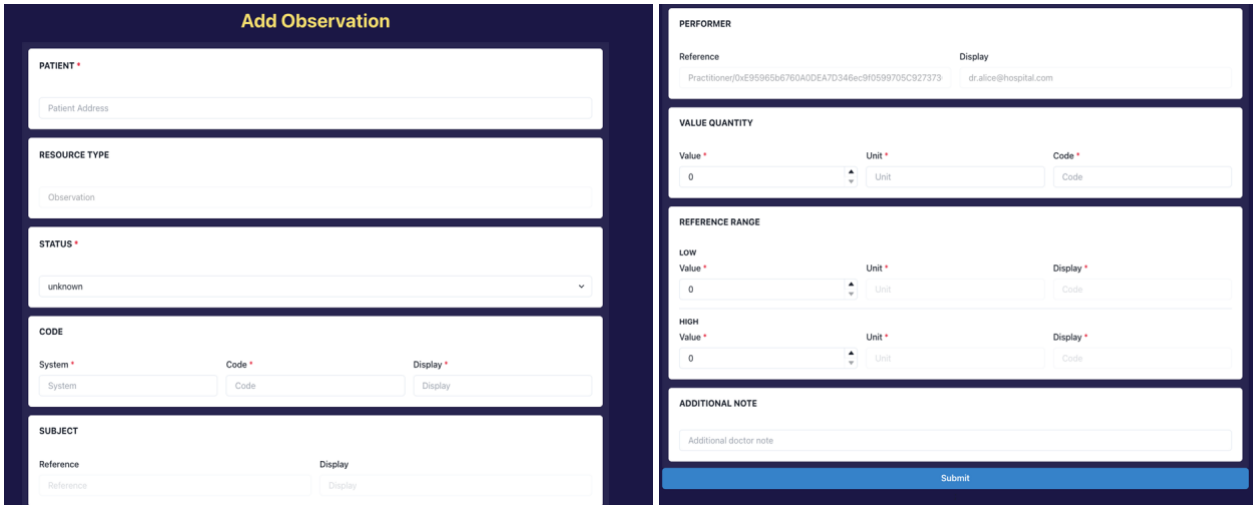

Figure 0.13 Add Observation Interface

4.2.6. Add Condition Module

The add condition module comprises input fields related to the FHIR Condition data type, such as resource type, subject, recorder, asserter, clinical status, verification status, category, severity, code, body site, onset and abatement date time, as well as additional administrative data such as patient address and extra note. Figure 4.14 shows the add condition module interface.

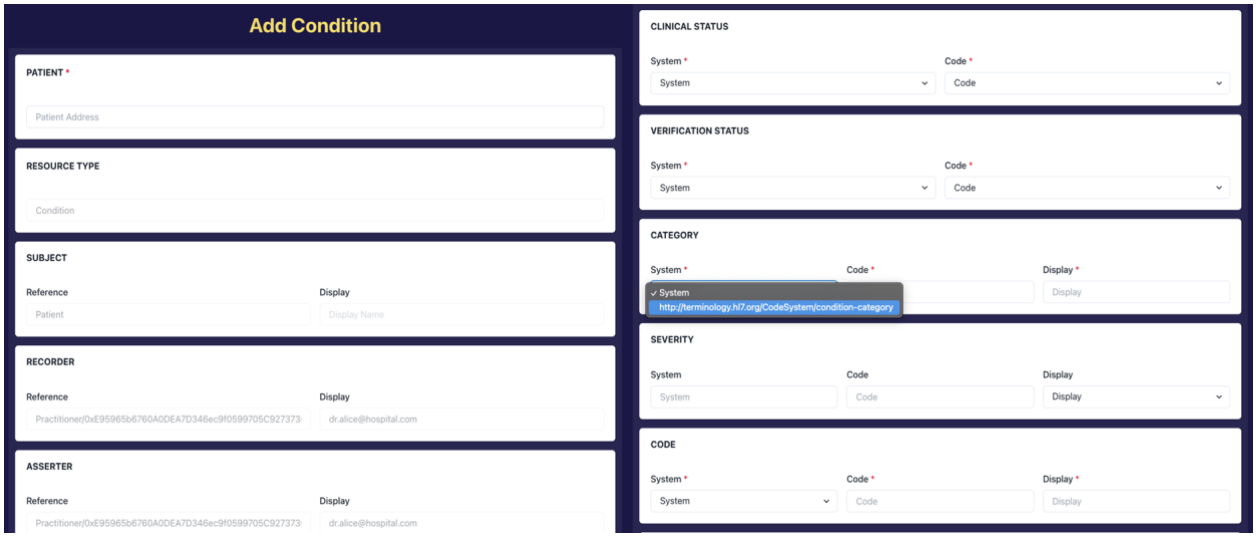

Figure 0.14 Add Condition Interface

4.2.6.1. Add Allergy Module

The add allergy module comprises input fields related to the FHIR Allergy Intolerance data type, such as resource type, patient, recorder, clinical status, verification status, code, reaction including the manifestation, exposure route, substance, description, and severity, recorded date, additional note, category, and criticality, as well as additional administrative data such as patient address. Figure 4.15 shows the add allergy module interface.

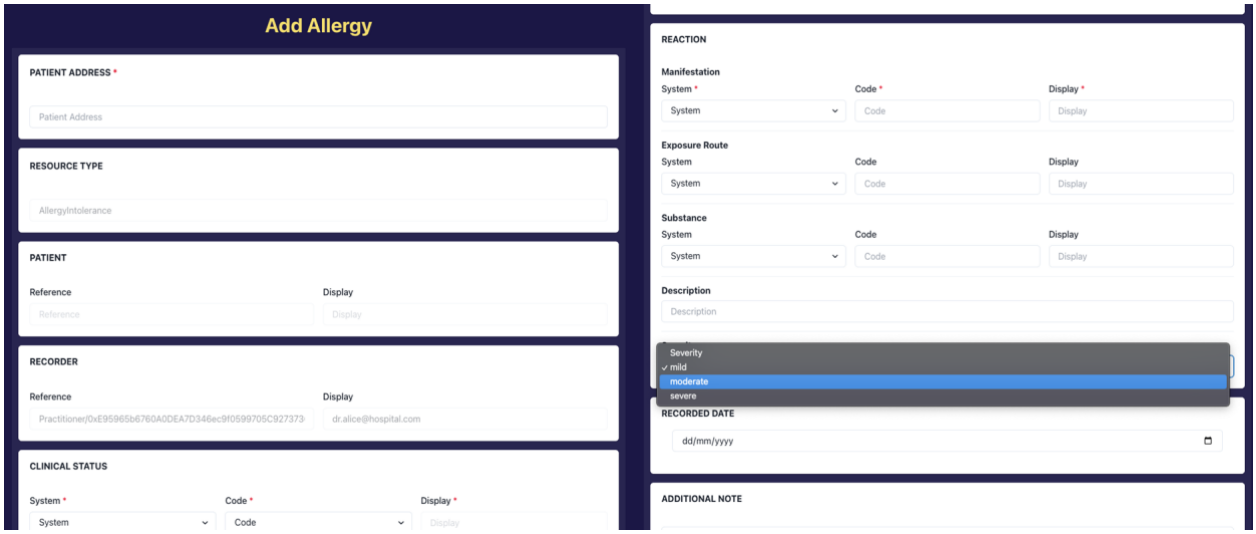

Figure 0.15 Add Observation Interface

4.2.6.2. Add Medication Module

The add medication module comprises input fields related to the FHIR Medication data type, such as resource type, status, code, form, manufacturer, ingredient, batch, as well as additional administrative data such as patient address and extra note. Figure 4.16 shows the add medication module interface. Furthermore, there is an additional AI implementation in this module, in the part of parsing the free-text clinical notes to a mapped input field. The doctor is able to insert a free-text medical prescription, and it will be parsed into a key-value mappings that are available in type of dosage, drug, duration, form, frequency, route, and strength. Figure 4.17 shows the AI-parsed module for the add medication.

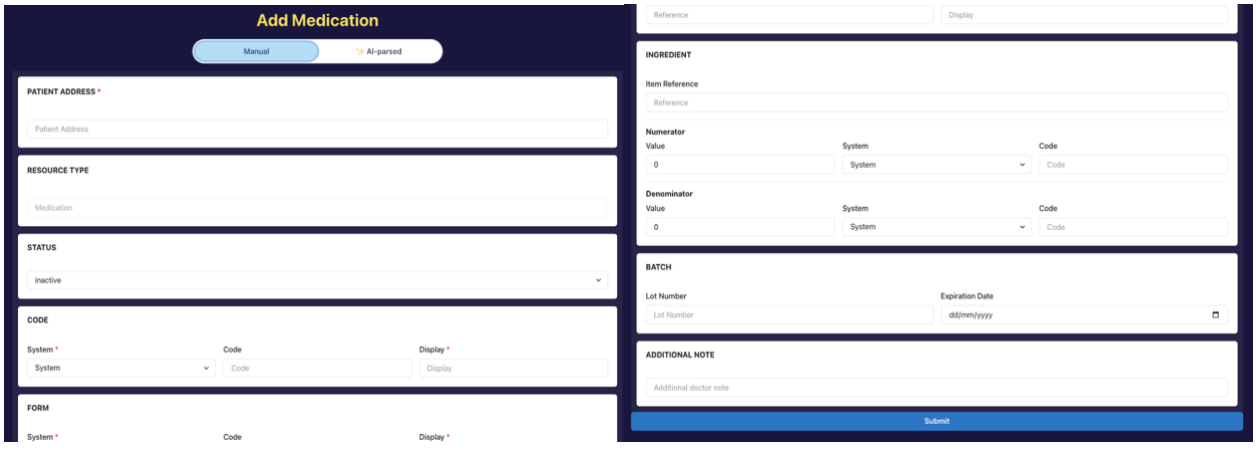

Figure 0.16 Add Medication Interface

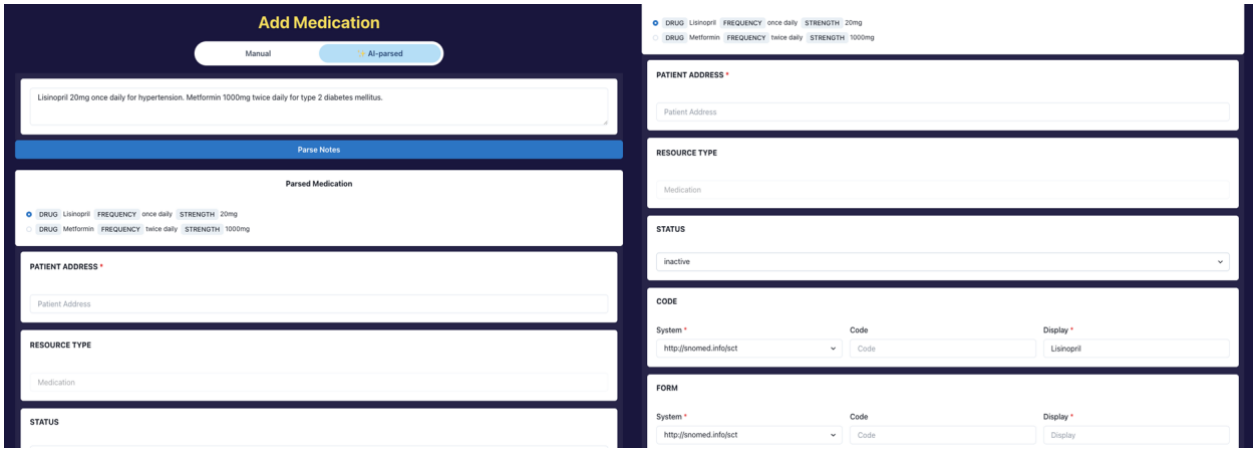

Figure 0.17 AI-Parsed Medication Interface

4.2.6.3. Patients List Module

The patients list module in the doctor pages comprises the list of patients whitelisted to the doctor. There is also a search query for the doctor to browse patients based on their name, address, email, or IC number. Figure 4.18 shows the interface of the patients list module.

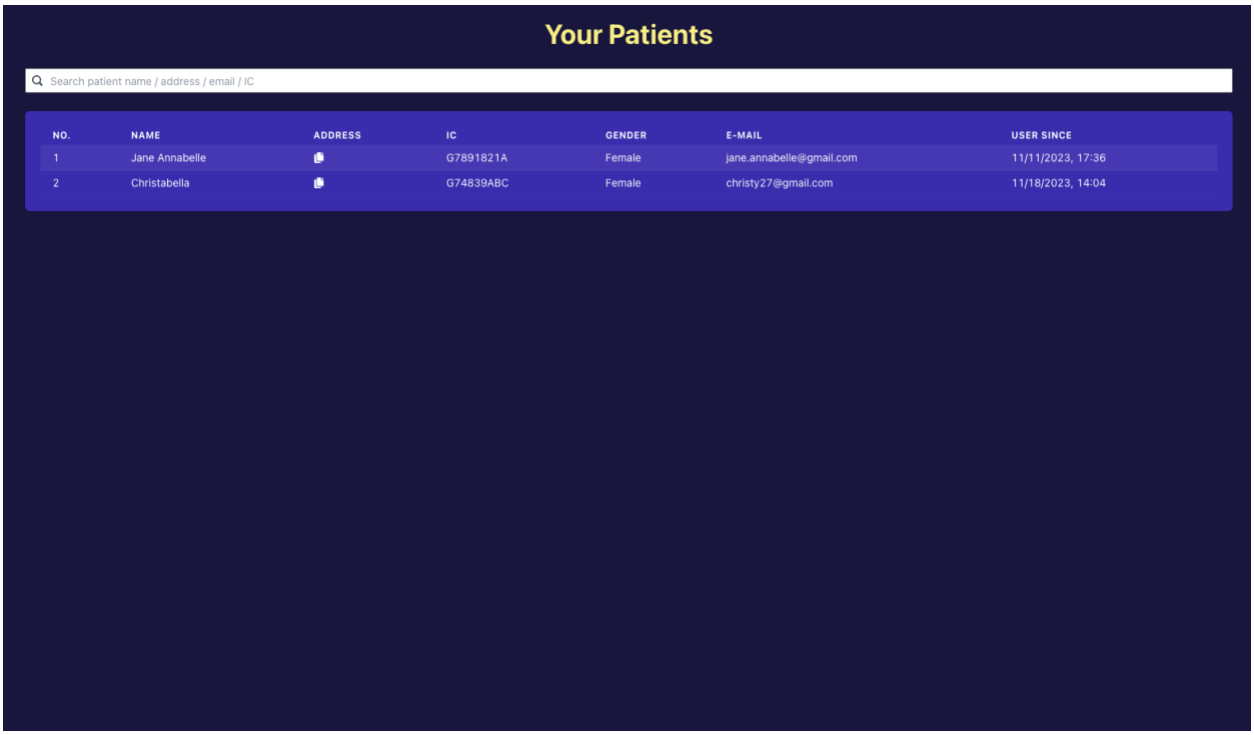

Figure 0.18 Patients List Interface (Doctor View)

4.2.7. Patient Pages

4.2.7.1. Dashboard

The patient dashboard comprises of shortcut to the to-be-approved medical records page requested by the whitelisted doctor, observations, conditions, medications, and allergies list. Figure 28 shows the interface of the patient's dashboard.

Figure 0.19 Patient Dashboard Interface

4.2.7.2. All Records

The all records module consists of the to-be-approved, approved, and declined medical records with each consisting of the details of the record date, issuer name, category, and name of the record. Figure 4.20 shows the patient's records interface.

Figure 0.20 Records of Patient Interface

4.2.7.3. Observations List Module

The observations list module comprises of list of observation cards that display the observation name, important values and units, issuer email, issued date, status of observation as well as the range bar that reflects the current disposition of the measurements. There is also a button to view a modal filled with the details of the observation in both field and raw model. Figures 4.21 and 4.22 show the interface of the observation list and the observation details modals respectively.

Figure 0.21 Observation List Interface

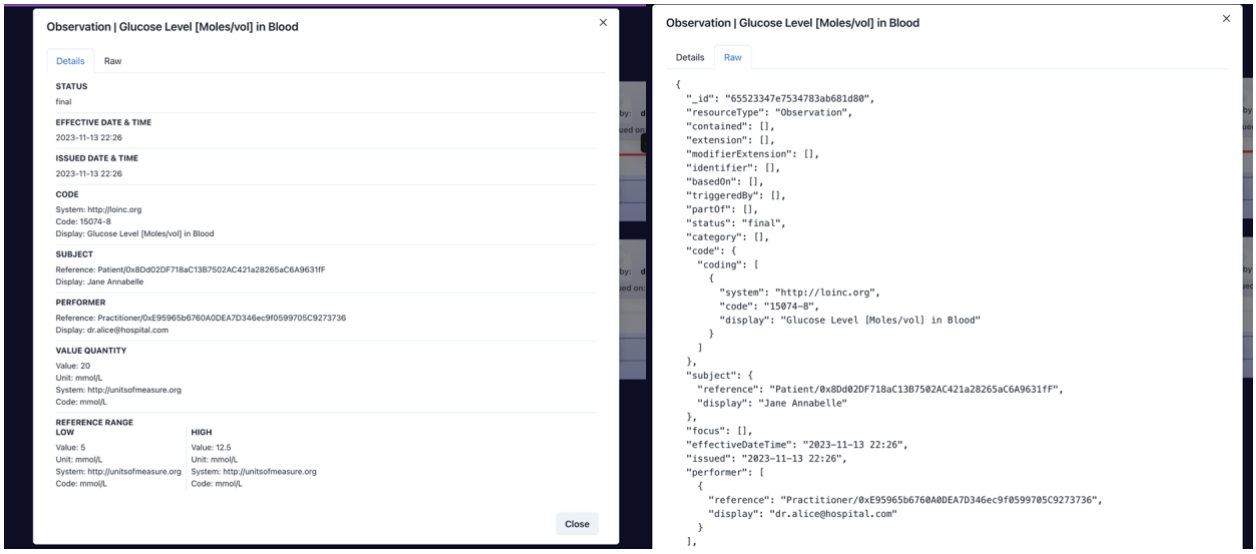

Figure 0.22 Observation Details Modal Interface

4.2.7.4. Conditions List Module

The conditions list module comprises of condition cards that consist of the condition name, clinical and verification status, asserter email, and recorded date. There is also a button to view the condition details modal in both the field and raw model. Figures 4.23 and 4.23 show the interface of the conditions list and the modals.

Figure 0.23 Conditions List Interface

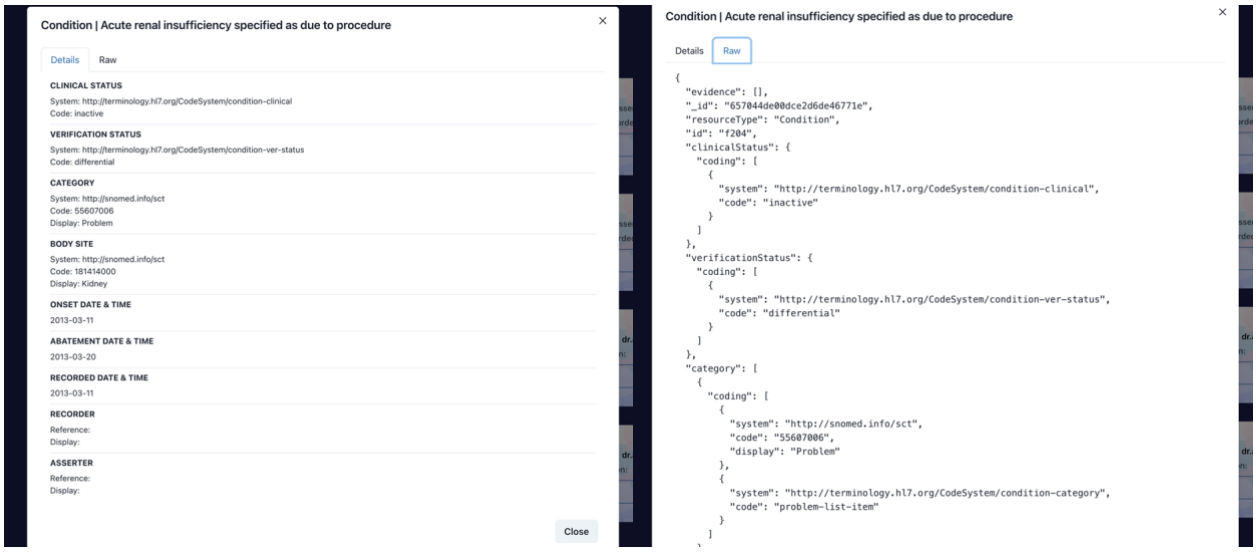

Figure 0.24 Condition Details Modal Interface

4.2.7.5. Allergies List Module

The allergies list module comprises of allergies card that consists of the allergy name, type, criticality, status, asserter name, and recorded date. Figures 4.25 and 4.26 show the interface of the allergies list module and modals.

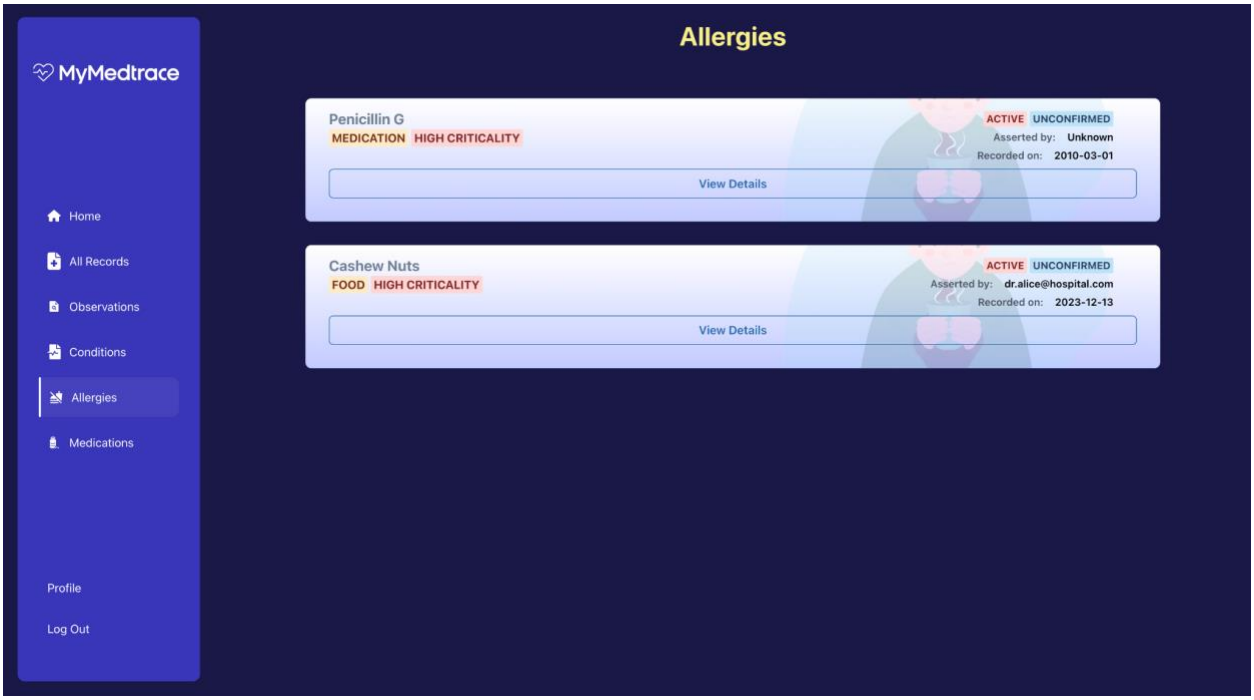

Figure 0.25 Allergies List Interface

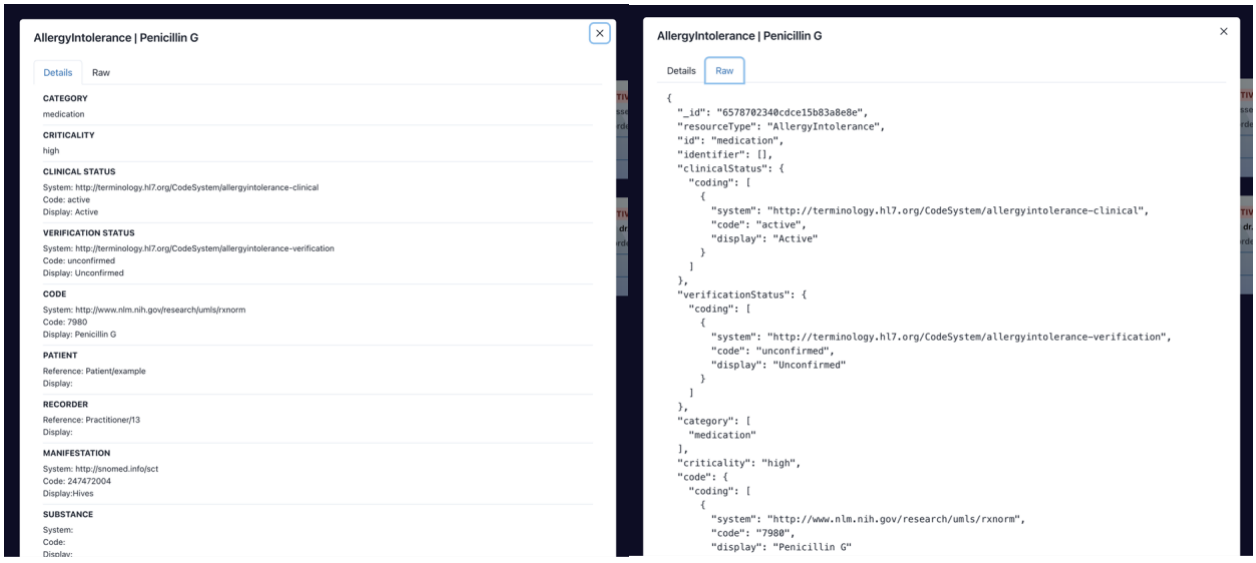

Figure 0.26 Allergy Details Modal Interface

4.2.7.6. Medications List Module

The medications list module comprises of medications card that consists of the medication name, status, and recorded date. Figures 4.27 and 4.28 show the interface of the medications list module and modals.

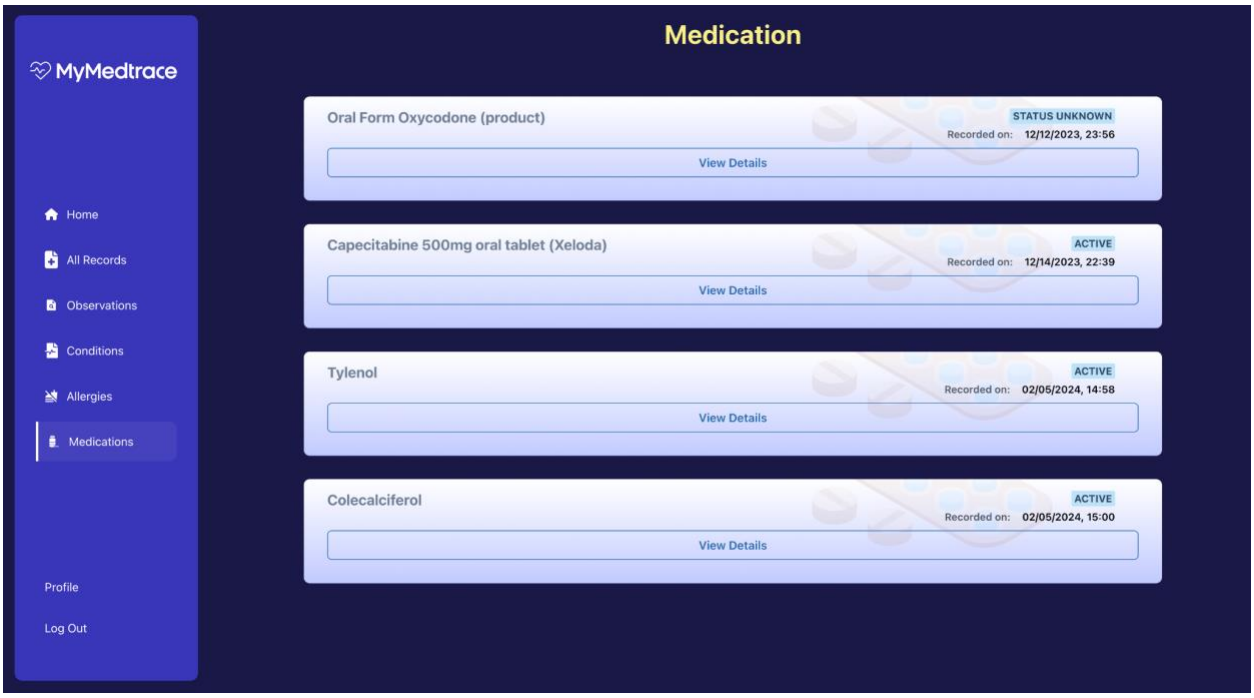

Figure 0.27 Medications List Interface

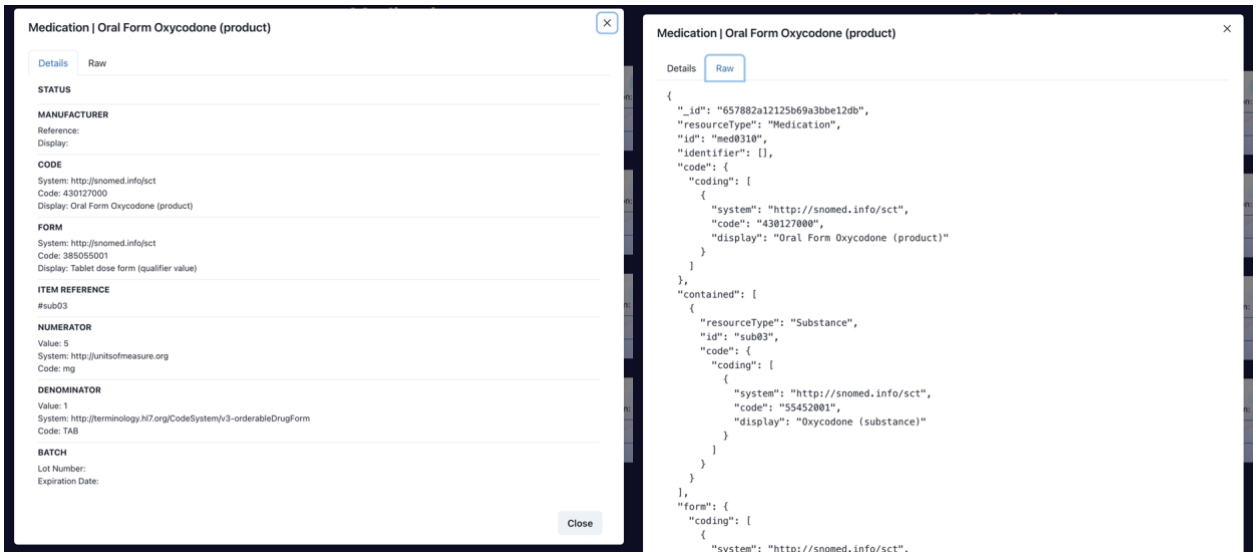

Figure 0.28 Medication Details Modal Interface

4.3. Backend

The backend or server side is responsible for processing the requests from the client side, managing the data and performing each business logic. It communicates with the blockchain and off-chain database to perform data retrieval or manipulation based on the functional request from

the client and return the requested response to the client side. The process of request and data validation, security, and authentication are also handled by the server.

For the implementation, a REST API (Representational State Transfer Application Programming Interface) is utilized to communicate between the client and server side. The REST API consists of standard HTTP methods (GET, POST, PUT, DELETE) that performs action on the resources, and can be represented in specific intended format such as JSON (JavaScript Object Notation). Furthermore, it is also stateless, meaning that every request also contains all the information necessary for the client side and a client state is not persisted on the server side.

The backend service utilizes Express.js as the web framework for building the APIs. It integrates well with the frontend framework, is highly flexible, and provides simplicity in design. To ensure that each requests and responses format are standardized, especially the format of the medical record data, a Typescript language and type library for the FHIR data is also used to enforce strict data types.

4.3.1. Login

In the login system, the server is responsible to perform a validation on the user's credential by utilizing the hashing comparison between the requested value and retrieved value from the database. Once the credential is validated, a JWT (JSON Web Token) is generated and formed into a bearer token with a specific expiry time and is sent to the client as an authorization cookie attached in the header. Using this time-sensitive authorization cookie that is valid for five days, a session user is able to be maintained on the client side. Furthermore, to ensure only authorized requests are sent from client side, every endpoint related to the data request needs to be encapsulated with the authorization cookie in the request header. Figure 4.29 and 4.30 shows the login API example using the POST request and the authorization cookie response respectively. To simplify the current development, the password sent in the body request is not being encrypted, and further encryption techniques such as asymmetric encryption using public and private key, or symmetric encryption using single secret key can be implemented to protect the data transfer.

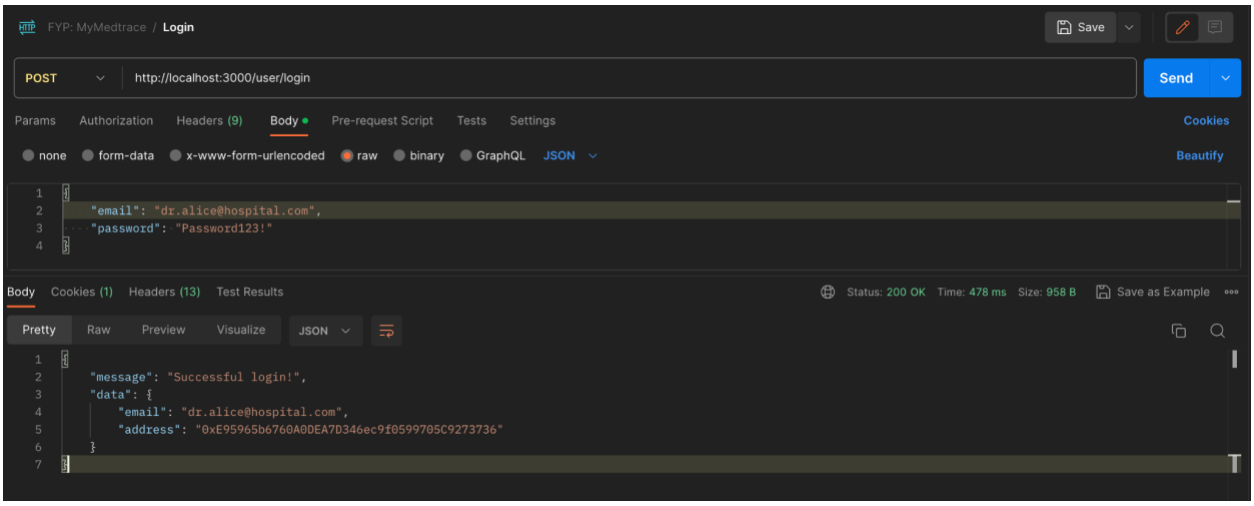

Figure 0.29 Login API Request and Response

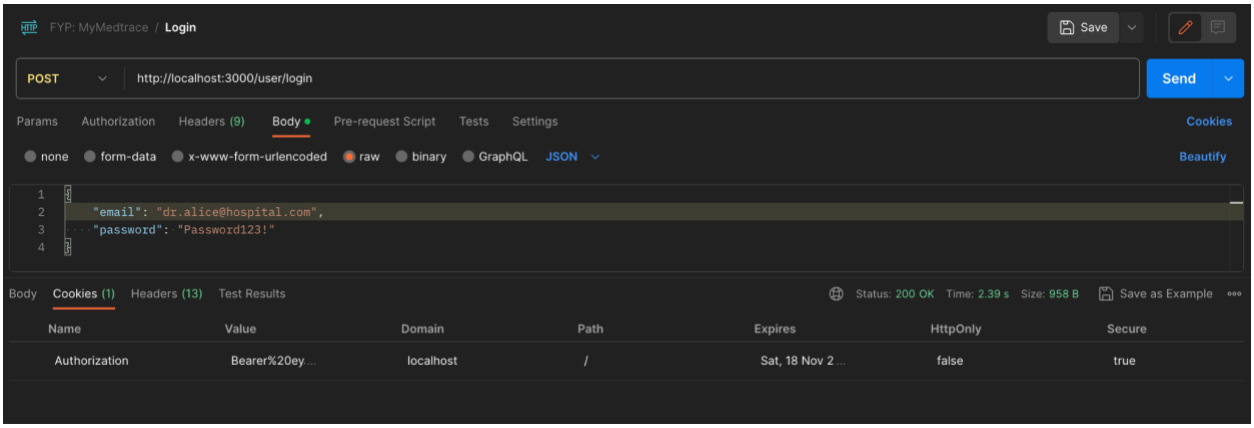

Figure 0.30 Login API Authorization Cookie Response

4.3.2. Sign Up

In the sign up API, the user's credentials including primary information and user type specific information are being validated before sent to the database. The user's password is being hashed using the *bcrypt* hash library with 10 salt rounds before being sent to the database. If the user type is doctor or patient, the information is also sent to the blockchain to be stored, while if the user type is administrator, the credentials are only stored in the off-chain database. Upon successful validation and signing up, a JWT token is also generated and sent to the client as an authorization cookie. Figure 4.31 and 4.32 shows the sign up API example using the POST request and the authorization cookie response respectively.

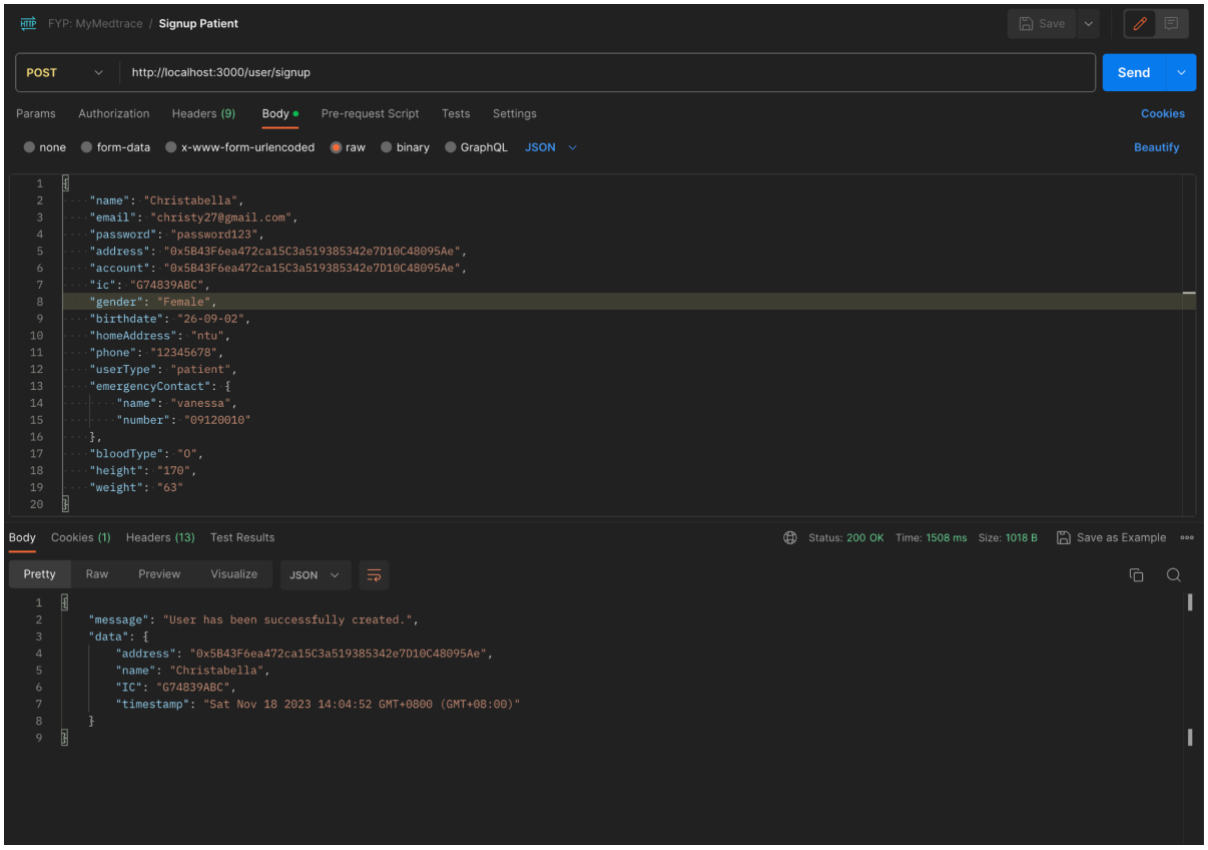

Figure 0.31 Sign Up API Request and Response

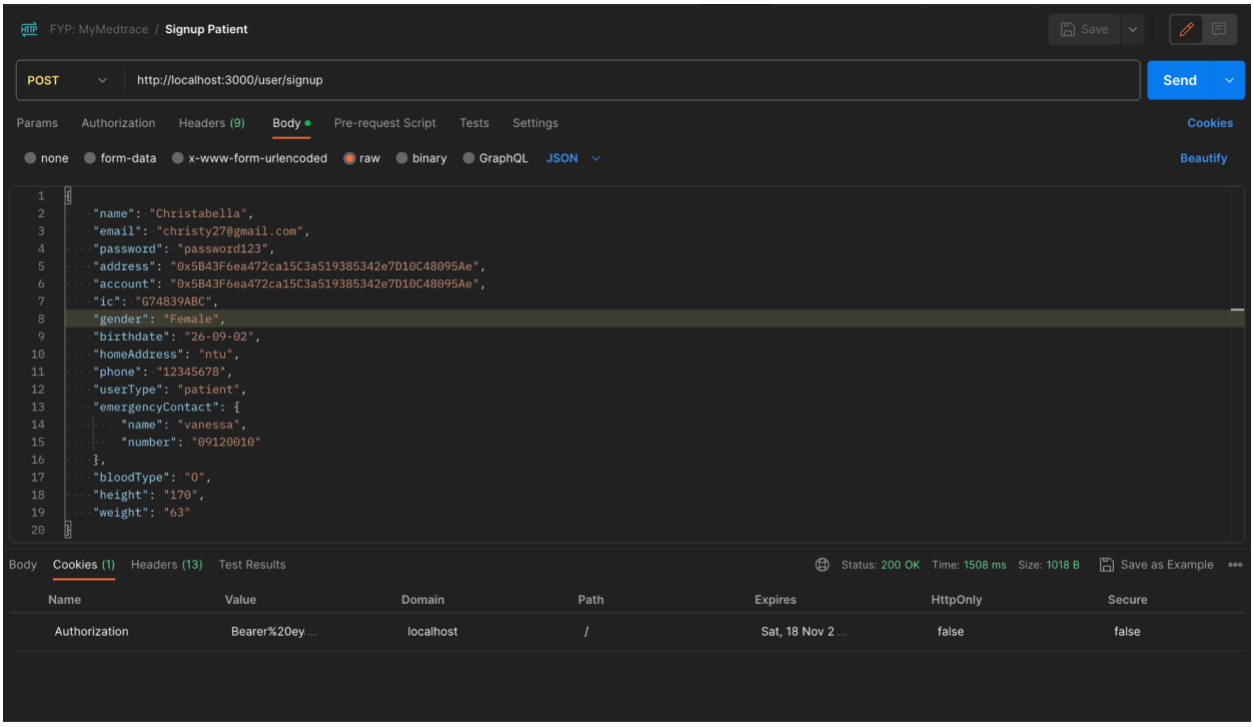

Figure 0.32 Sign Up API Authorization Cookie Response

4.3.3. Whitelist

The whitelist API serves the client request of enabling a specific doctor address to perform an action on a specific patient address. Additionally, the admin wallet address is also passed in the body request to sign the blockchain transaction. The API only interacts with the blockchain by adding the doctor address to the whitelisted doctor list in the patient's contract. Figure 4.33 shows the whitelist API using POST request and the example of patient's data after a successful whitelist.

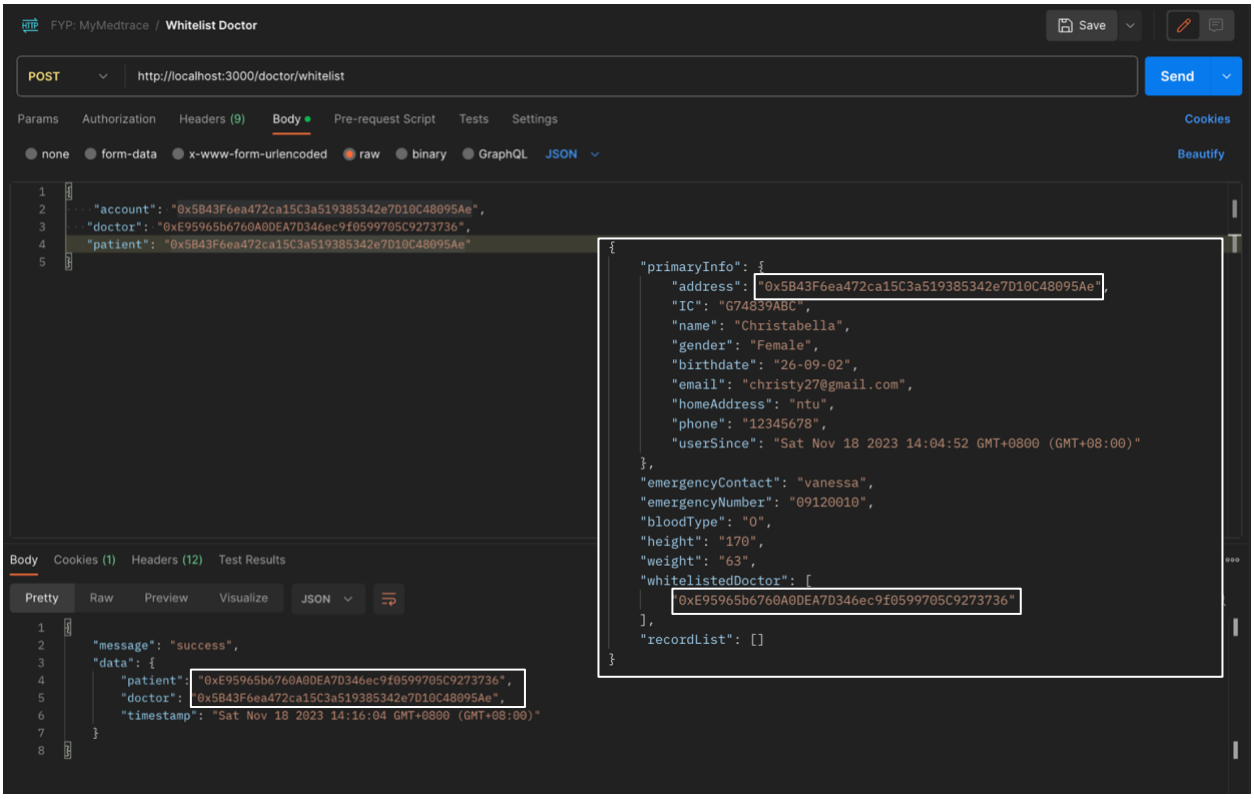

Figure 0.33 Whitelist API Request and Response Example and Patient's Data After Whitelist

4.3.4. Create Observation

The create observation API interacts with both the blockchain and off-chain database. The server will first validate if the patient exists and if the doctor is whitelisted. Upon success validation, the server generates a unique MongoDB object ID and performs a hash on the observation data. Further, it sends a transaction of the hashed data that is signed using the sender (doctor) address to the blockchain to create a new record and add it to the patient's record list attribute. The server

also sends a query to the database to create a new entry to the observation collection that consists of the full observation data with the same unique ID.

4.3.5. Get All Observations

Figure 4.34 shows the get observations API using a GET request that returns all the observation data types from the Observation collection in the database. The response is in a JSON format of a standardized Observation FHIR data type.

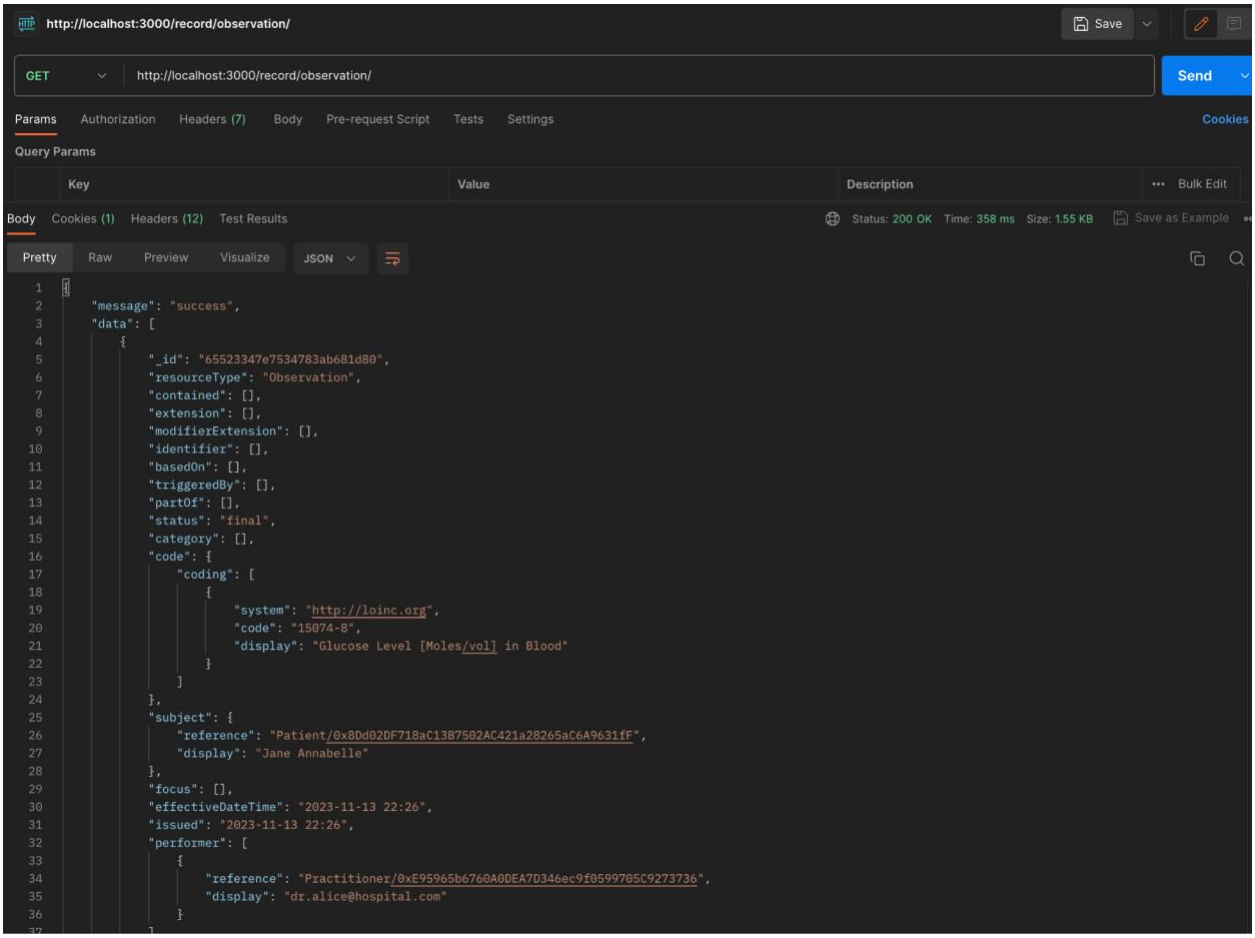

Figure 0.34 Get Observations API Response

4.3.6. Get Observation by ID

The API allows retrieving observations with a specific unique ID. The server interacts with both the blockchain and off-chain database to validate if there is any data discrepancy between the

blockchain and off-chain data. Figure 4.35 shows the get observation by ID example using the GET request.

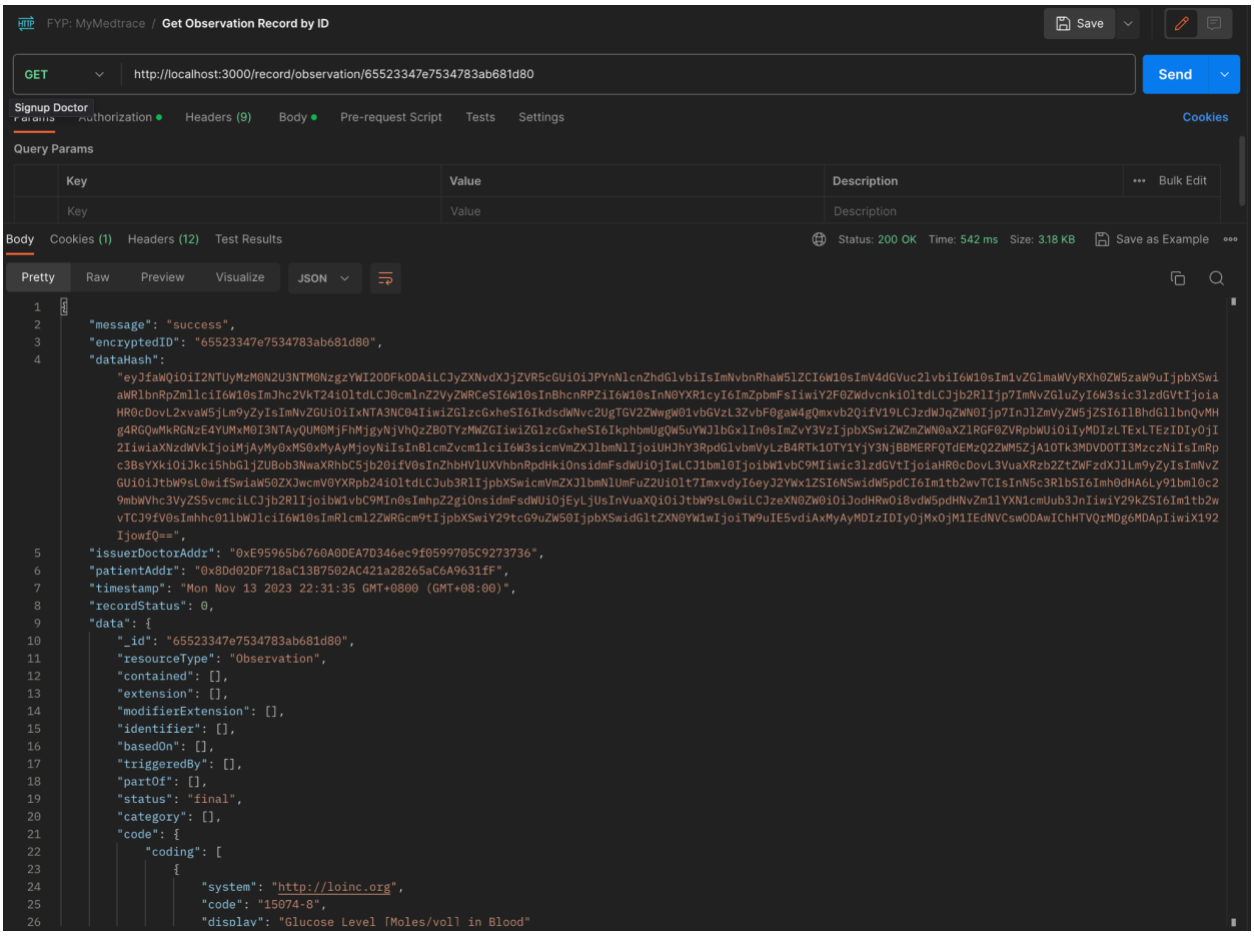

Figure 0.35 Get Observation By ID API Response

4.3.7. Get Observation Records of Patient by Patient Address

The API allows client to retrieve observation records of a specific patient. The server interacts with the blockchain and database and returns a response consisting of the observation metadata from blockchain such as issuer doctor address, patient address, creation timestamp, record status and the full data from the off-chain database. Figure 45 shows the get observation record of specific patient API response examples using the GET request.

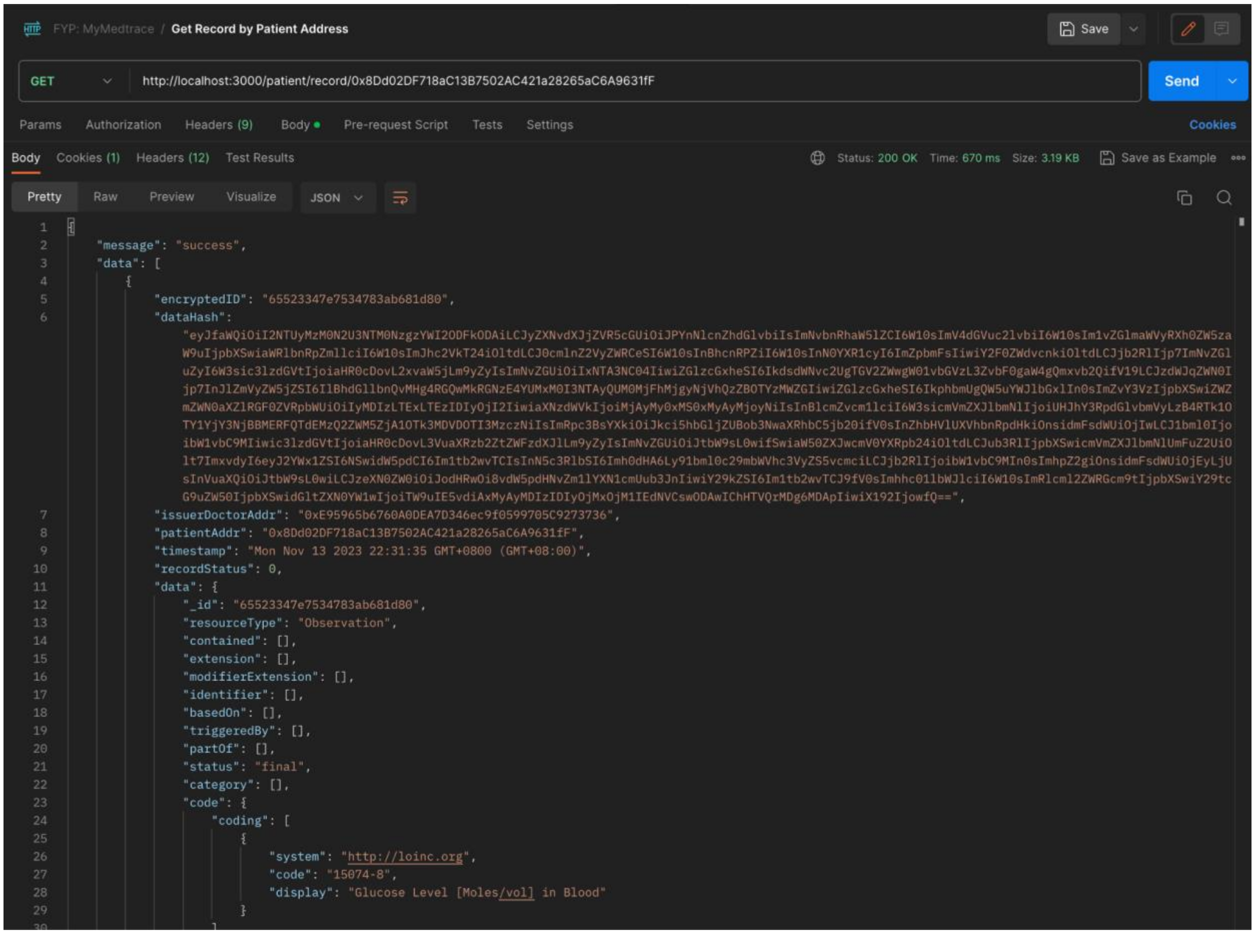

Figure 0.36 Get Observation By ID API Response

4.3.8. Create Condition

The create condition API interacts with both the blockchain and off-chain database. The server will first validate if the patient exists and if the doctor is whitelisted. Upon success validation, the server generates a unique MongoDB object ID and performs a hash on the condition data. Further, it sends a transaction of the hashed data that is signed using the sender (doctor) address to the blockchain to create a new record and add it to the patient's record list attribute. The server also sends a query to the database to create a new entry to the condition collection that consists of the full condition data with the same unique ID.

4.3.9. Get All Conditions

Figure 4.37 shows the get all conditions API using a GET request that returns all the condition records from the Condition collection in the database. The response is a JSON format of a standardized Condition FHIR data type.

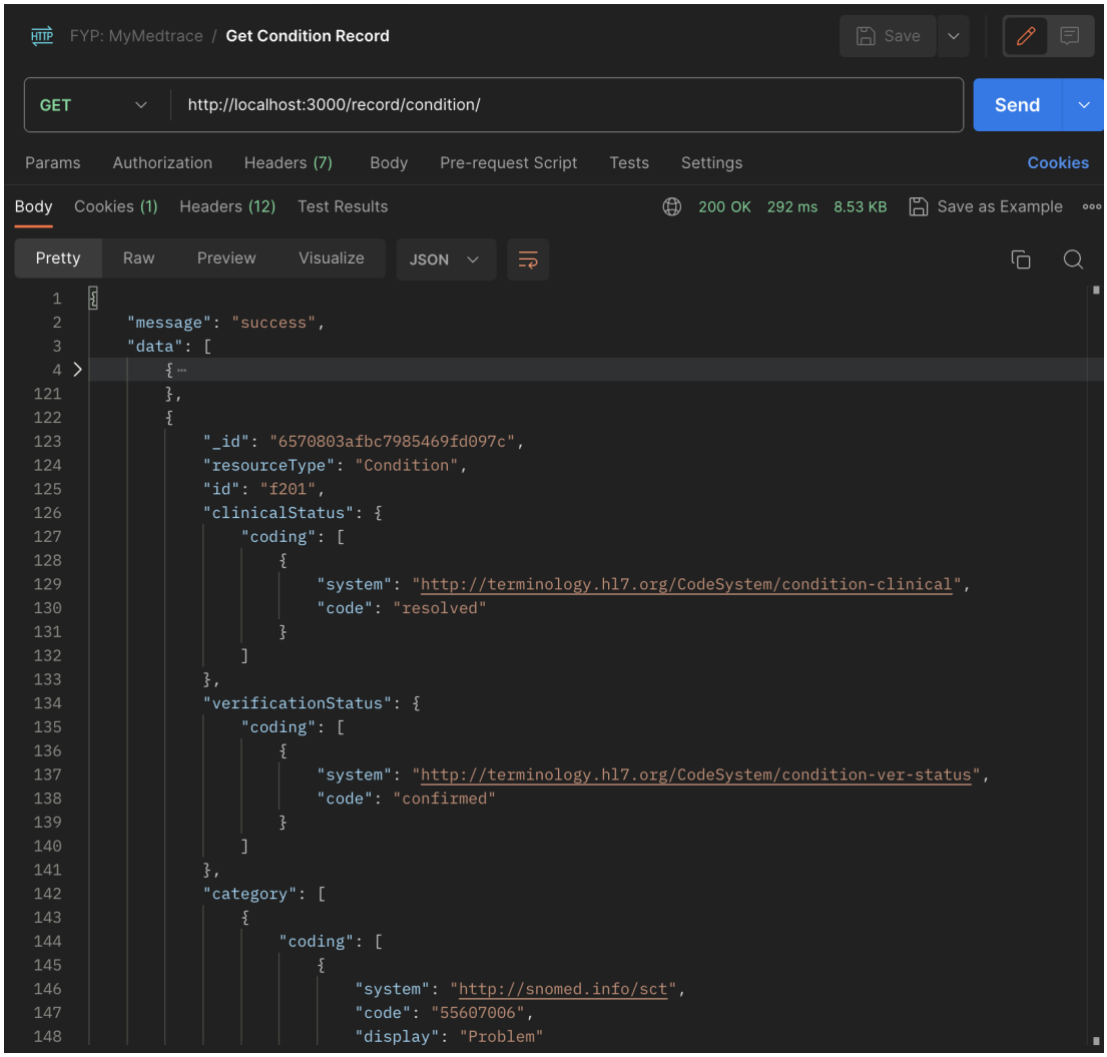

Figure 0.37 Get All Conditions API Response

4.3.10. Get Condition by ID

The API allows retrieving a condition with a specific unique ID. The server interacts with both the blockchain and off-chain database to validate if there is any data discrepancy between the blockchain and off-chain data by comparing the hashed values of both versions. Figure 4.38 shows the get a condition by ID example using the GET request.

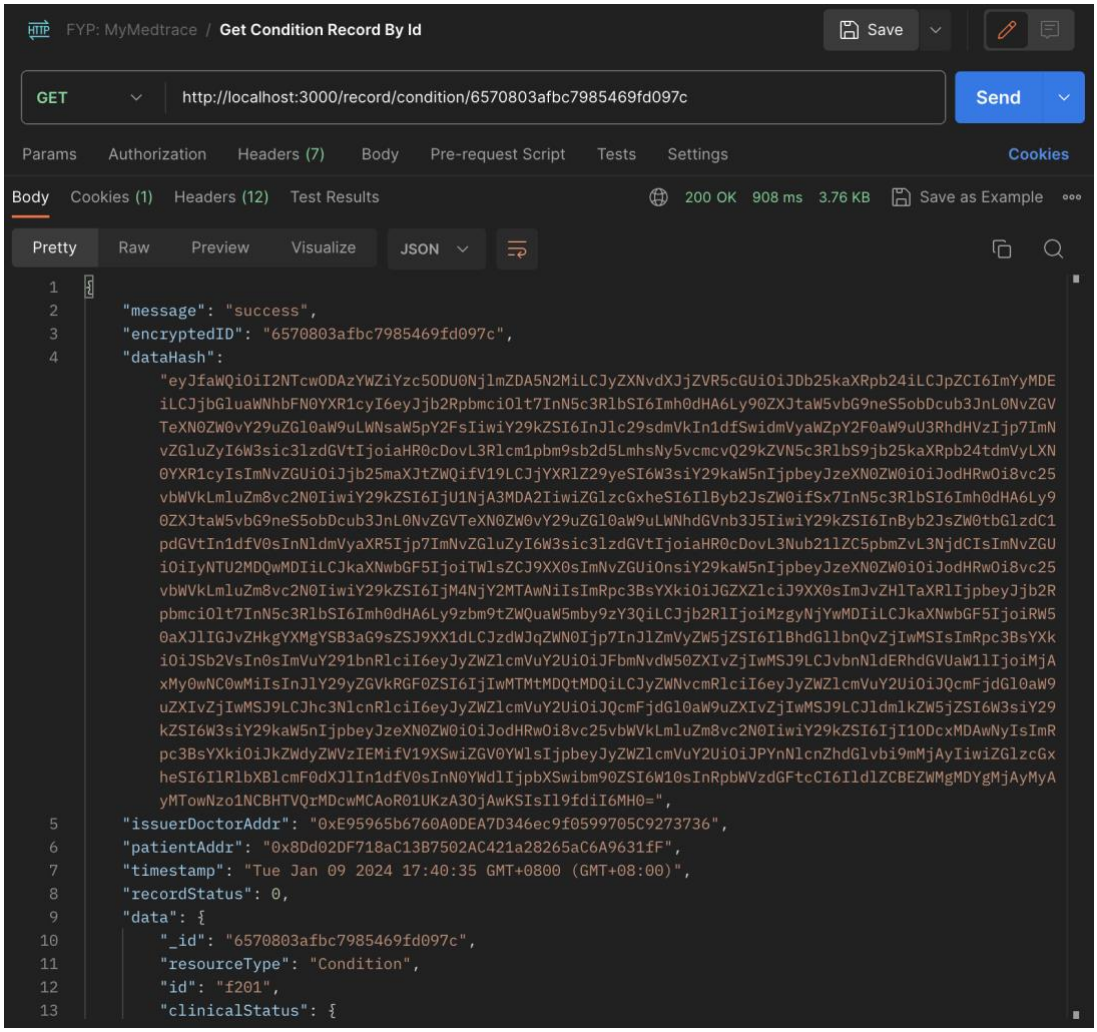

Figure 0.38 Get Condition By ID API Response

4.3.11. Get Condition Records of Patient by Patient Address

The API allows retrieving a condition list of patients with specific patient blockchain address. The server interacts with the blockchain and database and returns a response consisting of the conditions metadata from the blockchain such as issuer doctor address, patient address, creation timestamp, record status, and the full data from the off-chain database. Figure 4.39 shows the get condition records of specific patient ID API response examples using the GET request.

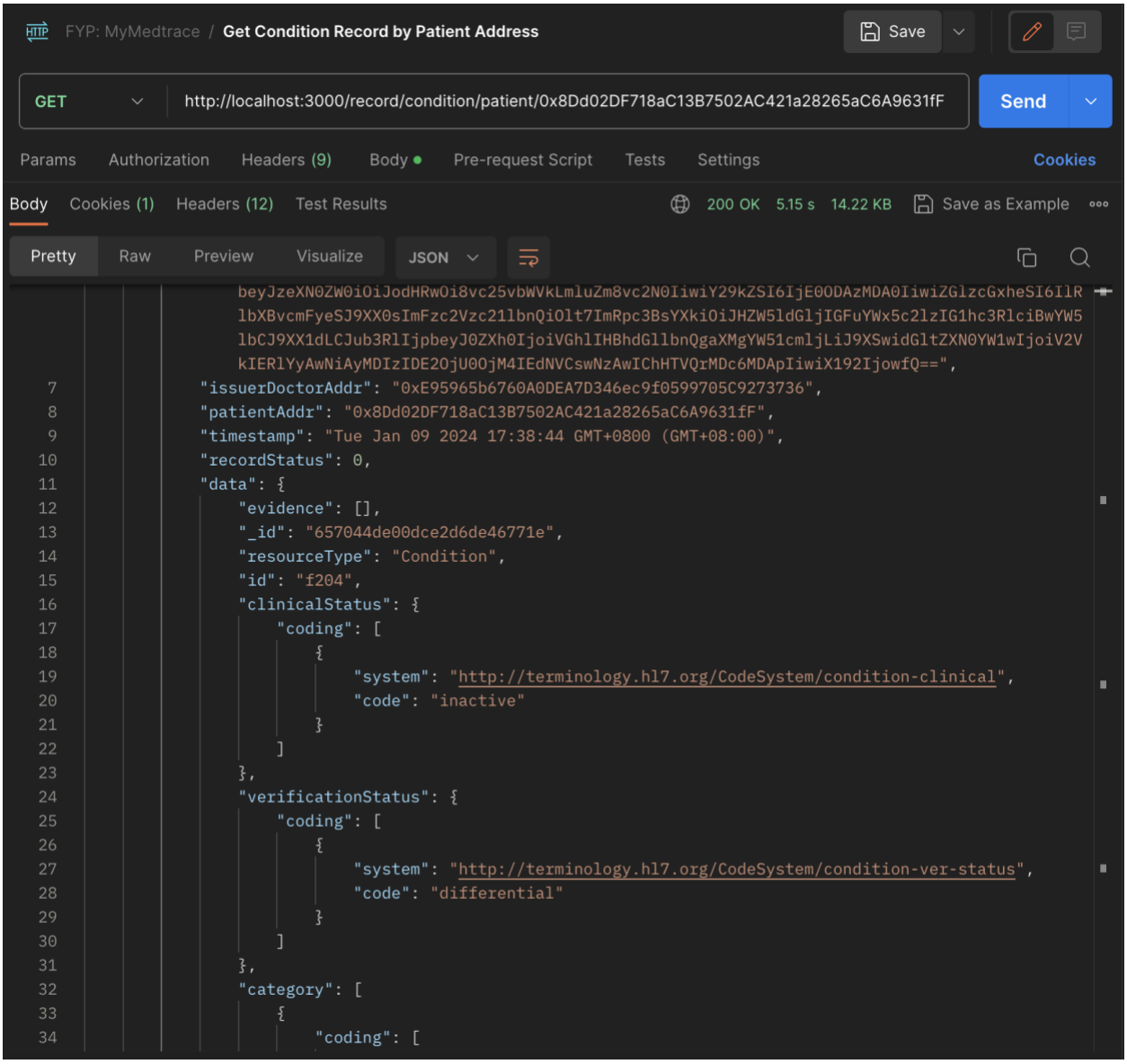

Figure 0.39 Get Condition By ID API Response

4.3.12. Create Allergy

The create allergy API interacts with both the blockchain and off-chain database. The server will first validate if the patient exists and if the doctor is whitelisted. Upon success validation, the server generates a unique MongoDB object ID and performs a hash on the allergy data. Further, it sends a transaction of the hashed data that is signed using the sender (doctor) address to the blockchain to create a new record and add it to the patient's record list attribute. The server also

sends a query to the database to create a new entry to the allergy collection that consists of the full allergy data with the same unique ID.

4.3.13. Get All Allergies

Figure 4.40 shows the get all allergies API using a GET request that returns all the allergy records from the Allergy collection in the database. The response is a JSON format of a standardized Allergy FHIR data type.

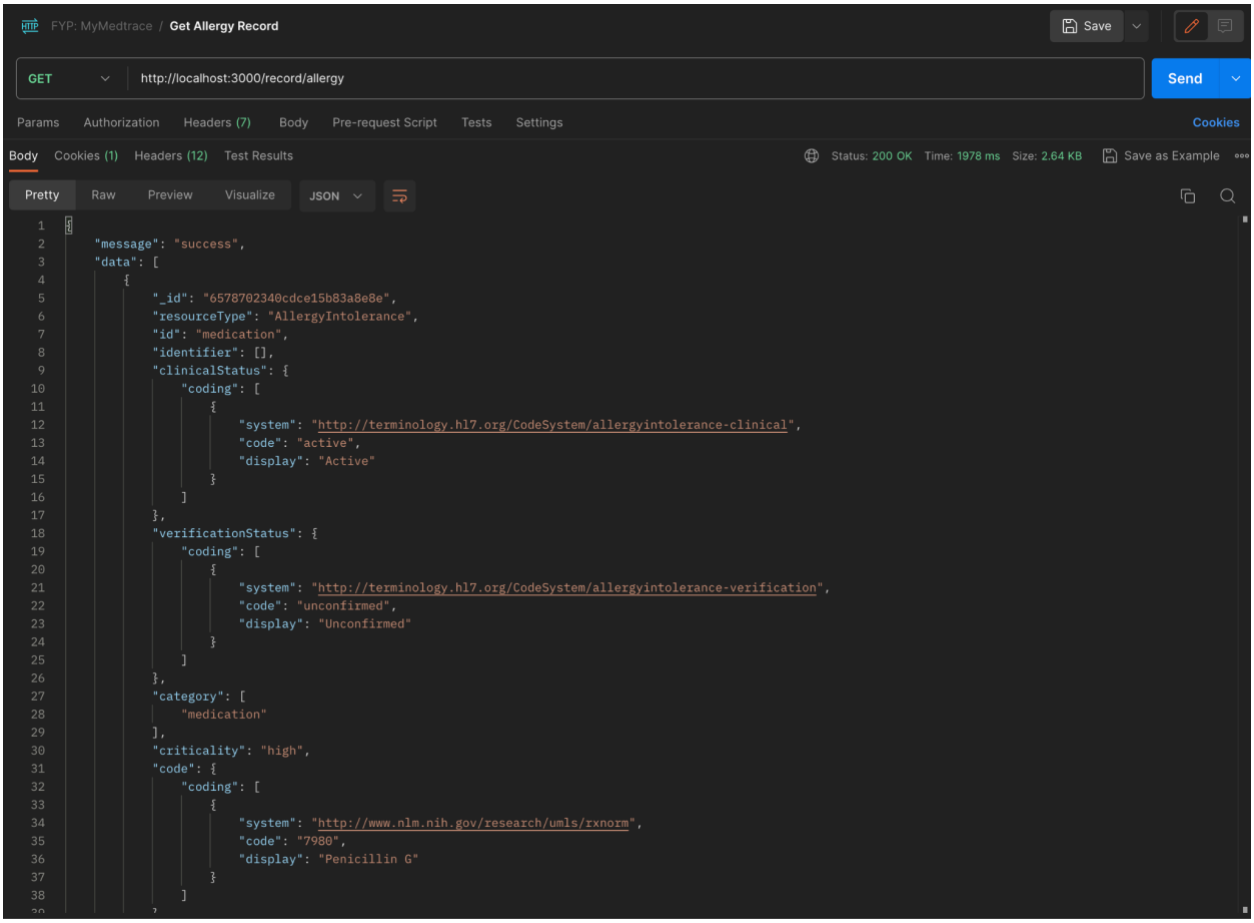

Figure 0.40 Get All Allergies API Response

4.3.14. Get Allergy by ID

The API allows retrieving an allergy with a specific unique ID. The server interacts with both the blockchain and off-chain database to validate if there is any data discrepancy between the

blockchain and off-chain data by comparing the hashed values of both versions. Figure 4.41 shows the get an allergy by ID example using the GET request.

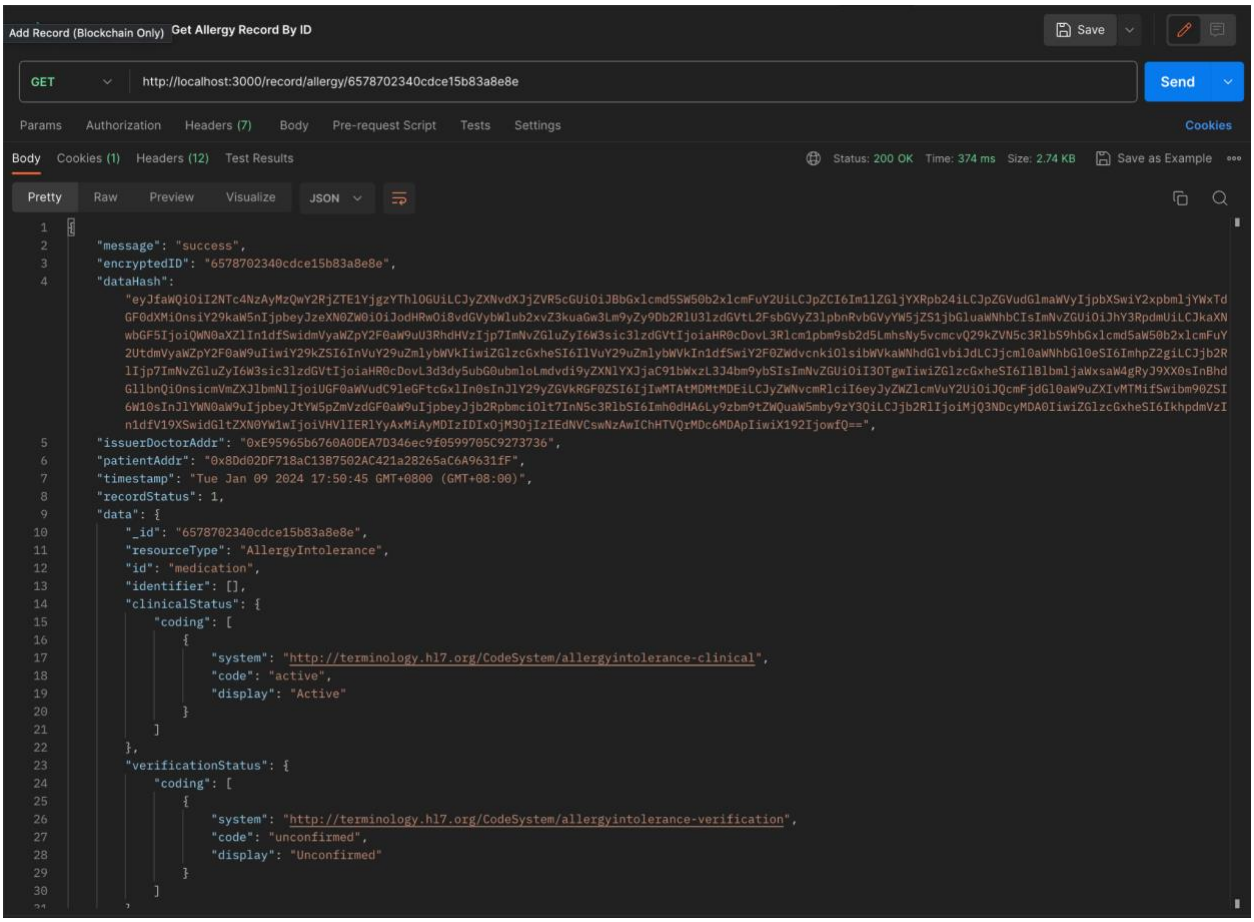

Figure 0.41 Get Allergies By ID API Response

4.3.15. Get Allergy Records of Patient by Patient Address

The API allows retrieving an allergy list of patients with specific patient blockchain addresses. The server interacts with the blockchain and database and returns a response consisting of the allergy metadata from the blockchain such as issuer doctor address, patient address, creation timestamp, record status, and the full data from the off-chain database. Figure 4.42 shows the allergy records of specific patient ID API response examples using the GET request.

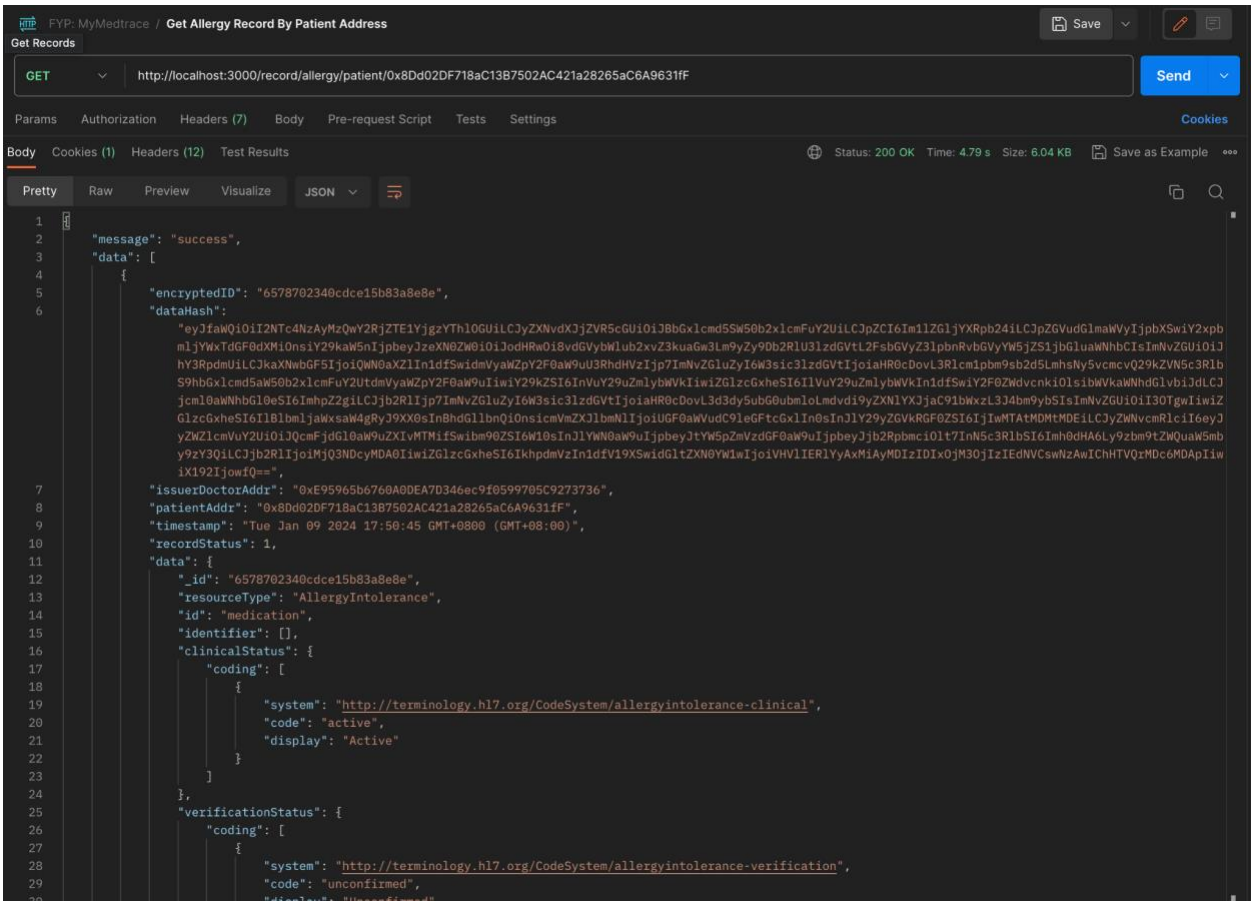

Figure 0.42 Get Allergies By ID API Response

4.3.16. Create Medication

The create medication API interacts with both the blockchain and off-chain database. The server will first validate if the patient exists and if the doctor is whitelisted. Upon success validation, the server generates a unique MongoDB object ID and performs a hash on the medication data. Further, it sends a transaction of the hashed data that is signed using the sender (doctor) address to the blockchain to create a new record and add it to the patient's record list attribute. The server also sends a query to the database to create a new entry to the medication collection that consists of the full medication data with the same unique ID.

4.3.17. Get All Medications

Figure 4.43 shows the get all medications API using a GET request that returns all the medication records from the Medications collection in the database. The response is a JSON format of a standardized Medication FHIR data type.

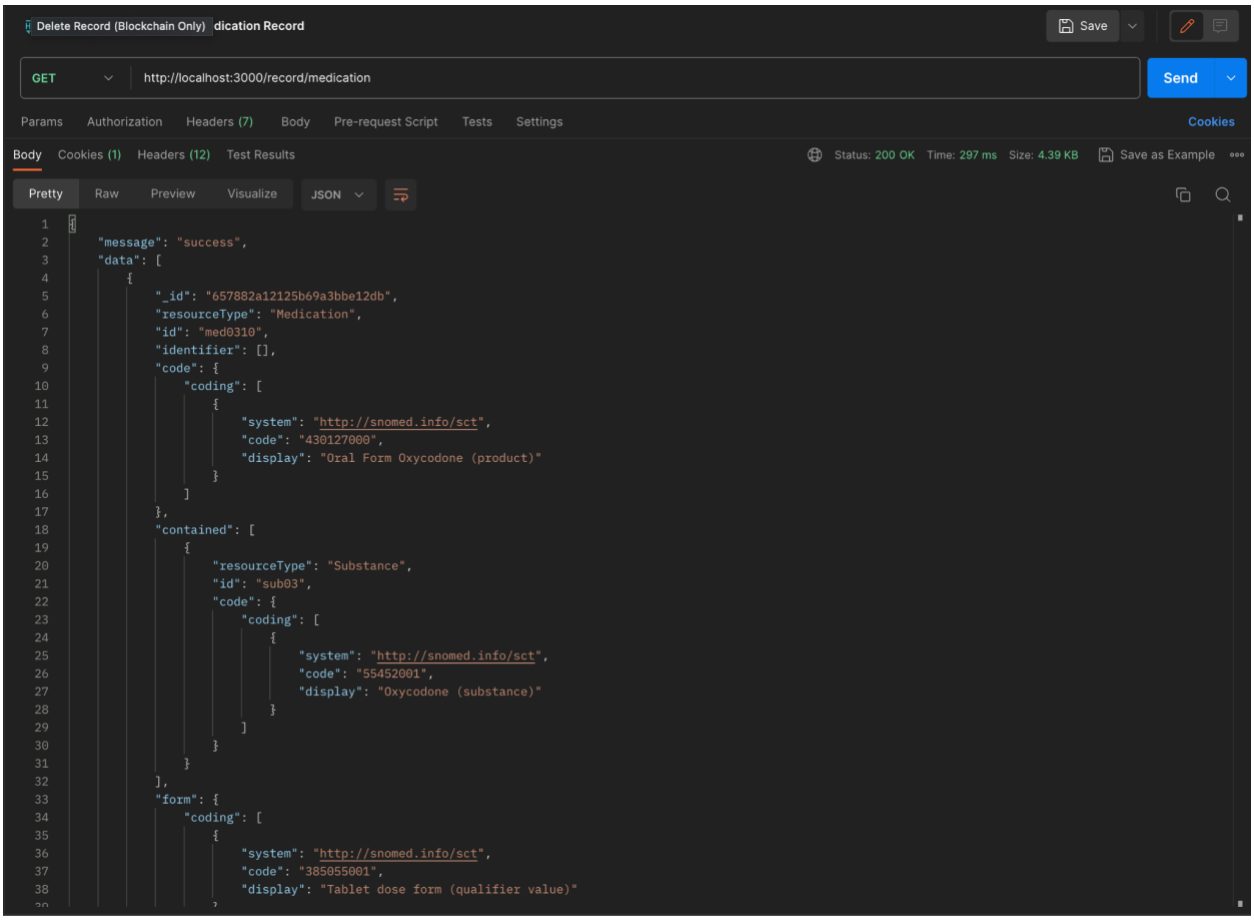

Figure 0.43 Get All Medications API Response

4.3.18. Get Medications by ID

The API allows retrieving a medication with a specific unique ID. The server interacts with both the blockchain and off-chain database to validate if there is any data discrepancy between the blockchain and off-chain data by comparing the hashed values of both versions. Figure 4.44 shows the get a medication by ID example using the GET request.

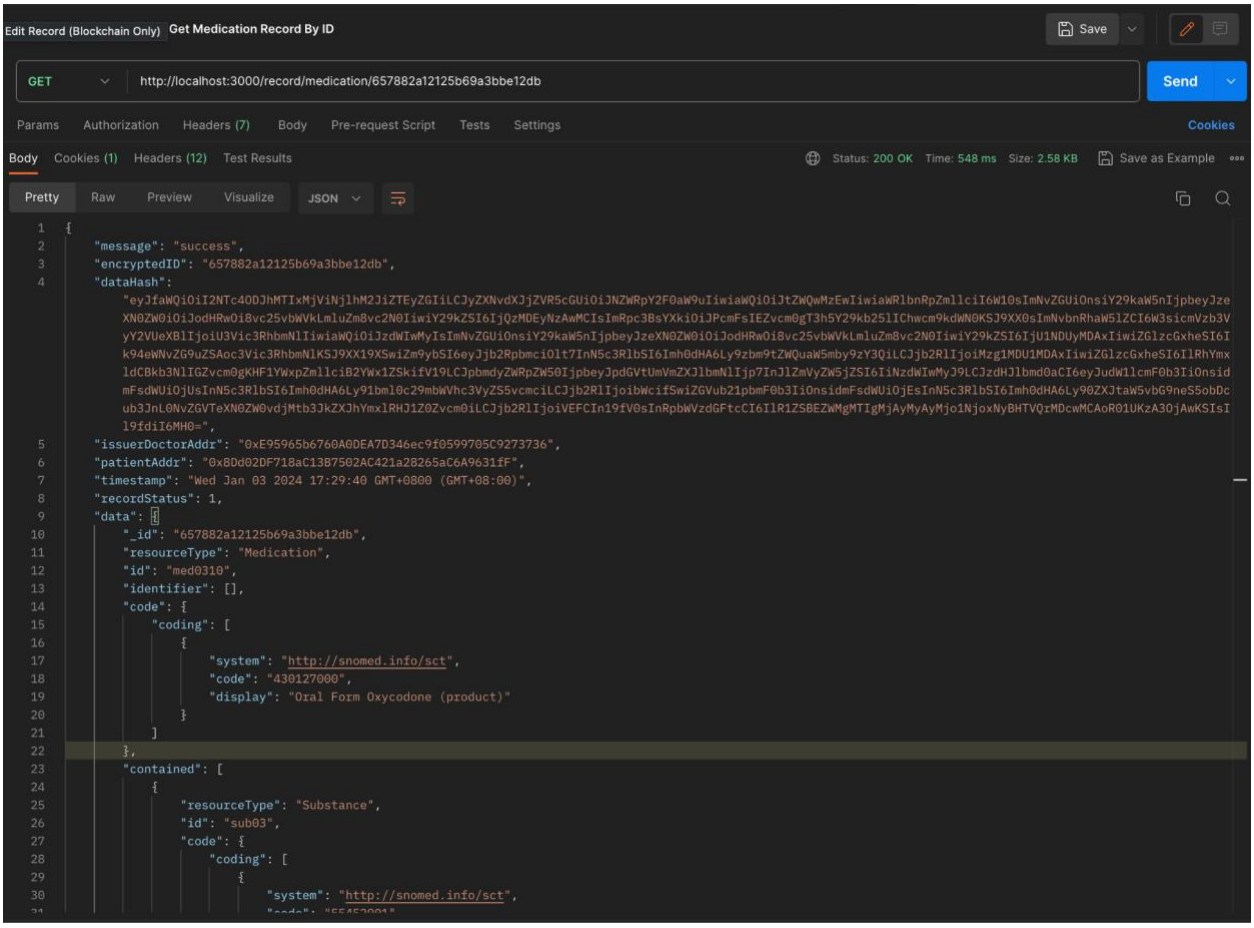

Figure 0.44 Get Medications By ID API Response

4.3.19. Get Medication Records of Patient by Patient Address

The API allows retrieving a medication list of patients with specific patient blockchain addresses. The server interacts with the blockchain and database and returns a response consisting of the medication metadata from the blockchain such as issuer doctor address, patient address, creation timestamp, record status, and the full data from the off-chain database. Figure 4.45 shows the medication records of specific patient ID API response examples using the GET request.

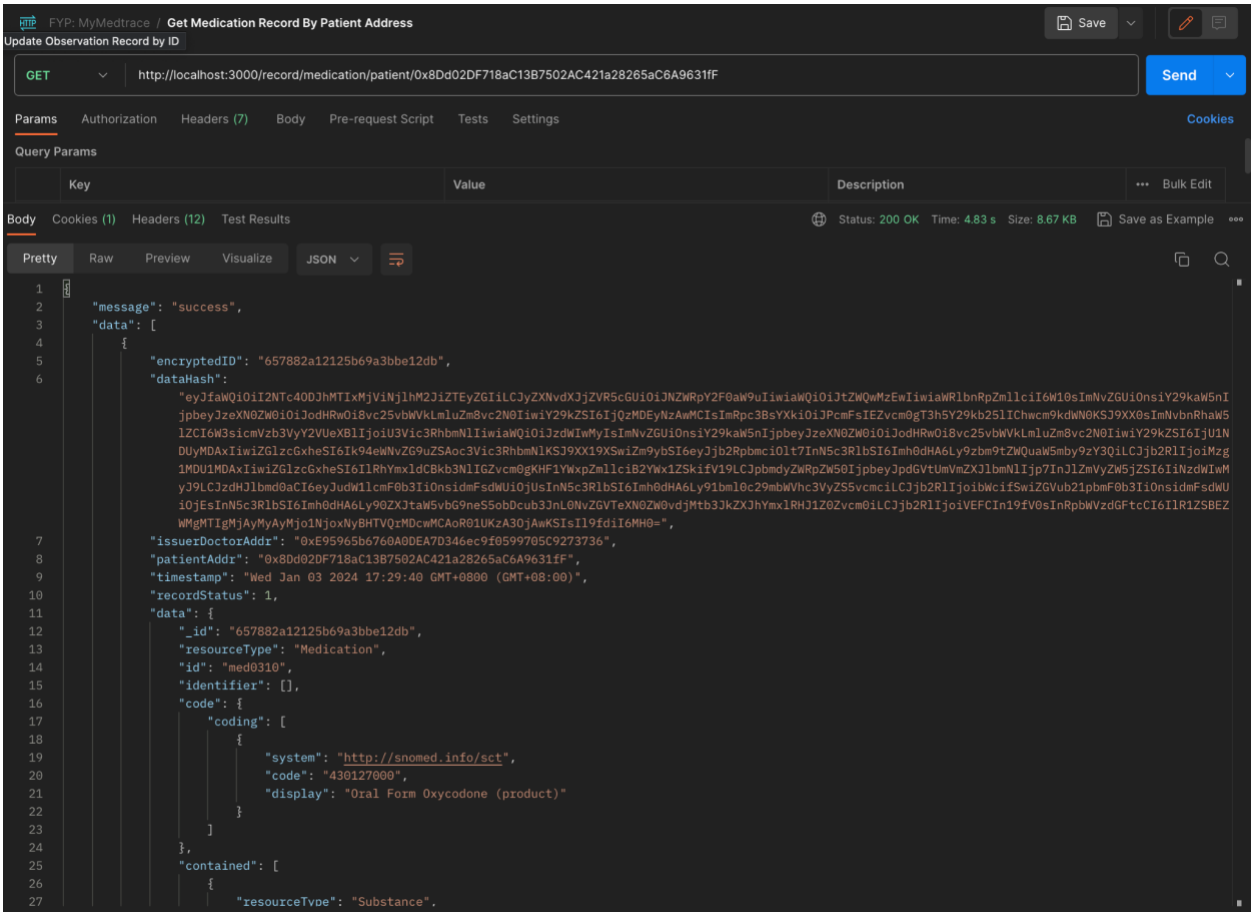

Figure 0.45 Get Medications By Patient Address API Response

4.3.20. Get All Patients

The API allows the client to retrieve all the patients lists from the blockchain. Figure 4.46 shows the response to the GET request that returns the total patients and metadata of each patient such as the patient's blockchain address, identification card number, name, gender, date of birth, email, home address, phone number, registration date, emergency contact number and name, blood type, height, weight, whitelist doctor, and record ID list.

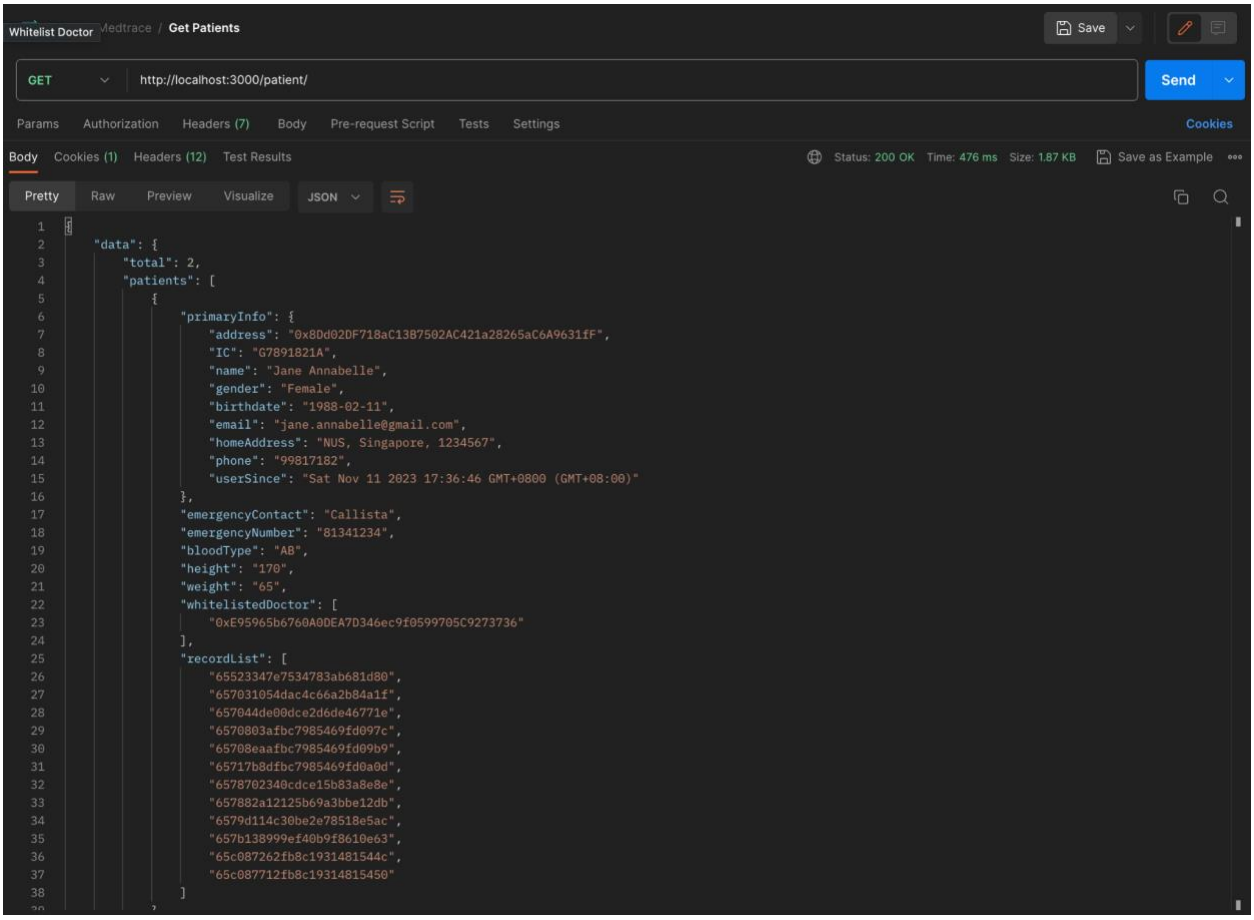

Figure 0.46 Get All Patients API Response

4.3.21. Get Patients of Doctor by Doctor Address

The API allows the client to retrieve the patients list of doctor with specific blockchain address. Figure 4.47 shows the response to the GET request consisting of the same structure as the get all patients API in the previous section.

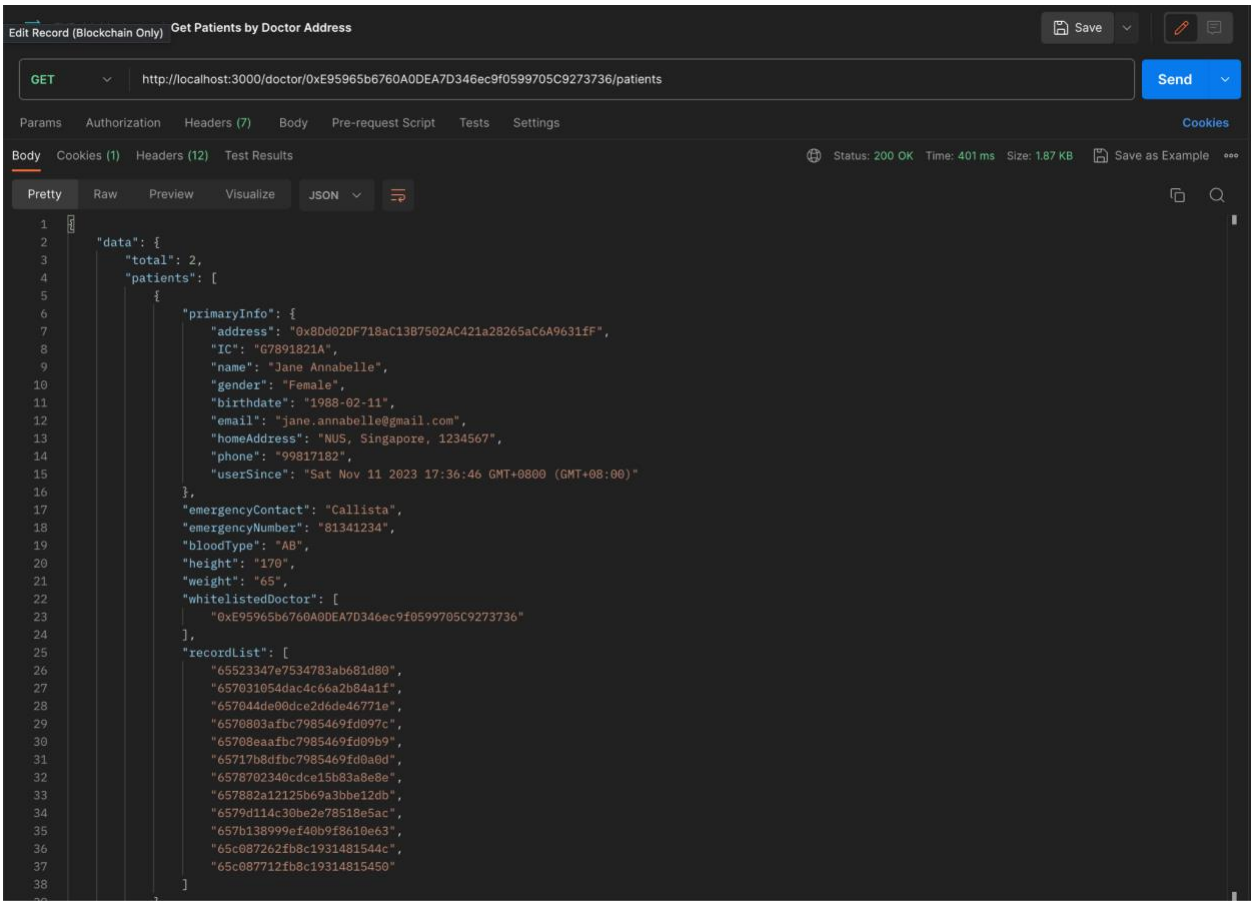

Figure 0.47 Get All Patients by Doctor Address API Response

4.3.22. Get All Doctors

The API allows the client to retrieve the list of doctors from the blockchain storage. Figure 4.48 shows the response to the GET request that consists of the doctor's blockchain address, identification card number, name, gender, date of birth, email, home address, phone number, user registration date, qualification, and major.

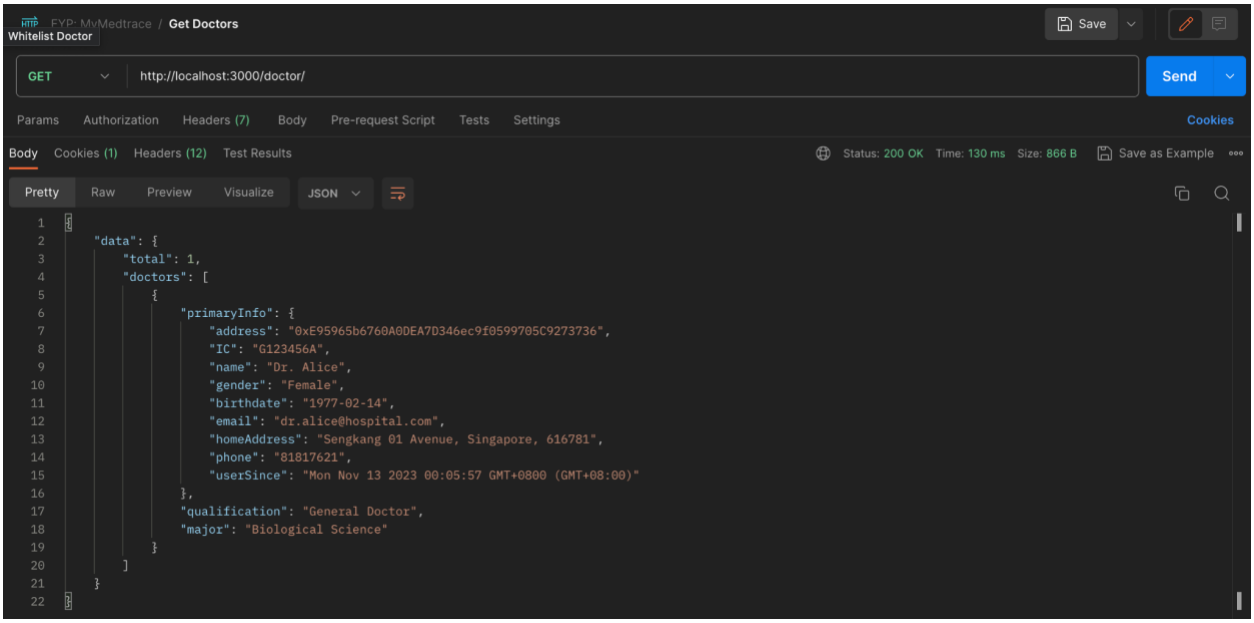

Figure 0.48 Get All Doctors API Response

4.4. Smart Contract

The smart contract is built using the Solidity language and Ethereum framework. A compiler namely Truffle [24] is used to compile the smart contract. A local blockchain emulator, Ganache [25], is also utilized to ease the smart contract development process, testing, and debugging. Ganache includes several account addresses that are pre-loaded with Ether (ETH), a native cryptocurrency for Ethereum platform, that enables it to easily simulate transactions. There are 2 smart contracts built in this project, the Main Contract and Record Contract.

4.4.1. Main Contract

The main contract serves as the database for the users. Figure 4.49 shows the overview of the main contract viewed using Ganache platform. It comprises six mappings which are doctor list, patient list, doctor addresses, patient addresses, *isDoctor*, and *isPatient*. The doctor list and patient list mappings maintain an address to doctor and patient struct respectively. The doctor addresses and patient addresses maintain an index to address mappings, and the *isDoctor* and *isPatient* maintain an address to Boolean mappings. These mappings are useful for storing the patients and doctors list, as well as perform an efficient validation.

There are three different structures in the main contract. The primary one is user struct that comprises the user's wallet address, IC number, name, gender, date of birth, email, home address, phone number, and timestamp of the user establishment. The doctor struct comprises primary information in the form of user struct, emergency contact name, emergency contact number, blood type, height, and weight, an array of whitelisted doctor, and an array of record list. Lastly, the doctor struct comprises primary information in the form of user struct, qualification and degree information.

Furthermore, there are also several functions that support the query of the blockchain database. The patient related functions are to create patient, edit patient details, get patient details, and delete patient. The doctor related functions are create doctor, edit doctor details, get doctor details, delete doctor, and whitelist doctor. When these functions are invoked, they provide modifications to the related mappings.

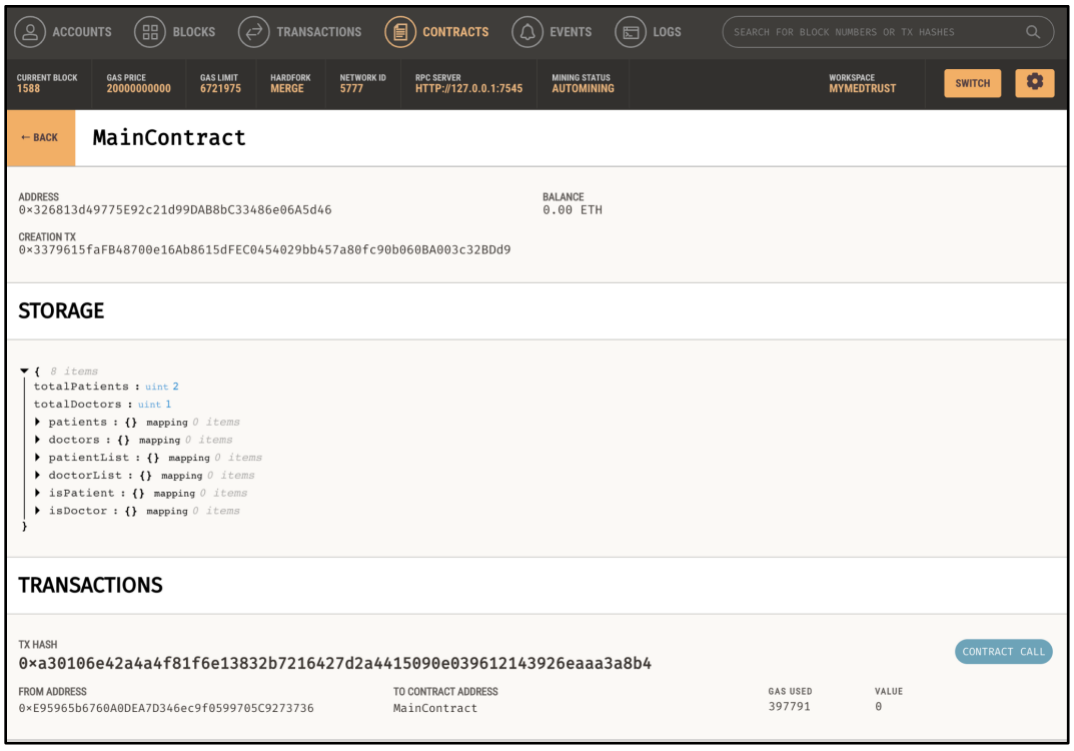

Figure 0.49 Main Contract Viewed using Ganache

4.4.2. Record Contract

The record contract serves as a database for the patient's record related information. Figure 4.50 shows the overview of the record contract viewed using Ganache platform. It includes a mapping namely record list that maintains an unique ID string to record struct mapping. There is also a record struct that comprises unique ID, hashing value of the record data, issuer doctor address, patient address, record creation timestamp, and the status of the record which is an enumeration of pending, completed, or declined status.

Additionally, there are record related functions that perform modifications to the record list mapping. They are mainly the basic CRUD (Create, Edit, Update, Delete) functions such as create record, edit record, get record details, and remove record.

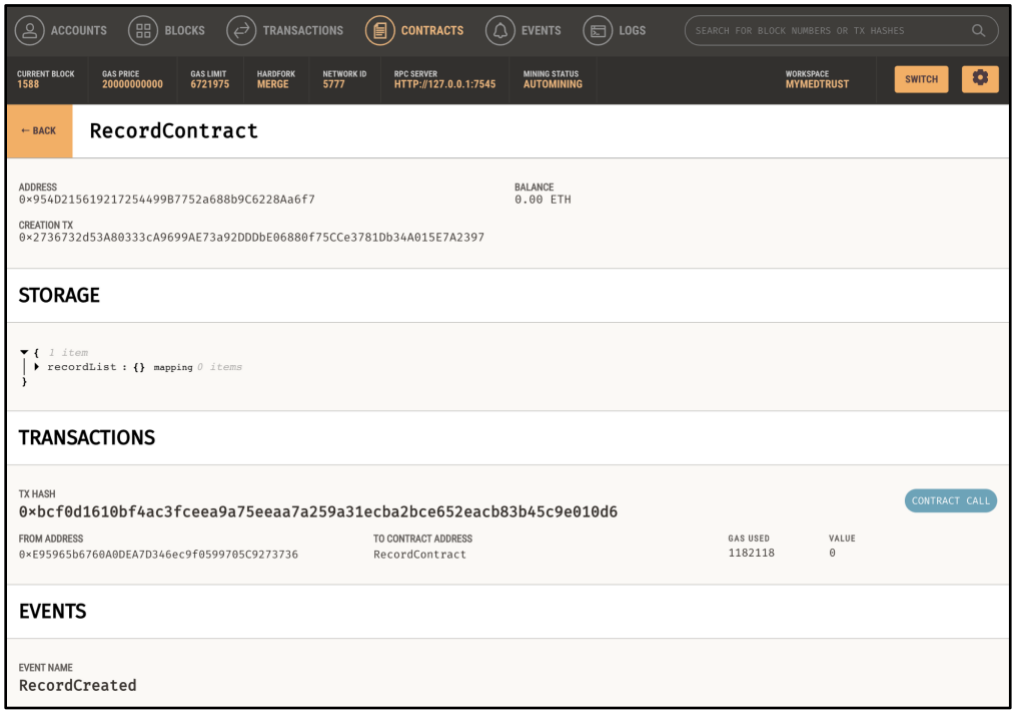

Figure 0.50 Record Contract Viewed using Ganache

4.5. Off-Chain Database

The off-chain database is utilized to increase the scalability of the blockchain-based platform. An open source NoSQL database, MongoDB [26], is utilized for the off-chain database

implementation. MongoDB is widely used for its ability to handle large volumes of flexible and dynamic documents, such as JSON-like documents. The document-oriented key feature is able to support the standardization of the medical record data types. It also provides high performance allowing for efficient queries and complex data transformations.

Currently, there are 2 collections maintained under the MongoDB schema that is used in this project, namely the users and observations data. In the further implementation, more collections based on different resource types are going to be used. Figure 4.51 shows the MongoDB schema, collections, and their entries.

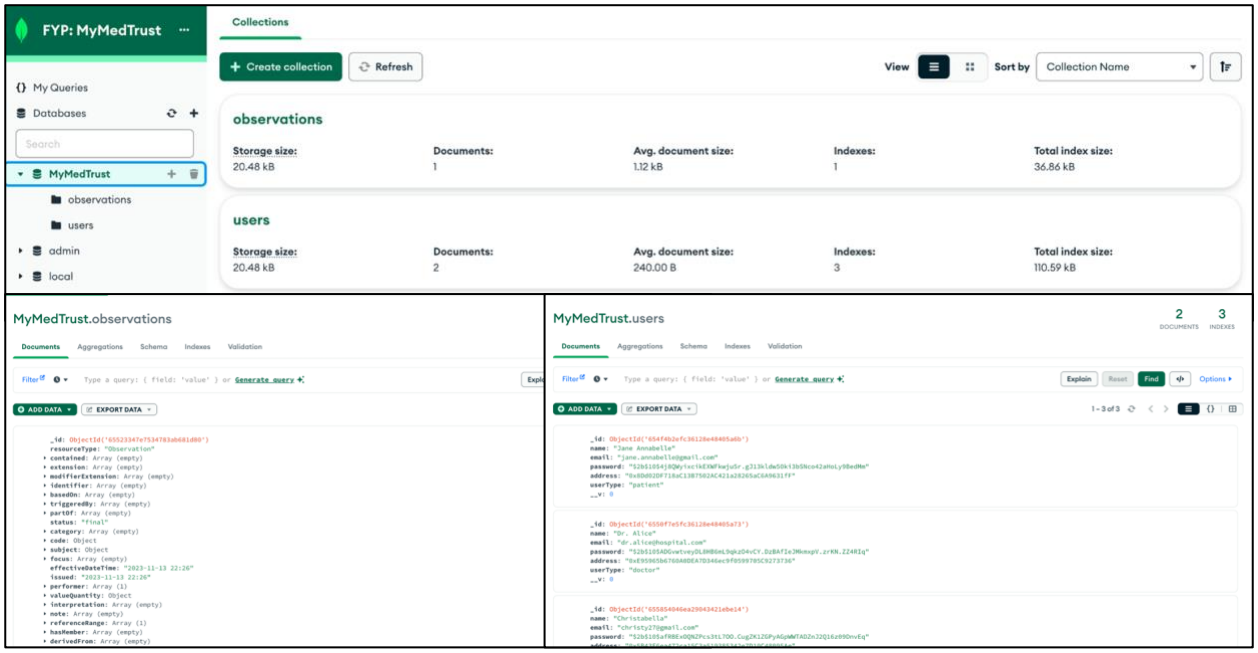

Figure 0.51 Get Observation By ID API Response

4.6. Artificial Intelligence

In the initial use case of adding new record, the doctor user needed to key in the entry of each field manually. To enhance this, an Artificial Intelligence (AI) is integrated by using the Natural Language Processing (NLP) model. One of the usages of NLP in the biomedical areas is the Named Entity Recognition (NER) [19]. In the biomedical field, NER involves in detecting and classifying important information in free texts such as medical records. This project utilized NER

model named Med7 [20] that focuses on extracting 7 medication information such as dosage, drug names, duration, form, frequency, route of administration and strength. Med7 is trained using the MIMIC-III, one of the largest datasets that includes more than 60,000 health records, and is available on the *spaCy* library in Python. Due to the limited availability of the appropriate model and entity mapping, the Med7 model is only utilized in the Add Medication module for extracting the FHIR Medication data type.

4.6.1. Backend

To use the Med7 model, a backend service is created using Flask, a Python-based framework. The API allows the client to parse free text medical notes in the request body and send the response of array of object consisting the key value pairs of the entity type and classified text. Figure 4.52 shows the parse medication text request and response example using the POST request.

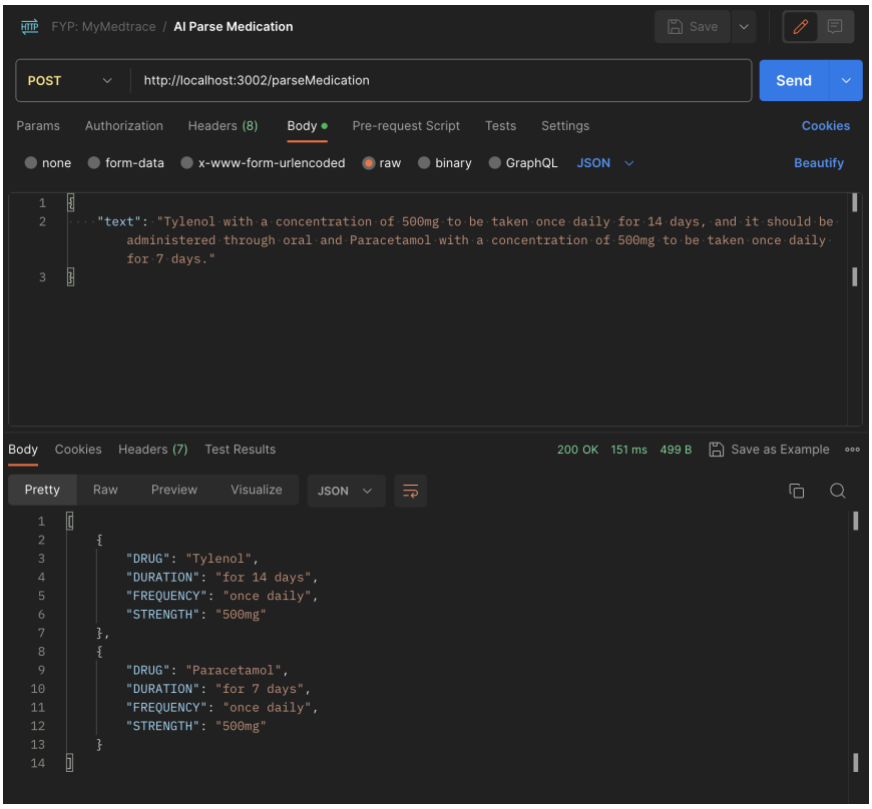

Figure 0.52 Parse Medication using NLP Request and Response

Chapter 5 Results and Discussions

In this chapter, the outcome of the MyMedtrace application is discussed in terms of the important key implementations and testing methodologies. The main goal is to ensure that the final application satisfies the main application requirements and to evaluate the overall quality and performance.

5.1. Key Implementations

The primary goal of the MyMedtrace application is to provide a user-friendly and patient-centric blockchain-based Electronic Health Record (EHR) management platform that store patient's data in scalable, interoperable, and secure manner. In order to achieve this goal, several key features are implemented. Firstly, utilizing blockchain smart contracts to store the patient's medical record provides security, immutability, transparency, and decentralization behaviour, that enhances the patient's trust and privacy. Additionally, to improve the scalability of the medical record storage, an additional MongoDB database is utilized as the off-chain storage to store the full record data alongside the blockchain as the on-chain storage that stores the record metadata. Finally, a user-friendly MyMedtrace web application is developed to allow patients, doctors and admins to perform various medical record queries in an efficient and intuitive manner.

5.2. Testing Methodologies

The testing methodologies should ensure all the goal aspects are tested and meet the expectations. The testing can be segmented into different approaches, which are Whitebox and Blackbox testing.

5.1.1. Whitebox Testing

Whitebox testing focuses on validating that the internal logic, code paths, and data structures used are working as expected. Figure 5.1 to 5.3 below shows the activity diagram of different scenarios tested.

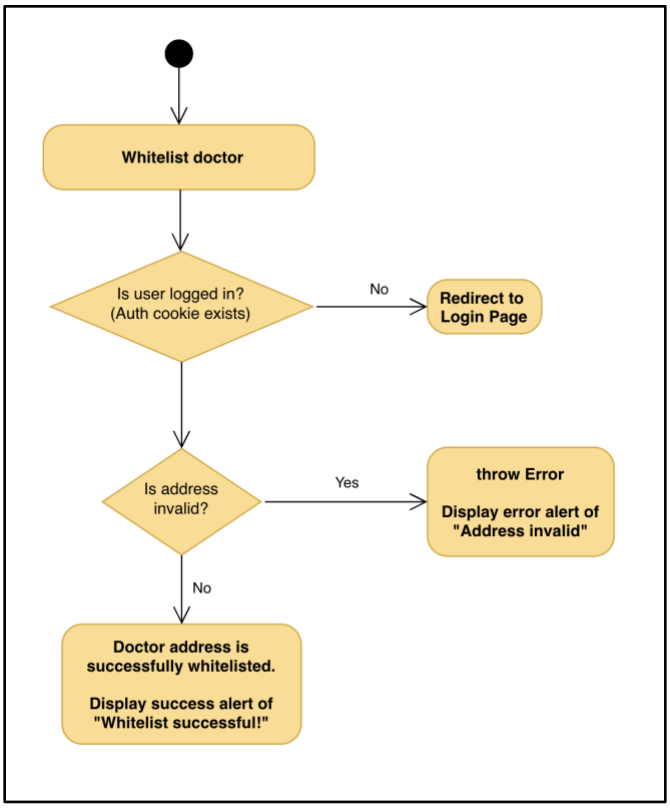

Figure 0.1 Whitelist Doctor Activity Diagram

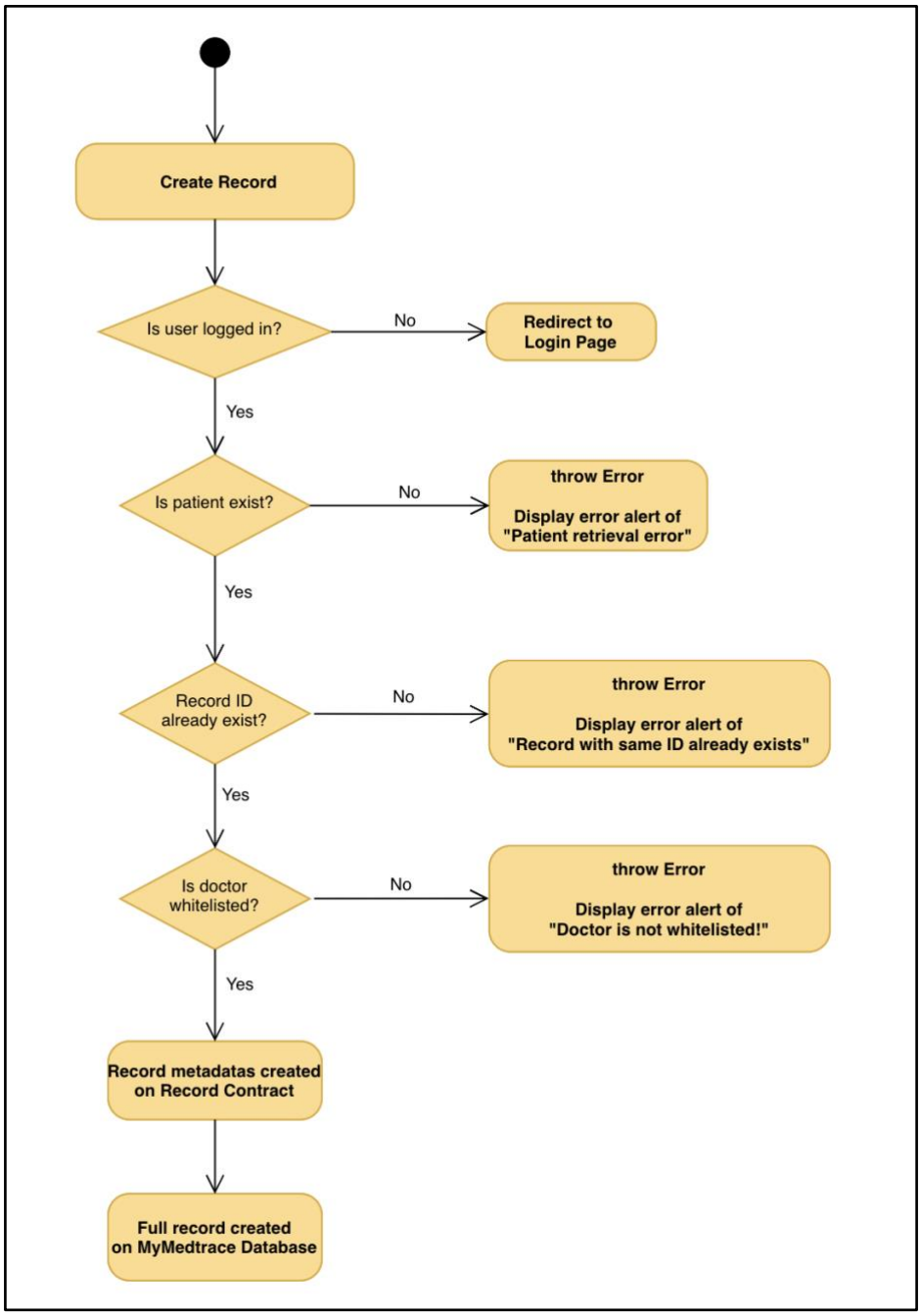

Figure 0.2 Create Record Activity Diagram

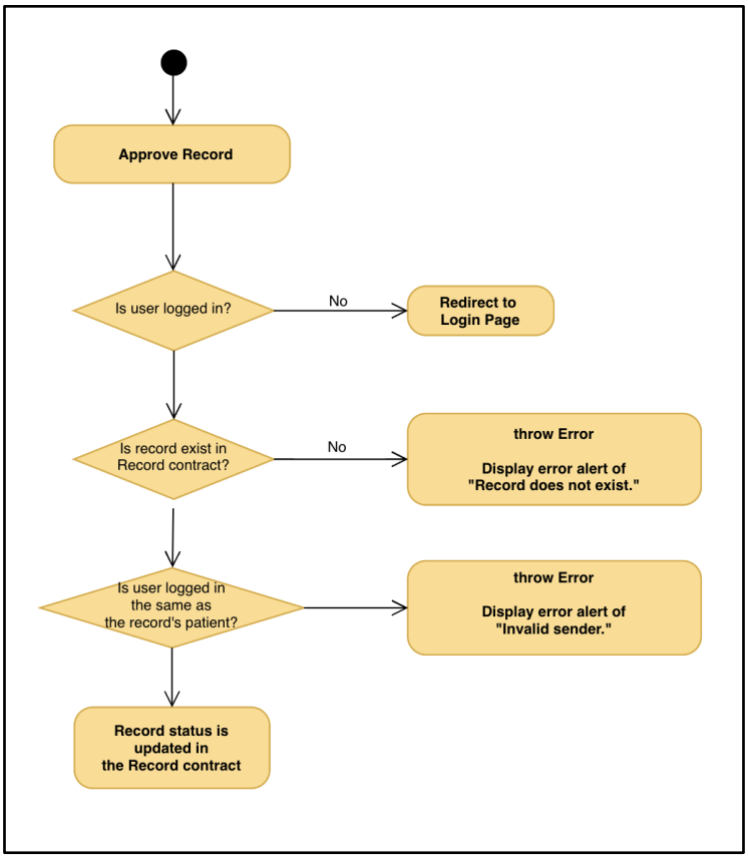

Figure 0.3 Approve Record Activity Diagram

In conclusion, the Whitebox testing enhances the development process by uncovering potential points of issues within the internal code. By comprehensively analysing the code's critical paths and code structure, the application's performance and reliability have been improved resulting into proper error handling and making the applications more reliable for the users.

5.1.2. Blackbox Testing

Blackbox testing is a testing method without prior knowledge of the internal structure. It focuses on external behaviour such as verifying the functional requirements and user flows to meet the expected criteria. These include user interface testing, performance testing, and compatibility testing.

5.1.2.1. Login

Test Case ID / Name	1 / Login		
Test Date	4 March 2024		
Purpose	To verify the login functionality		
Pre-Conditions	There is registered username ‘jane.annabelle@gmail.com’ and password ‘Password123!’		
Conditions	Input	Expected Result	Actual Result
1. Valid username & password	Username: jane.anabelle@gmail.com Password: Password123!	Successful login. The user is redirected to the dashboard page.	Successful login. The user is redirected to the dashboard page.
2. Valid username & invalid password	Username: jane.anabelle@gmail.com Password: Halo123!	Unsuccessful login. There is an error toast message of “Wrong password!”	Unsuccessful login. There is an error toast message of “Wrong password!” under the login fields.
3. Invalid username	Username: patient123!@gmail.com Password: Halo123!	Unsuccessful login. There is an error toast message of “User not found!”	Unsuccessful login. There is an error toast message of “User not found!”
4. Empty	Username: “”	A warning tooltip	A warning

username/password	Password: “”	under the input box appears with the message “Please fill in this field.”	tooltip under the box appears with the message “Please fill in this field.”

Table 5.1 Login Test Case

5.1.2.2. Sign Up

Test Case ID / Name	2 / Sign Up		
Test Date	4 March 2024		
Purpose	To verify the new user can successfully sign up for new account		
Pre-Conditions	1. New user is allocated with address 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Another account created previously with email "melissa@gmail.com" and address 0xE95965b6760A0DEA7D346ec9f0599705C9273736 3. The admin key is abc123		
Conditions	Input	Expected Result	Actual Result
1. All informations are filled in correctly	Name: Jane Annabelle Email: jane.anabelle@gmail.com Password: Password123! Confirm Password: Password123! Contact Address: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF IC: A1234567B Gender: Female Phone: 12345678	Successful sign up. The user is redirected to the dashboard page.	Successful sign up. The user is redirected to the dashboard page.

	Date of Birth: 27/02/2002 Home Address: 20 Nanyang, Singapore, 639809 User Type: Patient Emergency Contact Name: Felicia Emergency Contact Number: 67891234 Blood Type: O Height: 170 Weight: 69		
2. Email is already used	Username: melissa@gmail.com and other credentials	Unsuccessful sign up. There is an error toast message of "Email already used!"	Unsuccessful sign up. There is an error toast message of "Email already used!"
3. Address already used	Contact Address: 0xE95965b6760A0DEA7D346ec9f0599705C9273736 and other credentials	Unsuccessful sign up. There is an error toast message of "Address already used!"	Unsuccessful sign up. There is an error toast message of "Address already used!"
4. Empty credentials	Any empty field	A warning tooltip under the input box appears with	A warning tooltip under the box appears

		the message “Please fill in this field.”	with the message “Please fill in this field.”
5. Wrong admin key	Admin key: abc234	A warning tooltip under the input box appears with the message “Invalid adminKey: abc234”	A warning tooltip under the input box appears with the message “Invalid adminKey: abc234”

Table 5.2 Sign Up Test Case

5.1.2.3. Whitelist Doctor

Test Case ID / Name	3 / Whitelist doctor		
Test Date	4 March 2024		
Purpose	To verify the admin can do a whitelist for a doctor to a patient		
Pre-Conditions	1. Valid registered Patient A with an address of 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Valid registered Doctor A with an address of 0xE95965b6760A0DEA7D346ec9f0599705C9273736 that is not 3. Valid registered Doctor B with an address of 0x00F474F809999A015C327033F26D4EB51766A045 and already whitelisted to Patient A previously		
Conditions	Input	Expected Result	Actual Result
1. All information are filled in correctly	Patient Address: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF Doctor Address: 0xE95965b6760A0DEA7D346ec9f0599705C9273736	Successful whitelist. There is a success toast message of "Whitelist successful" under the field. The input fields are cleared.	Successful whitelist. There is a success toast message of "Whitelist successful" under the field. The input fields are cleared.
2. Doctor is already whitelisted	Patient Address: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF	Unsuccessful whitelist. There is an error toast message of	Unsuccessful whitelist. There is an error toast message of

	Doctor Address: 0x00F474F809999A015C327033F26D4EB51766A045	“Doctor has already been whitelisted!”	“Doctor has already been whitelisted!”
3. Address invalid	Patient Address: 0x8Dd02DF Doctor Address: 0xE95965b	Unsuccessful sign up. There is an error toast message of “Address invalid!”	Unsuccessful sign up. There is an error toast message of “Address invalid!”

Table 5.3 Whitelist Doctor Test Case

5.1.2.4. Dashboard and Navigation Bar Display for Different User Type

Test Case ID / Name	4 / Dashboard and Navigation Bar for Different User Type		
Test Date	4 March 2024		
Purpose	To verify the display of dashboard and navigation side bar are rendered correctly based on the user type		
Pre-Conditions	There are registered users types of patient, doctor, and admin.		
Conditions	Input	Expected Result	Actual Result
1. Loading state	N.A.	Dashboard and navigation bar display loading skeletons.	Dashboard and navigation bar display loading skeletons.
2. Admin user is logged in	N.A.	1. Navigation side bar is displayed correctly showing home, patients, doctors, records, and whitelist buttons. 2. Dashboard is displayed correctly showing the name of admin, total patients, doctors, and records, statistics of patients and records numbers, and shortcut to patients list.	1. Navigation side bar is displayed correctly showing home, patients, doctors, records, and whitelist buttons. 2. Dashboard is displayed correctly showing the name of admin, total patients, doctors, and records, statistics of patients and records numbers, and shortcut to patients list.

3. Patient user is logged in	N.A.	1. Navigation side bar is displayed correctly showing home, all records, observations, conditions, allergies, and medications buttons. 2. Dashboard is displayed correctly showing the name of patient, shortcuts to the pending of approval records, observations, conditions, allergies, and medications.	1. Navigation side bar is displayed correctly showing home, all records, observations, conditions, allergies, and medications. 2. Dashboard is displayed correctly showing the name of patient, shortcuts to the pending of approval records, observations, conditions, allergies, and medications.
4. Doctor user is logged in	N.A.	1. Navigation side bar is displayed correctly showing home, add observation, add condition, add allergy, add medication, and patients buttons. 2. Dashboard is displayed correctly showing the name of doctor, patients and records created	1. Navigation side bar is displayed correctly showing home, add observation, add condition, add allergy, add medication, and patients buttons. 2. Dashboard is displayed correctly showing the name of doctor, patients and records

		statistics, and shortcuts to add observation, add condition, add allergy, and add medication.	created statistics, and shortcuts to add observation, add condition, add allergy, and add medication.

Table 5.4 Dashboard and Navigation Bar for Differetn User Type Test Case

5.1.2.5. Add Observation

Test Case ID / Name	5 / Add Observations		
Test Date	4 March 2024		
Purpose	To verify the add new observation record functionality		
Pre-Conditions	1. There is a registered doctor user logged in. 2. There is a registered patient named Jane Annabelle with address 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF		
Conditions	Input	Expected Result	Actual Result
1. Initial state	N.A.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Observation' and the doctor's email and blockchain address respectively.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Observation' and the doctor's email and blockchain address respectively.
2. Patient address is filled in correctly	Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF	1. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input	1. There is a spinner box indicating 'Fetching user details…' and a spinning

		field. 2. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.	box inside the Subject input field. 2. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.
3. Patient address does not exist	Patient: 0x00F474F809999A015C327033F26D4EB51766A045	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the

		indicating retrieval error. 2. Subject field is still empty.	Patient input field indicating retrieval error. 2. Subject field is still empty.
3. Doctor is not whitelisted	N.A.	There is an error alert displayed indicating "Doctor is not whitelisted!" after submission.	There is an error alert displayed indicating "Doctor is not whitelisted!" after submission.
4. Units in Value Quantity is filled	Value Quantity: - Units: 'mmHg'	The units in Reference Range Low and High should be automatically filled following the value in units in Value Quantity field	The units in Reference Range Low and High should be automatically filled following the value in units in Value Quantity field
5. All required information are filled in correctly	1. Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Status: any options from dropdown	Successful add observation request. There is a modal indicating 'Observation is successfully added!'	Successful add observation request. There is a modal indicating 'Observation is successfully

	3. Code: a. System http://loinc.or b. Code 85354-9 c. Blood pressure: Blood pressure panel with all children optional 4. Subject (retrieved from the address in Patient field): a. Reference: Patient/0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF b. Display: Jane Annabelle 5. Value Quantity a. Value: 120 b. Unit: mmHg	with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home page.	added!' with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home page.

	c. Code: mmHg 6. Reference Range Low: a. Value: 60 b. Unit: mmHg c. Code: mmHg 7. Reference Range High: a. Value: 150 b. Unit: mmHg c. Code: mmHg		

Table 5.5 Add Observation Test Case

5.1.2.6. Add Condition

Test Case ID / Name	6 / Add Condition		
Test Date	4 March 2024		
Purpose	To verify the add new condition record functionality		
Pre-Conditions	1. There is a registered doctor user logged in. 2. There is a registered patient named Jane Annabelle with address 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF		
Conditions	Input	Expected Result	Actual Result
1. Initial state	N.A.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Condition' and the doctor's email and blockchain address respectively.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Condition' and the doctor's email and blockchain address respectively.
2. Patient address is filled in correctly	Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF	1. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input	3. There is a spinner box indicating 'Fetching user details…' and a spinning

		field. 2. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.	box inside the Subject input field. 4. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.
5. Patient address does not exist	Patient: 0x00F474F809999A015C327033F26D4EB51766A045	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the

		indicating retrieval error. 2. Subject field is still empty.	Patient input field indicating retrieval error. 2. Subject field is still empty.
3. Doctor is not whitelisted	N.A.	There is an error alert displayed indicating “Doctor is not whitelisted!” after submission.	There is an error alert displayed indicating “Doctor is not whitelisted!” after submission.
4. All required informations are filled in correctly	1. Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Clinical Status: a. System: http://terminology.hl7.org/CodeSystem/condition-clinical b. Code: resolved 3. Verification Status: a. System:	Successful add condition request. There is a modal indicating 'Condition’ is successfully added!’ with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and	Successful add condition request. There is a modal indicating 'Condition’ is successfully added!’ with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close

	http://terminology.hl7.org/CodeSystem/condition-ver-status b. Code: confirmed 4. Category a. System: http://snomed.info/sct b. Code: 123456 c. Display: Problem 5. Body Site: a. System: http://snomed.info/sct b. Code: 38266002 c. Display: The entire body as a whole 6. Onset Date & Time: 2023-12-01 11:12 7. Abatement Date & Time: 2023-12-06 22:08 8. Recorder and Asserter are correctly retrieved from doctor that is logged in	all input fields are emptied. When a home button is clicked, user is redirected to the home page.	modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home page.

Table 5.2.6 Add Condition Test Case

5.1.2.7. Add Allergy

Test Case ID / Name	7 / Add Allergy
Test Date	4 March 2024
Purpose	To verify the add new allergy record functionality
Pre-Conditions	3. There is a registered doctor user logged in. 4. There is a registered patient named Jane Annabelle with address 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF

Conditions	Input	Expected Result	Actual Result
3. Initial state	N.A.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Allergy' and the doctor's email and blockchain address respectively.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Allergy' and the doctor's email and blockchain address respectively.
4. Patient address is filled in correctly	Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF	5. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input	7. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input field.

		field. 6. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.	8. Patient address is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.
9. Patient address does not exist	Patient: 0x00F474F809999A015C327033F26D4EB51766A045	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field indicating	1. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field indicating retrieval error. 2. Subject field is

		retrieval error. 2. Subject field is still empty.	still empty.
3. Doctor is not whitelisted	N.A.	There is an error alert displayed indicating "Doctor is not whitelisted!" after submission.	There is an error alert displayed indicating "Doctor is not whitelisted!" after submission.
4. All required information are filled in correctly	1. Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Category: food 3. Criticality: high 4. Clinical Status: a. System: http://terminology.hl7.org/CodeSystem/allergyintolerance-clinical b. Code: active c. Display: active 5. Verification Status: a. System:	Successful add condition request. There is a modal indicating 'Allergy' is successfully added!' with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is	Successful add condition request. There is a modal indicating 'Allergy' is successfully added!' with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home

	http://terminology.hl7.org/CodeSystem/allergyintolerance-verification b. Code: unconfirmed c. Display: unconfirmed 6. Code a. System: http://snomed.info/sct b. Code: 227493005 c. Display: Cashew 7. Manifestation: a. System: http://snomed.info/sct b. Code: 39579001 c. Display: Anaphylactic reaction 8. Substance: a. System: http://www.nlm.nih.gov/res	redirected to the home page.	page.

	earch/umls/rxnorm b. Code: 1160593 c. Display: cashew nut allergenic extract Injectable Product 9. Exposure Route: a. System: http://snomed.info/sct b. Code: 34206005 c. Display: Subcutaneous route 10. Description: a. Severe reaction to subcutaneous cashew extract. 11. Severity: severe		

Table 5.7 Add Allergy Test Case

5.1.2.8. Add Medication

Test Case ID / Name	8 / Add Medication		
Test Date	4 March 2024		
Purpose	To verify the add new medication record functionality		
Pre-Conditions	1. There is a registered doctor user logged in. 2. There is a registered patient named Jane Annabelle with address 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF		
Conditions	Input	Expected Result	Actual Result
1. Initial state	N.A.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Medication' and the doctor's email and blockchain address respectively.	Input fields are all empty with some static fields pre-filled in such as resource type and performer pre-filled with 'Medication' and the doctor's email and blockchain address respectively.
2. Patient address is filled in correctly	Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF	1. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input field. 2. Patient address	1. There is a spinner box indicating 'Fetching user details…' and a spinning box inside the Subject input field. 2. Patient address

		is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.	is successfully retrieved to fill in the field Subject. There is a green box under the Patient field indicating the successfully parsed patient's name.
3. Patient address does not exist	Patient: 0x00F474F809999A015C327033F26D4EB51766A045	3. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field indicating retrieval error. 4. Subject field is still empty.	4. Unsuccessful patient's name retrieval because the address is not exist. There is a red box under the Patient input field indicating retrieval error. 5. Subject field is still empty.
4. Doctor is not whitelisted	N.A.	There is an error alert displayed indicating "Doctor is not	There is an error alert displayed indicating "Doctor is not

		whitelisted!" after submission.	whitelisted!" after submission.
5. Parsing medication notes using AI	Prescribed 500mg of amoxicillin to be taken orally twice daily for ten days to treat the bacterial infection.	1. There is a Parsed Medication result box of a mapped key values pairs such as: DRUG amoxicillin, DURATION for ten days, FREQUENCY twice daily, ROUTE orally, STRENGTH 500mg. 2. Users are able to choose the result they want to put into the input fields based on the parsed results. 3. Upon choosing, the medication code display,	1. There is a Parsed Medication result box of a mapped key values pairs such as: DRUG amoxicillin, DURATION for ten days, FREQUENCY twice daily, ROUTE orally, STRENGTH 500mg. 2. Users are able to choose the result they want to put into the input fields based on the parsed results. 3. Upon choosing, the medication code display,

		and ingredients are automatically filled based on the result. Duration and frequency are inserted into the additional notes fields.	and ingredients are automatically filled based on the result. Duration and frequency are inserted into the additional notes fields.
6. All required information are filled in correctly	1. Patient: 0x8Dd02DF718aC13B7502AC421a28265aC6A9631fF 2. Status: active 3. Manufacturer: a. Reference: Org/12812 b. Display: HeloMed Inc. 4. Code: a. System: http://snomed.info/sct b. Code:	Successful add condition request. There is a modal indicating 'Allergy' is successfully added!' with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home page.	Successful add condition request. There is a modal indicating 'Allergy' is successfully added!' with the details of the record ID and the timestamp. There is also a button to close the modal or go to home page. When a close modal button is clicked, modal is closed and all input fields are emptied. When a home button is clicked, user is redirected to the home page.

	38690600 1 c. Display: Capecitabine 500mg oral tablet (Xeloda) 5. Form: a. System: http://snomed.info/sct b. Code: 385055001 c. Display: Tablet dose form (qualifier value) 6. Item Reference: Med012131 7. Numerator a. Value: 340 b. System: http://unitsofmeasure.org		

	c. Code: mg 8. Denominator a. Value: 3 b. System: http://terminology.hl7.org/CodeSystem/v3-orderableDrugForm c. Code: TAB 9. Batch a. Lot Number: 9494788 b. Expiration Date: 2025-06-17		

Table 5.8 Add Medication Test Case

5.1.2.9. List Table

Test Case ID / Name	9 / List Table		
Test Date	4 March 2024		
Purpose	To verify the list table of patients, doctors, and records are displayed properly		
Pre-Conditions	N.A.		
Conditions	Input	Expected Result	Actual Result
1. Loading state	N.A.	Table displays loading skeleton	Table displays loading skeleton
2. Patients table	N.A.	1. Patients table correctly displays the list of patients with and their information such as name, blockchain address, identity card number, gender, email and registered date. 2. Search bar functions correctly to query for patient's name, blockchain address, email, and identity card number.	1. Patients table correctly displays the list of patients with and their information such as name, blockchain address, identity card number, gender, email and registered date. 2. Search bar functions correctly to query for patient's name, blockchain address, email, and identity card number.
3. Doctors table	N.A.	1. Doctors table correctly displays the list of doctors with and their information such as	1. Doctors table correctly displays the list of doctors with and their information such as

		name, blockchain address, medical field, identity card number, gender, email and registered date. 2. Search bar functions correctly to query for doctor's name, blockchain address, medical field, email, and identity card number.	name, blockchain address, medical field, identity card number, gender, email and registered date. 2. Search bar functions correctly to query for doctor's name, blockchain address, medical field, email, and identity card number.
4. Records table	N.A.	1. Records table correctly displays the list of medical records with the key information such as recorded date, patient name, patient address, issuer doctor name, category and record display name. 2. Search bar functions correctly to query for doctor's name, patient's address, patient's name, and record category. 3. Upon clicking each	1. Records table correctly displays the list of medical records with the key information such as recorded date, patient name, patient address, issuer doctor name, category and record display name. 2. Search bar functions correctly to query for doctor's name, patient's address, patient's name, and record category.

		entry, a modal is displayed to show the record details.	3. Upon clicking each entry, a modal is displayed to show the record details.

Table 5.9 List Table Test Case

5.1.2.10. Medical Records List

Test Case ID / Name	10 / List Table		
Test Date	4 March 2024		
Purpose	To verify the list of medical records in the patient's module are properly displayed		
Pre-Conditions	N.A.		
Conditions	Input	Expected Result	Actual Result
1. Loading state	N.A.	Page displays loading card skeleton	Page displays loading card skeleton
2. Correct medical record card	N.A.	For every medical record type (i.e. observation, condition, allergy, and medication), correct medical record card design and content are displayed.	For every medical record type (i.e. observation, condition, allergy, and medication), correct medical record card design and content are displayed.
3. View medical record details	N.A.	Each card contains the 'View details' button. Upon clicking the button, a modal is displayed to show the medical record details divided into 2 tabs, Details and Raw. The Details tab shows the details in structured version, and the Raw tab shows the details in JSON format.	Each card contains the 'View details' button. Upon clicking the button, a modal is displayed to show the medical record details divided into 2 tabs, Details and Raw. The Details tab shows the details in structured version, and the Raw tab shows the details in JSON format.

Table 5.10 Medications Record List

5.1.2.11. Approval of Medical Record Request

Test Case ID / Name	11 / Approval of Medical Record		
Test Date	4 March 2024		
Purpose	To verify the approval of medical record functionality		
Pre-Conditions	N.A.		
Conditions	Input	Expected Result	Actual Result
1. Correct display of records	N.A.	Page displays the correct pending, approved, and declined records. There are approve and decline buttons in the pending records.	Page displays the correct pending, approved, and declined records. There are approve and decline buttons in the pending records.
2. Correct approval or decline button behaviour	N.A.	If the approval or decline button is clicked, a modal is displayed showing the record details including the doctor issuer and issued date. There are confirm approval or decline and close buttons in the modal.	If the approval or decline button is clicked, a modal is displayed showing the record details including the doctor issuer and issued date. There are confirm approval or decline and close buttons in the modal.
3. Successful approval of request	N.A.	If the confirm approval button is clicked and approval is successful, a success alert is displayed indicating “Thank you! Approval success” and the page will reload.	If the confirm approval button is clicked and approval is successful, a success alert is displayed indicating “Thank you! Approval success” and the page will reload.
4. Successful decline of	N.A.	If the confirm decline button is clicked and decline is	If the confirm decline button is clicked and decline is

request		successful, a success alert is displayed indicating “Thank you! Decline success” and the page will reload.	successful, a success alert is displayed indicating “Thank you! Decline success” and the page will reload.

Table 5.11 Approval of Medical Record Test Case

In conclusion, the Blackbox testing provides valuable insights in terms of the features functionality and behaviour of the MyMedtrace web application from the external perspective. Through executions of various test cases across different modules, the quality of the application is assessed in terms of usability, user-friendliness, ability to meet all the requirements, and proper unexpected situations handling. The Blackbox testing plays a significant role in improving the application in terms of the overall quality and satisfaction in the user’s perspective.

Chapter 6 Conclusions and Future Work

6.1. Conclusions

In this project, a comprehensive analysis of challenges in the EHR management system are identified and various studies related to the possibilities of blockchain utilization in this field are explored in the effort to deliver an effective solutions. Furthermore, through iterative process of developments and evaluations, a blockchain-based EHR management web application, MyMedtrace, is built as a final product to tackle the challenges present in the realm of EHR management system.

With the use of blockchain technology, a secure, immutable and decentralized EHR storage system is achieved. Further, through smart contract that encapsulates secure automated transactions and tamper-proof record keeping, both service performance and development efficiency are improved. Additionally, due to the blockchain limitation in storage capacity and scalability, a combination of blockchain and off-chain database are able to increase the scalability of the applications resulting into more effective queries and performance.

Furthermore, the utilization of FHIR (Fast Healthcare Interoperability Resources) data types to store all the records in standardized manner is able to provide interoperability. Through its flexible and adaptable formats, the patients are able to efficiently present their records to wide range health care systems.

6.2. Recommendations in Future Work

Although usages of blockchain with an additional off-chain storage and FHIR data types in the MyMedtrace platform has tackled the existing challenges of EHR storage system in terms of the security, scalability, and interoperability, there remains few areas for future enhancement due to time constraint and resources. These areas are worth exploring to significantly enhance various aspects of the overall application.

1. Data Transfer Security

The flow of the overall application involves in transferring data from client to server and to blockchain and MongoDB database side. The data transferred includes important user credentials including the blockchain address and password, and also the full record to be stored. These information are possibly tampered if there is an Man-in-the-Middle (MitM) attack that intercepts the communication and perform data modification before forwarding it to the intended recipient. This action is able to compromise the confidentiality and integrity of the data in transit, hence can result into credential misusage or inconsistent data. Therefore to prevent this data security issue, it will be a secure practice to encrypt every data transferred from the client to the server side. Various cryptographic techniques are worth exploring to find the best and optimal solutions to enhance the data transfer security.

2. Smart Contract Optimization

There is a limitation in the bytecode size of the smart contract in Solidity, which is up to 24kb. Due to the size limitation, some functions are not be able to implemented or need to be simplified, hence resulting into significant impact on the query performance. Therefore, in this case, the contracts are broken down into two contracts, which are the *MainContract* that stores the user credentials and the *RecordContract* that mainly stores the hash values of full record and some metadata such as the owner and issuer address. In *RecordContract*, there is a record list mapping that use the key of record ID for easier constant-time access of individual record details. However, for querying the whole records available, as exist in the All Records page in admin module, the performance is slower, as we are not able to directly get the mapping size. One way to improve this is to maintain another array in the record contract that stores all the record IDs and iterate the record IDs array to access the record list mapping. Furthermore, more exploration on the smart contract data types and good practices are able to optimize the current smart contract behaviour.

3. Features Enhancement

Currently there are limitations in the application feature especially in terms of the adding record flow that requires the doctor to insert the data field one by one and record approval mechanism.

The closest approach to improve the manual record creation flow is to implement NLP model to parse the doctor free-text clinical notes to respective field which is currently only applicable in the Medication data type. Furthermore, in the current version, if the patients want to approve the doctor's request of adding new record, they need to do it through the platform. One possible way to optimize this is to implement approval through email or implement push notification mechanism on the mobile app version in the future.

Reflection on Learning Outcome Attainment

- Engineering knowledge

  Through this final year project, I'm able to exercise my engineering knowledge, especially in the field of Software Engineering by implementing the full-stack development in building this blockchain-based web application. During the process, several software engineering lifecycle stages are implemented such as planning, problem analysis, establishing the design, implementation, testing and integration, and as well as iteratively gather feedbacks from third party such as supervisors and friends to continuously improve the application. In addition, another software engineering knowledges such as use case diagram, user activity diagram, and sequence diagrams are implemented in order to visualize the overall design of the application better.

- Design/development of Solutions

  Through this final year project, I'm able to exercise the design and development stages of providing solutions to a real world problem. In this case is to implement a blockchain-based web application as a solution to an existing Electronic Health Record management system problem in security, interoperability, and scalability. Since this is an individual project, I'm able to get involved in the end-to-end project development starting from the requirements gathering and design of the system, the actual development using several technology stacks, bug fixing, and until the testing and finalize the product.

- Modern Tool Usage

  In this final year project, some cutting edge technologies are utilized. This includes, React Typescript in the frontend module, as well as the open source UI libraries such as ChakraUI and AntDesign, Express.JS and Python Flask in the backend modules, including MongoDB Atlas as the database server. In addition, since this is a blockchain-based application, the Web3 smart contract is also built using the Solidity and Ethereum. Hence, the whole FYP journey definitely develop my technical skills in gaining a hands-on experience with these tools.

Project No: A3053-231

Acknowledging/Declaring the Use of GAI

Please refer to NTU's Current Policy & Guidelines on the Use of Generative AI available in NTUlearn home page and the link:
https://entuedu.sharepoint.com/sites/Student/dept/ctlp/SitePages/Exploring-the-Impact-of-Generative-Artificial-Intelligence-(GAI)-Tools-on-Education.aspx

1. Complete the following declaration if applicable.
2. Create a Paper Trail to document the input prompt, output obtained, and how you have used it.

I Vanessa Christy Chandra, vanessac001@e.ntu.edu.sg honestly and sincerely make the following declaration in relation to the following course submission.

1. Name of course: Final Year Project
2. Course Code: IM4080
3. Instructor: Assoc Prof Chua Hock Chuan
4. Title of Assignment/Project Submission: *MyMedtrace:* A Scalable Patient-Centric Blockchain-based Electronic Health Records Management Platform

In relation to the foregoing I hereby declare that, fully and properly in accordance with the Assignment/Project Instructions I have (check where appropriate):

i.	Used GAI as permitted to assist in generating key ideas only	☐
ii.	Used GAI as permitted to assist in generating a first text only.	☐
	And/or	
iii.	Used GAI to refine syntax and grammar for correct language submission only.	☑
	Or	
iv.	As it is not permitted: Not used GAI assistance in any way in the development or generation of this assignment or project.	☐

I also declare that I have:

a. Fully and honestly submitted the digital paper trail required under the assignment/project instructions; and that
b. Wherever GAI assistance has been employed in the submission in word or paraphrase or inclusion of a significant idea or fact suggested by the GAI assistant, I have acknowledged this by a footnote; and that,
c. Apart from the foregoing notices, the submission is wholly my own work.

Vanessa Christy Chandra — 11 April 2024

Student Name & Signature — Date

References

[1] M.Y. Ali, S. Ahmed, M.I. Hossain, A.B.M. Alim al Islam, and J. Noor, "Electronic Health Record's Security and Access Control Using Blockchain and IPFS," in *Proceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, Lecture Notes in Networks and Systems*, vol. 447, pp. 493-505, 2023.

[2] S. Garg, R. K. Kaushal, and N. Kumar, "Blockchain-based Electronic Health Record and Open Research Challenges," *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2022, pp. 1-5.

[3] R. Pilla, T. Oseni, and A. Stranieri, "A Study Into the Impact of Data Breaches of Electronic Health Records," in *Proceedings of the 2023 Australasian Computer Science Week (ACSW '23)*, January 2023, pp. 252-254.

[4] I. Tham, "Personal info of 1.5m SingHealth patients, including PM Lee, stolen in Singapore's worst cyber attack," The Straits Times, 20 July 2018. [Online] Available: https://www.straitstimes.com/singapore/personal-info-of-15m-singhealth-patients-including-pm-lee-stolen-in-singapores-most. (Accessed: 21 September 2023)

[5] Y. Zhuang, L. R. Sheets, Y. -W. Chen, Z. -Y. Shae, J. J. P. Tsai and C. -R. Shyu, "A Patient-Centric Health Information Exchange Framework Using Blockchain Technology," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 8, pp. 2169-2176, Aug. 2020.

[6] A. A. Mamun, S. Azam and C. Gritti, "Blockchain-Based Electronic Health Records Management: A Comprehensive Review and Future Research Direction," in *IEEE Access*, vol. 10, pp. 5768-5789, 2022.

[7] R. G. Sonkamble, S. P. Phansalkar, V. M. Potdar, and A. M. Bongale, "Survey of Interoperability in Electronic Health Records Management and Proposed Blockchain Based Framework: MyBlockEHR," in *IEEE Access*, vol. 9, pp. 158367-158401, 2021.

[8] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen and E. Dutkiewicz, "Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities," in *IEEE Access*, vol. 7, pp. 85727-85745, 2019.

[9] W. Gu, J. Li and Z. Tang, "A Survey on Consensus Mechanisms for Blockchain Technology," 2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA), Xi'an, China, 2021, pp. 46-49, doi: 10.1109/CAIBDA53561.2021.00017.

[10] R. K. Kaushal, N. Kumar and S. N. Panda, "Blockchain Technology Its Applications and Open Research Challenges", *Journal of Physics: Conference Series*, vol. 1950, no. 1, pp. 12030, 2021.

[11] Ethereum. “Ethereum: What is Ethereum?” Ethereum. Available: https://ethereum.org. (Accessed: November 11, 2023).

[12] S. Thakur, B. Gupta, U. Mathur and D. Bansal, "Electronic Health Record Systems for Enhanced Medical Care: A Survey," 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Coimbatore, India, 2023, pp. 257-262.

[13] Y. Mohammad and L. Stergioulas, "Building an information security strategy for EHR: Guidelines for assessing the current situation," 2010 Annual International Conference of

the IEEE Engineering in Medicine and Biology, Buenos Aires, Argentina, 2010, pp. 3919-3922.

[14] M. T. Quasim, A. A. E. Radwan, G. M. M. Alshmrani and M. Meraj, "A Blockchain Framework for Secure Electronic Health Records in Healthcare Industry," 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 2020, pp. 605-609, doi: 10.1109/ICSTCEE49637.2020.9277193.

[15] D. C. Nguyen, P. N. Pathirana, M. Ding and A. Seneviratne, "Blockchain for Secure EHRs Sharing of Mobile Cloud Based E-Health Systems," in IEEE Access, vol. 7, pp. 66792-66806, 2019.

[16] P. A. Lobo and V. Sarasvathi, "Distributed File Storage Model using IPFS and Blockchain," 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2021, pp. 1-6.

[17] M. George and A. M. Chacko, "A Patient-Centric Interoperable, Quorum-based Healthcare System for Sharing Clinical Data," 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, 2022, pp. 1-6.

[18] W. J. Gordon, C. Catalini, "Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability," *Computational and Structural Biotechnology Journal*, vol. 16, pp. 224-230, 2018.

[19] P. Mishra and A. L. Anjana, "Named Entity Recognition on Biomedical Text," in Advancements in Communication and Systems, Ed. by A. K. Tripathi and V. Shrivastava, Computing and Intelligent Systems, SCRS, India, 2023, pp. 197-207.

[20] K. Ormilitzin, “Med7: Clinical Information Extraction System in Python and spaCy”. [Online] Available: https://kormilitzin.medium.com/med7-clinical-information-extraction-system-in-python-and-spacy-5e6f68ab1c68. (Accessed: March 4, 2024).

[21] “Chakra UI - a simple, modular and accessible component library that gives you the building blocks you need to build your react applications.,” Chakra UI: Simple, Modular and Accessible UI Components for your React Applications., https://chakra-ui.com/. [Online] (Accessed: Mar. 22, 2024).

[22] “Ant Design,” The world’s second most popular React UI framework. [Online] https://ant.design/ (Accessed: Mar. 22, 2024).

[23] “React-apexchart - a react chart wrapper for Apexcharts.js,” ApexCharts.js. [Online] https://apexcharts.com/docs/react-charts/ (Accessed: Mar. 22, 2024).

[24] Home - Truffle Suite. [Online] https://archive.trufflesuite.com/ (Accessed: Mar. 22, 2024).

[25] Ganache - Truffle Suite. [Online] https://archive.trufflesuite.com/ganache/ (Accessed: Mar. 22, 2024).

[26] “The developer Data Platform,” MongoDB. [Online] https://www.mongodb.com/ (Accessed: Mar. 22, 2024).