FINAL REPORT

Software Security - A Secure Hospital System

CSE 545: Spring, 2022

Submitted to:

Professor Stephen S. Yau

Ira A. Fulton School of Engineering Arizona State University

Submitted by (GROUP 11):

Mohammed Albasha (Leader)

Dhiraj Kumar (Deputy Leader)

Tazreen Khan

Smit Dharmeshbhai

Singh

Girija Rani

Eshwar Pallem

Sahith Doma

Pratyush Narayan

Mar 30 , 2022

Table of Contents

1. INTRODUCTION

The human body is an extremely perplexing and refined structure with many great capacities. As science and innovation advance, medicine turns into a fundamental piece of research. As of today, the health sector contains major medical establishments like medical clinics, hospitals, and so forth that have provided research and advancement foundations to the knowledge of our bodies. Hence, therefore, these medical establishments seek to provide the very best facilities they can, specifically their services.

The aim of the project is to provide an overview of requirements within this report. This document will also provide detailed information about different components within the function such as the design, functionality, and behavior.

1.1. Goals and Objectives

The goal of the project is to efficiently design a secure hospital system framework that will implement various required components, as well as security conditions for secure healthcare data, patient data, transactions, and user information. The hospital security system empowers better tolerant consideration, patient security, patient classification, proficiency, decreased expenses, and a better administration data framework. It gives simple admittance to basic data hence empowering the administration to make better choices on scheduling as well. By end of the project, we hope to:

- Provide more security
- Avoid both redundancy and inconsistency
- user-friendly
- Accessible
- Reliable

1.2. Statement of Scope

Many current medical facilities seem to facilitate using paper-based systems. This is inefficient as it is too slow and cannot constantly provide updated lists of patients, as well as general hospital records within a reasonable timeframe. The system intends to improve this issue by reducing overtime pay and causing an increase in the number of patients that can be treated accurately. Requirement statements in these documents are both functional and non-functional as many aspects are needed to be implemented.

The scope of this document is to provide a high level description of different elements and how they work within a secure and developed system given the subtleties who use it. These subtleties provide the hospital staff, doctors, and patients their subtleties given appointments, transactions, participation, etc.This document will be used by the developer as a reference to how this secure hospital system (SHS) will function, as well as its implementation during its'

development.In additions, the document would be updated depending on the further changes in project or feedback recieved.

1.3. About Service

The main problem with handling medical information is making sure that it securely stays confidential to the public (based on who you are, like general staff or doctor), in also the most efficient way possible in terms of factors like cost and time. In this section, the critical problems that are pertinent to the environment will be identified and discussed. The reader should be able to understand the overview of the framework as it should not be too difficult to navigate through.

1.4. Major Constraints

There are various factors that may prevent healthcare from cybersecurity. One of those being the deep specialization of security, so trying to minimize the number of vulnerabilities within a hospital, as well as providing implementation and maintenance require a lot of time and resources.Factors like these were taken into consideration when discussing attacks that might occur within a system.This section will describe how product/ service constraints affect the design, functionality, behavior, implementation, and testing of software.

2. OVERVIEW OF THE PROJECT

The following are the contributions given by every single person on the team:

2.1. Mohammed Albasha's contributions to the project:

- Had initialized the firebase, flask app, and GitHub.
- Had designed a firestore database layout.
- Had worked on setting up a development plan.
- Had worked on setting up an app for routing and navigation.
- Had worked on the implementation of a window to create prescriptions.
- Had worked on the implementation of a window for viewing patients.
- Had worked on the implementation of a window to create a new employee.
- Had worked on the implementation of a window for viewing employees.
- Had worked on designing data structures, system architecture, and use case diagrams as part of the design documentation.
- Had worked on the implementation of the window to gather full patients information for the first time (UI & Backend).
- Had worked on setting up firebase functions to host Node JS API and creating OTP verification in node JS API and creating OTP window input UI.
- Had created React Hooks, to store selected patents.
- Had worked on the setting auth layer in react-router.
- Had worked on code and components integrations and bug fixes and deployment issues (Chatbot API)
- Had worked on the deployment of react frontend to firebase hosting.
- Had worked on implementing database security rules in firestore and firebase Auth session management
- Had worked on creating supplementary react UI components.
- Had worked on creating an email POST endpoint to send dynamic emails.
- Had worked on implementing role-based access for each user.
- Had worked on wrapping up hospital staff features and linking all components.
- Had worked on the implementation of a window for patients to request reports.
- Had Implemented Admin Internal File Views and Functionality.
- Had Implemented mechanism to prevent malicious sign-in.
- Had Implemented user role-based access control.
- Had assisted the team in technical tasks, technology-stack setup and configuration.

2.2. Dhiraj Kumar Karthikeyan's contributions to the project:

- Had initialized the communication channel and Jira.
- Had worked on setting up a development plan.

- Had worked on the implementation of a window to create transactions.
- Had worked on writing and designing System UI descriptions in the design documentation.
- Had worked on the implementation of the window to view Lab-Tests (UI & Backend).
- Had worked on the revised transactions List to be displayed using the new Unified data table component.
- Had worked on fixing and editing the transactions logic.
- Had worked on the implementation of session management using Redux.
- Had worked on the enhancement to the view lab test page based on user types.
- Had helped teammates with React and functional flow.
- Had overviewd user roles testing and penetration testing.
- Ensured proper input validation across all functionalities.
- Designed and implemented transaction workflow.
- Initialized Linux VM to host blockchain api
- Had worked on creating and testing the blockchain api from the application

2.3. Tazreen Khan's contributions to the project:

- Scheduled all team meetings to work on the project.
- Had worked on the implementation of a login UI.
- Had worked on the implementation of a singing up UI
- Had worked on the implementation of a window to recommend a lab for patients.
- Had worked on research and used technology limitations in the design documentation.
- Had worked on the implementation of the window to recommend lab .
- Had worked on the implementation of the window for staff hospitals to view patients' appointments.

2.4. Smit Dharmesh Bhai Shah's contributions to the project:

- Had worked on the implementation of a window for appointments (Hospital Staff) (UI & Backend).
- Had worked on the implementation of a window to view appointments.
- Had worked on preparing testing plans and strategies for the design documentation.
- Had worked on the implementation of a window to create new appointments.
- Had worked on the implementation of the window to view appointment Details.
- Had worked on the implementation of the window for insurance staff to see pending transactions.
- Had worked on creating an appointment history view.
- Had worked on the implementation of the window to allow patients to see their past .

- Had worked on implementing the delete report functionality and updating the test report.
- Had worked on the implementation of view doctor recommendations as a popup inside lab tests details view.
- Had worked on creating the test report table view and implementation of test reports table view.
- Ensured all lab staff functionality are well maintained and fixed all discovered bugs.

2.5. Girija Rani Nimmagadda's contributions to the project:

- Had worked on the implementation of a window for prescriptions.
- Had worked on the implementation of a window to view transactions.
- Had worked on organizing design documents and authored the introduction.
- Have worked on the implementation of the window to Complete Transactions.
- Had worked on the implementation of the window to allow patients to see their prescription details (UI & Backend).
- Had ensured proper input validation across all functionalities.
- Had worked on the implementation of the revised prescription table UI.
- Had worked on implementation of a window to serve as a landing page for patients.
- Had worked on the final report of the project.

2.6. Eashwar Pallem's contributions to the project:

- Had worked on the implementation of a Dashboard UI.
- Had worked on the implementation of a window to view diagnosis details
- Had worked on the descriptive system Ui components for the design documentation
- Had worked on the implementation of a window For Patients To Request Lab Tests.
- Had worked on the implementation of transactions view for insurance staff.
- Had worked on the implementation of transaction details view for insurance.

2.7. Sahith Doma's contributions to the project:

- Had worked on the designing class & activity diagrams for the design documentation
- Had worked on identifying the requirements and functionalities to be added.
- Had worked on the user guide of the project.
- Had worked on the testing in production.
- Had worked on editing the DB diagram and system architecture diagram.
- Had worked on the implementation of the help and support page.

2.8. Pratyush Narayan's contributions to the project:

- Had worked on setting up a development plan.
- Had worked on the implementation of a window for Diagnosis .
- Had worked on the implementation of Chatbot backend
- Had worked on the designing sequence and misuse diagrams for the design document.
- Had worked on fixing the diagnosis list and changing the displayed data.
- Initialized hyperledger.
- Started Independent NodeJS API for blockchain and deployed it.
- Had worked on creating a blockchain to store transactions.
- Had worked on Chatbot UI and testing ML in production.
- Tested production against session tampering and spoofing.
- Ran penetration testing on production.
- Had worked on the patient profile form to edit and save personal details.

3. DESIGN DOCUMENT

3.1. Data Structures

The SHS will have several different data-structures in the front-end. These data-structures are to be transferred between views and serve in making smooth view load with minimal backend reliance and less backend requests. We are aiming towards having a light process on the main thread and faster load time, with regards to data security.

Internal software data structure

Internally we have constant templates of what data to be transferred, with regards to the data current state at a given transfer time.

- Patient info:
    - Contains:
        - Name
        - Patient-ID
        - User-ID
    - Views that use it:
        - Transactions View
        - Diagnosis View
        - Lab Tests View
        - Create Appointment View
        - Create/Edit Transaction View
        - Create Prescription View
        - Create Diagnosis

- Employee Info:
    - Contains:
        - Name
        - Employee-ID
        - User-ID
    - Views that use it:
        - Appointment View
        - Create Appointment
        - Create Patient Record
        - Diagnosis View
        - Lab Tests View
        - Appointments View
        - Transactions View
        - Create Transaction View

        - Employees View
        - Employee Info View
        - Logs View

- Transaction Info:
    - Contains:
        - Amount
        - Transaction-ID
        - Patient-ID
        - Status
    - Views that use it:
        - Transaction View

- Appointment Info:
    - Contains:
        - Appointment-ID
        - Patient-ID
        - Date
        - Status
        - Doctor
        - Type
    - Views that use it:
        - Appointment View
        - Create Prescription View
        - Create Diagnosis View
        - Create Lab Test View

- Claim Info:
    - Contains:
        - Claim-ID
        - Patient-ID
        - Amount
        - Date
        - Status
    - Views that use it:
        - Transaction View
        - Claim View

- Lab Test Info:
    - Contains:
        - Test-ID
        - Test Name
        - Date
        - Doctor-ID
        - Patient-ID
        - Type
        - Status
        - Comments
    - Views that use it:
        - Lab Test View

- Prescription Info:
    - Contains:
        - Prescription-ID
        - Patient-ID
    - Views that use it:
        - Prescription View

- OTP Code Data:
    - Contains:
        - Created Date
        - Expiration Date
        - Email
        - Hashed-iv
        - Hashed-content
        - Status
        - User-ID
    - Views that use it:
        - Transactions View
        - New Appointment View

- Payments Info:
    - Contains:
        - Amount
        - Date
        - Patient-ID

        - Transaction-ID
    - Views that use it:
        - Transaction Info View
        - Transactions Table View

- Doctor Recommandations Info:
    - Contains:
        - Date
        - Description
        - Employee-ID
        - Lab Name
        - Lab Type
        - Patient-ID
        - Recommandation-ID
    - Views that use it:
        - Labs Table View
        - Lab Details View
        - Recommend Lab View

- System Failure Info:
    - Contains:
        - Action / Failure Cause
        - Date
        - Error
        - User
    - Views that use it:
        - All Views

- System Logs Info (User Actions / Interactions:
    - Contains:
        - Action
        - Date
        - Log-ID
        - User
    - Views that use it:
        - All Views

Global data structure

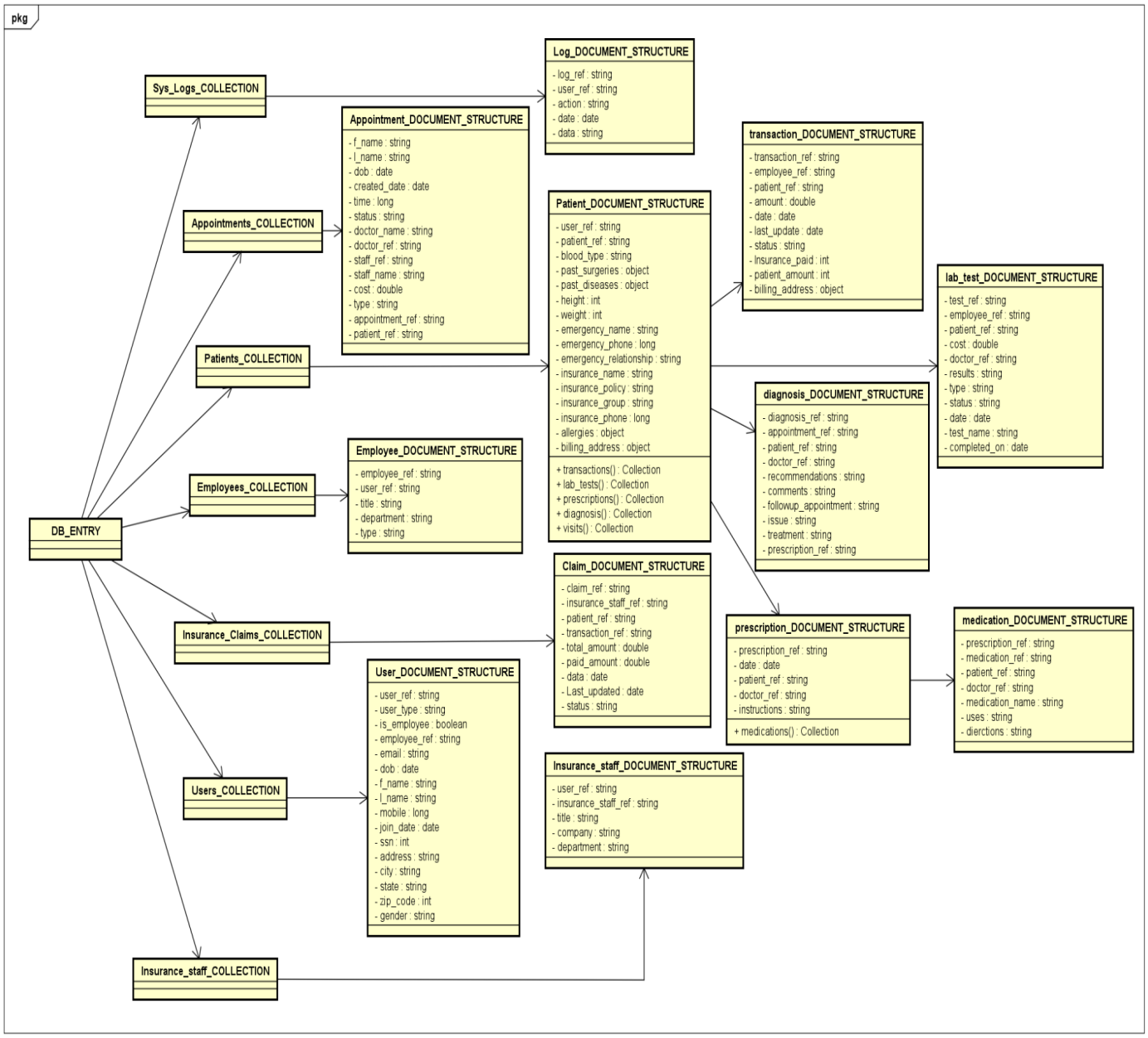
By utilizing Firebase Authentication service, we are able to get a user object that has the required and legitimate credentials to perform CRUD operations. Also we made react-hooks that store a constant object of other user metadata.

- Firebase User Object:
    - Contains:
        - Name
        - Email
        - Phone
        - Email verified
        - UID

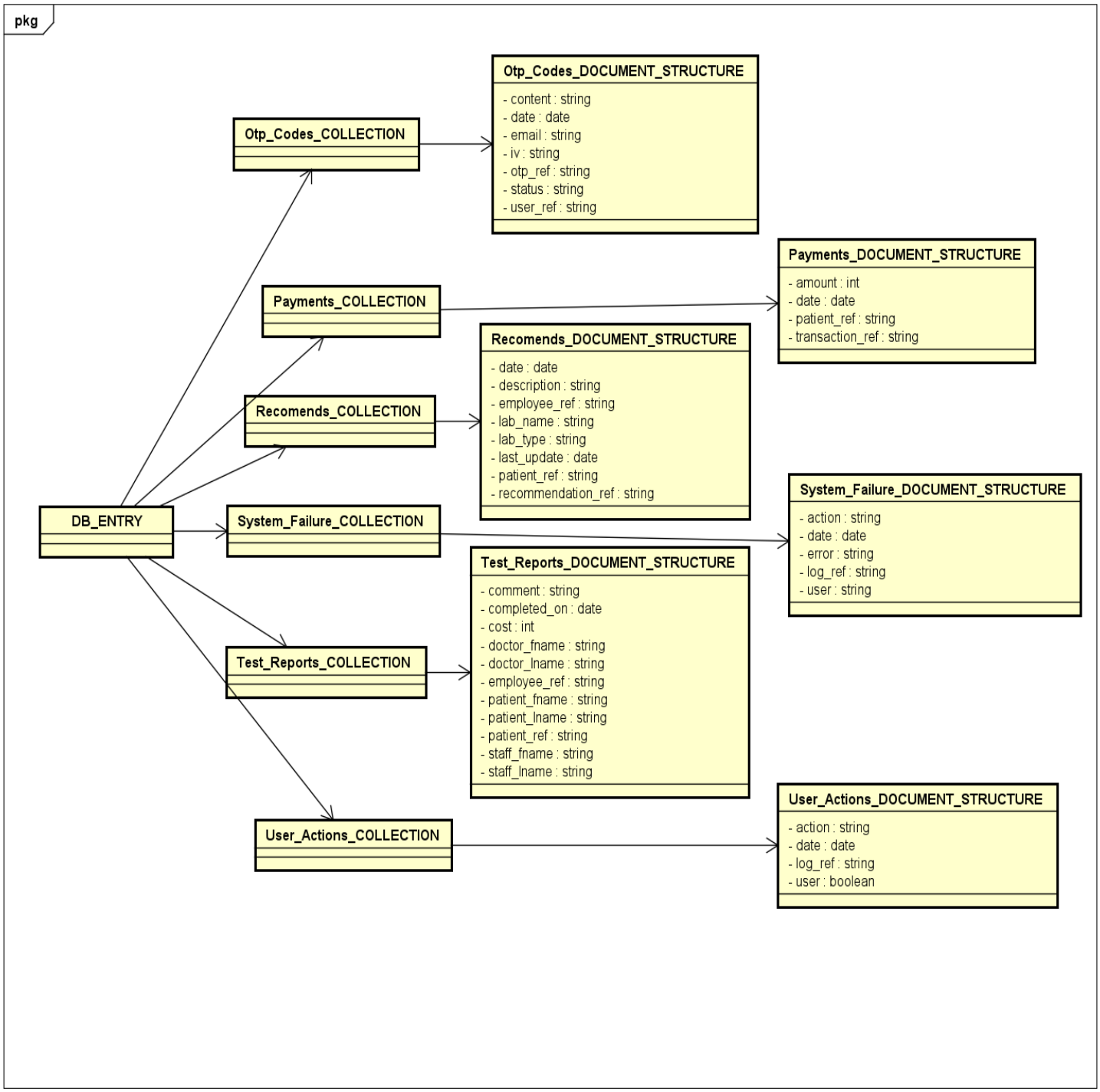
- User Type Hook:
    - Contains:
        - User-ID
        - UID
        - Type
        - Employee-ID
        - Patient-ID
        -

Database description

By using Firestore-DB, we came up with a flexible data storage and querying approach. As we have collections of documents for the main data-structures of the SHS. With regards to the metadata and needed data to track storage constraints and tracking each user type data. In the following database layout, are the details of our database structure.

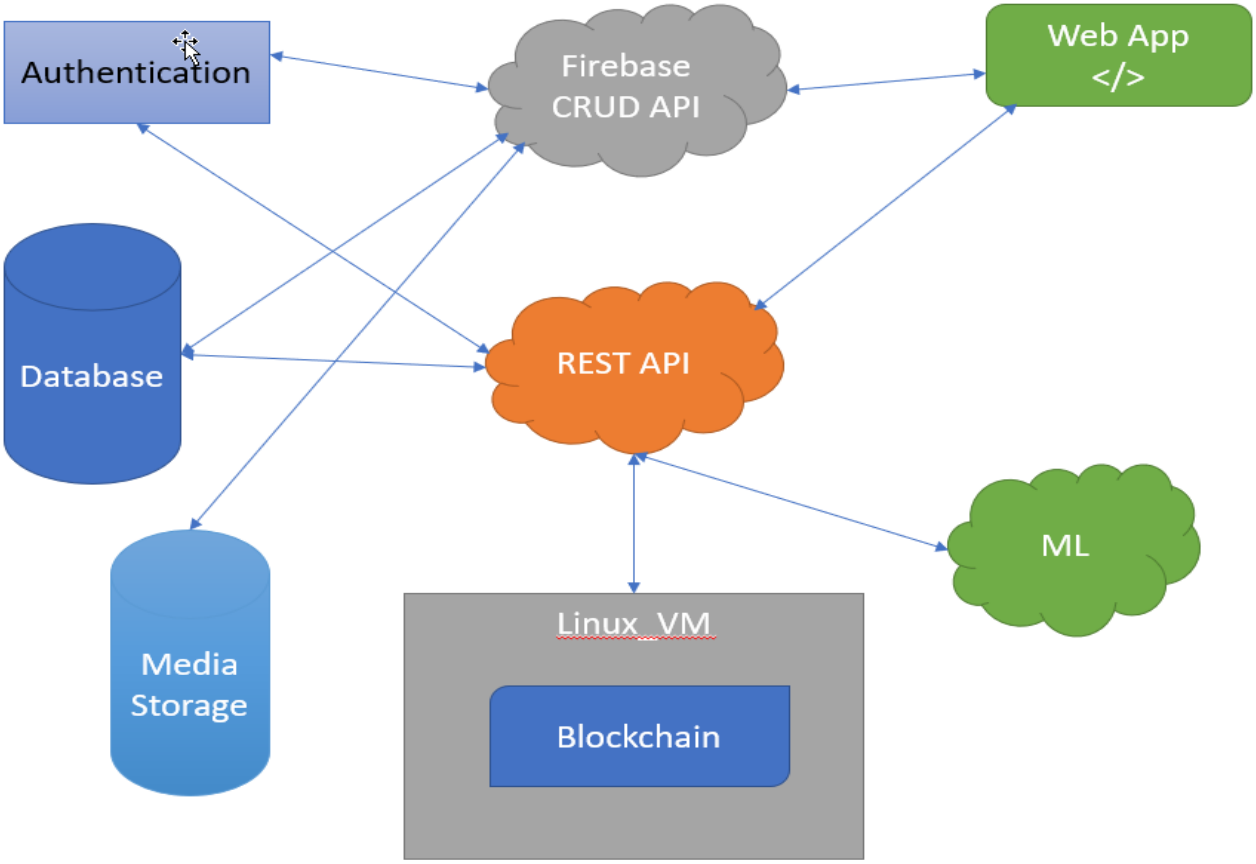

pkg
Sys_Logs_COLLECTION
Log_DOCUMENT_STRUCTURE
- log_ref : string
- user_ref : string
- action : string
- date : date
- data : string
Appointment_DOCUMENT_STRUCTURE
- f_name : string
- l_name : string
- dob : date
- created_date : date
- time : long
- status : string
- doctor_name : string
- doctor_ref : string
- staff_ref : string
- staff_name : string
- cost : double
- type : string
- appointment_ref : string
- patient_ref : string
transaction_DOCUMENT_STRUCTURE
- transaction_ref : string
- employee_ref : string
- patient_ref : string
- amount : double
- date : date
- last_update : date
- status : string
- Insurance_paid : int
- patient_amount : int
- billing_address : object
Appointments_COLLECTION
Patient_DOCUMENT_STRUCTURE
- user_ref : string
- patient_ref : string
- blood_type : string
- past_surgeries : object
- past_diseases : object
- height : int
- weight : int
- emergency_name : string
- emergency_phone : long
- emergency_relationship : string
- insurance_name : string
- insurance_policy : string
- insurance_group : string
- insurance_phone : long
- allergies : object
- billing_address : object
+ transactions() : Collection
+ lab_tests() : Collection
+ prescriptions() : Collection
+ diagnosis() : Collection
+ visits() : Collection
lab_test_DOCUMENT_STRUCTURE
- test_ref : string
- employee_ref : string
- patient_ref : string
- cost : double
- doctor_ref : string
- results : string
- type : string
- status : string
- date : date
- test_name : string
- completed_on : date
Patients_COLLECTION
diagnosis_DOCUMENT_STRUCTURE
- diagnosis_ref : string
- appointment_ref : string
- patient_ref : string
- doctor_ref : string
- recommendations : string
- comments : string
- followup_appointment : string
- issue : string
- treatment : string
- prescription_ref : string
Employee_DOCUMENT_STRUCTURE
- employee_ref : string
- user_ref : string
- title : string
- department : string
- type : string
Employees_COLLECTION
DB_ENTRY
Claim_DOCUMENT_STRUCTURE
- claim_ref : string
- insurance_staff_ref : string
- patient_ref : string
- transaction_ref : string
- total_amount : double
- paid_amount : double
- data : date
- Last_updated : date
- status : string
Insurance_Claims_COLLECTION
prescription_DOCUMENT_STRUCTURE
- prescription_ref : string
- date : date
- patient_ref : string
- doctor_ref : string
- instructions : string
+ medications() : Collection
medication_DOCUMENT_STRUCTURE
- prescription_ref : string
- medication_ref : string
- patient_ref : string
- doctor_ref : string
- medication_name : string
- uses : string
- dierctions : string
User_DOCUMENT_STRUCTURE
- user_ref : string
- user_type : string
- is_employee : boolean
- employee_ref : string
- email : string
- dob : date
- f_name : string
- l_name : string
- mobile : long
- join_date : date
- ssn : int
- address : string
- city : string
- state : string
- zip_code : int
- gender : string
Insurance_staff_DOCUMENT_STRUCTURE
- user_ref : string
- insurance_staff_ref : string
- title : string
- company : string
- department : string
Users_COLLECTION
Insurance_staff_COLLECTION

Extending the above diagram.



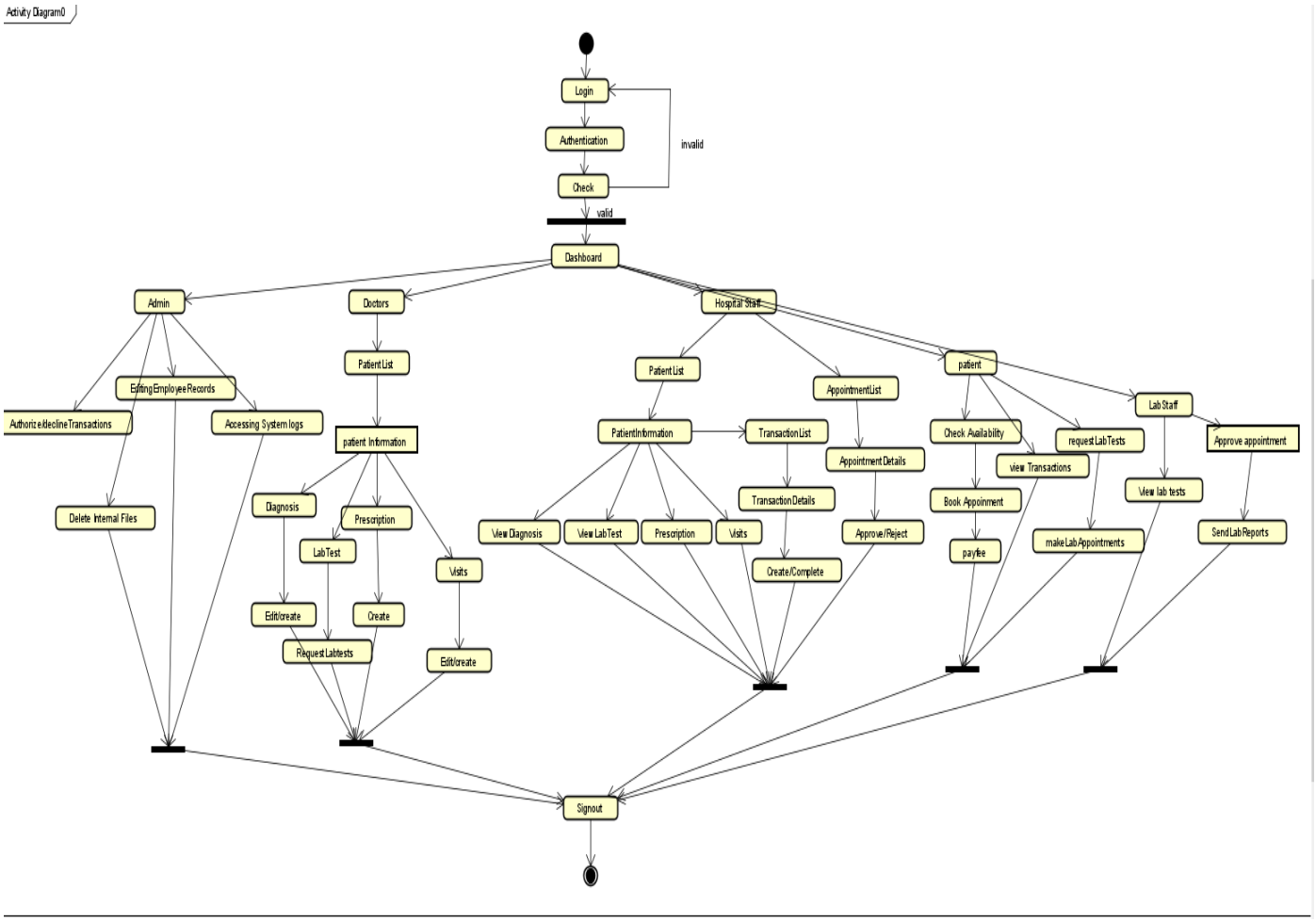
3.2. Architectural and Component Design

System Architecture Diagram

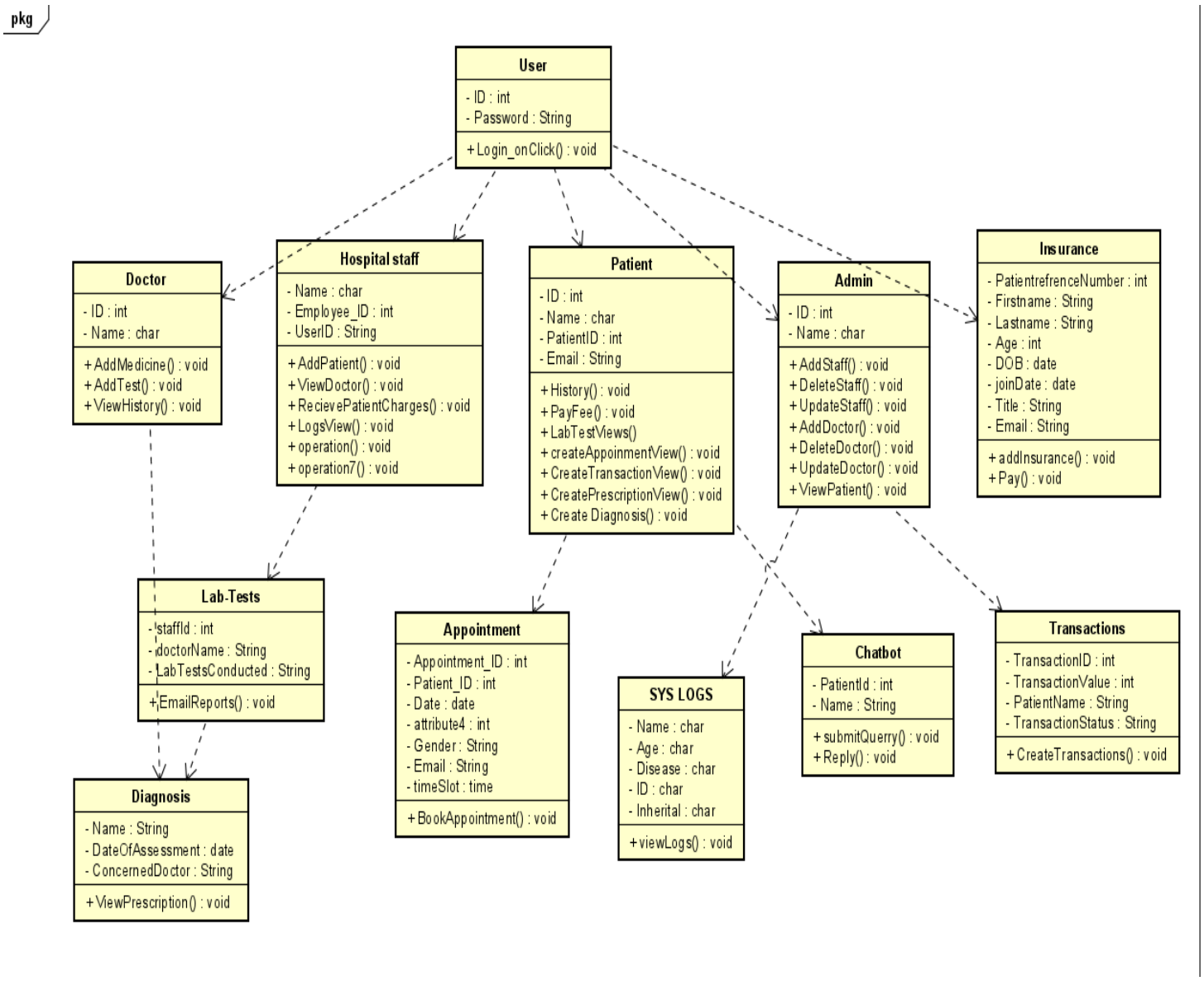


UML Diagrams (Below)

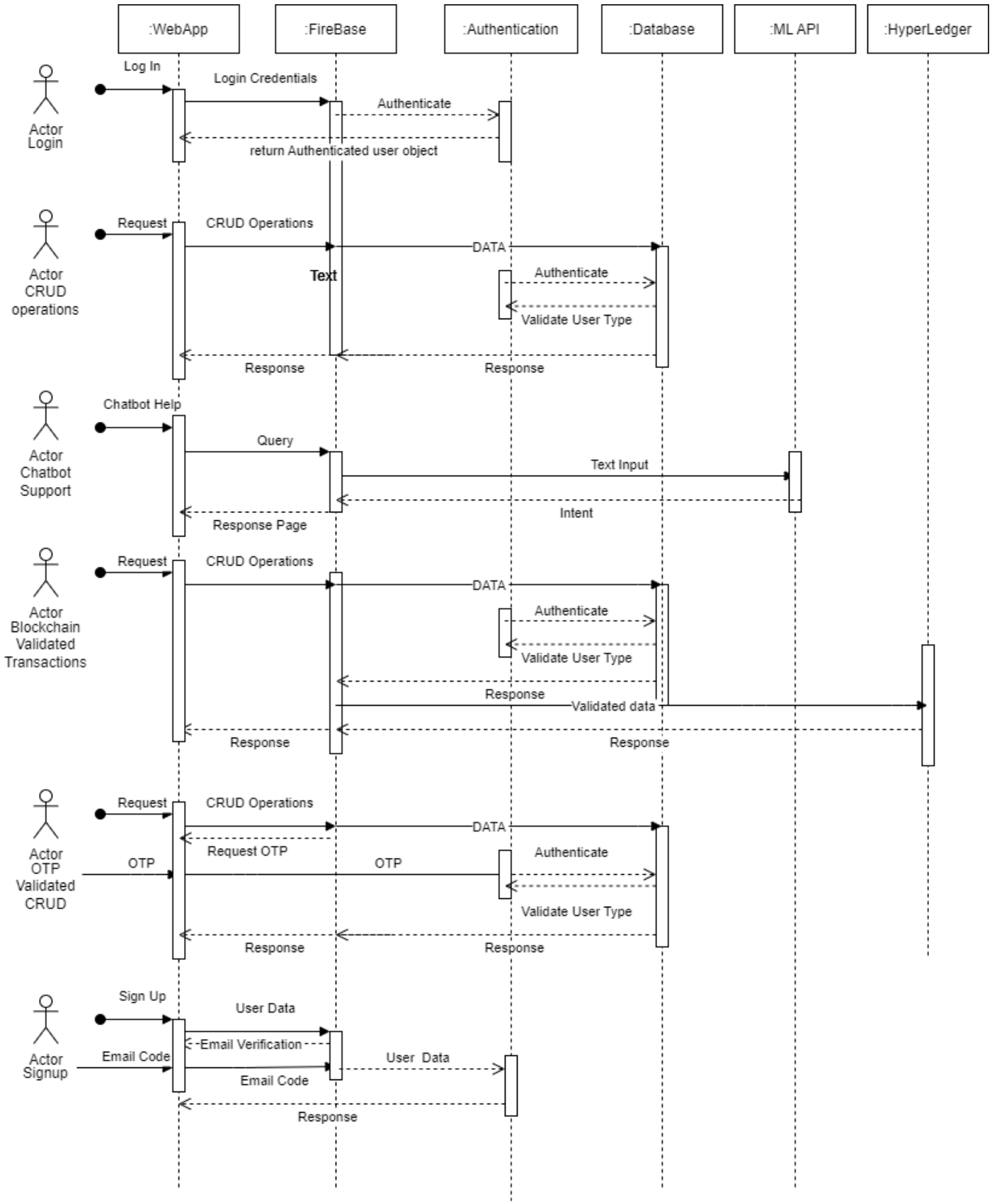
- **Activity Diagram**



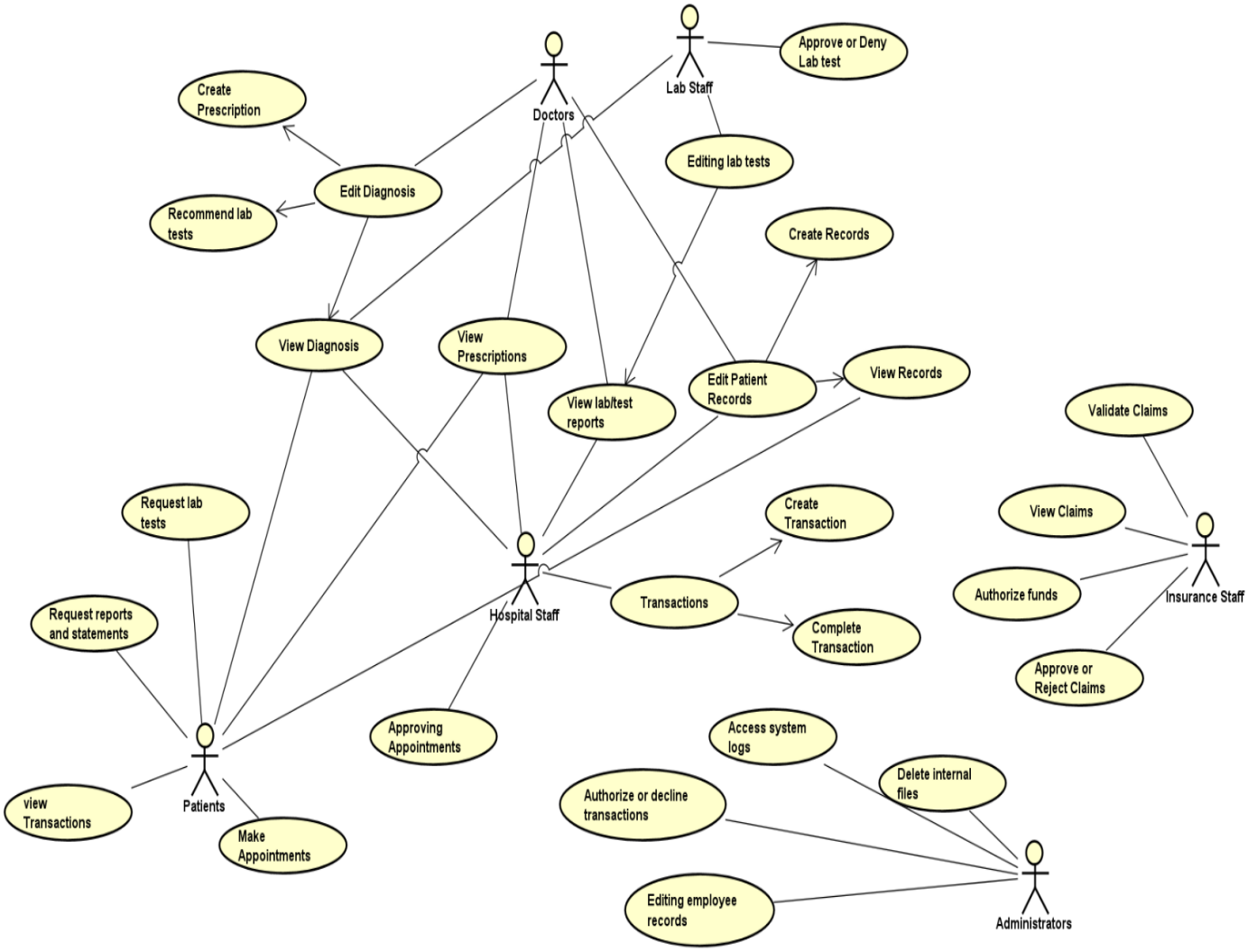
- **Class Diagram**



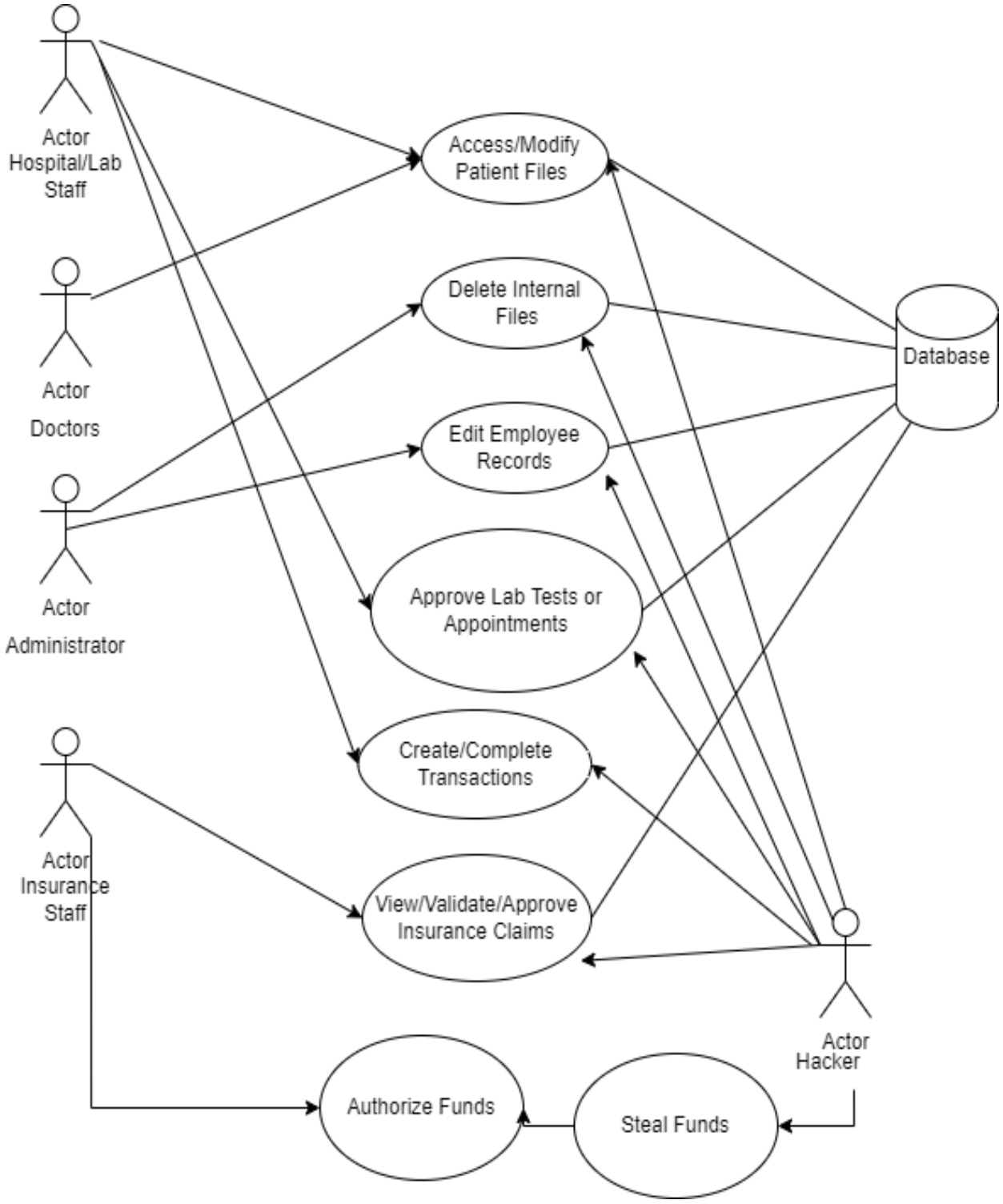
- **Sequence Diagram**



- **Use Case Diagram**



- **Mis-Use Case Diagram**



Misuse Case scenarios :

- Hacker Impersonating Hospital staff/ Doctor/ Lab staff to access or modify the patient files or records.
- Hacker authorizing invalid insurance claims.
- Hacker Authorizing insurance funds for invalid transactions stealing money from hospitals.
- Hacker performing invalid CRUD operations or sending huge volumes of garbage data to block the read/write port or deny the service.
- Hacker accessing log files, Employee information and deleting system files critical to application.

3.3. User Interface Design

Available Components

Our current implementation uses builtin components from the material UI library and also certain custom components. At the time of creating this document we have implemented the following custom components. Action-Button, Table, Textarea and Textfield.

Objects and actions

Login Page: Fields for username and password entry.
Generic sign-in functionality with the option to reset the password. Sign-up button to take the user to the Sign-Up page.

Sign-up page: Fields for creating a new username, password and email. Sign-up functionality

ENTITY	DESCRIPTION
Username and Password	Accepts a new or existing username and password depending on the context
Sign-In and Sign-Up	These are generic functionalities for existing or new users to gain access to the system

Patients home Page: An actions button showcasing the different operations that a patient can do. Which includes creating an appointment, viewing lab reports, transactions, diagnosis and prescriptions. The page also consists of a form consisting of the basic patient information where certain fields are editable. A Help icon to request for any further assitance and chat bot to help with general queries.

ENTITY	DESCRIPTION
Actions Button	This button has different operations that a patient can perform. Which includes creating an appointment, viewing lab reports, transactions, diagnosis and prescriptions.
Help	To request for any further assitance
Simple personal details form	basic patient information where certain fields are editable.
Chat bot	To help with general queries.

View Transactions: The page consists of a search bar that can search based on name , transaction value, patient name and transaction status. A button is provided to create a new transaction for an existing patient. The view also provides a one click functionality to create receipts on a transaction that is complete.

ENTITY	DESCRIPTION
Search Bar	search based on name, transaction value, patient name and transaction status
Create New Transaction	Navigates to the create transaction page
Create Receipts	one click functionality to create receipts on a transaction that is complete.
Chat bot	To help with general queries.

Create Transactions:This page consists of a form with few pre-filled values like the transaction number, hospital staff ID and patient ID. Once the other necessary details like transaction amount, due date and billing address are filled the transaction can be submitted to the admin for review.

ENTITY	DESCRIPTION
Basic Form	with few pre-filled values like the transaction number, hospital staff ID and patient ID.
Submit Transaction	The transaction is submitted for admin review.

Lab Report page: This page has the doctor and the lab staff details for the current report. The lab tests performed and an assessment report with a brief description of the results.

ENTITY	DESCRIPTION
Basic Form	Doctor and the lab staff details for the current report along with the date on which the report was taken.

Lab Test Conducted and Lab Report Assessment	Two text areas displaying the details of the test performed and its respective results.

Diagnosis Page: This page is a generic web page containing elements that take various inputs from the Doctor in order to document the diagnosis of a patient. This page consists of a textual space to enter the name of the patient, Ailment the patient is suffering from, concerned doctor, treatment recommended, assessment of the treatment and prescription. It also has a date entry space that takes the date as input to know the consultation and diagnosis date. This page is primarily controlled by the doctor but also has secondary access to the doctor and lab staff.

ENTITY	**DESCRIPTION**
Patient_Name	The patient name is a unique identifier required for the record to identify the patient uniquely. It is the doctor that can manipulate this space. Apart from that the admin, Patient, lab, and hospital staff can analyze it but not manipulate
Date_Of_Assessment	A date format that denotes the entry date of the patient into the system. It is the doctor that can manipulate this space. Apart from that, the admin, Patient, lab, and hospital staff can analyze it but not manipulate
Ailement_suffering_from	Denotes the ailment the patient is suffering from diagnosed by the doctor. It is the doctor that can manipulate this space. Apart from that, the admin, Patient, lab, and hospital staff can analyze it but not manipulate
Concerned_Doctor	The doctor's name is a unique identifier required to identify the doctor who treated the patient. It is the doctor that can manipulate this space apart from that the admin, Patient, lab and hospital staff can analyse it but not manipulate

Treatment	A textual space that denotes the treatment that the doctor gave the patient. It is the doctor that can manipulate this space apart from that the admin, Patient, lab and hospital staff can analyse it but not manipulate
Assessment_Result	A textual space denotes the output of the treatment provided by the doctor to the concerned patient. It is the doctor that can manipulate this space apart from that the admin, Patient, lab and hospital staff can analyze it but not manipulate
Prescription	A textual space that denotes the medicine prescribed to the patient. It is the doctor that can manipulate this space apart from that the admin, Patient, lab and hospital staff can analyze it but not manipulate

Appointment Page: This page is a generic web page that has elements that take input from the patient in order to book an appointment with the doctor. This page provides various textual spaces to enter the Full Name, Phone number,Address and email address. It also provides a dropdown menu to select the gender. Also select the date of birth and date of consulting the doctor. Time stamps are provided to select the comfortable time. This page is primarily manipulated by the patient and also has access to the doctor, admin, lab and hospital staff.

ENTITY	**DESCRIPTION**
Full_name	The Full Name is a unique identifier required for the record to identify the patient uniquely. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
Gender	Gender is an entity that denotes the gender of the patient. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only

	analyze.
Phn_no	It is a unique identifier to specify the phone number of the patient. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
DOB	It is an entity denoting the date of birth of the patient. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
Address	The address is a unique set of strings that denotes the patient's place of residence. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
Email	The email id of the patient is stored to create a point of contact. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
Appointment_Date	It is the date that denotes the date of appointment with the doctor. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.
Time_slots	A timestamp that denotes the time the patient had an appointment with the doctor on the intended date. The Patient and admin can manipulate the data here whereas the Doctor, lab, and hospital staff can only analyze.

Prescription Page: the prescription page provides textual space to enter the name of the medicine and strength. It provides dropdown menu options to select the type and per day count of the medicine. It also provides a check box to select the option for Refilling after a month. This page is primarily handled by the doctor and he can only manipulate it apart from the lab staff. Whereas the patient can analyze the page.

ENTITY	DESCRIPTION
Name	The Full Name is a unique identifier required for the record to identify the patient uniquely.
Type	Type consists of various categories of medicine to be prescribed to the patient
Strength	It is a string that denotes the amount of medicine to be prescribed to the patient.
Per_day_count	It is an integer that denotes the number of times the patient needs to take in the medicine per day.
Refill_After_A_Month	It is a check box that enables the patient to renew his subscription after every month.

Dashboard Page: The dashboard page gives an overall view of what the website as whole contains. It basically contains all the elements required for the web page in other words this page can access any other web page directly or indirectly. This page only contains buttons that navigate to different pages based on the button the user clicked. This page also contains some dropdown options for some buttons. The dashboard here is dynamic which is based on the type of user that logged only a certain web pages are accessible to the user.

ENTITY	DESCRIPTION
Admin	This is a unique identifier assigned to a button that does the job of navigating the user to the admin page or admin-controlled pages. Only the admin has access to this functionality.

Patient	This is a unique identifier assigned to a button that does the job of navigating the user to the patient web page. The patient is the primary user of this webpage and apart from that the doctor, admin, hospital, and lab staff also have access to this page.
View_Records	This is a dropdown option of the patient button. This button navigates you to the Diagnosis page. This page is a read-only page for the patient and is primarily controlled by the doctor
Take_An_Appointment	This is a dropdown option of the patient button. This button navigates you to the Appointment page. This page allows the patient to make changes to book an appointment with the doctor. The doctor, admin have access to this page too.
Doctor	This is a unique identifier assigned to a button that does the job of navigating the user to the doctor web page. The patient is the primary user of this webpage and apart from that the admin, hospital, and lab staff also have access to this page.
Patient_Diagnosis	This is a dropdown option of the doctor button. This button navigates you to the Diagnosis page. This is primarily controlled by the doctor and This page is a read-only page for the patient.
Prescriptions	This is a dropdown option to the doctor button that navigates you to the prescription page which is controlled by the doctor primarily. This page allows doctors to prescribe medicine to the patient. Apart from that this page can also be accessed by the admin, hospital staff.

Lab_Test	This is a dropdown option to the doctor button that navigates you to the prescription page which is controlled by the doctor primarily. This page allows doctors to prescribe medicine to the patient. This page is primarily controlled by both doctors and lab_staff.
Hospital_staff	This is a unique identifier assigned to a button that does the job of navigating the user to the Hospital staff page or Hospital Staff-controlled pages. Apart from Hospital staff the admin also has access to this page.
Lab_Staff	This is a unique identifier assigned to a button that does the job of navigating the user to the Lab staff page or Lab Staff-controlled pages. Apart from Lab staff the admin also has access to this page.
Insurance	This is a unique identifier assigned to a button that does the job of navigating the user to the Insurance. Apart from Lab staff the admin also have access to this page.

Restrictions and Constraints

1) When handling the firebase real-time DB in the future, we need to consider simultaneous responses into one single DB (in this case the firebase). Restrict the number of responses that are sent by a server to a single DB at any point. We need to consider how with each response pushed by the DB, it can send around thousands of responses per second to users. Although for our current server, only a few students would access it, it is a good idea to take into consideration this limit when developing the system.

2) For cloud messaging in the firebase, we should limit the number of reads or write functions triggered from a single function, which may trigger even more functions creating a domino effect and may mess things up.

3) While using React as our UI builder creating the interface of the system, we need to consider it. Since it focuses on the UI component out of all program

logic (MVC pattern), the backend implementation of the UI has been a struggle. Including integrating everyone's separate code.

4) Constraints also occurred at the beginning of the project due to a lack of knowledge behind JSX with its' React library. While working with JSX was not difficult, including the REACT library was hard to work with since JSX is not very intuitive and trying to work things out.

3.4. Testing Issues

Test strategy and preliminary test case specification are presented in this section. The goal of testing is to ensure that the system performs as per the functional and non-requirements specified in the document.

3.5. Unit Testing

Unit testing was performed on each task of the project before pushing code to the main git repository. The parts of code that were assigned to each team member were performed with unit testing. That way, it would be much easier to detect and fix any problems that were found by the person who had programmed the code. Validating each unit of code before pushing it to the main code helped to set up the overall layout of the application before moving on to the security application.

Classes of Tests

- **Black Box Testing**

1. Functional Testing: The goal of this testing is to validate the functional requirements/specifications of the system.

2. Security Testing: The goal of this testing is to make sure the system is secure and can protect confidential information against internal or external hackers. This includes the testing of blockchain platforms to secure medical records.

3. Regression Testing: The goal of this testing is to make sure that the code change or any addition has not broken any additional functionality.

4. Performance Testing: The goal of this testing is to make sure response time of any event does not exceed 3-5 seconds, the system can withstand user load of approximately 50 users per second and operate in a 24/7

environment.

5. Testing ML model for chatbot: The goal of this testing is to determine the accuracy and precision of the chatbot

- **White Box Testing**

1. Unit Testing: The goal of this testing is to make sure the code written for a single function is working as expected before continuing. This will be done by the individual developer to identify the bugs in early stages before integrating the functionality with the system.

2. Integration Testing: The goal of this testing is to make sure when two individual components/modules are integrated, their functionalities as well as the communication between them and the external system work as expected.

3. Statement coverage: The goal of this testing is to make sure that each line of the source code has been executed and tested at least once.
4. Memory Leak Testing: The goal of this testing is to make sure that any sort of memory leaks is identified to optimize the application and protect it from a security standpoint.

Performance Bounds

1) Response time of any event should not exceed the standard response time for hospital applications (3 to 5 seconds).

2) System should withstand a user load of approximately 50 users per second.

3) System should operate in a 24/7 environment.

Identifying Critical Components

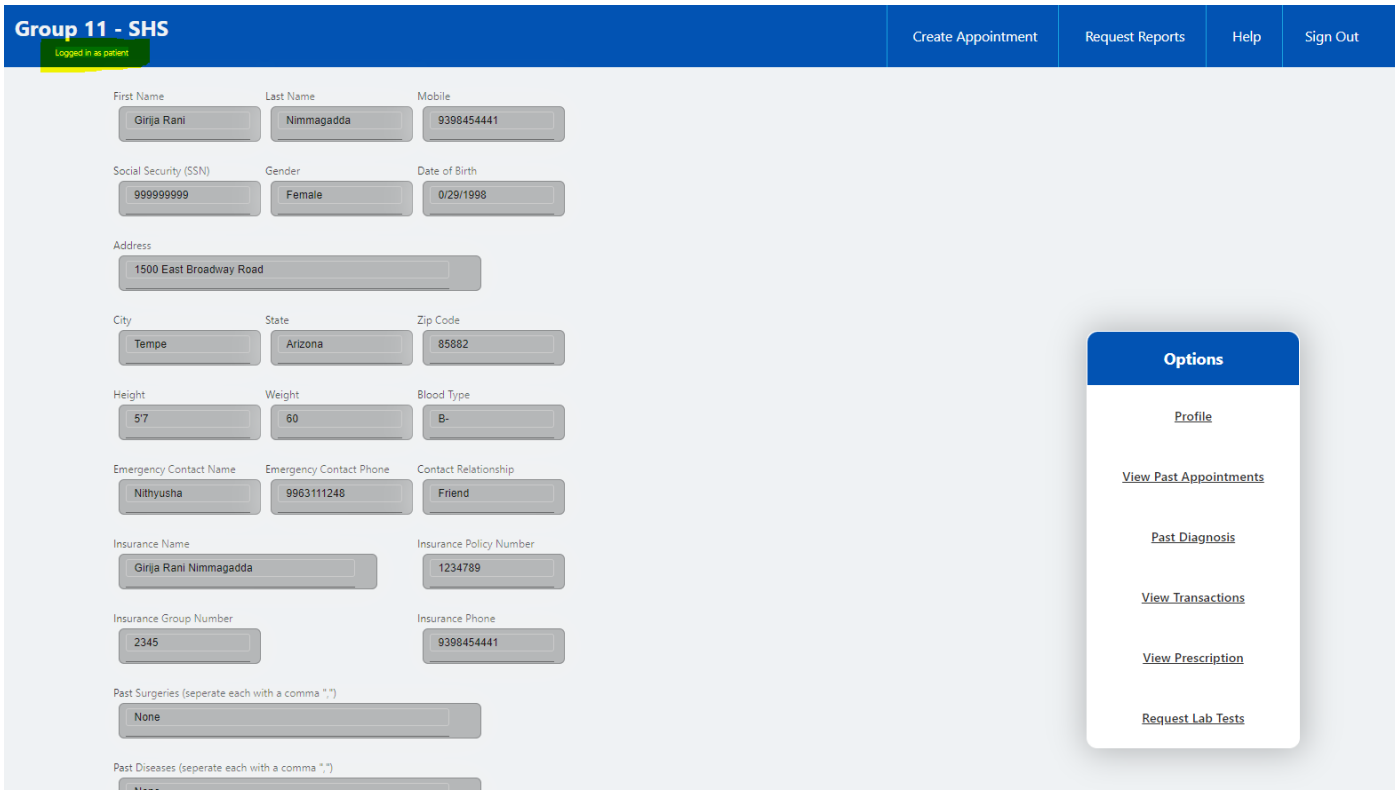
The following components are critical and demand particular attention during testing:

Auth Functionality(Sign-in, Sign-up and OTP Verification)
A patient should be able to book an appointment for a particular doctor or a general appointment

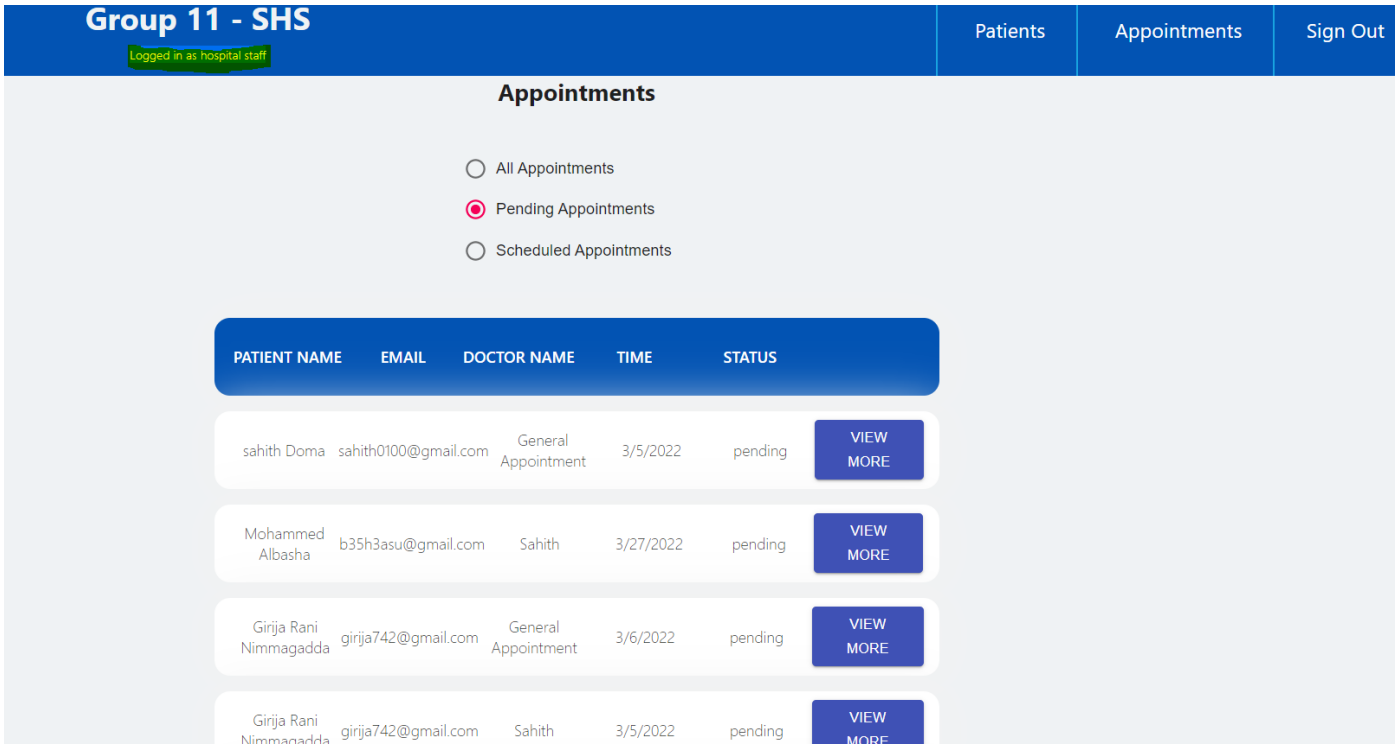
Hospital Staff can view, approve/deny appointments based on doctor’s availability
A window where patients can write symptoms and health issues to the doctor
A window where doctor can provide his/her diagnosis to the patient
A patient can request a lab test based on the recommendations of the doctor
Lab staff can view, approve/deny lab test requests based on the recommendation of the doctor
A patient can request an insurance claim
Insurance staff can view, approve/deny insurance claim requests
Once the insurance request is approved, insurance staff can authorize to transfer funds to the patient’s account
A patient can update his/her personal and contact information
A help box to assist patients with queries
A chat box to help users with general queries
The system should be secured with public key certificate
All medical records including diagnosis reports, approved insurance claim requests and approved transactions are captured in Hyperledger Blockchain Platform

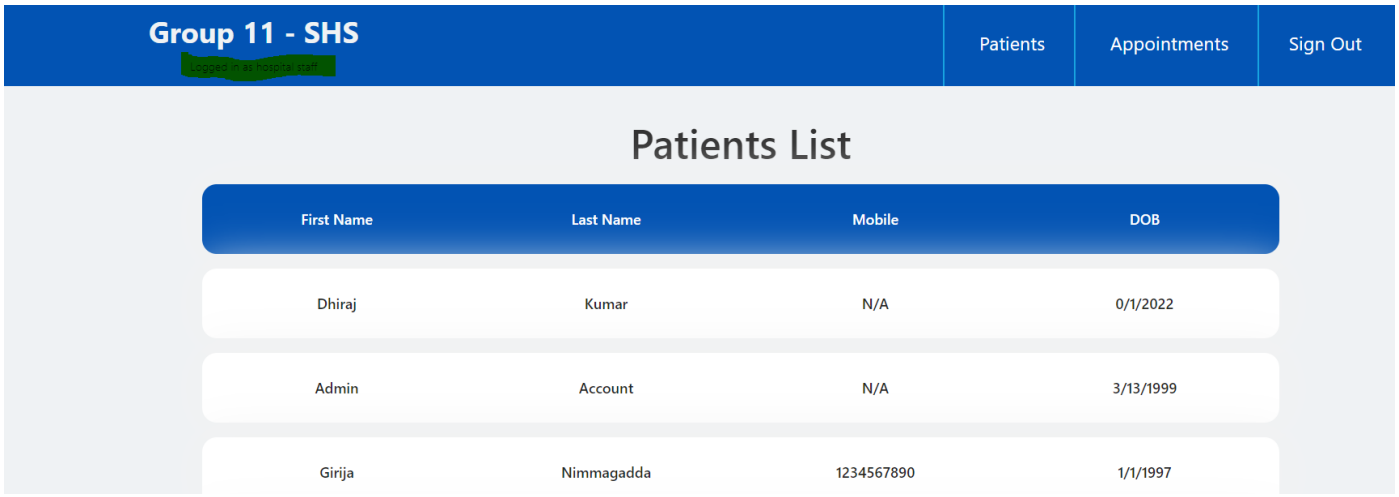
4. APPLICATION SCREENSHOTS

- **Patient dashboard**

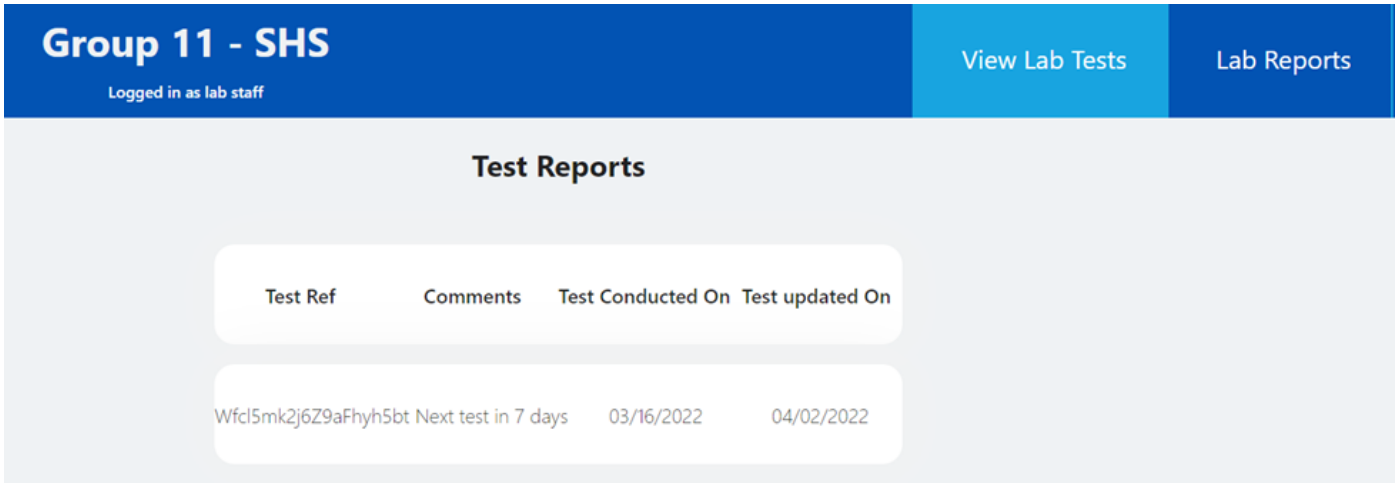


- **Hospital-Staff dashboard**

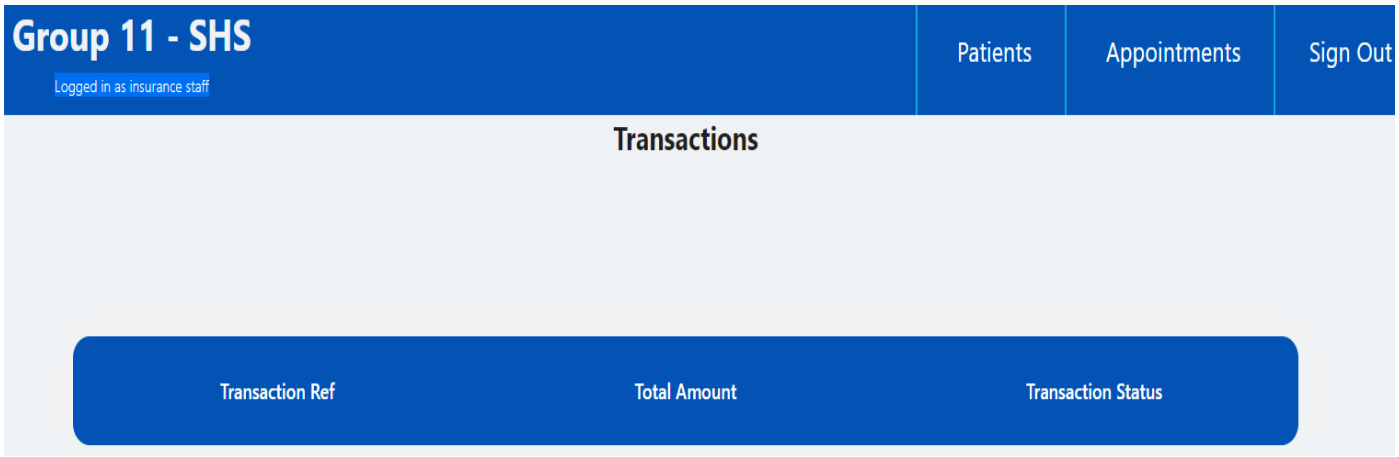




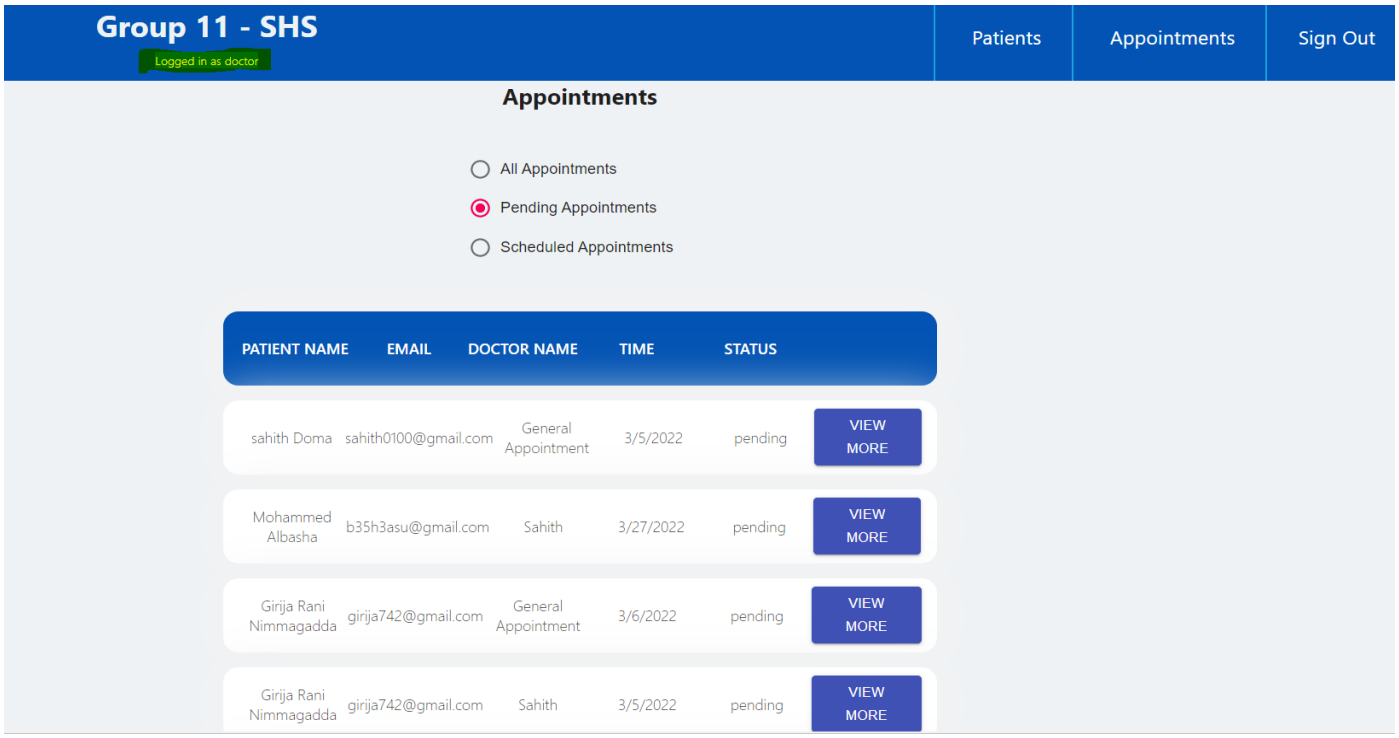
- **Lab-Staff dashboard**



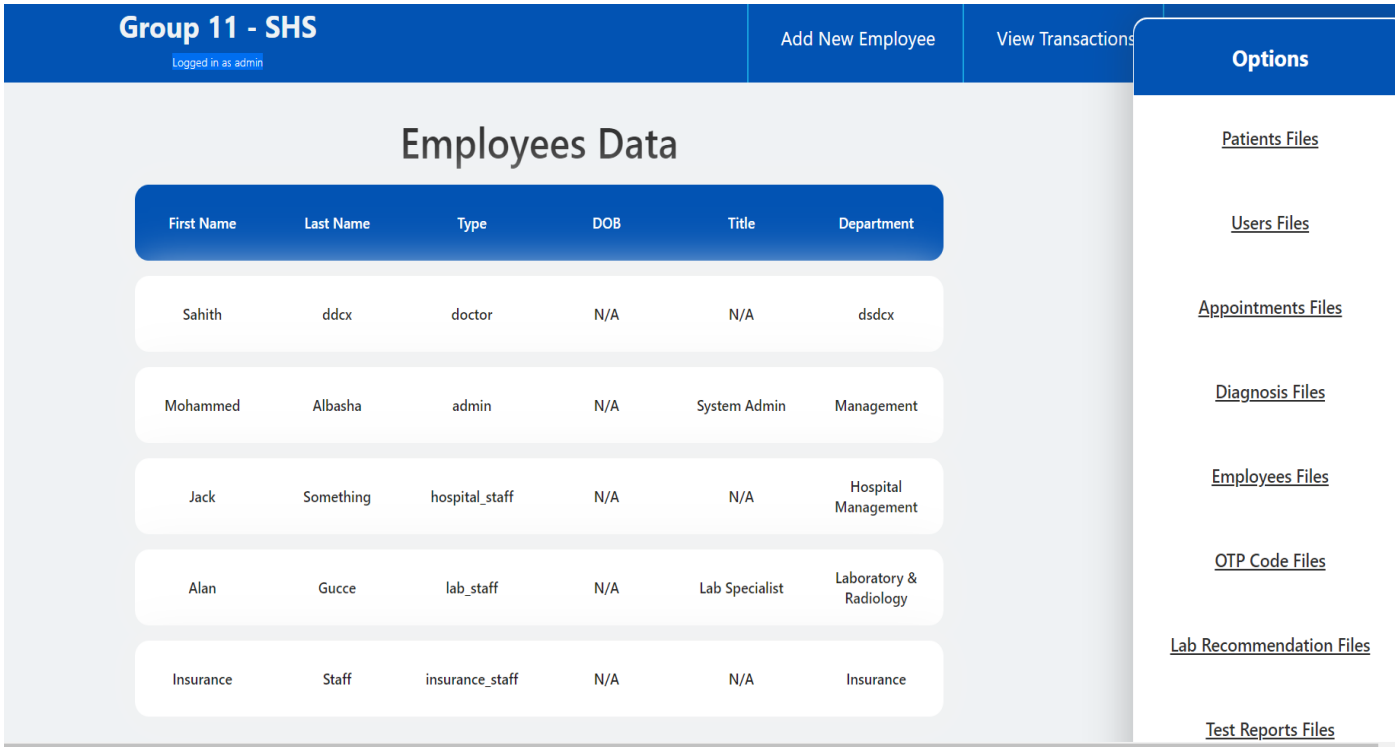
- **Insurance-Staff dashboard**



- **Doctor dashboard**

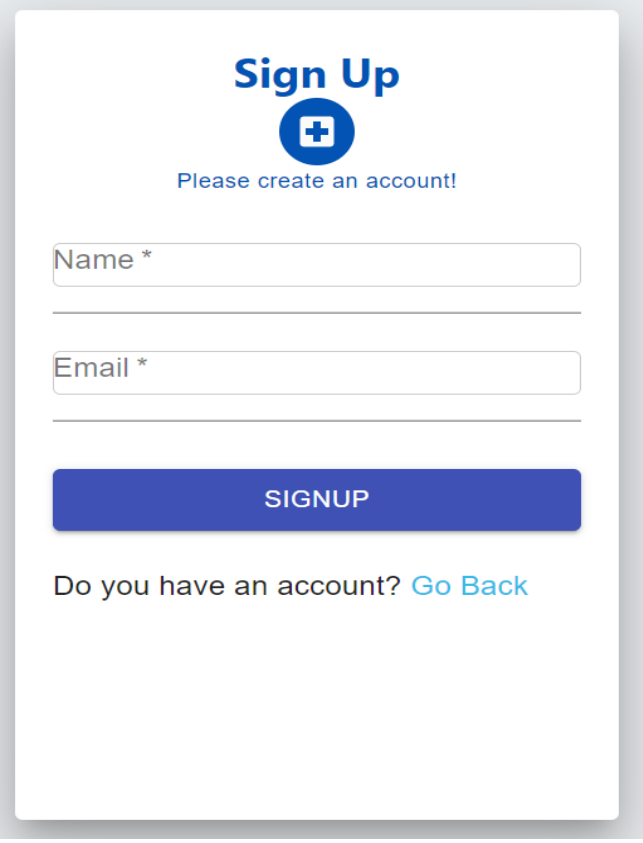


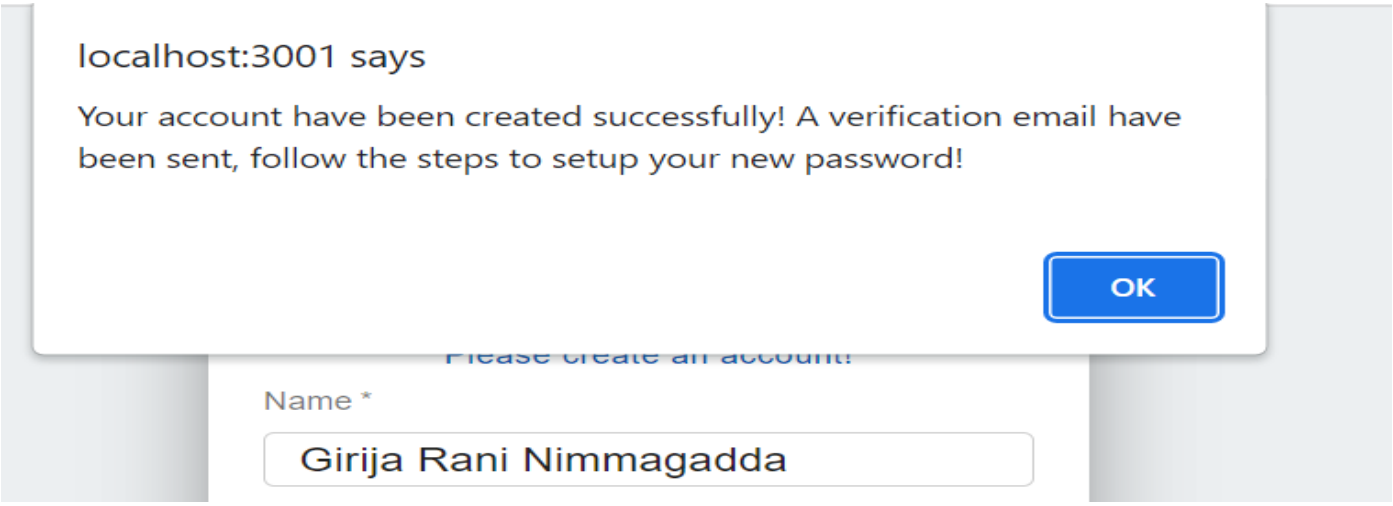
- **Admin dashboard**



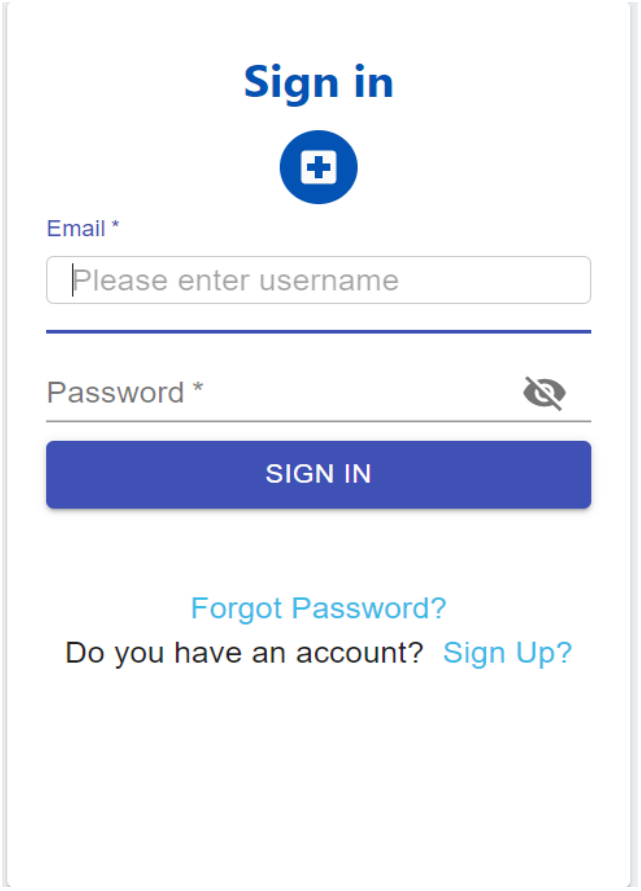
- **Sign-in / Sign-Up**

Signup:

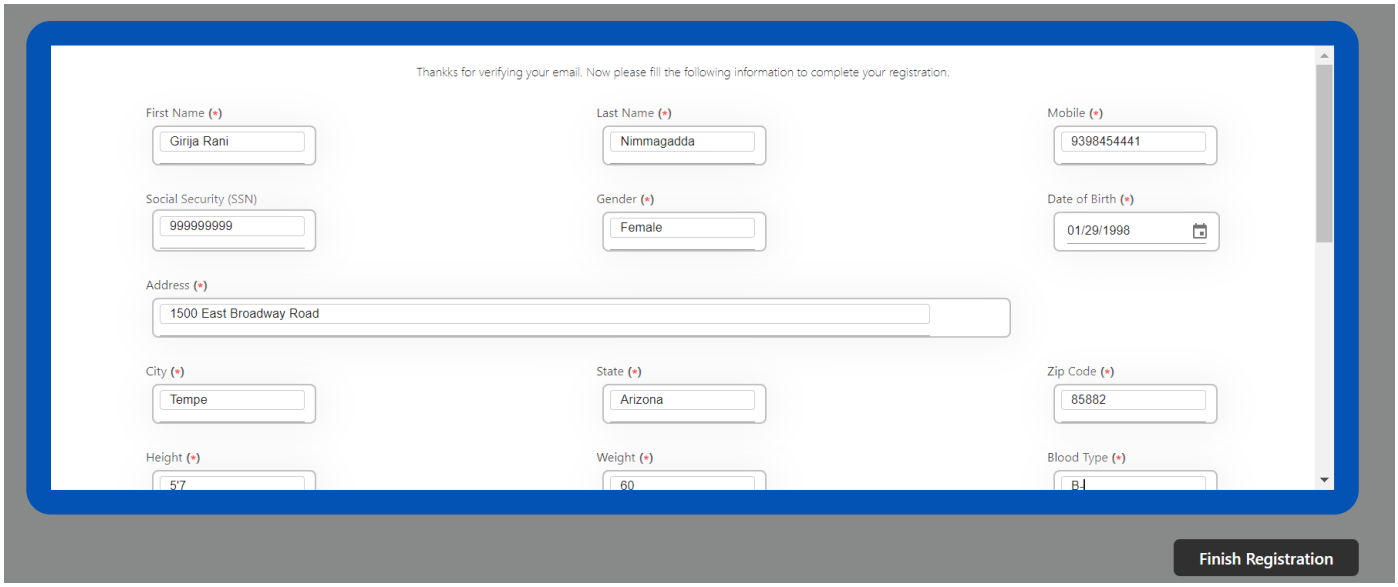


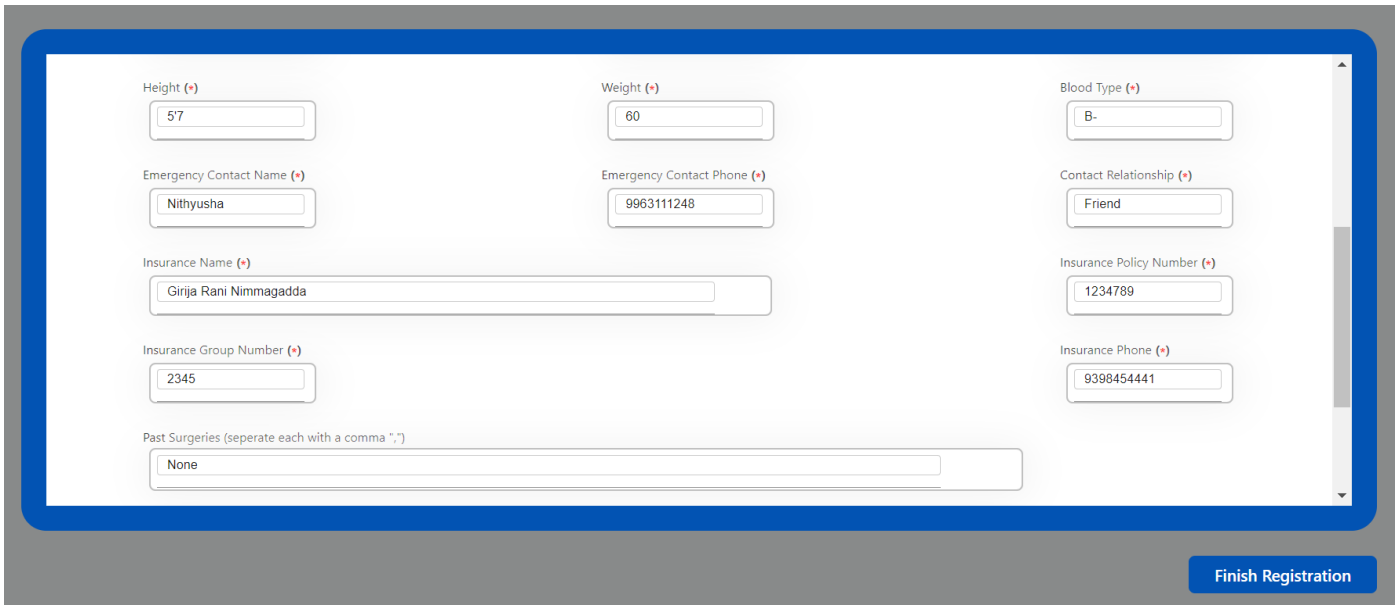


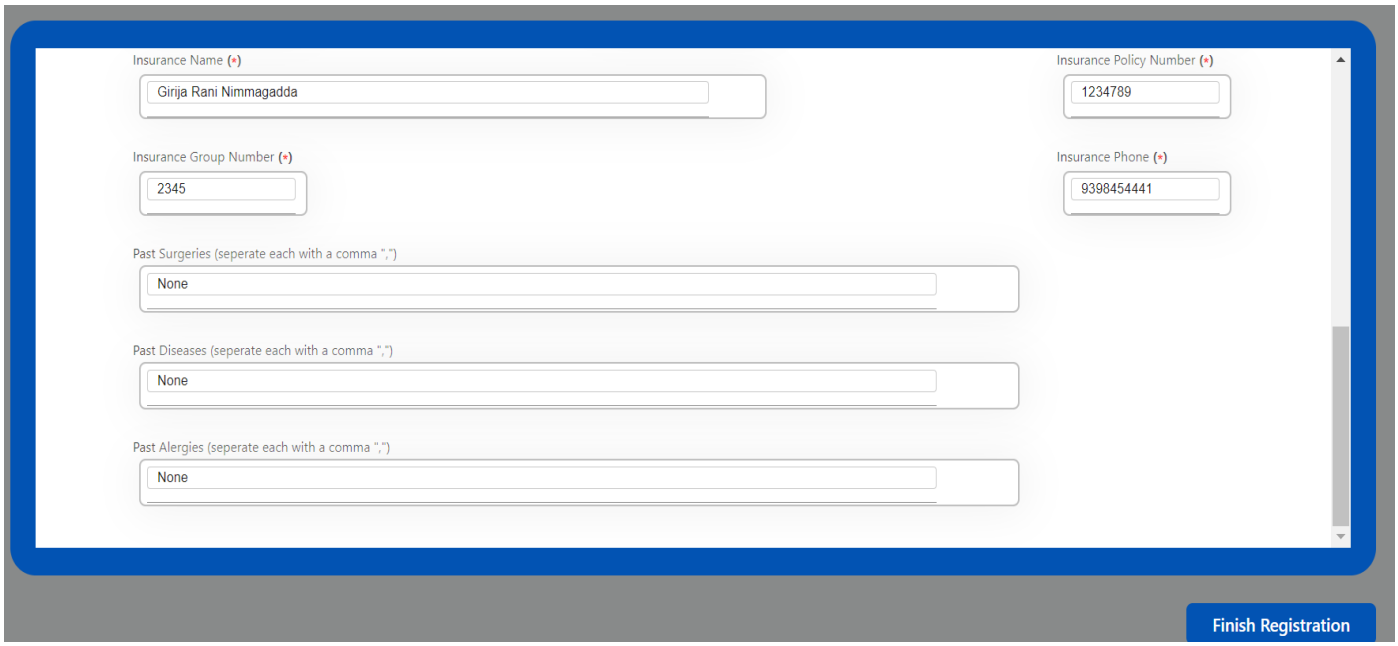
Sign in :

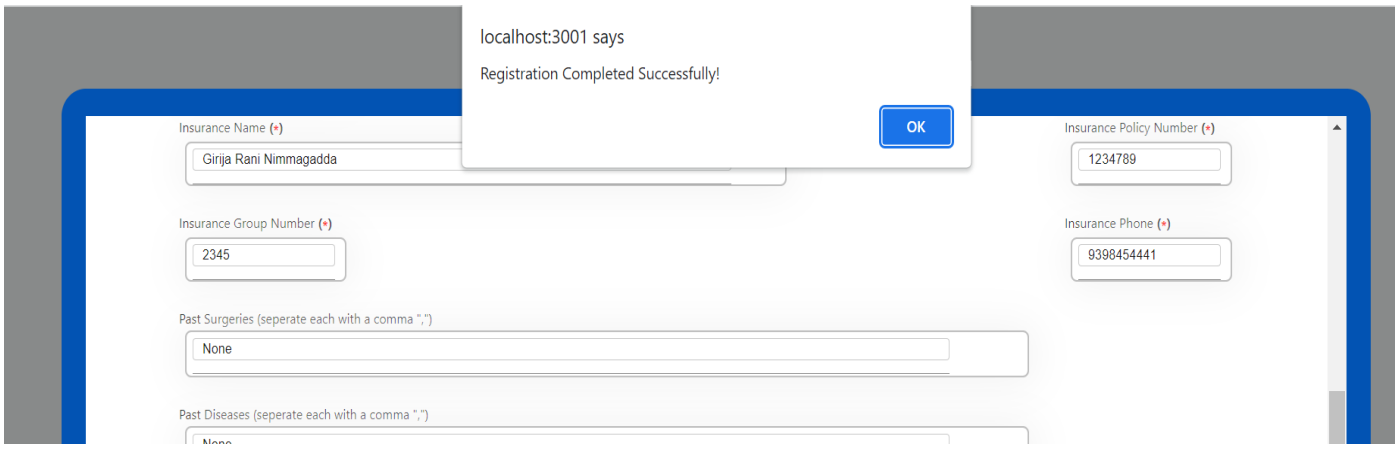


- **New Patient Registration View**

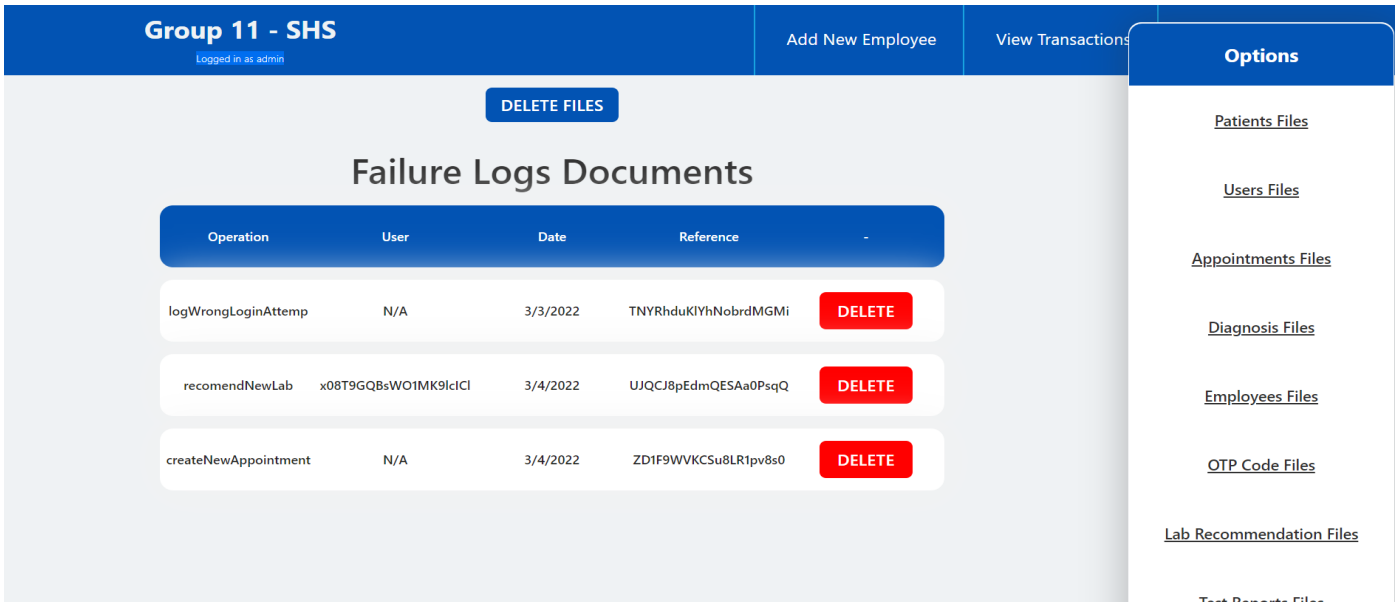


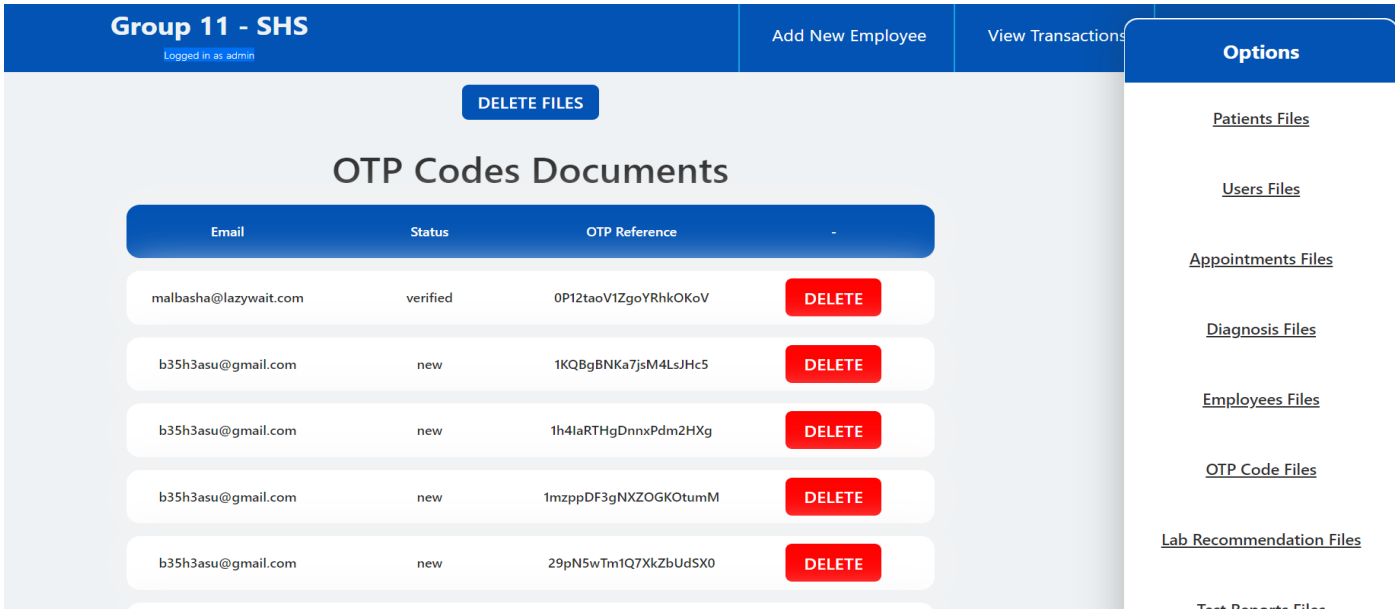

Height (*)
5'7
Weight (*)
60
Blood Type (*)
B-
Emergency Contact Name (*)
Nithyusha
Emergency Contact Phone (*)
9963111248
Contact Relationship (*)
Friend
Insurance Name (*)
Girija Rani Nimmagadda
Insurance Policy Number (*)
1234789
Insurance Group Number (*)
2345
Insurance Phone (*)
9398454441
Past Surgeries (seperate each with a comma ",")
None
Finish Registration


Insurance Name (*)
Girija Rani Nimmagadda
Insurance Policy Number (*)
1234789
Insurance Group Number (*)
2345
Insurance Phone (*)
9398454441
Past Surgeries (seperate each with a comma ",")
None
Past Diseases (seperate each with a comma ",")
None
Past Alergies (seperate each with a comma ",")
None
Finish Registration

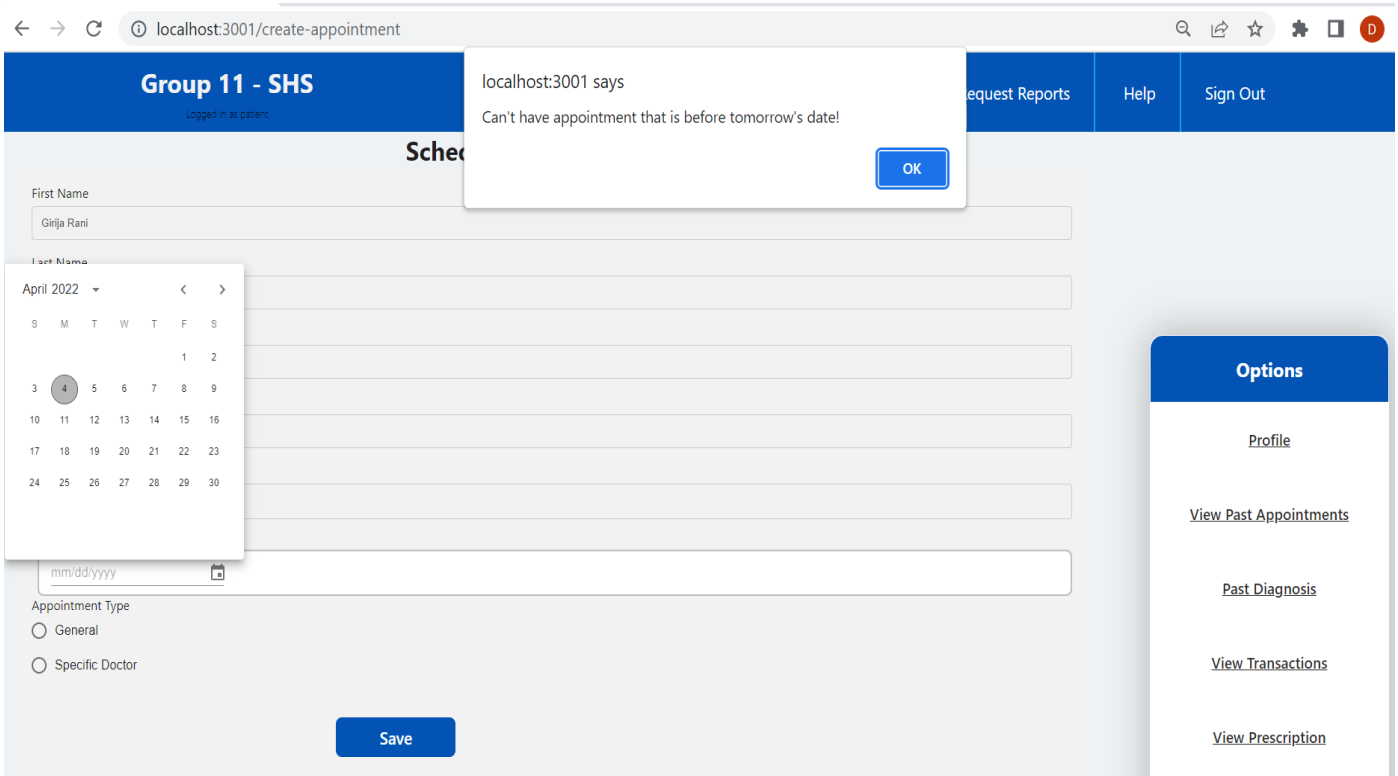

localhost:3001 says
Registration Completed Successfully!
OK
Insurance Name (*)
Girija Rani Nimmagadda
Insurance Policy Number (*)
1234789
Insurance Group Number (*)
2345
Insurance Phone (*)
9398454441
Past Surgeries (seperate each with a comma ",")
None
Past Diseases (seperate each with a comma ",")

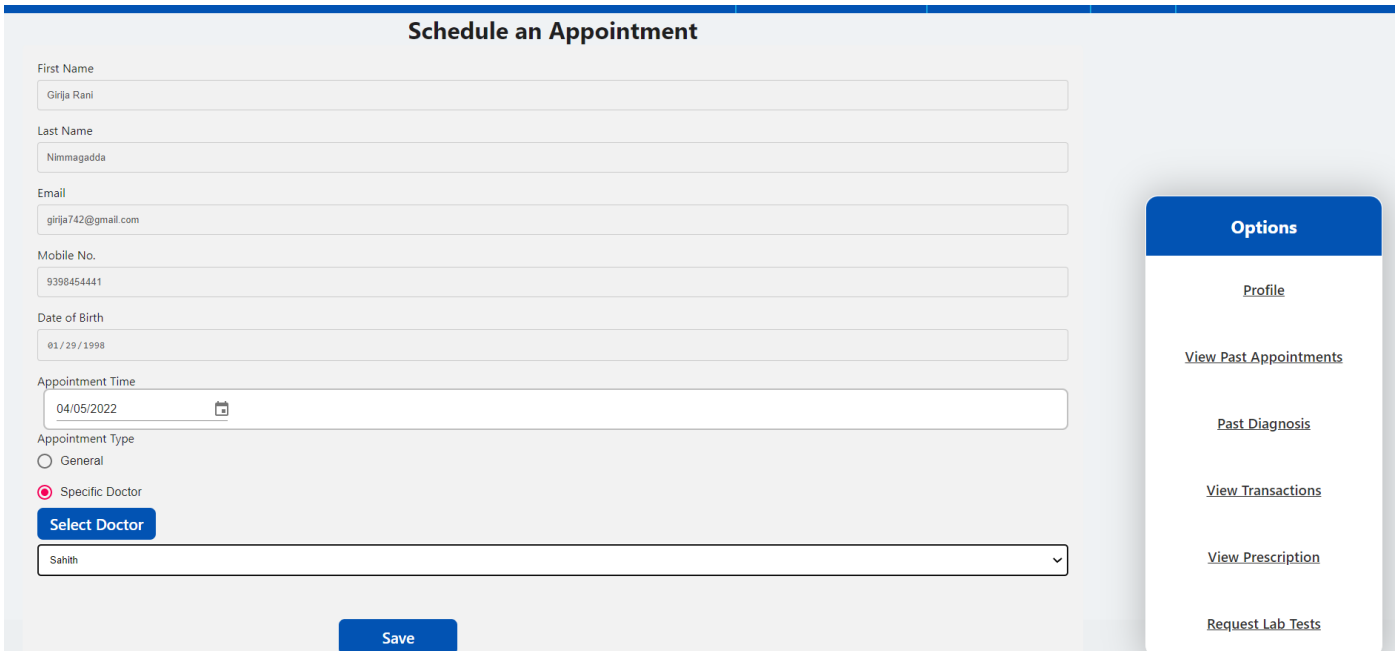
- **Admin control over internal files**

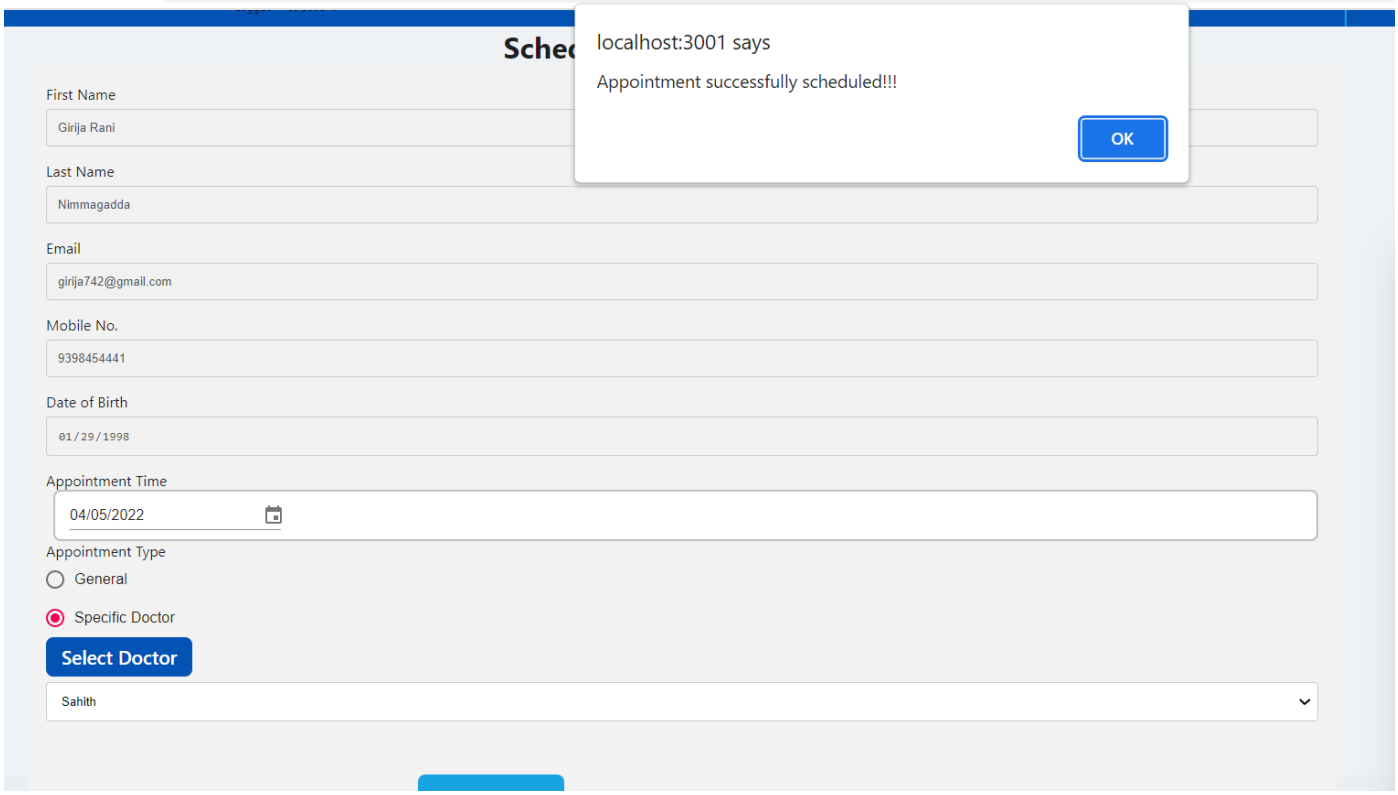




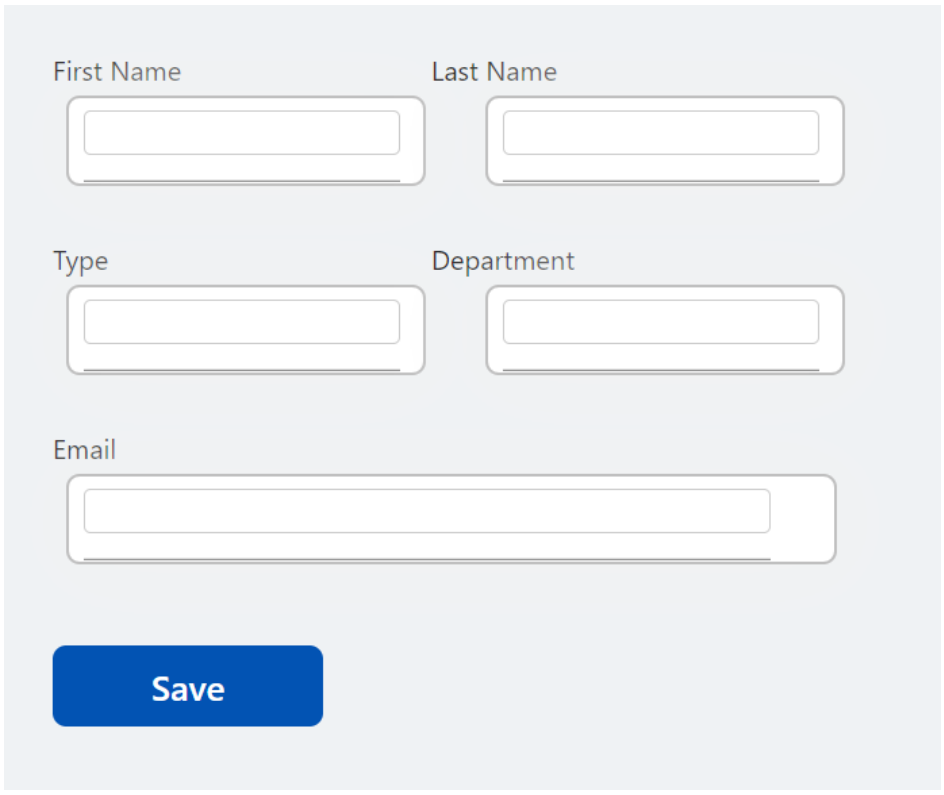
- **Create appointment**

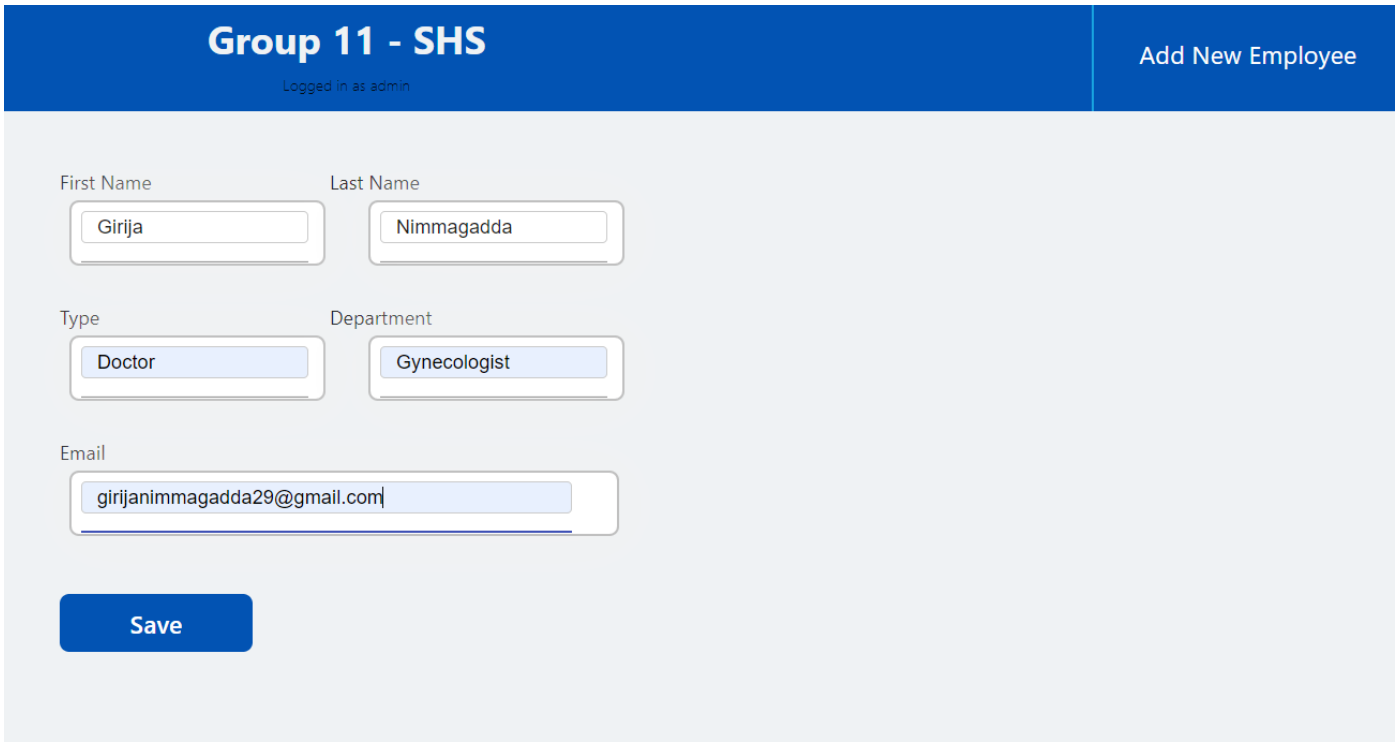


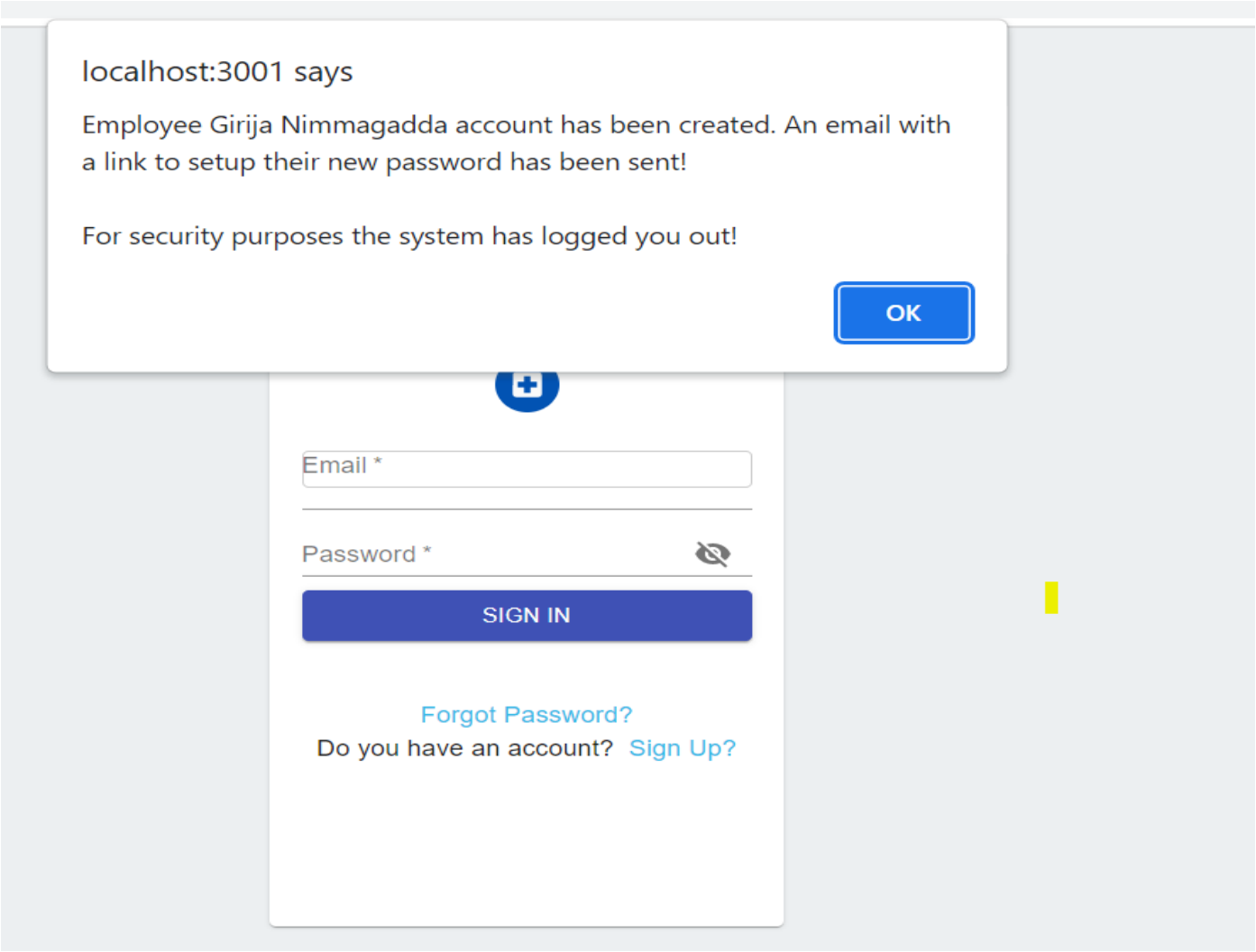

Schedule an Appointment
First Name
Girija Rani
Last Name
Nimmagadda
Email
girija742@gmail.com
Mobile No.
9398454441
Date of Birth
01/29/1998
Appointment Time
04/05/2022
Appointment Type
General
Specific Doctor
Select Doctor
Sahith
Save
Options
Profile
View Past Appointments
Past Diagnosis
View Transactions
View Prescription
Request Lab Tests


localhost:3001 says
Appointment successfully scheduled!!!
OK
First Name
Girija Rani
Last Name
Nimmagadda
Email
girija742@gmail.com
Mobile No.
9398454441
Date of Birth
01/29/1998
Appointment Time
04/05/2022
Appointment Type
General
Specific Doctor
Select Doctor
Sahith

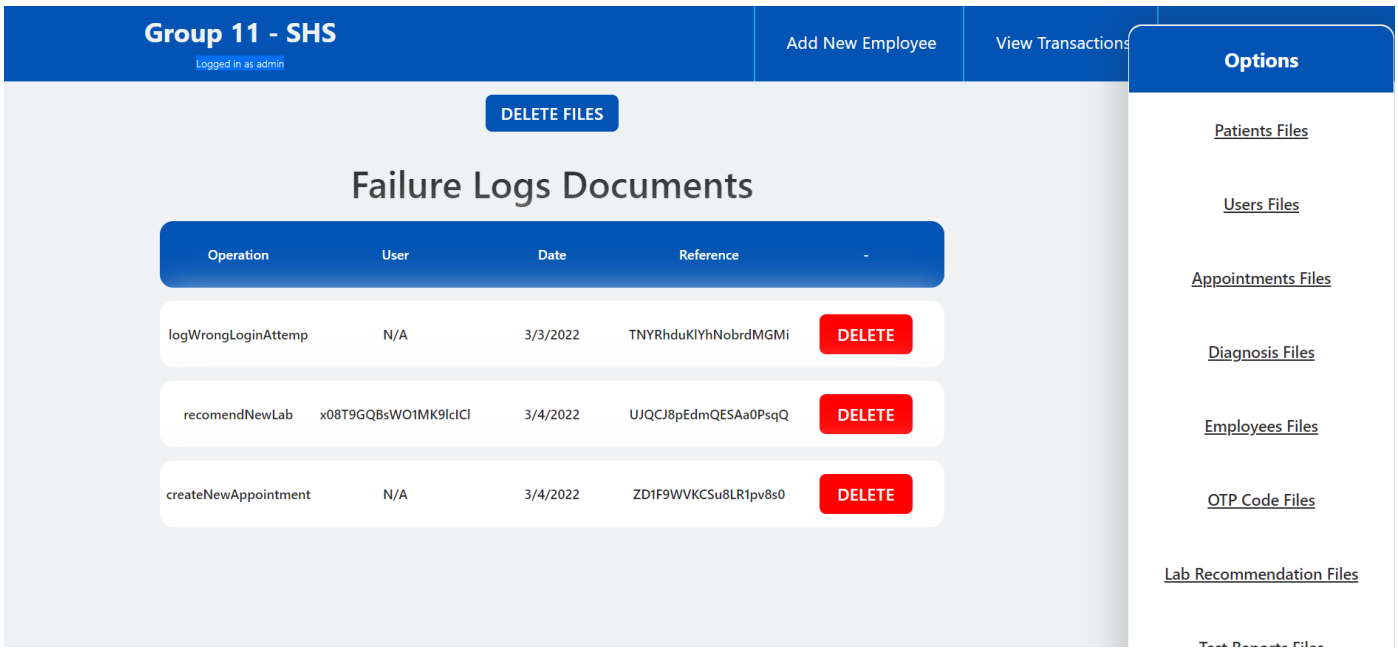
- **Create Employee**

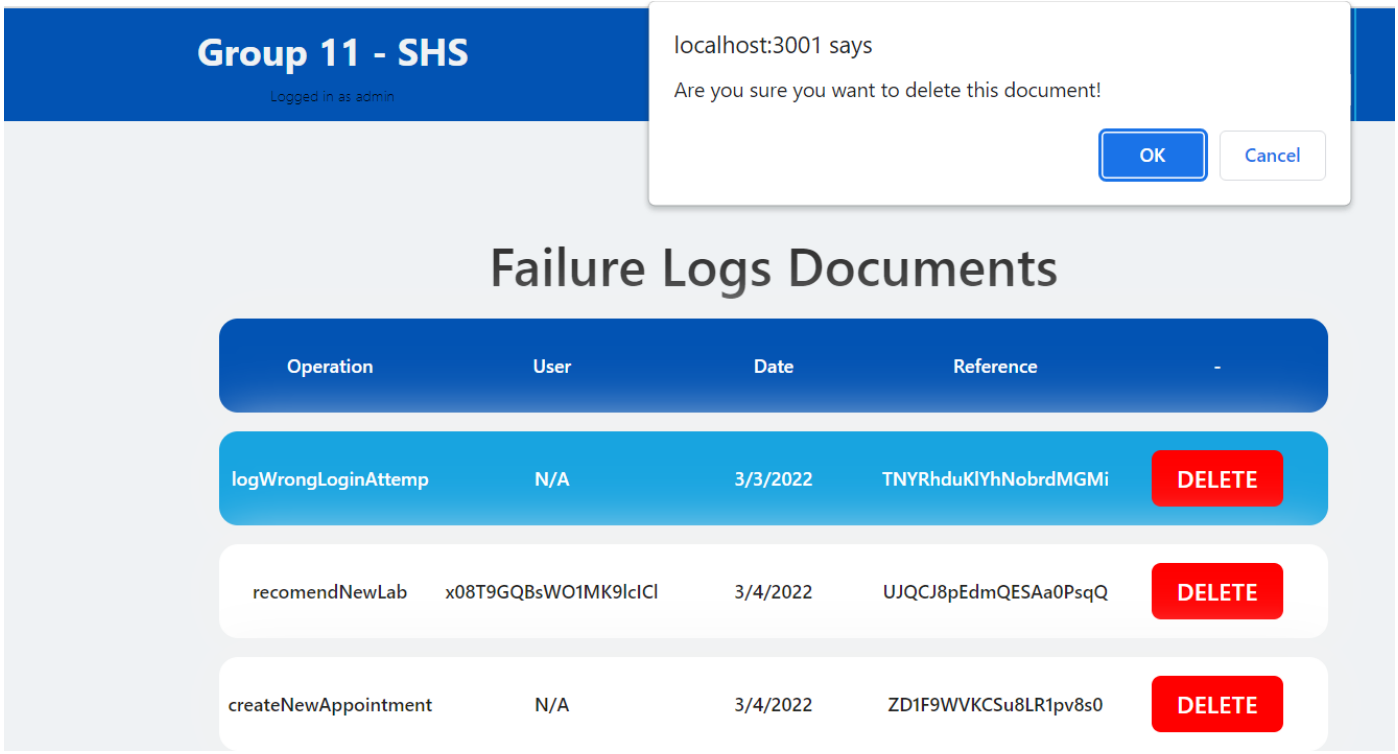


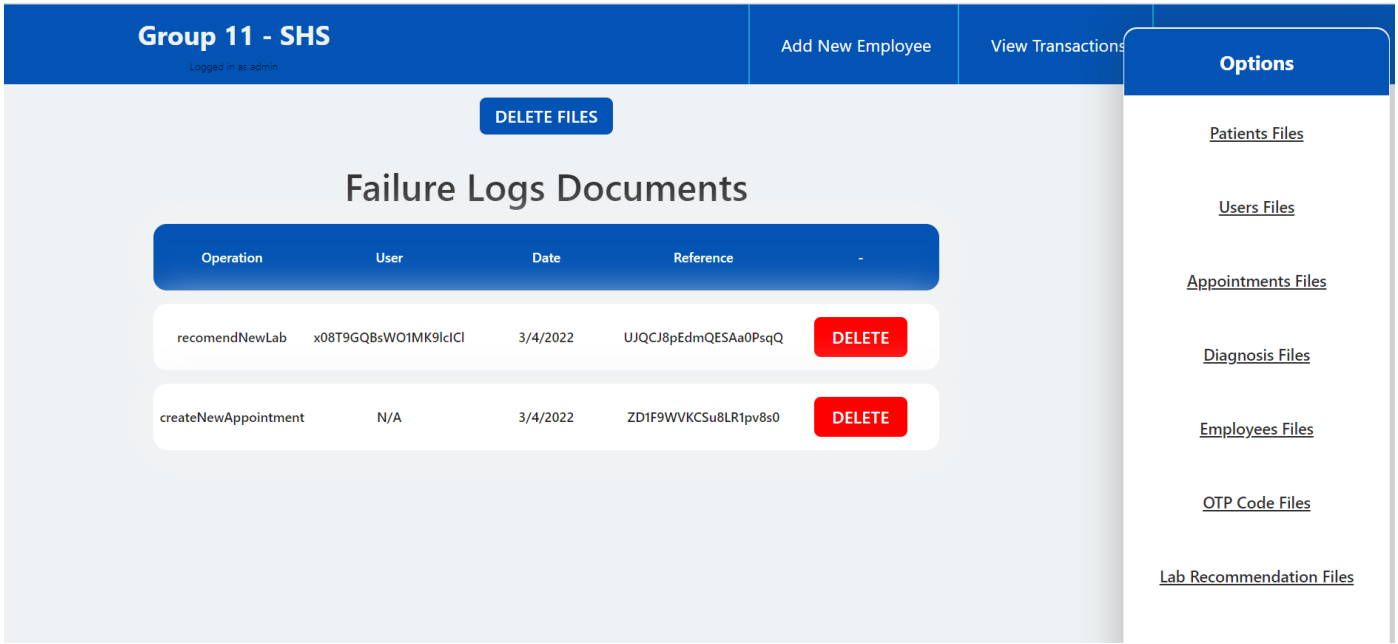




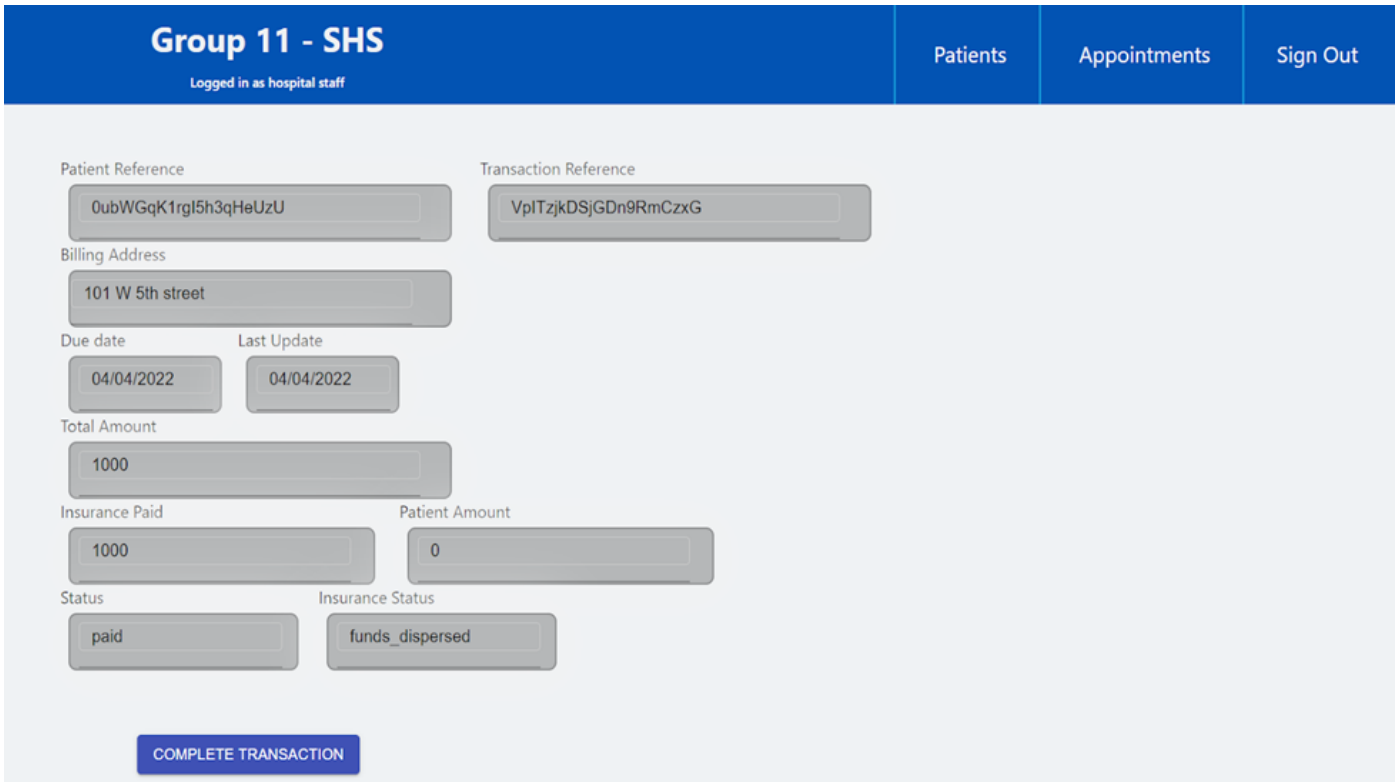
- **Delete Internal Files**

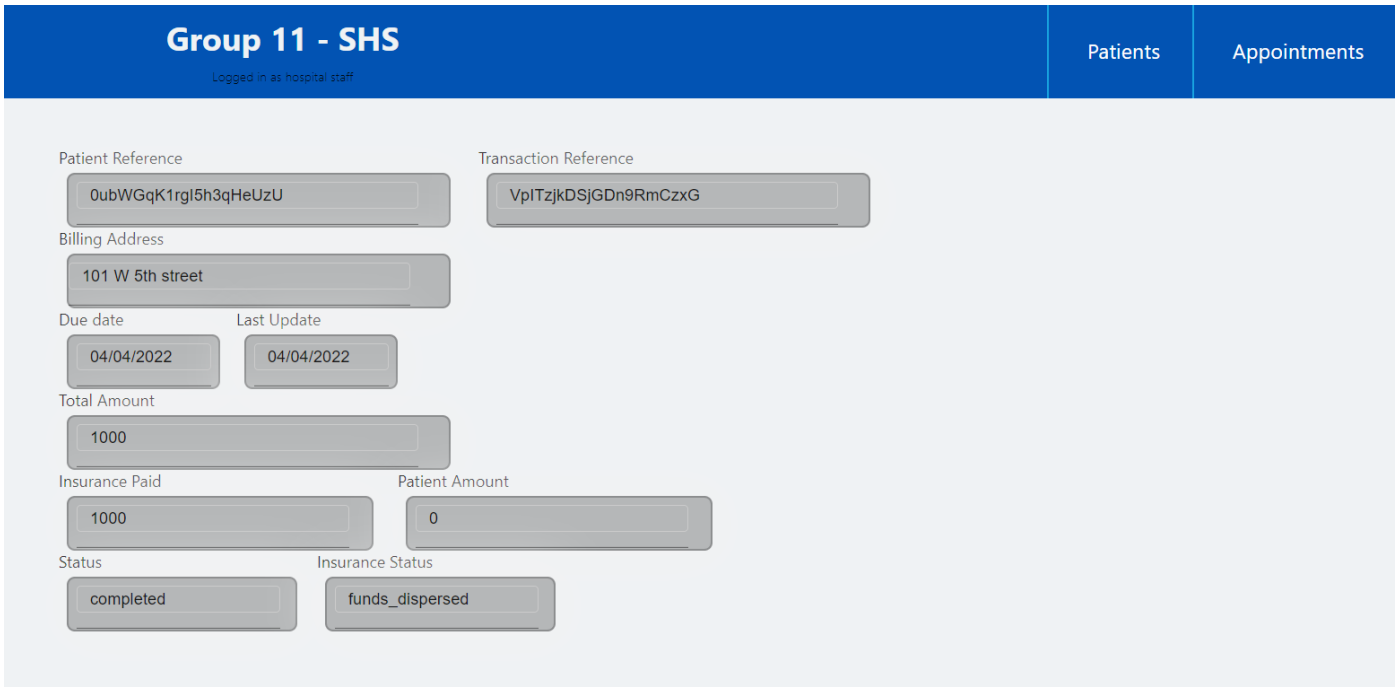



Group 11 - SHS
Logged in as admin
localhost:3001 says
Are you sure you want to delete this document!
OK
Cancel
Failure Logs Documents
Operation
User
Date
Reference
-
logWrongLoginAttemp
N/A
3/3/2022
TNYRhduKlYhNobrdMGMi
DELETE
recomendNewLab
x08T9GQBsWO1MK9lcICl
3/4/2022
UJQCJ8pEdmQESAa0PsqQ
DELETE
createNewAppointment
N/A
3/4/2022
ZD1F9WVKCSu8LR1pv8s0
DELETE

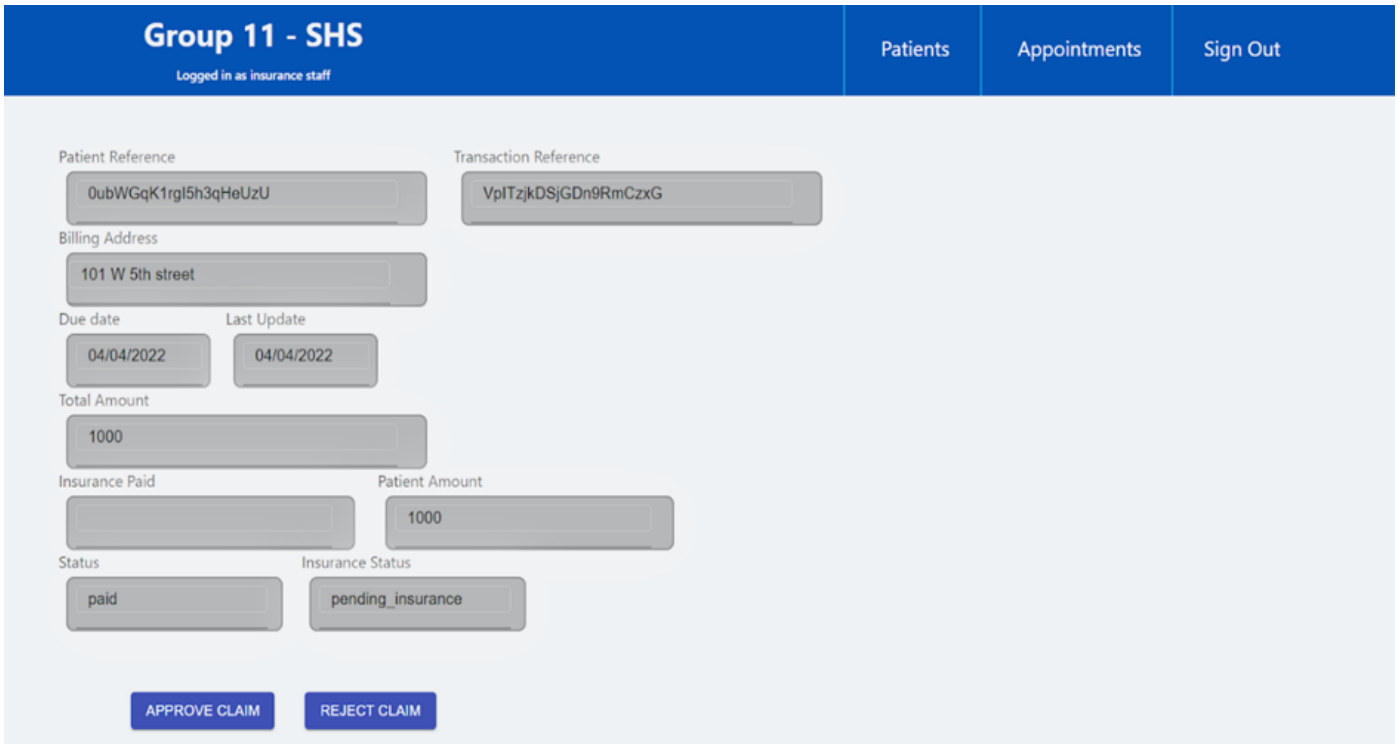

Group 11 - SHS
Logged in as admin
Add New Employee
View Transactions
Options
DELETE FILES
Failure Logs Documents
Operation
User
Date
Reference
-
recomendNewLab
x08T9GQBsWO1MK9lcICl
3/4/2022
UJQCJ8pEdmQESAa0PsqQ
DELETE
createNewAppointment
N/A
3/4/2022
ZD1F9WVKCSu8LR1pv8s0
DELETE
Patients Files
Users Files
Appointments Files
Diagnosis Files
Employees Files
OTP Code Files
Lab Recommendation Files

- **Complete Transaction**

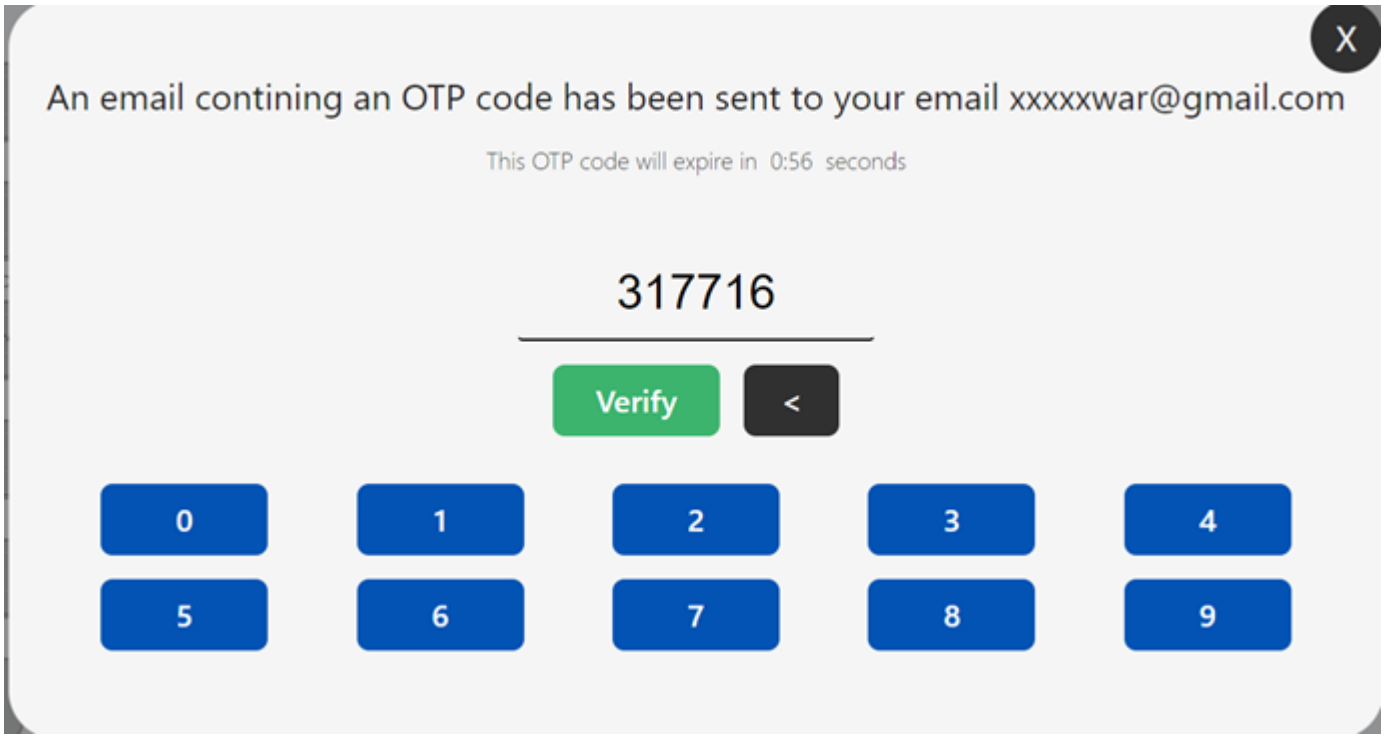




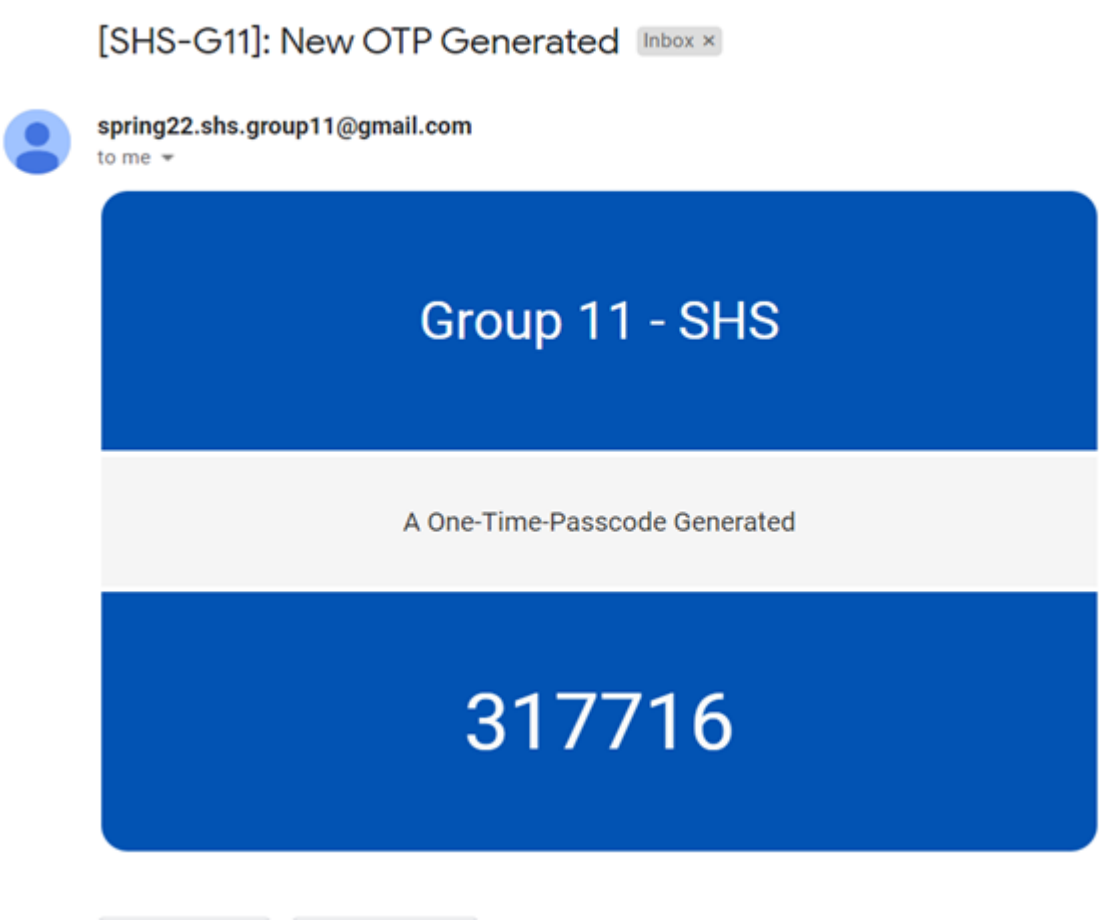
- **Approve Claim**



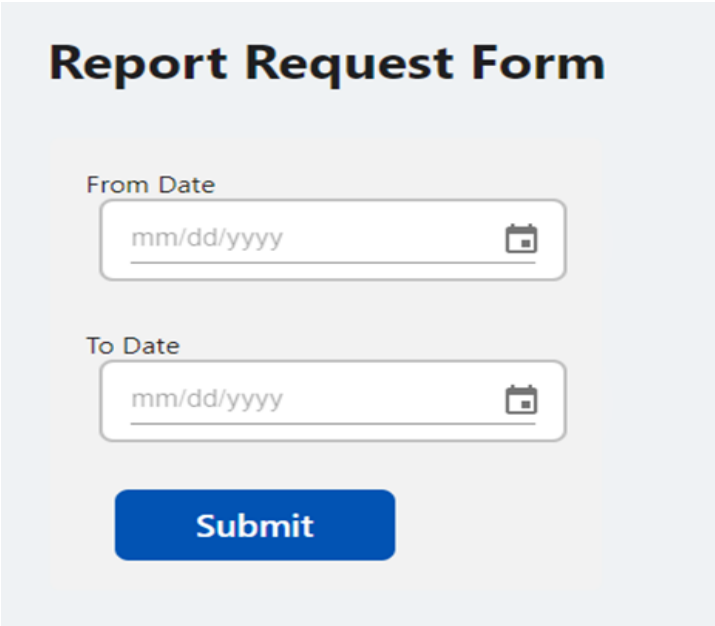
- **OTP View**



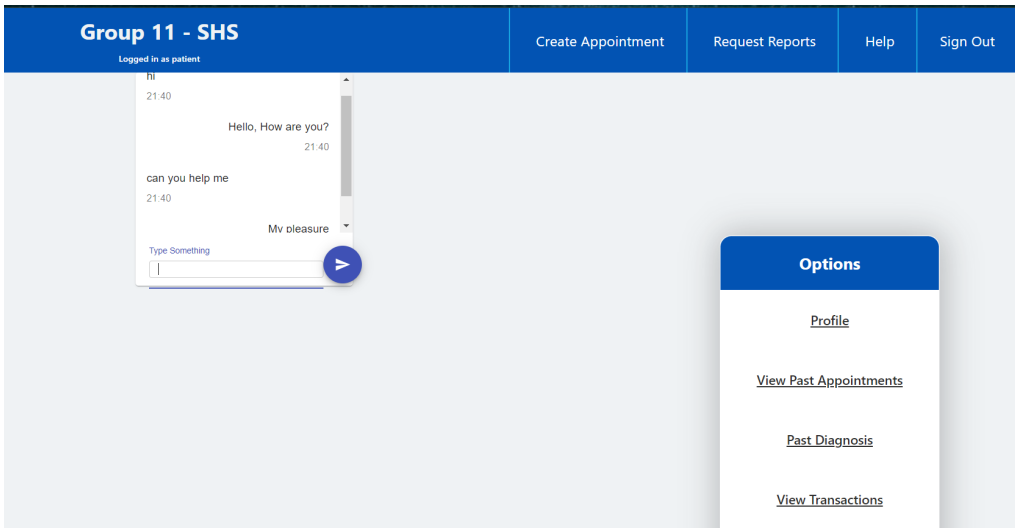
- **OTP Email**



- **Lab Report sample PDF in downloadable format**



- **ChatBot**


Group 11 - SHS
Logged in as patient
Create Appointment
Request Reports
Help
Sign Out
hi
21:40
Hello, How are you?
21:40
can you help me
21:40
Type Something
Options
Profile
View Past Appointments
Past Diagnosis
View Transactions

5. OVERVIEW OF SECURITY FEATURES

When electronically entrusting your health records to an establishment, it should be important for that establishment in making sure that the privacy of the patients and their security of information is guarded with an imperative barrier. These services must protect the confidentiality of hospital records, as well as provide integrity and information availability for anyone granted access to these records. Otherwise, creating an easily penetrable service can cause great damage to not the only reputation of the company, but also the trust of the patients that have entrusted their sensitive information to you. That is why the main focus should be on implementing security barriers in a service that are not only difficult to hack but also operationally efficient for creators to maintain. So some factors need to be considered are:

- Identify any known and potential threats
- Work and retrieve advice to further build a secure system
- Figure out security solutions
- Set up ongoing support and service
- Etc.

5.1. Role-Based Access Security

To login into the application, users must first sign-in, in or create an account to access the hospital server. Logging in with the password of the system is fairly simple and secure, as that enables a host-based verification to log in. Firebase will be the main backend source of offering a real-time database for storing information from the server. In this application, the user will only have access to pages that are permitted as per the user role. For instance, the patient user can not access the admin dashboard page. We also implemented a malicious login control management. If the user has exceeded 5 failed login attempts, thes user account will be freezed for 1 day.

5.2. Added Features

OTP Verification

A one-time password or (OTP) was implemented to authenticate their access in a transaction or session logged in. If the user does not use the OTP within two minutes of receiving the email, it will expire and deny you access. How it is done is in the back end, the date and time the OTP was generated are logged. The moment they tried to verify the OTP, it will compare and find if they took more than two minutes from when the OTP was logged.

Blockchain Transactions

Since blockchain allows us to organize and store processed data in a present blockchain Network, this will be used to store all approved transactions within the hospital service including records like reports, Insurance claim requests, and other approved transactions securely. To protect the chain, safeguarding the data in the API is the best thing to do to protect the transactions.

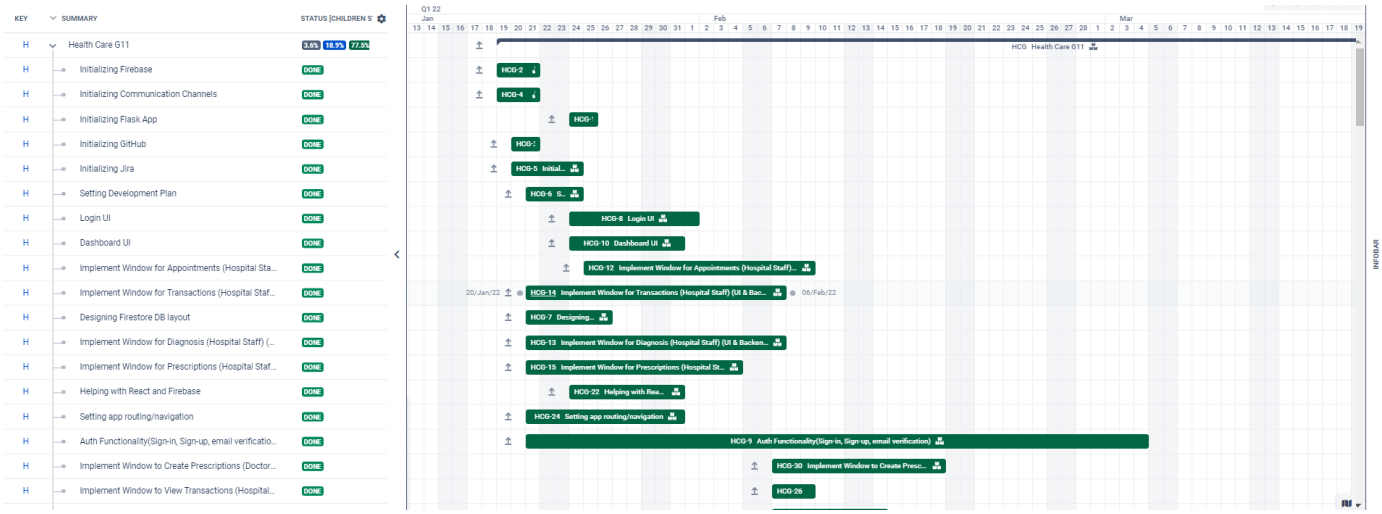
Session Management

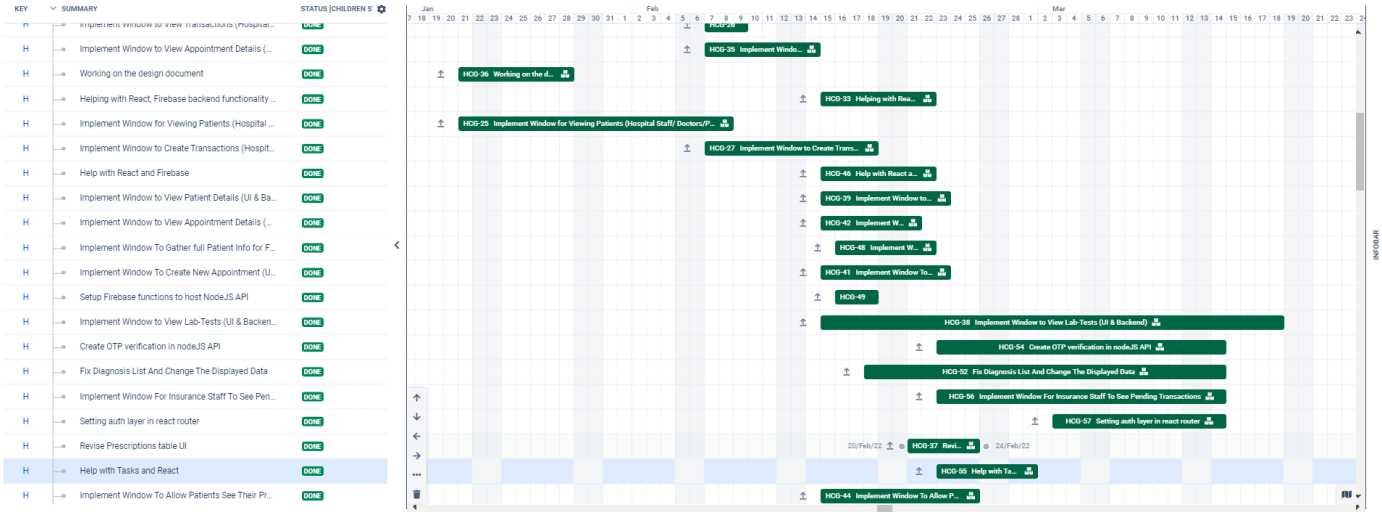
In health care systems, users can access multiple applications simultaneously with the system. However, users have to log in to each app individually, and it is difficult to maintain the same context between these apps. We are building a session management system for web-based applications that use a directory application, which will allow you to sign in to multiple web-based applications, and maintain the same context between those user applications.

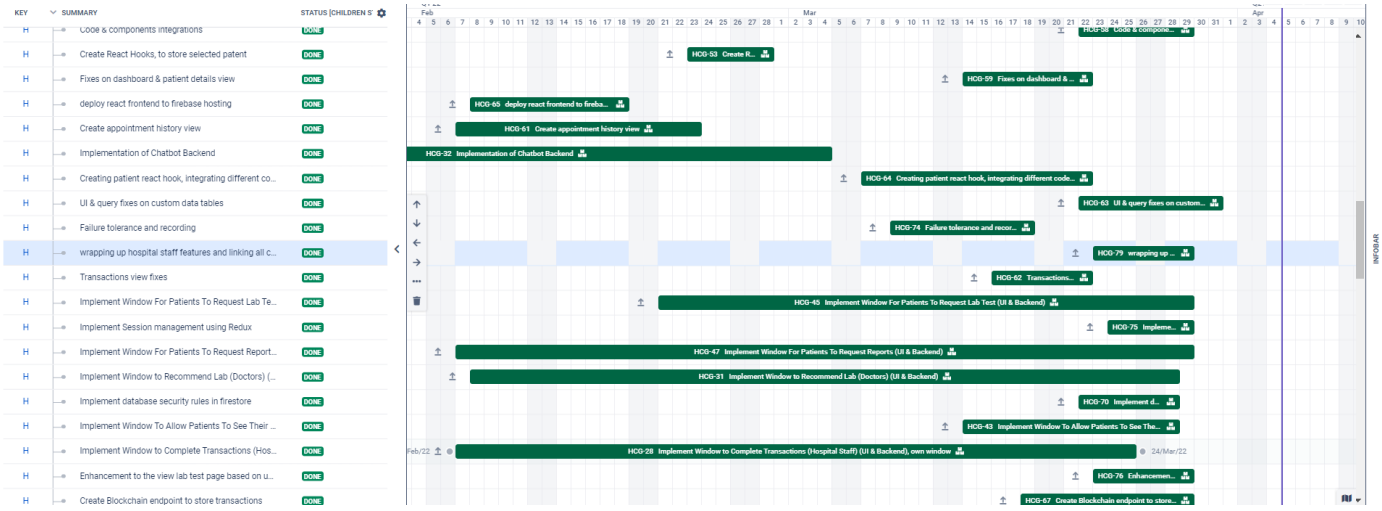
Build and Deploy Without Source Code Map

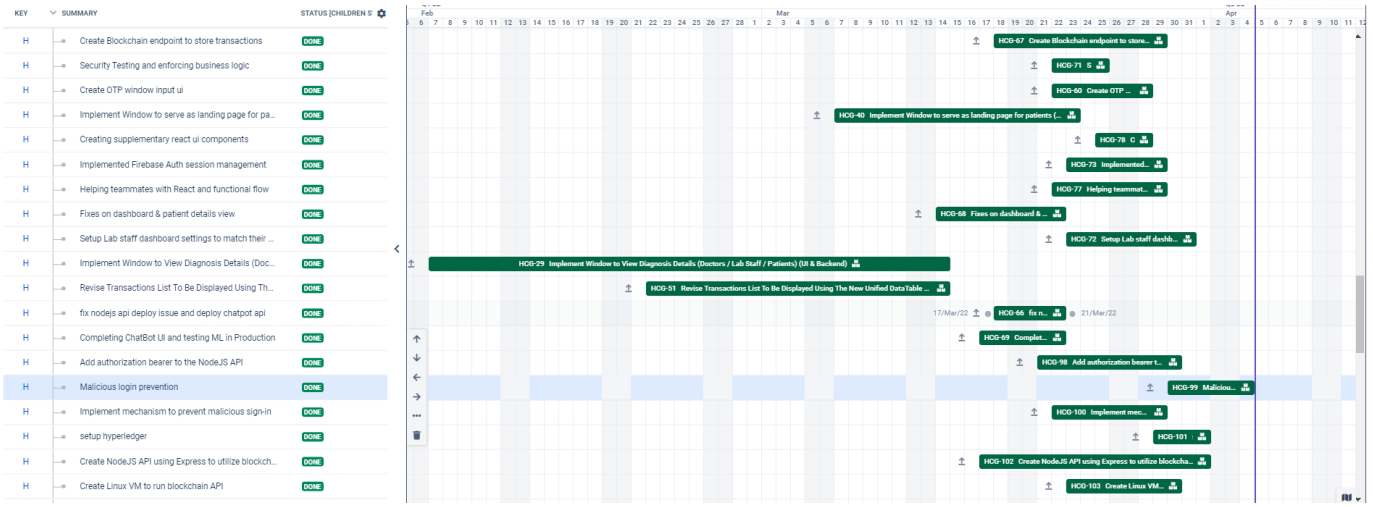
While debugging, source codes are used to display your original JavaScript, which is far easier to read than minified production code. Source maps are, in a way, the decoder ring for your hidden (minified) code. During development, source maps might be useful. They make it easier to troubleshoot programs because you can still look at the actual code rather than a generated one. They're useful even in production since they let you debug problems while presenting a client-friendly version of your program.

6. FINAL GANTT CHART

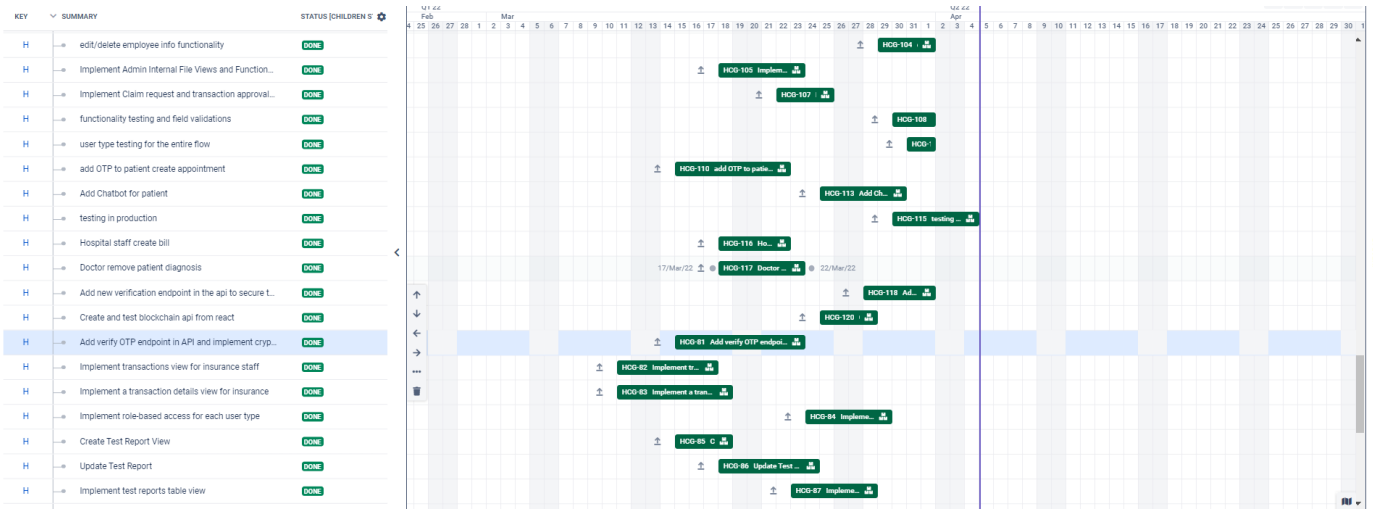




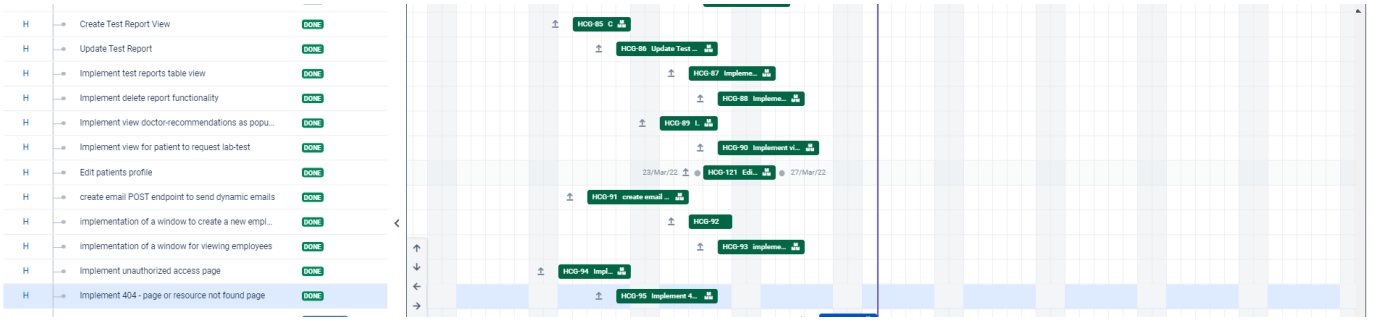




KEY	SUMMARY	STATUS [CHILDREN S
H	Create Blockchain endpoint to store transactions	DONE
H	Security Testing and enforcing business logic	DONE
H	Create OTP window input ui	DONE
H	Implement Window to serve as landing page for pa...	DONE
H	Creating supplementary react ui components	DONE
H	Implemented Firebase Auth session management	DONE
H	Helping teammates with React and functional flow	DONE
H	Fixes on dashboard & patient details view	DONE
H	Setup Lab staff dashboard settings to match their ...	DONE
H	Implement Window to View Diagnosis Details (Doc...	DONE
H	Revise Transactions List To Be Displayed Using Th...	DONE
H	fix nodejs api deploy issue and deploy chatpot api	DONE
H	Completing ChatBot UI and testing ML in Production	DONE
H	Add authorization bearer to the NodeJS API	DONE
H	Malicious login prevention	DONE
H	Implement mechanism to prevent malicious sign-in	DONE
H	setup hyperledger	DONE
H	Create NodeJS API using Express to utilize blockch...	DONE
H	Create Linux VM to run blockchain API	DONE



KEY	SUMMARY	STATUS [CHILDREN S
H	edit/delete employee info functionality	DONE
H	Implement Admin Internal File Views and Function...	DONE
H	Implement Claim request and transaction approval...	DONE
H	functionality testing and field validations	DONE
H	user type testing for the entire flow	DONE
H	add OTP to patient create appointment	DONE
H	Add Chatbot for patient	DONE
H	testing in production	DONE
H	Hospital staff create bill	DONE
H	Doctor remove patient diagnosis	DONE
H	Add new verification endpoint in the api to secure t...	DONE
H	Create and test blockchain api from react	DONE
H	Add verify OTP endpoint in API and implement cryp...	DONE
H	Implement transactions view for insurance staff	DONE
H	Implement a transaction details view for insurance	DONE
H	Implement role-based access for each user type	DONE
H	Create Test Report View	DONE
H	Update Test Report	DONE
H	Implement test reports table view	DONE



H	Create Test Report View	DONE
H	Update Test Report	DONE
H	Implement test reports table view	DONE
H	Implement delete report functionality	DONE
H	Implement view doctor-recommendations as popu...	DONE
H	Implement view for patient to request lab-test	DONE
H	Edit patients profile	DONE
H	create email POST endpoint to send dynamic emails	DONE
H	implementation of a window to create a new empl...	DONE
H	implementation of a window for viewing employees	DONE
H	Implement unauthorized access page	DONE
H	Implement 404 - page or resource not found page	DONE

7. BONUS FEATURES

7.1. Failure detection and logging

Helps to detect any failures within the application to detect any families or crashes of the nodes. Using this will give our system a backup to do much better.

7.2. Robot Script Calls Prevention When Creating New Employee

In admin, when they create a new employee within the server, the service will automatically sign them out. Doing this will prevent robot script calls from creating a new employee within the program.

7.3. Transaction Notification

Whenever a transaction is being made to or from the patient within the server, a notification will be sent to them about the action. Specifically, an email will be sent to the patient in question, along with information about the transaction, such as the date when it happened.

7.4. Hashed OTP

As stated before, another additional feature is the OTP being hashed and treated the same as the password within the database. For further elaboration, OTP is being generated and sent to the user no one can know what it is. Even as system administrators, there is no way to know what that OTP is as it is going to be hashed inside the database. The contents of the OTPs are going to be like a hash function, but when looking into it, there is no way to know how the numbers will be. So only the end-user will know what the OTP is, which is an extra layer of security.

7.5. Authorization

An authorization bearer will be invoked on calls to the NodeJS API to increase security by authenticating and authorizing the request. Using a JWT (JSON Web Token), it will securely transfer between two JSON objects. Each user will be assigned their own JWT and used within the server to allow access to different things within the server. Using this will provide us with high-end security, to ensure that our API will not be vulnerable to non-authenticated / authorized users.

7.6. Email Notification

The email notification feature is added for the patient component. The email response will be sent to the patient when the patient schedules a new appointment or any transactions are updated.

7.7. Report Download In PDF

The patient reports can be generated and downloaded into pdf format for their future purpose.

7.8. Logging User Operations And Interactions

Whenever the user logins, it records time and stores all the metadata as a document at the backend in the database with all the information of the user.

8. REFERENCES

1. Complications/Limitations that may occur using a firebase DB
2. Understanding JSX w/ React library
3. More in-depth about limitations using firebase/firestore
4. Storing data in Firebase
5. React JS Tutorial: Building Firebase Chat App (React Hooks) (djamware.com)
6. FirebaseUI React Components (firebaseopensource.com)