CSE 545: Software Security

Secure Hospital System

Design Document – Group 9

Yash Bhokare (L)	Kiran Shanthappa (DL)	Anthony Mac
Sai Manoja Gadde	Shashank Baradi	Karan Naik
Jayanth Goritha	Bhaskara Atluri	Rajesh Badugula

1 Introduction

With advances in healthcare informatics, one can provide better means to process patient records and therefore speed up the treatment, which in turn reduces the overall cost. Many tools exist for facilitating patient record processing: from assisting data entry to manipulating records, from generating output in required form to transferring it to other physicians for further examination, or to save it digitally for future use. The project adopts waterfall implementation methodology. This design document incorporates a software engineering approach during development, and incorporates many aspects from database design, web deployment, and security. While developing our design, we follow formal patient privacy requirements for user account management and secure transactions. Also adopts ML based chat bot and malicious login prevention, to provide a secure hospital management system.

Electronic medical records (EMRs) contain an individual patient's extremely sensitive confidential information including identifying information and personal medical records. However, these EMRs are created, maintained, and stored across isolated hospital and insurance providers creating "data silos" that cause major patient linkage and data sharing problems in healthcare. This work extends Hyperledger Fabric as a solution to overcome the problems of traditional records management in hospital/medical system such as accessibility, security, and transaction/approval records.

1.1 Scope

The aim of the course project is to develop a skeleton secure hospital system (SHS) with limited functional, performance, and security requirements for secure hospital management, user account management and secure transactions. Multiple users should be able to securely use this system from any place and at any time with the availability of Internet access and web browser.

Hosting the secure hospital system on AWS with minimal resource to highlight the functionality and security design learnt throughout the course. The free tier AWS i.e., micro, or small EC2 instance will be used for hosting the application. This system will not be capable of diverting DDoS attacks.

2 Overview

SHS is a greenfield development requiring requirement analysis with design and development in a new clean environment. Various security measures such as communication over secure channel, data encryption, session management, secure login with ML based malicious login prevention and, blockchain are considered at software design to reduce vulnerabilities in the system.

2.1 Member Responsibilities

Team Member	**Project Responsibilities**
Yash Bhokare (Leader)	1. Responsible for managing the team and working together. 2. Responsible to oversee the front-end and integration of it with the back end. Also responsible for including a chatbot. 3. Implemented the UI design for doctor, patient, hospital, lab, and admin. 4. Created the angular application that included data binding with the backend team using the Rest API calls. Also hosted the final application on Amazon S3 bucket. 5. Researched and implemented the malicious login and security features for the front-end. 6. Implemented the chatbot using DialogFlow and integrated it inside our application. Also hosted it using an ec-2 instance with SSL enabled. 7. Team meeting organisation and providing co-ordination among team members. Monitoring team progress as per the schedule. 8. Weekly report and project report review. 9. Supported for project report and user guide preparation. 10. Project demo preparation.
Kiran Shanthappa (Deputy Leader)	1. Design of the blockchain network for storage of approved transactions. 2. Setup of Hyperledger Fabric on AWS EC2 instance. 3. Implementation of chaincode in GoLang and java API for interacting with the peer. 4. AWS Java SDK for interacting with SQS.

	5. Testing of the blockchain functionality and lab staff. 6. Preparation of template for design document, user guide and project report. Review of all the documents and formatting the structure/organisation of content. 7. Testing of the hospital staff and lab staff access control and functionality. 8. Testing of the blockchain transaction, limitations, data validity. 9. As deputy leader supporting leader in the weekly meeting organisation, team progress monitoring and weekly report preparation. 10. Supporting team members with application functionality design.
Anthony Mac	1. Primarily was responsible for configuring and deploying the database in PostgreSQL. 2. Secondary responsibility of supporting database integration with the backend by providing necessary queries. 3. Research included research into PostgreSQL stuff like live modification without dropping and recreating objects, automation of generating sample data, configuration of remote listening, and hashing/encryption at rest with pgcrypto. 4. All user roles access control testing at database level. 5. Supported for project report and user guide preparation – database and transaction history management.
Sai Manoja Gadde	1. Primarily was responsible for configuring and deploying the database in PostgreSQL. 2. Secondary responsibility of supporting database integration with the backend by providing necessary queries. 3. Research included research into PostgreSQL stuff like live modification without dropping and recreating objects, automation of generating sample data, configuration of remote listening, and hashing/encryption at rest with pgcrypto. 4. All user roles access control testing at database level. 5. Supported for project report and user guide preparation – database and transaction history management.

Shashank Baradi	1. Research into potential backend technologies for the application from the requirements and security perspective. 2. Designed the hierarchy for backend system design and implemented the code in “spring-boot” 3. Research and implementation of the JWT authentication for user sign in/signup 4. Responsible for integrating the backend code for interactions with the database. 5. Responsible for writing several APIs that are consumed by the front end for all requirements. 6. Supported for project report and user guide preparation – Application frameworks, tools, and integration.
Karan Naik	1. Primarily responsible for UI development of the patient, doctor, Insurance Staff as well as Hospital Staff in Angular. 2. Secondary responsibility was of integrating the backend and frontend by calling the API from the frontend and showing the data on the frontend as per the design. 3. Implemented Angular factories for using angular services like $http and $resource to make RESTful API calls. 4. Helped in UI-Router for binding data to different states and rendering different templates. 5. Worked in creating Angular modules, controllers, directives, and filters to deploy the entire business logic. 6. Supported for project report and user guide preparation – Hospital Staff, and lab staff access control and functionality.
Jayanth Goritha	1. Support for Hyperledger Fabric implementation on AWS EC2 instance. 2. Implementation of chaincode in GoLang and java API for interacting with the peer. 3. Testing of the blockchain functionality and lab staff. 4. Testing of the blockchain transaction, limitations, data validity. 5. Supported for project report and user guide preparation – Hospital Staff, and lab staff access control and functionality.

Bhaskara Atluri	1. Researched and Designed Insurance Policies and Claim Requests 2. Setting up AWS and launching ec2-instances for various functionalities 3. Collaborated with the team on JWT token-based login authentication. 4. Implemented two-factor authentication using Google authenticator. 5. Researched and implemented ML based malicious login with the team. 6. Worked on Final Report, User Guide and Design Documentations.
Rajesh Badugula	1. Involved in weekly team meeting. 2. Login page design. 3. Support for design document preparation, project report and user guide preparation.

3 Detailed Design

3.1 Software Design

SHS can be divided into a 3-tier architecture, involving web, app, and data tier. All tiers will be hosted on AWS. The hosting the web and app tier components is through spring boot. The data tier has two components for storing the hospital data. Postgre SQL for storing the hospital data and Hyperledger blockchain to capture the approved transaction details.

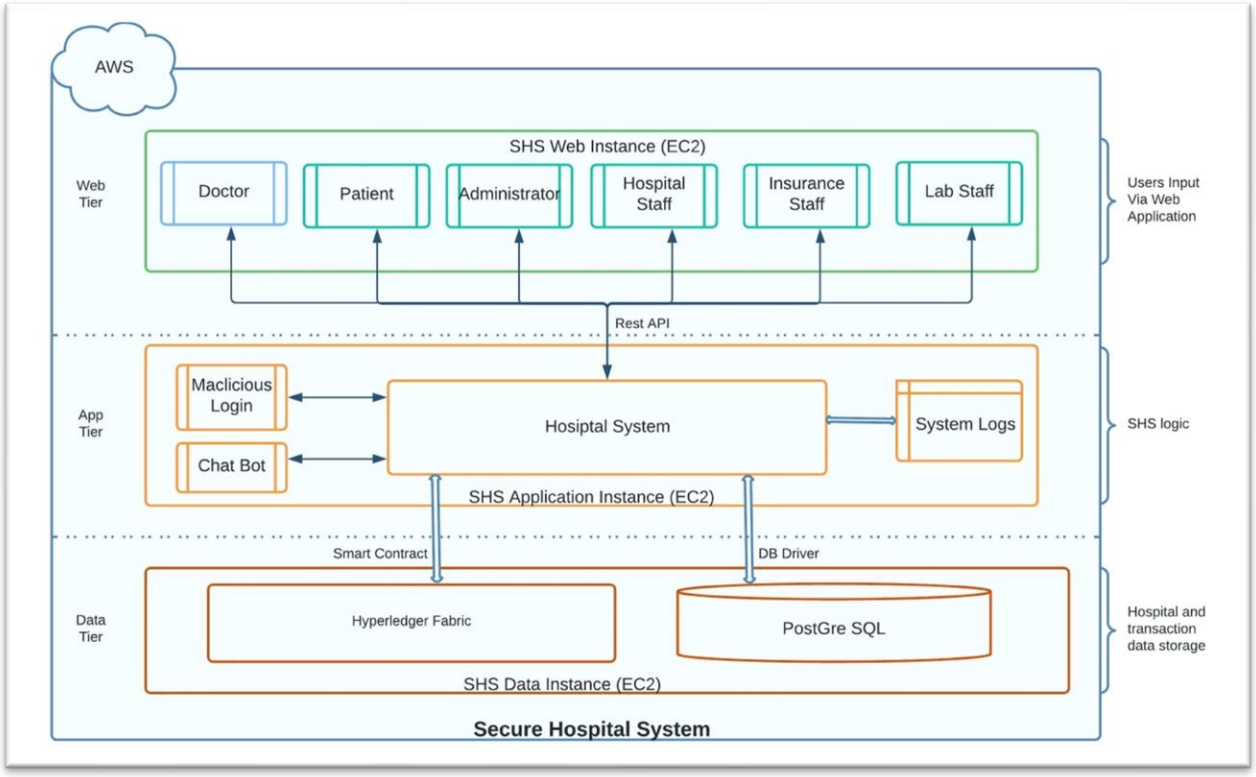


Fig. Secure Hospital System Architecture

The Web application are hosted as website. Web pages are designed using angular, as it is highly preferable as a robust frontend tool supplying components that assist people to write easy-to-use, readable, and maintainable code. Angular has built-in support to help prevent two common HTTP vulnerabilities, cross-site request forgery (CSRF or XSRF) and cross-site script inclusion (XSSI). Both must be mitigated primarily on the server-side, but Angular provides helpers to make integration on the client-side easier.

A security group acts as a virtual firewall for SHS EC2 instances to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. When component launched as an instance, you can specify same security groups. All components access across tier will be through secure protocol such as HTTPS. The secure hospital system will use a certificate for the web application for accessing webpages. A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance. Amazon EC2 stores the public key on your instance, and you store the private key.

Blockchain technology is based on the principle of providing an individual control over their data and information, and hence potentially appropriate as part of a solution to patient ownership of their health data. Blockchain can create a decentralized identification that allows health care providers and patients to interact with one another directly without the need for an intermediary. The immutable audit trail which allows all changes of personal information to be tracked and traced. The increased security of patient's medical information. Not only is the data stored in multiple nodes across the distributed ledger; but it can also be encrypted with the only key being in the hands of the patient. Due to limited functionality the feature of key with patient will be considered as future work. Blockchain applications may, therefore, provide solutions to the current problems of interoperability of medical records, incomplete patient data at the point of service, integrate insurance claim, and lack of access to personal records while ensuring security.

3.1.1 Data Management

The data in SHS can be divided into sensitive and regular. The sensitive data represents the personal data of the patients such as SSN, Insurance number, and phone number or user password. The patient personal data are prevented from direct view by unprivileged users by data masking. The user password is encrypted and stored to prevent any hackers from easily accessing password. All other data are considered as regular data and stored as raw information in the database.

The various transaction and user data are captured through the data in the Database. The relation between tables can be seen in the ER diagram below. Here Postgre SQL will be hosted in AWS EC2 instance. With the access from the application for data processing.

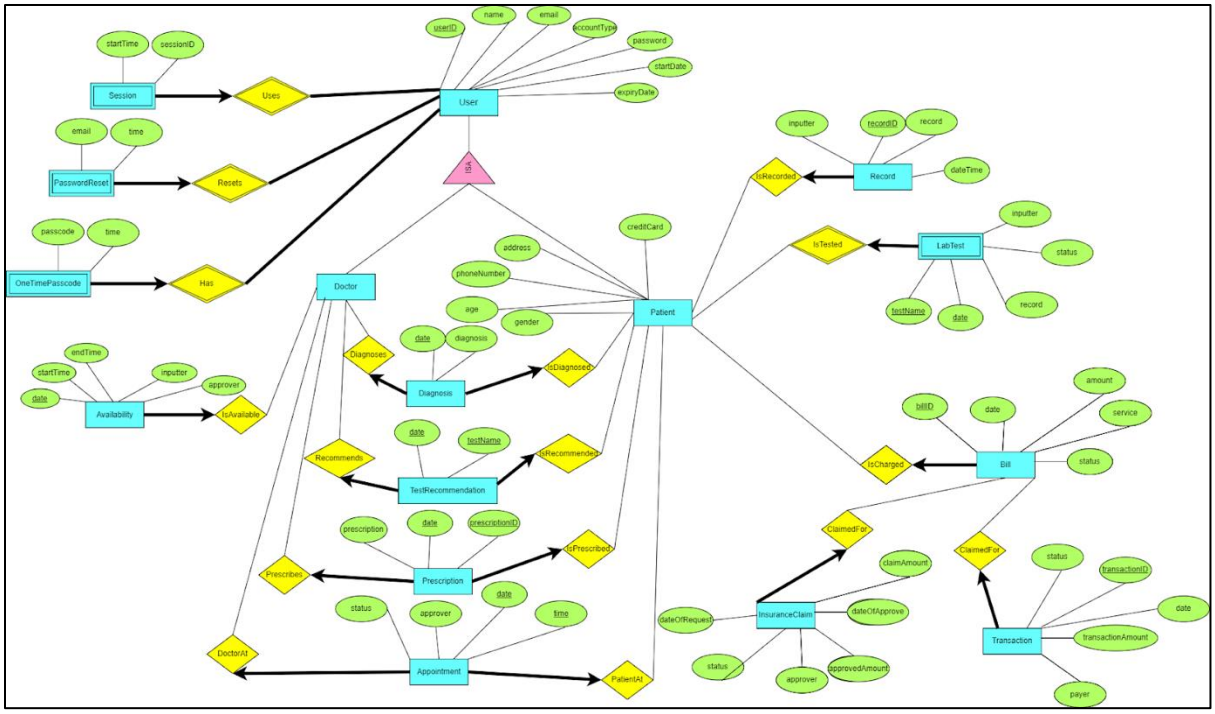


Fig. SHS – ER Diagram

3.1.2 Hyperledger Network

The network consists of infrastructure from organisation R1 and organization R2. The initial request to input request is considered as the A1 and approval request will be considered as A2. P1 and P2 represents the peer of organisation R1 and R2, respectively. CA1 and CA2 represents the certificate of organisation R1 and R2, respectively. L1 and L2 represents the ledger of organisation R1 and R2, respectively. O4 is the ordering service with CA4 as its certificate. All the instances hosted on a micro instance of AWS EC2.

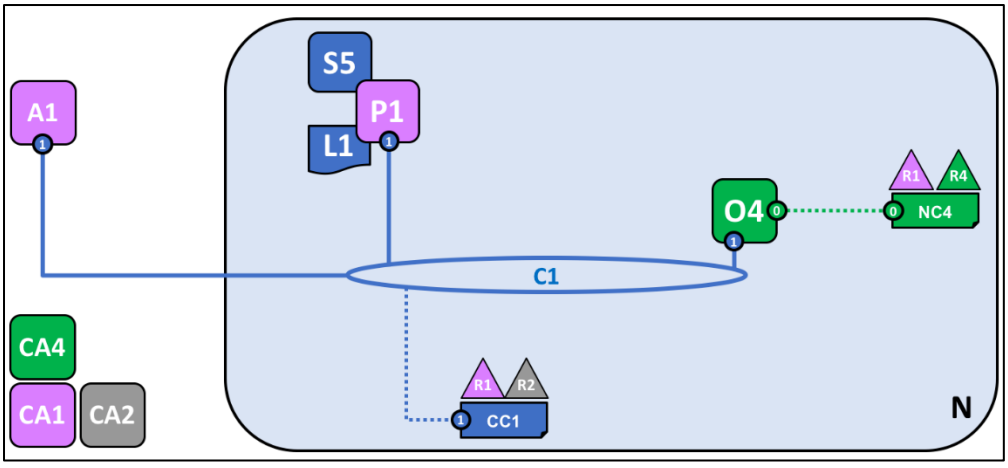


Fig. Blockchain Network for SHS

A smart contract S5 has been installed onto P1. Client application A1 in organization R1 can use S5 to access the ledger via peer node P1. A1, P1 and O4 are all joined to channel C1 i.e., they can all make use of the communication facilities provided by that channel. Specifically, R2 has added peer node P2, which hosts a copy of ledger L1, and chaincode S5. R2 approves the same chaincode definition as R1. P2 has also joined channel C1, as has application A2. A2 and P2 are identified using certificates from CA2. All of this means that both applications A1 and A2 can invoke S5 on C1 either using peer node P1 or P2.

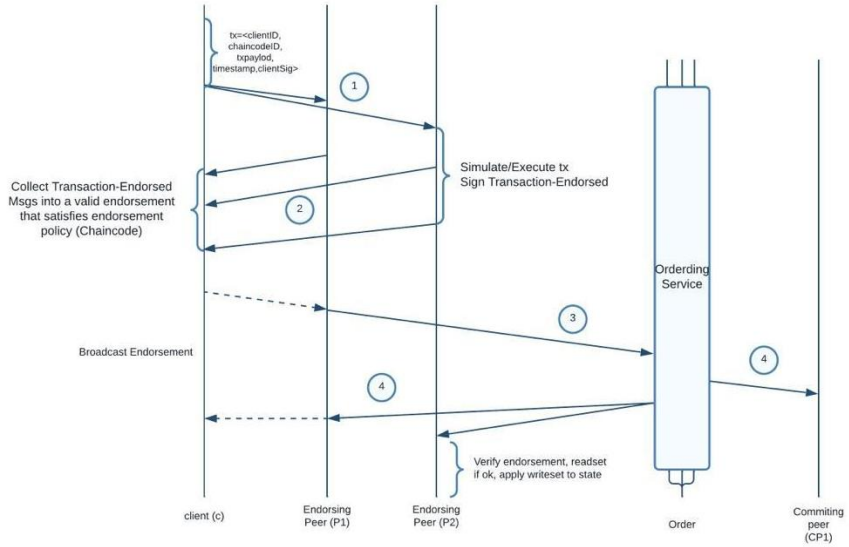


Fig. Smart Contract flow in Hyperledger

Below screenshot provides running of peer and nodes of the blockchain on the AWS:

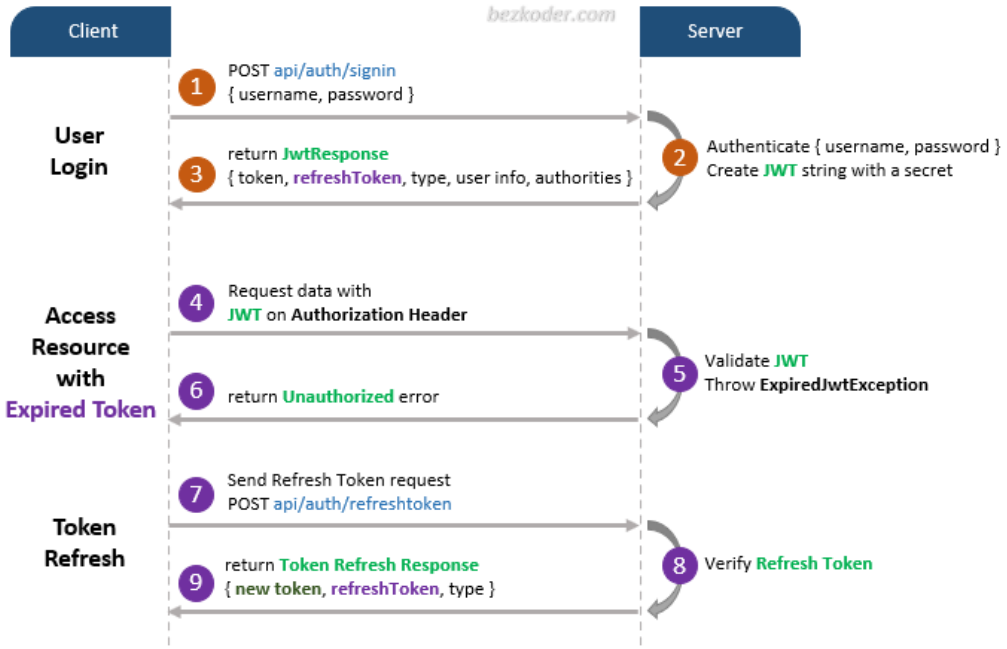
```
ubuntu@ip-172-31-27-100:~/fabric-samples/test-network$ docker ps -a
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS          PORTS
                                  NAMES
c2f52c010f84   hyperledger/fabric-tools:latest     "/bin/bash"              9 seconds ago    Up 8 seconds
                                  cli
0c09bf11837a   hyperledger/fabric-peer:latest      "peer node start"        10 seconds ago   Up 9 seconds    0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.
0:19051->19051/tcp, :::19051->19051/tcp   peer0.org2.example.com
94e99c9cb41b   hyperledger/fabric-peer:latest      "peer node start"        10 seconds ago   Up 8 seconds    0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:17051->1
7051/tcp, :::17051->17051/tcp             peer0.org1.example.com
58fbb2b0a6b0   hyperledger/fabric-orderer:latest   "orderer"                10 seconds ago   Up 8 seconds    0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:17050->1
7050/tcp, :::17050->17050/tcp             orderer.example.com
09ed3a9dcc13   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   18 seconds ago   Up 16 seconds   0.0.0.0:8054->8054/tcp, :::8054->8054/tcp, 7054/tcp, 0.0.0.
0:18054->18054/tcp, :::18054->18054/tcp   ca_org2
099a07d7f3c7   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   18 seconds ago   Up 16 seconds   0.0.0.0:7054->7054/tcp, :::7054->7054/tcp, 0.0.0.0:17054->1
7054/tcp, :::17054->17054/tcp             ca_org1
f67bb1fa492a   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   18 seconds ago   Up 16 seconds   0.0.0.0:9054->9054/tcp, :::9054->9054/tcp, 7054/tcp, 0.0.0.
0:19054->19054/tcp, :::19054->19054/tcp   ca_orderer
ubuntu@ip-172-31-27-100:~/fabric-samples/test-network$
```

Application communicates with the Blockchain through the Amazon SQS. The data is captured through five basic fields for all the transaction. The ID structure will have transaction ID appended with the type of transaction such as “ins,” “tns,” and “lab.” Data captured in the blockchain involve dateTime, transaction/insurance/bill amount, status of the record, and authoriser details.

3.1.3 Application Framework

The framework used for backend development of the application is ‘spring boot’ (Java). We have used spring boot because it has particularly satisfactory performance with respect to security for the application. JWT authentication is currently used for signup/signing in for various users. The backend also uses the JDBC template for interactions with the database. The crypto encoder class from maven helps to hash the sensitive fields such as passwords before storing them in the database.

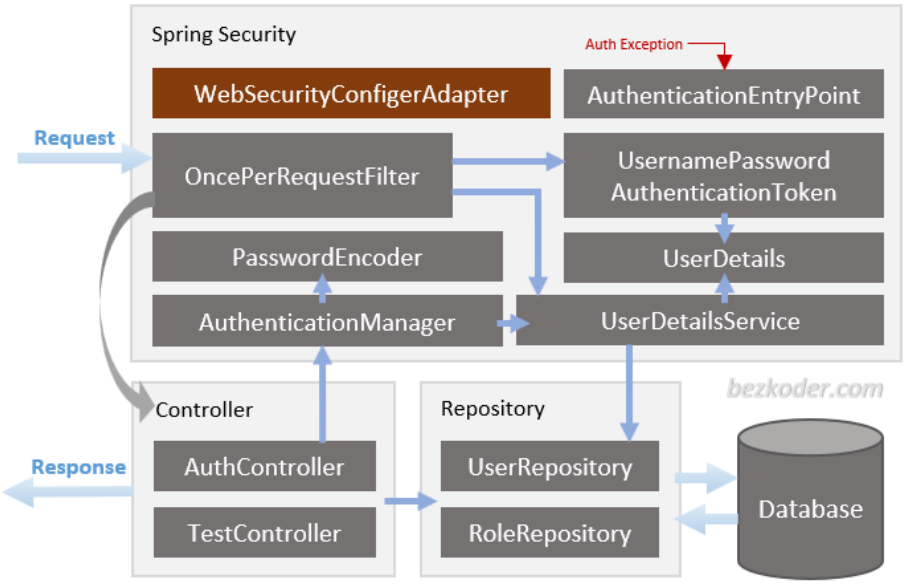
For managing the session management, we have implemented the refresh token functionality with spring boot. There is a value for the time (which can be present from the code) in which the token expires, and after that if the user is still logged in, the JWT authenticator sends in a new refresh token which will hold the session alive. This is depicted in the figure below:



‘WebSecurityConfigerAdapter’ is the main class that manages the security part of the application for sign in/signup. Once a request is generated from the front end, it is passed on to the AuthenticationEntryPoint class which extracts the username and password from the request. The password is then hashed using the crypto encoder class and saved into the userDetails class format along with the username.

Any interactions with the database from the front end go through two steps. First the request is passed on to the controllers (Patient controller, Hospital staff controller, etc.) which extracts the

JSON format from the request along with any path parameters. These parameters are then passed to the repository which executes the SQL query. The response from the database is forwarded to the frontend in the same structure. These interactions are depicted in the following figure.



The flow of secure hospital transaction requests and response can be understood with below data flow diagram and class diagram.

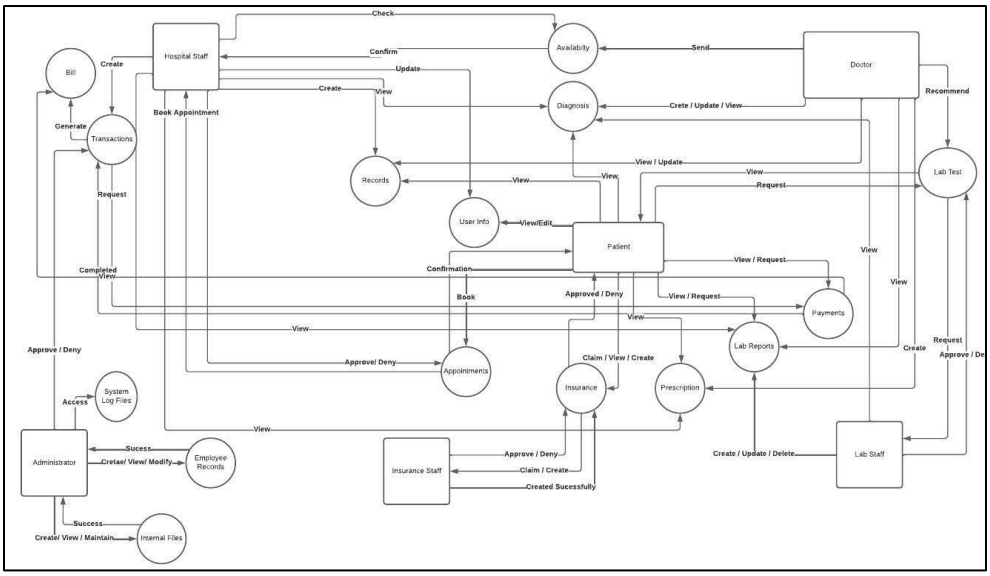


Fig.4 SHS – Data Flow Diagram

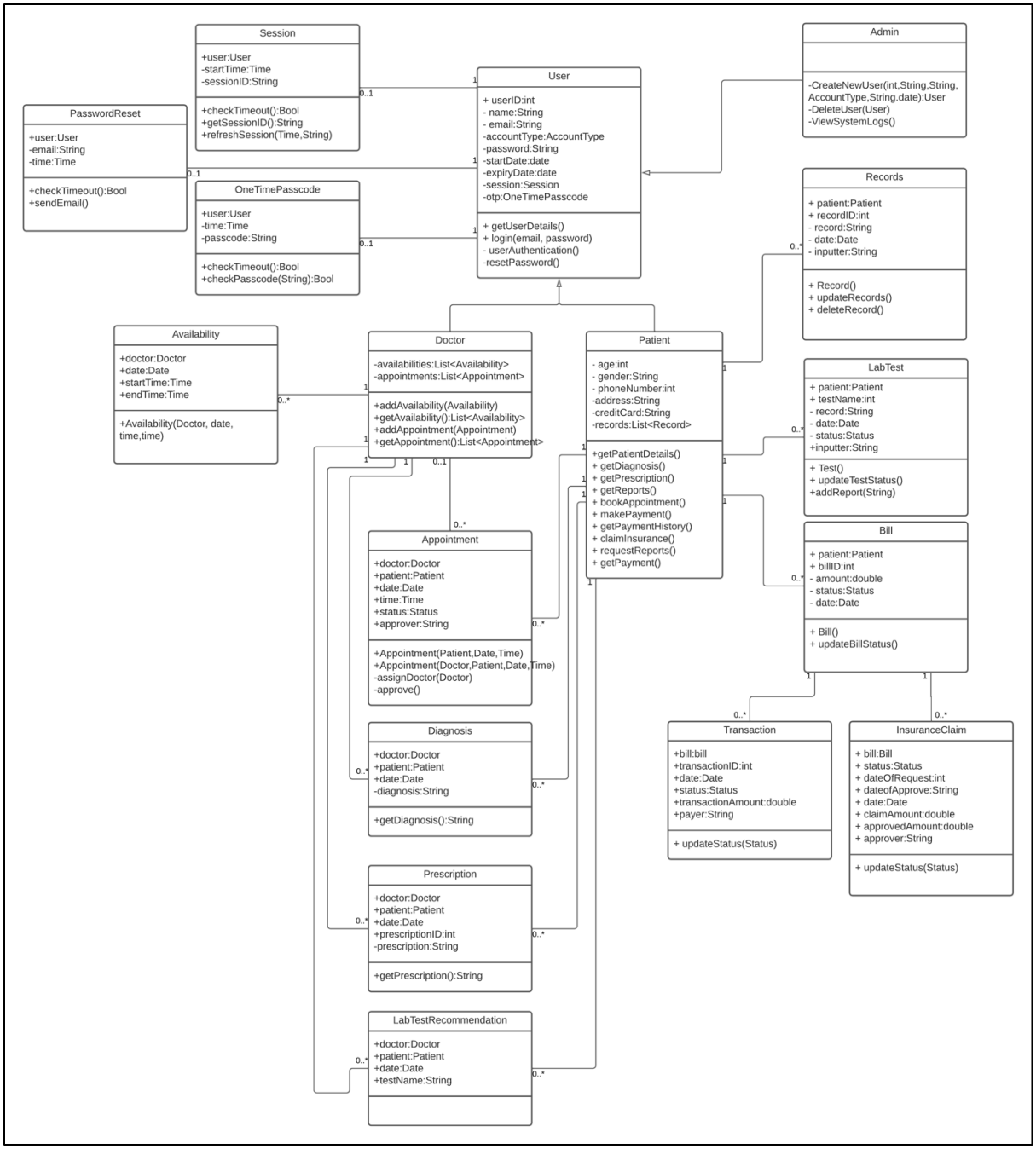


Fig.5 SHS – Class Diagram

3.2 Functionality Design

3.2.1 Hospital Staff

Hospital staff are at the centre of the transaction management and engage in smooth functionality of the hospital digital system. They can approve appointment requests based on doctor availability, create patient records, update personal information of patient on request by patient, view patient diagnosis, view prescription, view lab test reports, create transaction requests, complete transactions upon approval and, create receipts and bills.

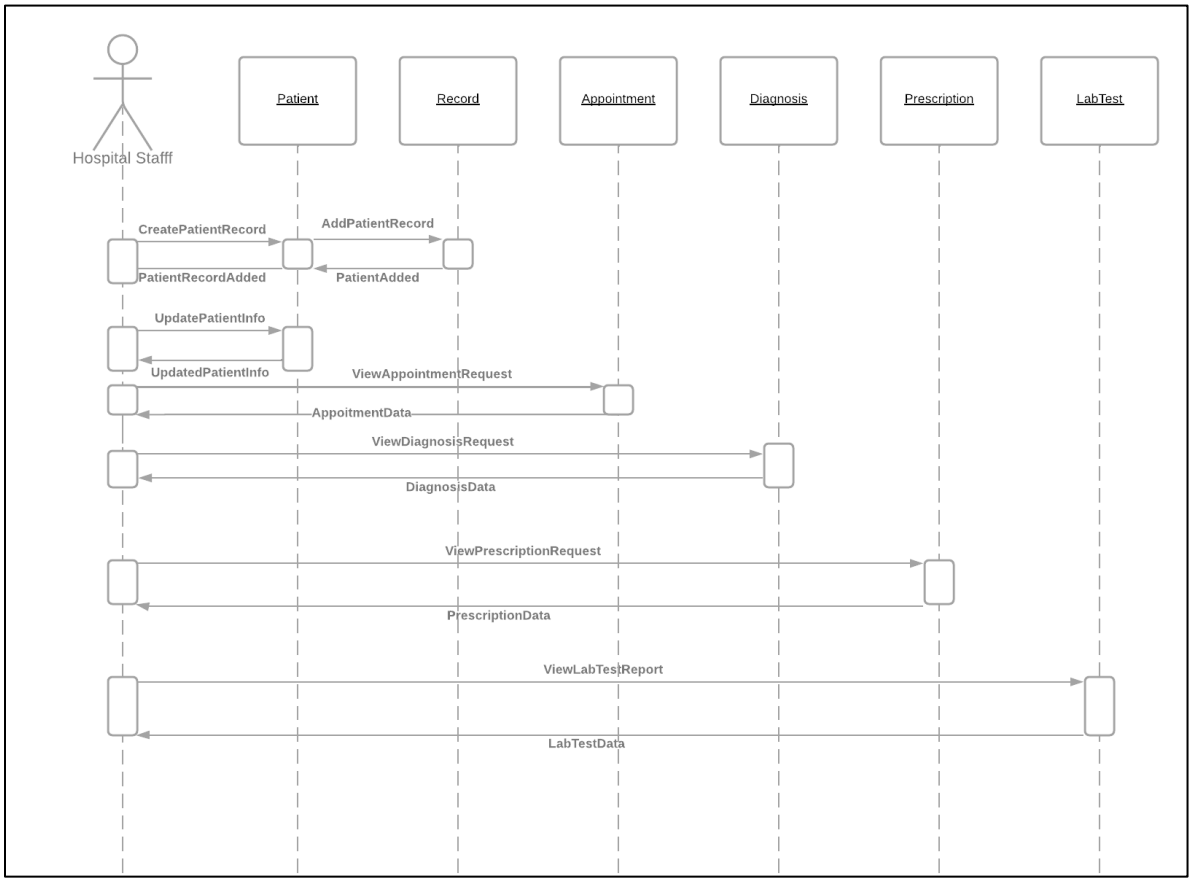


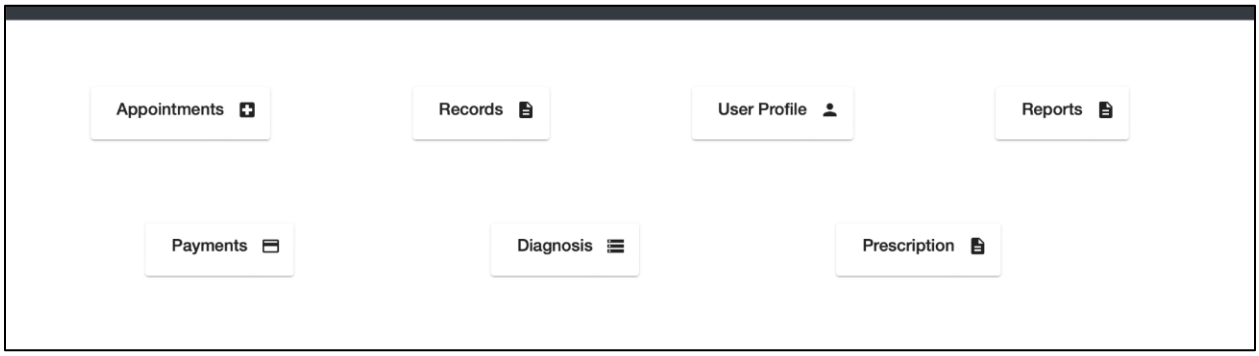
Fig.6 Hospital Staff Sequence diagram

The hospital staff has data models such as appointments, records, user profile, reports, payments, diagnosis, prescription. The hospital staff will receive the appointment requests from the patient and depending upon the availability of the doctor and the priority of the patient, the appointment for the patient will be approved/denied with the doctor. Following that, appointments are scheduled based on the availability of the doctor in that particular time period. After confirmation, the appointment gets updated in the patient portal depending upon the decision

taken by the hospital staff. The hospital staff can write a record for the patient depending upon the diagnosis, prescription and lab tests details he gets from various departments. Furthermore, the hospital staff receives information and can view the information from the corresponding diagnosis, prescription, record of the patient and lab test report about their diagnosis, prescription, reports of the lab test and the complete record of the patient. Furthermore, the hospital staff are allowed to change the personal information of the patient upon the patient's request. And through the payments tab, the hospital staff can create transaction requests for the required appointments and lab tests for the patient and upon the approval, the hospital staff can change the status of the transaction to complete and generate a bill of the completed transaction for the patient.

Here as we can see in the above fig. is the dashboard for the hospital and it shows how the hospital staff interacts with various other entities for the system. We will go through each of the entities and give you a detailed flow of how the hospital staff uses the system.

Frontend Structure:



Here as we can see in the above fig. is the dashboard for the hospital and it shows how the hospital staff interacts with various other entities for the system. We will go through each of the entities and give you a detailed flow of how the hospital staff uses the system.

Records : Here the patient’s profile diagnosis, prescription are being viewed and record information of the patient can be updated. The information for the record for the patient is being fetched from the patient id from the record table.

User Profile: Here the patient’s profile diagnosis, prescription can be viewed, and record information can be viewed and updated. The information for the user’s profile can be viewed by the hospital staff and has the option to edit the information of the patient upon patient’s request. The patient’s data is being fetched from the patient id from the record table.

Reports : Here, the report tabs show the information regarding the patient reports from the lab test. It shows the complete reports of the lab test as well as has the option to create a new report for the patient depending upon the lab test and add the report to the respective patient id. These reports are made after the doctor diagnoses the lab test and is updated by the hospital staff which can be viewed by the patient. Lab Test is conducted by Lab Staff and puts the information in the lab test model which is then shown to the hospital staff for creating the new record.

Diagnosis: Here the patient's profile diagnosis, prescription is being viewed and record information can be viewed and updated. The information for the diagnosis for the patient is being fetched from the patient id from the diagnosis table.

Prescription: Here the patient's profile diagnosis, prescription is being viewed and record information can be viewed and updated. The information for the prescription for the patient is being fetched from the patient id from the prescription table.

3.2.1.1 Appointment Approval

The patient books the appointment for the doctor the request is being updated in the appointment table and is open to decision whether to approve/ deny for the hospital staff. The decision of approval/deny comes from the hospital staff. They check whether there is availability of the doctor on the time the patient requested the appointment. If it is available, then the hospital staff approves the appointment request from the patient otherwise the request is denied from hospital staff.

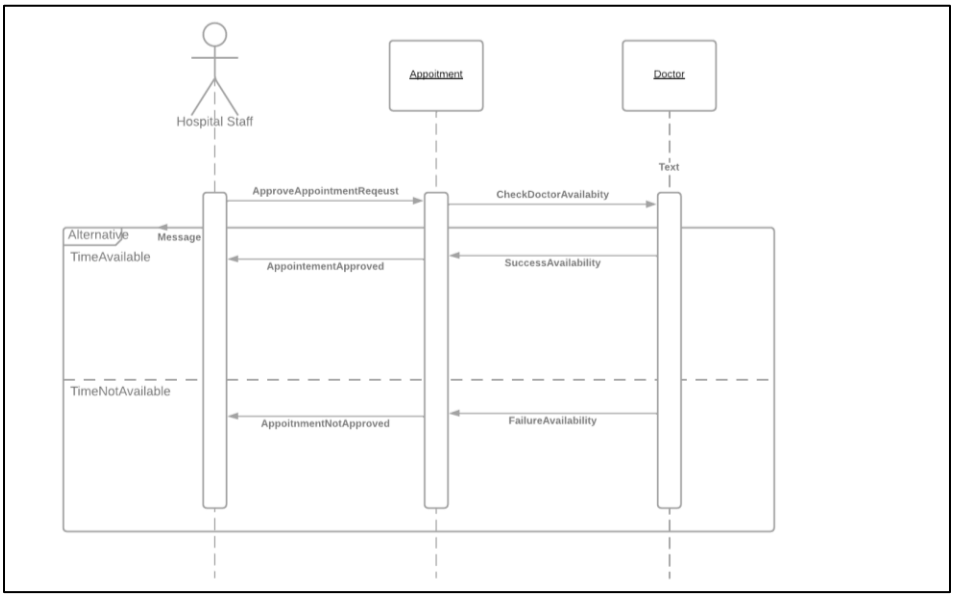


Fig.7 Appointment approval by hospital staff – Sequence diagram

3.2.1.2 Transaction Approval

An appointment/lab Test for a patient is confirmed by the hospital staff, the hospital staff will create a transaction request for the patient and whenever the patient does the payment, it moves to the complete Transactions tab. Here, the hospital staff will have the option to approve the transaction after the payment is received from the patient. When the transaction is approved, the bill gets generated, and it will be available to view in the patient portal and the hospital staff has the option to download the bill as well.

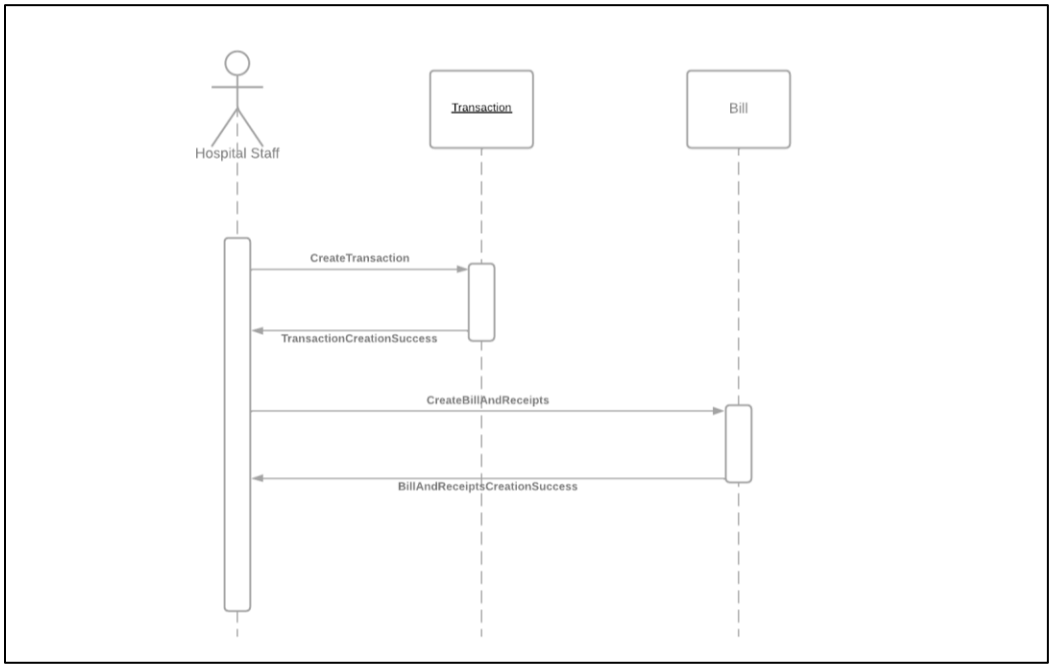


Fig.8 Hospital staff transaction approval – Sequence diagram

3.2.2 Patient Sequence Diagram

The patient uses data models such as diagnostic, lab test, prescription, insurance policy, transaction, appointment, and record. The patient schedules the appointment, which is saved in the appointment model. Following that, patients are scheduled based on their availability in the particular time period. Furthermore, the transaction's records are saved in the record model. Furthermore, the patient receives information from the corresponding diagnosis, prescription, and lab test model about their diagnosis, prescription, and lab test. Furthermore, the insurance policy model provides the patient with information about the current insurance policy as well as the status of previous claim insurance policies. Finally, the transaction model provides information on

numerous transactions conducted by the patient. Following screenshot of the dashboard of the application will help you understand how the patient interact with various other models:

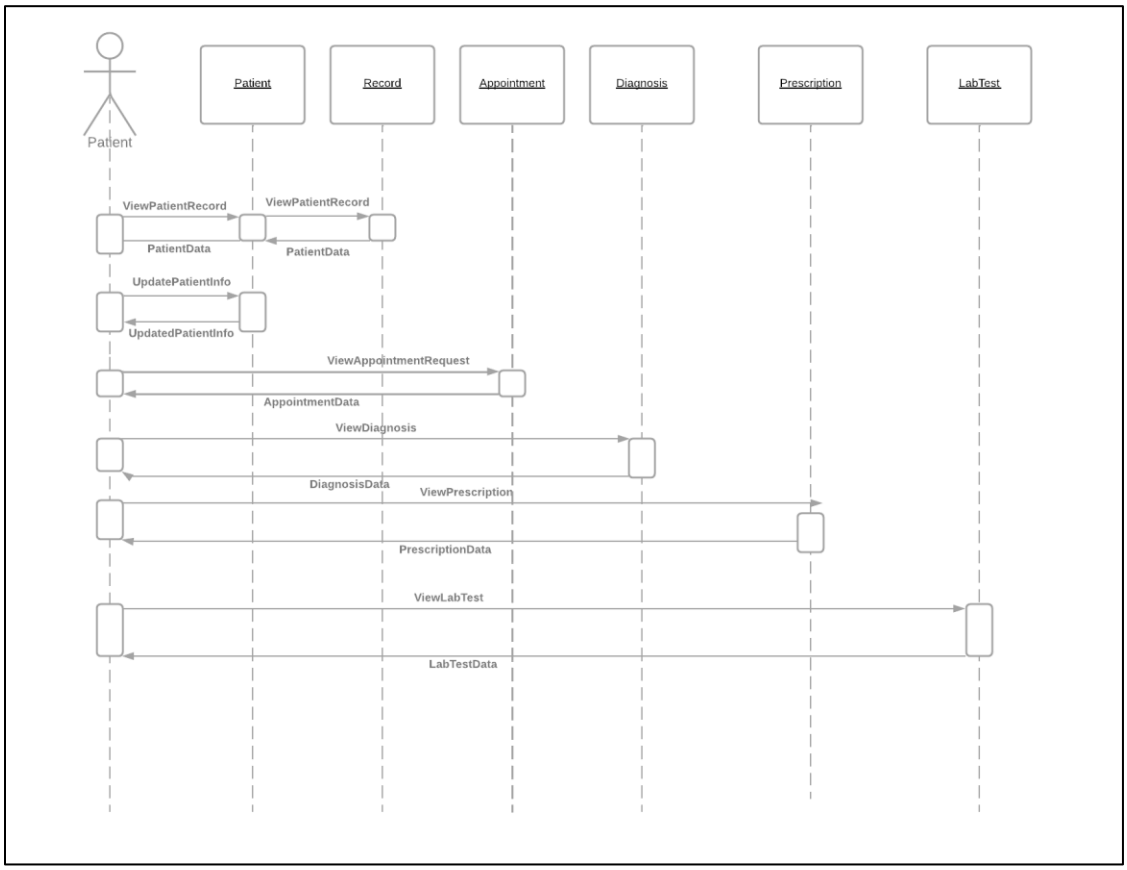
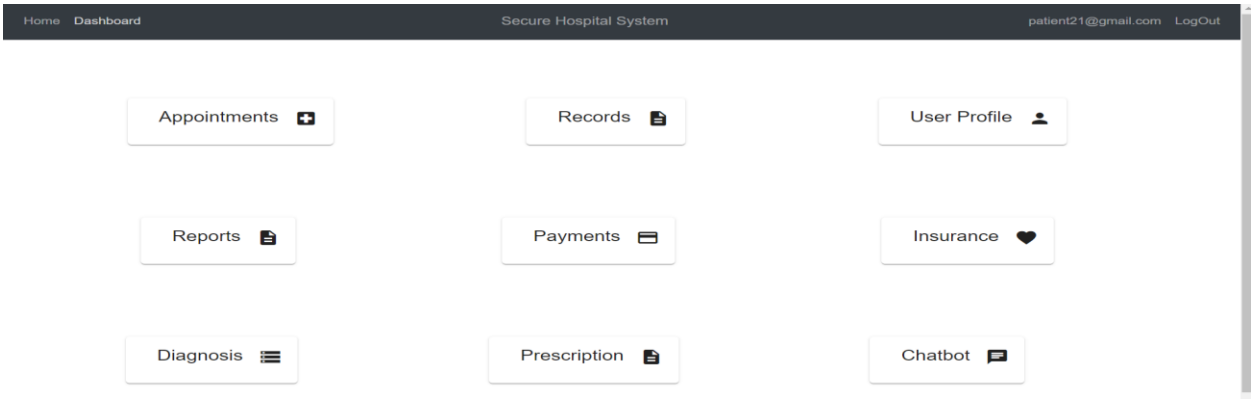


Fig.9 Patient Sequence diagram

Frontend Structure:

The dashboard for the patient and it shows how the patient interacts with various other entity for the system. We will go through each of the entity and gave you detailed flow of how patient uses the system.



Records : Here the patient's profile diagnosis, prescription and record information are being viewed. The information for the record for the patient is being fetched from the patient id from the record table.

User Profile: Here the patient's profile diagnosis, prescription and record information are being viewed. The information for the user's profile for the patient is being fetched from the patient id from the patient table.

Reports : Here, the report tabs show the information regarding the patient reports from the lab test. It shows the complete reports of the lab test as well as has the option to download the report in the form of pdf. These reports are made after the doctor diagnoses the lab test on the patient portal and advises the patient to do lab tests for the particular diseases. Lab Test is conducted by Lab Staff and puts the information in the lab test model which is then shown to the patient.

Diagnosis: Here the patient's profile diagnosis, prescription and record information are being viewed. The information for the diagnosis for the patient is being fetched from the patient id from the diagnosis table.

Prescription: Here the patient's profile diagnosis, prescription and record information are being viewed. The information for the prescription for the patient is being fetched from the patient id from the prescription table.

Chatbot: Here the chatbot helps user in navigation through the application as well as provide the patient with information such as appointment, insurance policy etc.

3.2.2.1 Appointment Request and Approval

The patient books the appointment for the doctor and wait for the confirmation of whether the appointment is booked or not. The request approval comes from the hospital staff. They check whether there is availability of the doctor on the time the patient requested the appointment. If it is available, then patient gets the appointment otherwise the request is denied, and patient have to book the appointment again. The appointment also gives information regarding the past appointment request date as well as the status of the appointment.

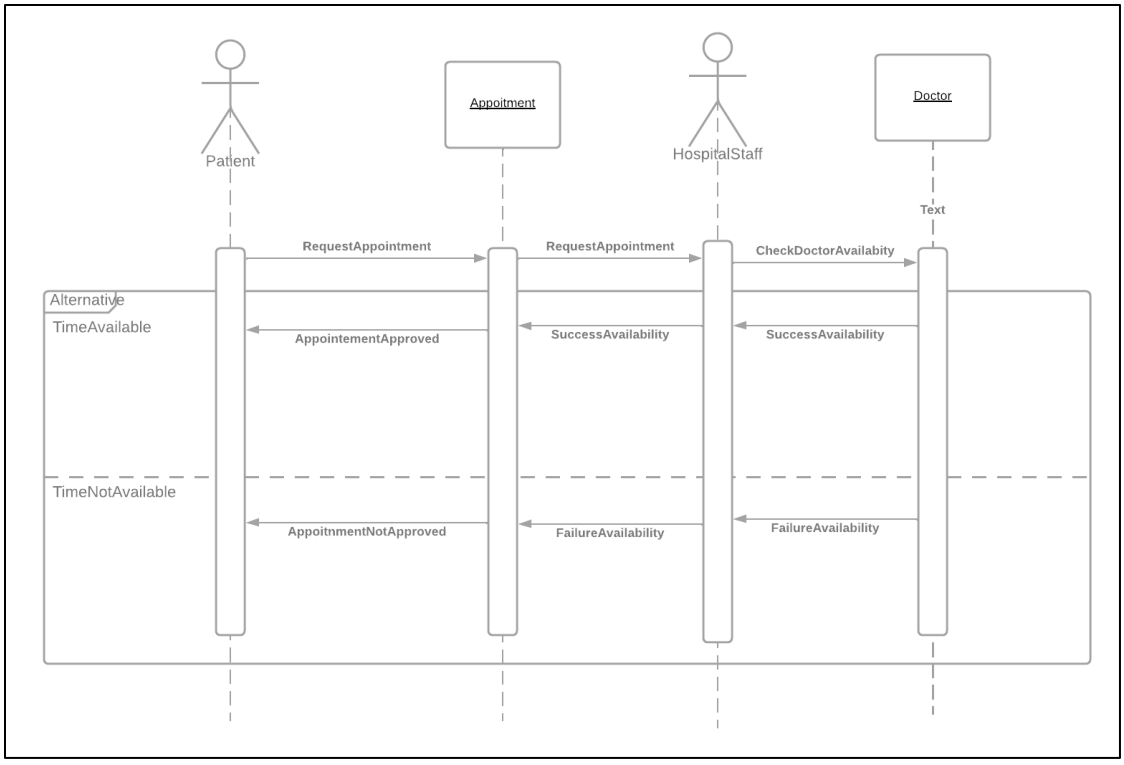


Fig.10 Appointment request and approval – Sequence diagram

3.2.2.2 Insurance Claim and Bill payment

Payments: Here, the patient uses the payment tab to make the payment for the fees to the hospital. Patients have three options to make the payment through cash, credit card and insurance. If the insurance is used as a mode of payment, then on that basis of the insurance claim of the user the fee to the hospital is paid.

Insurance: Here the insurance tab shows the user the current insurance policy which is being taken by the user as well as the past insurance claimed by the user. First of all, the policy which was claimed in the past has information regarding the claimed amount and approved amount by the insurance staff for the patient. Also, the request data as well as the approved date. Here the fig. shows the insurance page with the tabs for the past insurance claims and current insurance policy

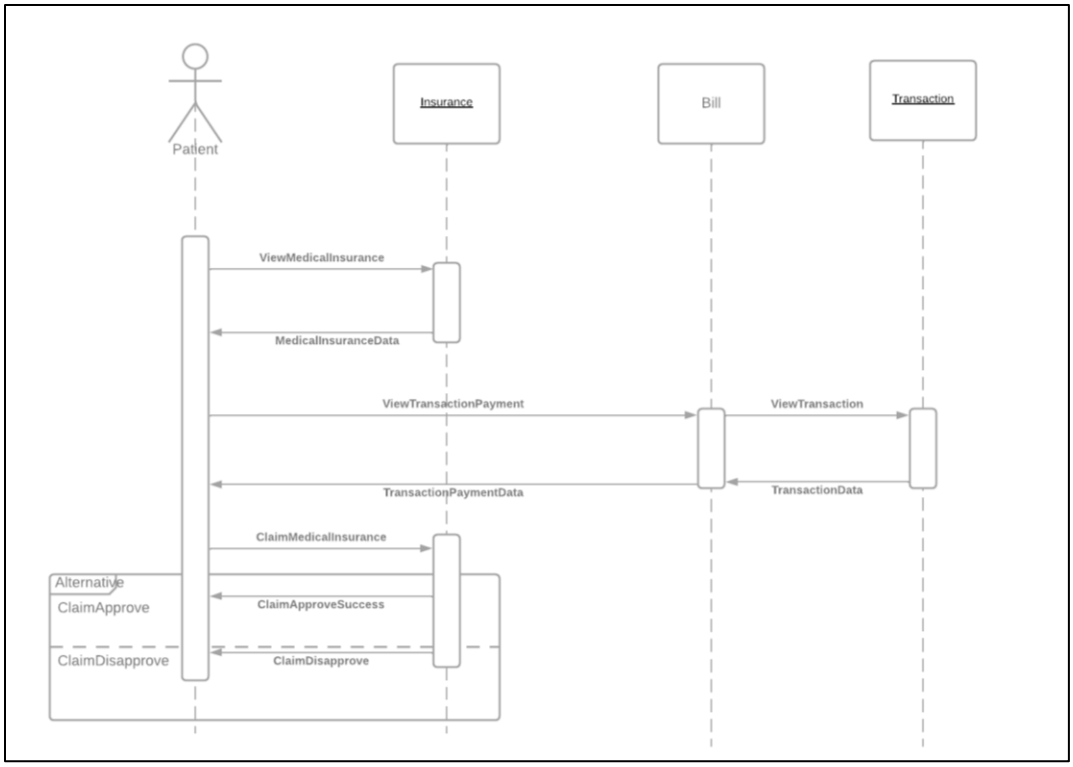


Fig.11 Insurance claim request and Bill payment – Sequence diagram

3.2.3 Doctor Sequence Diagram

Database Structure: The doctors use the appointment, availability, diagnosis, lab Tests, prescriptions, and records tables as database models, and we use api calls from the Postgres and PGadmin to access the necessary data from the database. The Doctors have the privileges to view and update patient records. Can create, update, and remove patient diagnosis. He has the ability to create prescriptions, recommend lab tests, and view lab reports. The Doctor has all the options from editing the records table to prescriptions table along with the diagnosis table from the front end which we build using angular.

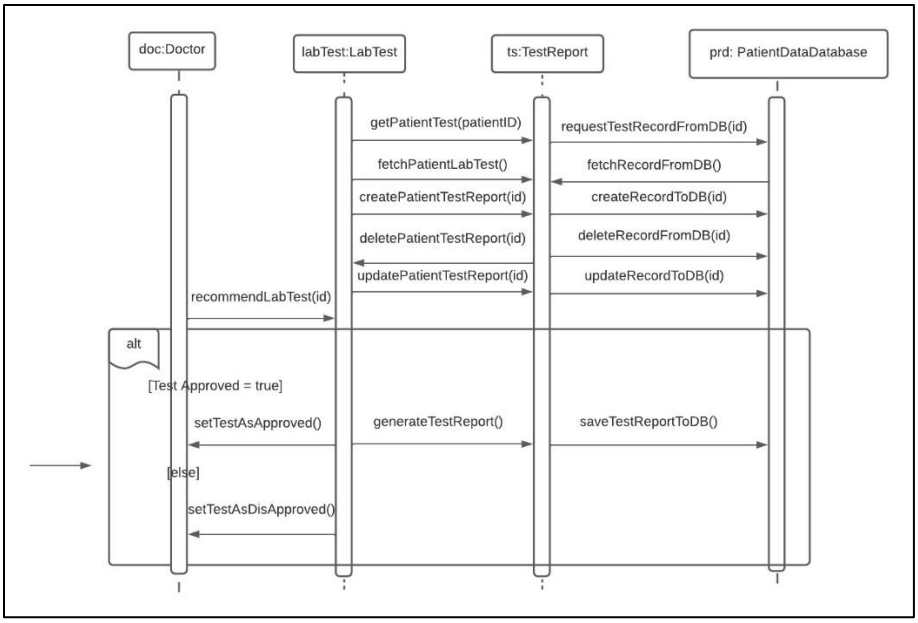


Fig. Lab test request processing – Sequence diagram

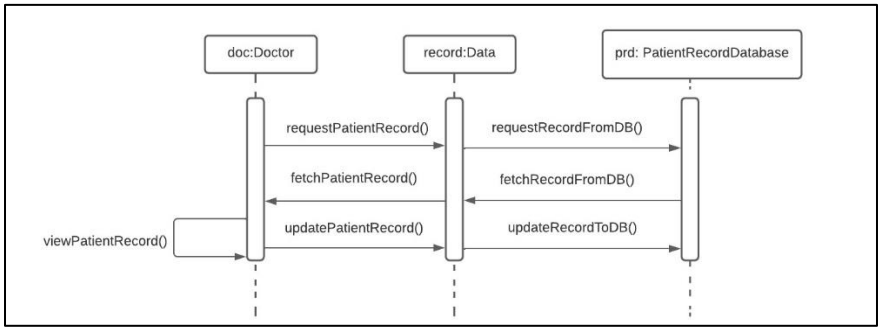


Fig. View patient history by doctor – Sequence diagram

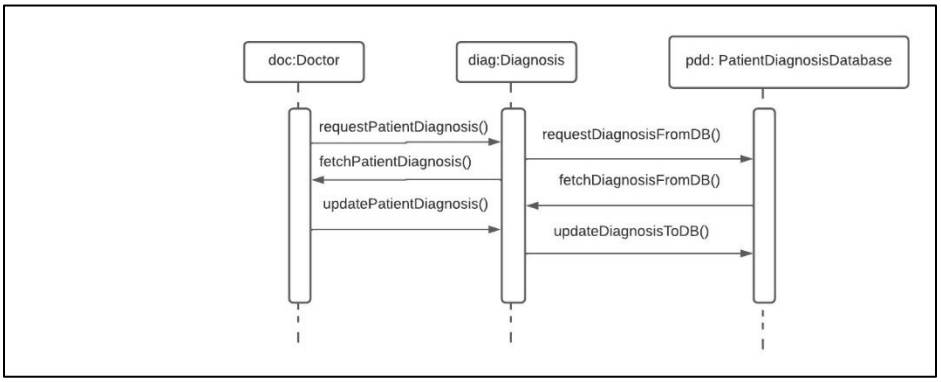


Fig. Patient diagnosis – Sequence diagram

3.2.4 Insurance Staff Sequence Diagram

The insurance staff utilizes the following database models which are the bill, insurance Claim, insurancePolicy and the transaction using Postgresql and PGadmin. The hospital staff generates corresponding bills for each patient. Each patient might be related to multiple bills. So, we use transactionID as a primary key to uniquely track the bills. The patient can choose for which bills he can raise an insurance claim. Once the patient requests an insurance claim, it gets populated in the insuranceClaim table which then the Insurance Staff can access to get the pending insurance claims. The insurance staff has the permissions to edit the insurancePolicy table as well. They have the ability to create new insurance policies and also view the existing ones from the insurancePolicy table.

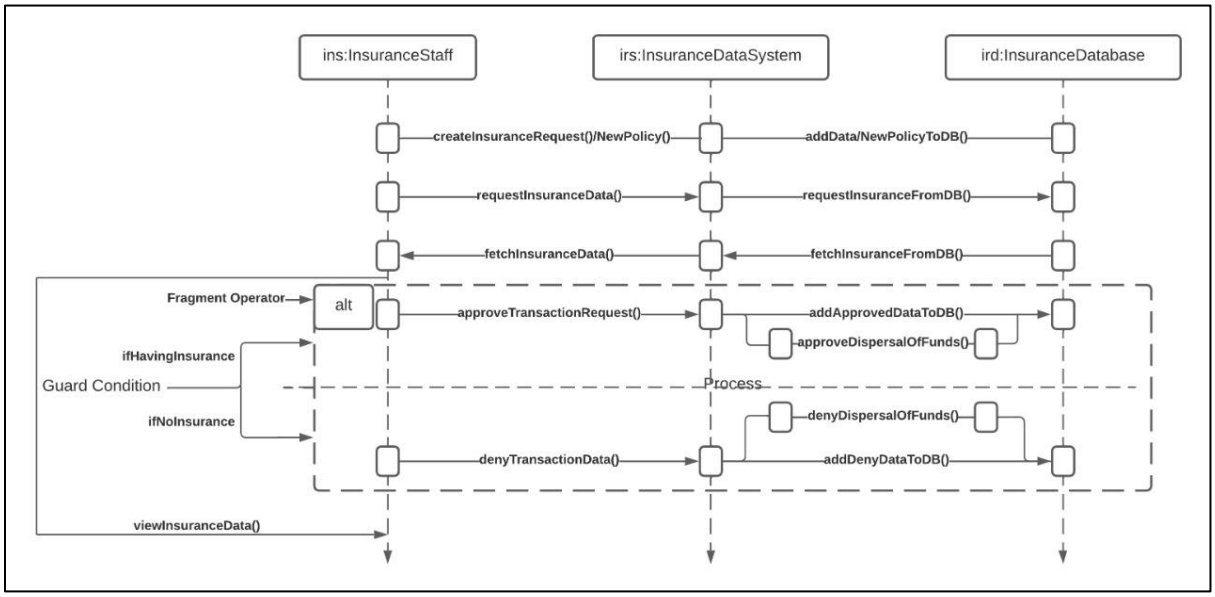
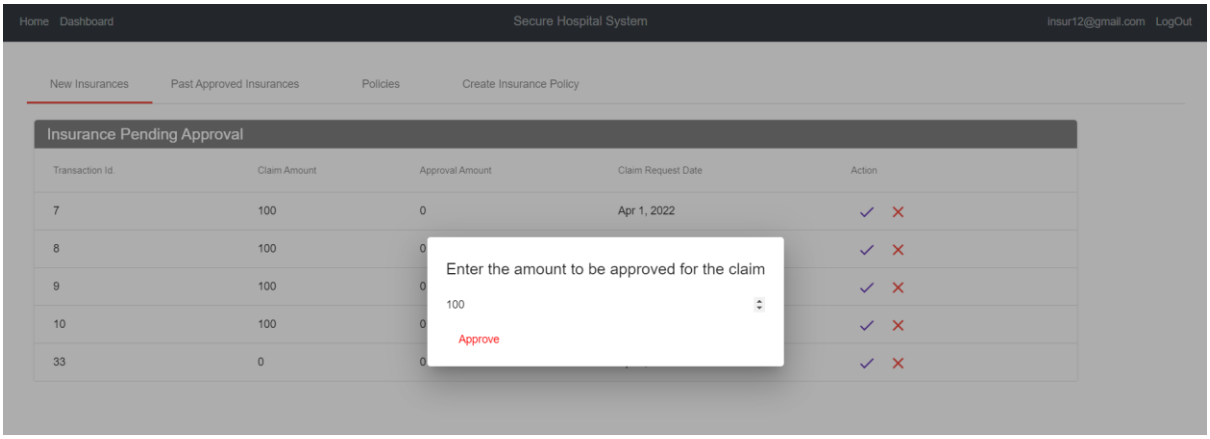


Fig.15 Insurance claim process – Sequence diagram

Front-End Structure: When the user logs in as an insurance staff, they get the InsuranceStaff tab which contains the different functionalities of Viewing the active insurance claims, which the staff can approve or deny. The next tab is the View approved insurances which lets the staff see all the records of the approved claims. The following tab is the one which contains all the records for the insurance policies of every patient and the final tab would be for the insurance staff to add new insurance policies for new or existing patients based on requirements.



In the New Insurances section, we access the bills from the bill table that the patient raises a claim request for along with the patient id, and thus letting the insurance staff uniquely identify the bills of a particular patient and decide on whether to approve or deny the request. On denying the request, the claim request will be removed from the insuranceClaim table and will be repopulated under the pending section of the bill table, allowing the patient to re-request a claim again or pay for himself. But if the insurance staff decides to approve the claim, they get to choose the amount they want to approve and once approved it is updated in the insuranceClaim table and the transaction table is updated accordingly with the transaction being completed.

In the Past Approved Insurances Section, we use the insuranceClaim table and request all the data that has the approver details and approvedAmount updated. Which means we get back all the insurance claims that has been approved by the staff; this helps the staff to maintain the records of the approved insurance claims.

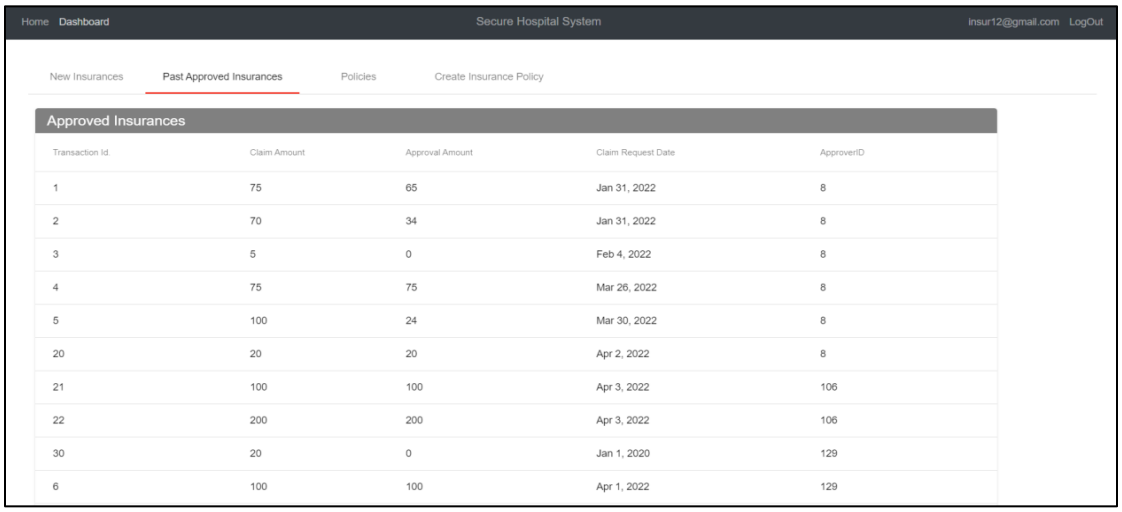


Fig. Insurance Staff can view all the approved insurance claims

Once the patient requests an insurance claim, the following factors will be initialised which are the transactionID, deteOfRequest, claimedAmount, date and status. When the patient has requested the claim, the status is made as requested. When the insurance staff sees all pending claims, he can either approve/deny these claims. If he chooses to deny the claim, the patient will be notified that the claim has not gone through. We will not allow the patient to request the same bill again for insurance as the insurance staff have their reasons to reject the insurance claim. But when the insurance staff chooses to approve the claim, he will be prompted to enter the amount of money that he wishes to approve. Once he enters the amount and clicks on approve, the insuranceClaim table with the specific transactionID gets updated with the new values of dateOfApprove, approvedAmount and approver and the status changing to approved. Later on, the Hospital staff verifies this and changes the status to insurance_transaction. We maintain the records of all the approved insurance claims to validate them with the insurance policies to make future reimbursements. We made sure that the insuranceClaim is not entirely viewable by the patients but only by the Insurance Staff. The insurance staff will be able to see on the bills for which the patient has requested insurance claims. We would like to entend this functionality to once the patient approves and enables permission for the insurance staff to access thier entire bills table, the staff could cross-verify every bill and notify the patient which bill can be claimed by insurance and how much will be covered even before requesting the claim, making the process a lot more easier and efficient and hassle free for the patient.

The insurance staff is also responsible for creating and maintaining the insurance policies that are uniquely curated for individual patients. We work with the insurancePolicy table to manage all the policy queries. Each patient can be associated with multiple insurance policies, but the table make sures that the same policy is not duplicated for the same patient. This way, the staff can verify the policies and validate the claims accordingly. With the maximum amount of money that can be spent on a policy for a particular patient, we use this data in validating the insurance claim requests and this assists the staff members to make easy decisions.

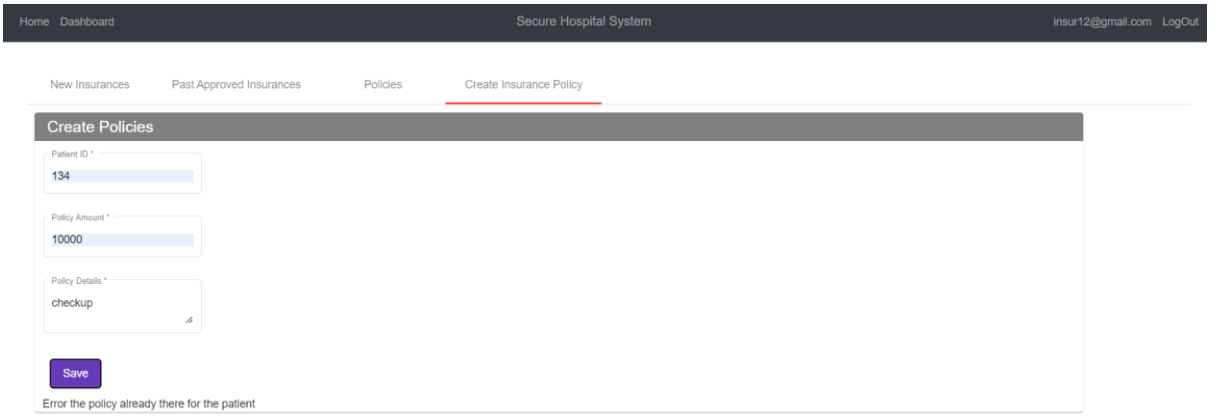


Fig. Insurance Staff can create insurance policy for patients

In the following tab (Policies), we access the insurancePolicy to get all the policy details of each and every patient. It has all the details of maximum amount that can be spent on a patient for a particular policy. This aides the staff in determining if a claim has to be approved or denied based on the policies that are set to that particular patient. We have implemented such a way that we do create policies for patient whenever necessary, but we do not delete them but keep them for record.

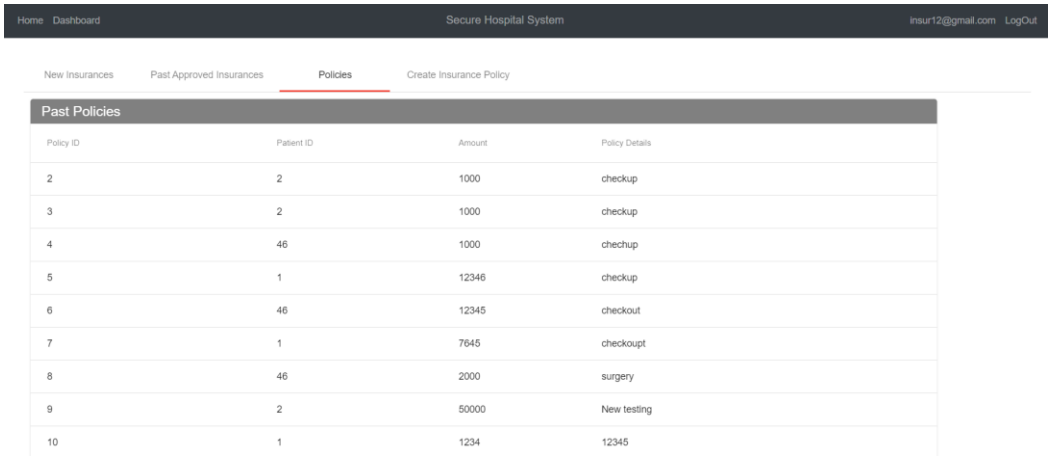


Fig. Insurance Staff can view all insurance policies

In the last tab, which is the Create Insurance Policy, we access the insurancePolicy table to create and maintain the policies for each patient. This table ensures that a patient can have any number of policies associated to him but not the same policy repeated more than once. This functionality helps to add policies to the insurancePolicy table which the staff later access to view all the available policies for the patient while approving their insurance claims.

3.2.5 Administrator Sequence Diagram

The administrator can create, view, modify, and delete employee records. He maintains all internal files. Authorize (approve or deny) transaction requests. He is provided system user to access system log files (System log files are only accessible by administrators).

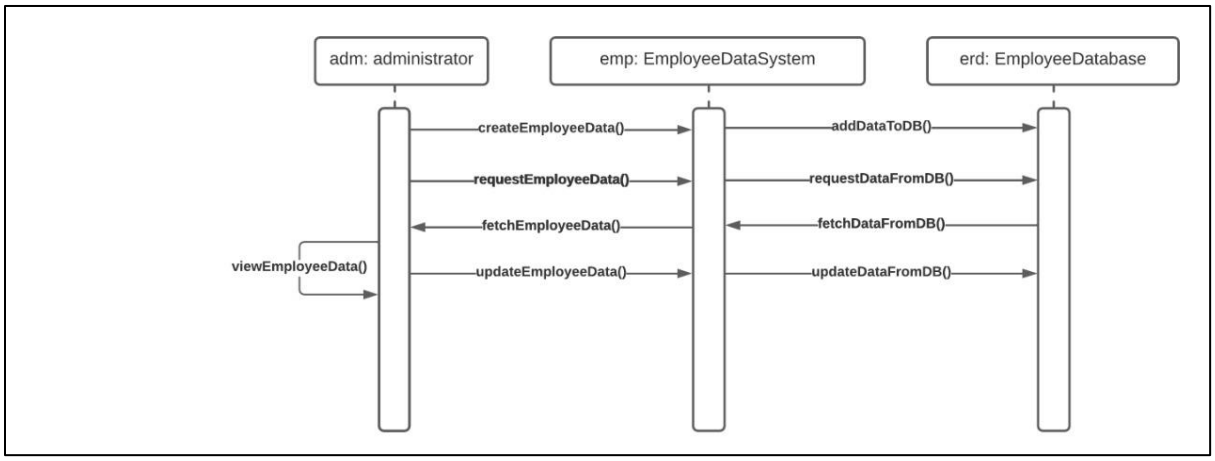


Fig.16 Administrator access to employee data – Sequence diagram

Database Structure: The administrator is responsible to handle registration and profiles of employee records, transactions, and access to system logs. The data of all users are fetched using the user table which contains all the fields. The name,email,accountType and password fields are added to the database whenever a new employee is being created. Admin can also view and delete the records of employees from the profile page of all users.

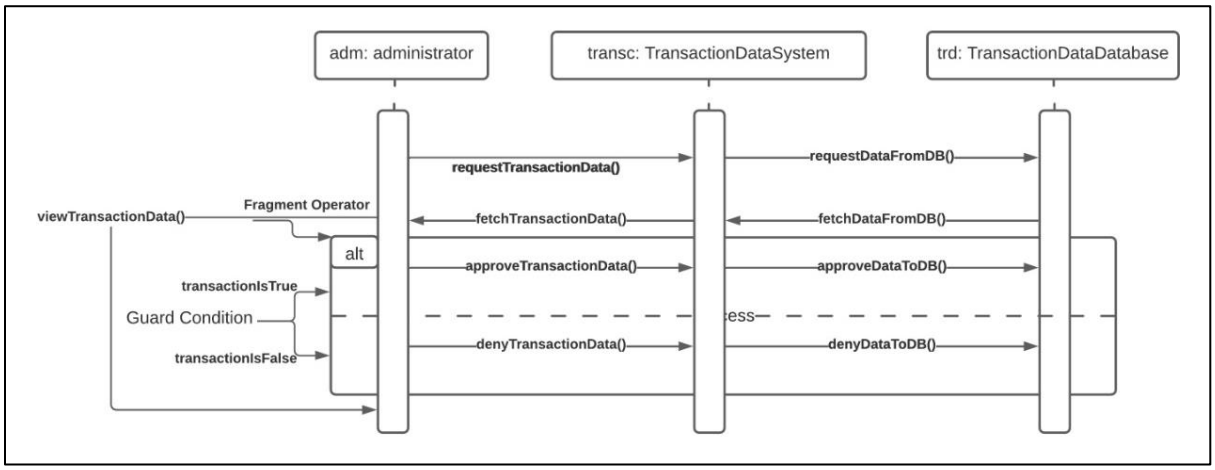


Fig. Administrator access to transaction data – Sequence diagram

Front-End Structure:

When the admin clicks on the Registeration button it opens up a page as shown below to enter the details of the employee along with its role. Once the user registers an account it gets stored in the data base with the given permissions.

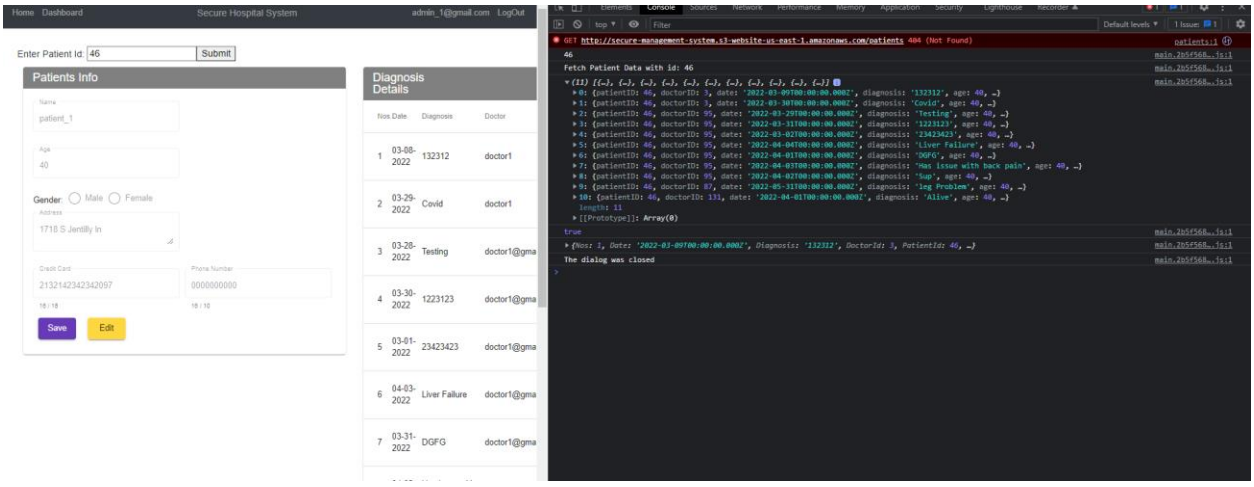
Transactions: Admin has the ability to reject or approve the transaction created by the hospital staff for a patient. Once the admin approves the transaction then it moves for payment.

All Users

No.	User ID	Name	Email	Account Type	Action
1	0	generaldoctor	fake_email@gmail.com	doctor	✕
2	1	user1	user1@gmail.com	patient	✕
3	2	user2	user2@gmail.com	patient	✕
4	3	doctor1	user3@gmail.com	doctor	✕
5	4	doctor2	user4@gmail.com	doctor	✕
6	5	doctor3	user5@gmail.com	doctor	✕
7	6	hstaff	user6@gmail.com	hospital_staff	✕
8	7	lstaff	user7@gmail.com	lab_staff	✕
9	8	istaff	user8@gmail.com	insurance_staff	✕
10	9	admin	user9@gmail.com	admin	✕
11	12	shashank	shasankrddy@gmail.com	admin	✕
12	19	shash1	shash1@gmail.com	admin	✕
13	23	abhishek	abhi@asu.edu	admin	✕
14	26	shash8	shash8@asu.edu	admin	✕
15	28	mounika1	mounika1@gmail.com	patient	✕
16	29	mounika22	mounika2@gmail.com	patient	✕
17	30	test	test@gmail.com	patient	✕

UserProfile: Admin also has access to all data of the patient and can edit any details for it.

System Logs: Admin can access and view the system log files.



3.2.6 Help & Support Centre

Web page to provide the emergency contact details with location. FAQ to include:

1. Instruction for appointment booking.
2. Insurance Claim
3. Facilities available at the Hospital

3.3 Chatbot Design

We are using an Amazon Lex service with advanced natural language models to design, build, test, and deploy conversational interfaces in our application.

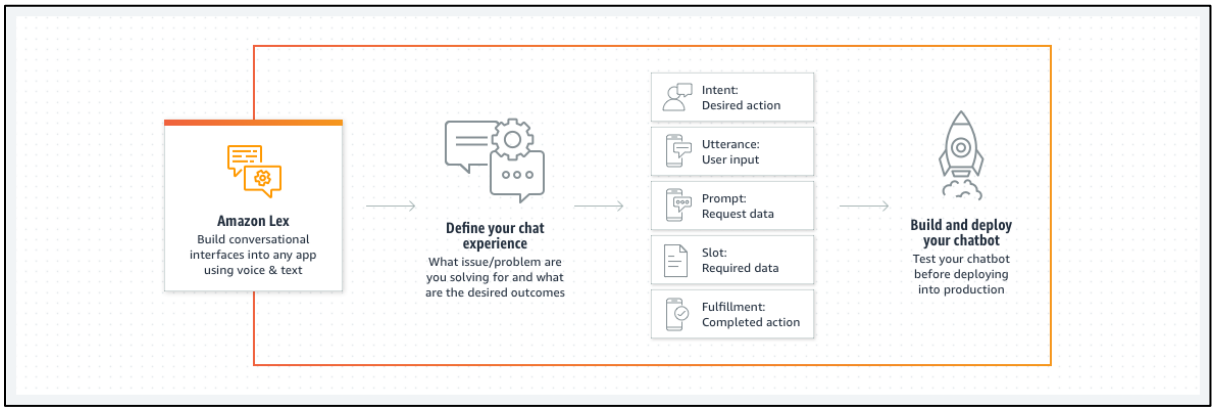


Fig.18 Chat bot – data flow diagram

Amazon Lex follows the basic chatbot architecture adhered to by all the big cloud based Conversational AI providers. We will need to create intents, entities, script (dialog wording), and then the dialog flow.

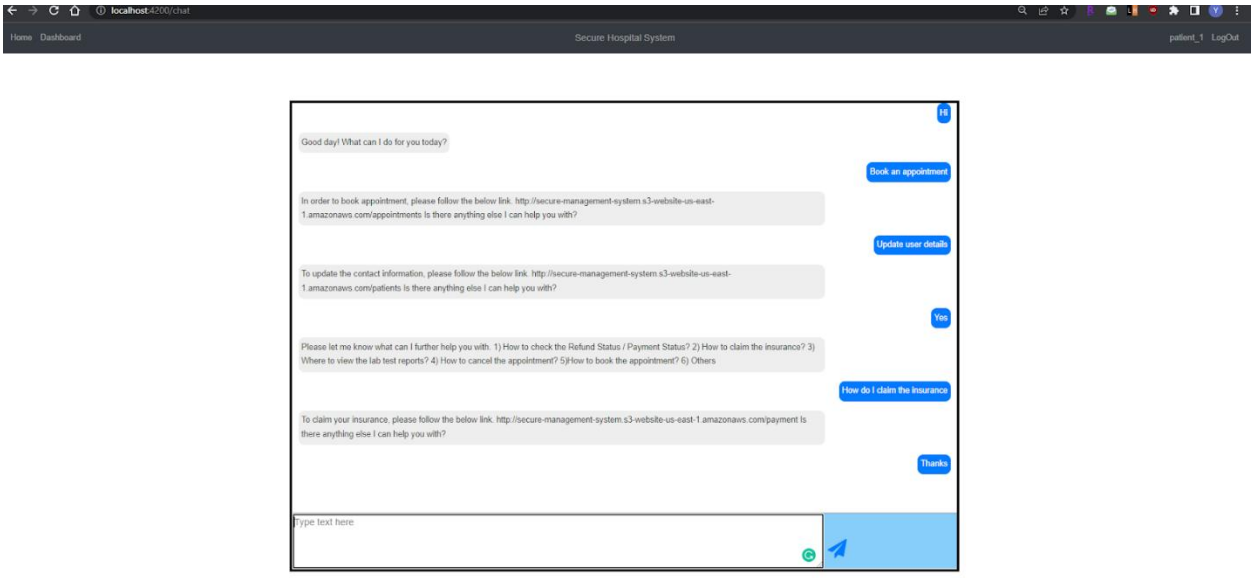


Fig.19 Sample of chat bot

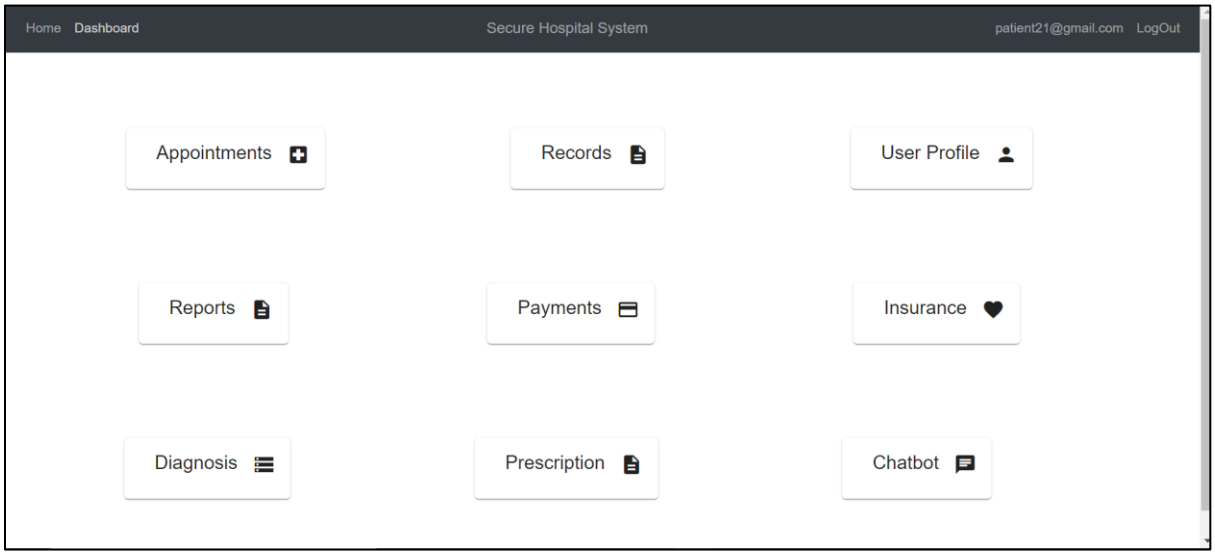
4 User Interface

For accessing secure hospital system, user has to sign-up with relevant role. Login to the SHS using login authentication and two-factor authentication. Following subsection describes the functionality and navigation for various user roles.

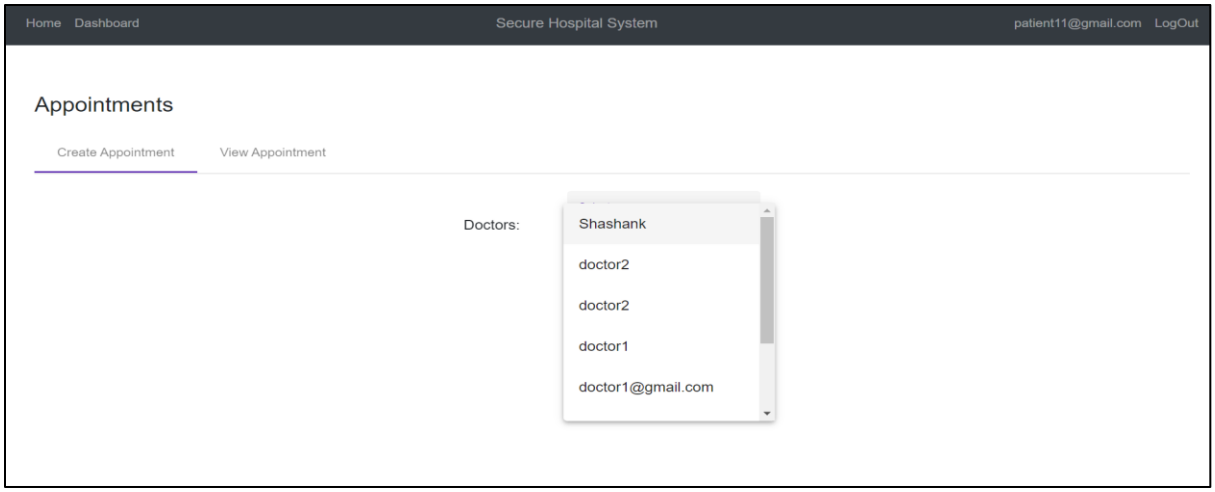
6.1 User: Patient

Login to the SHS webpage with the user role as patient. Below is the functionality and navigation through the patient user.

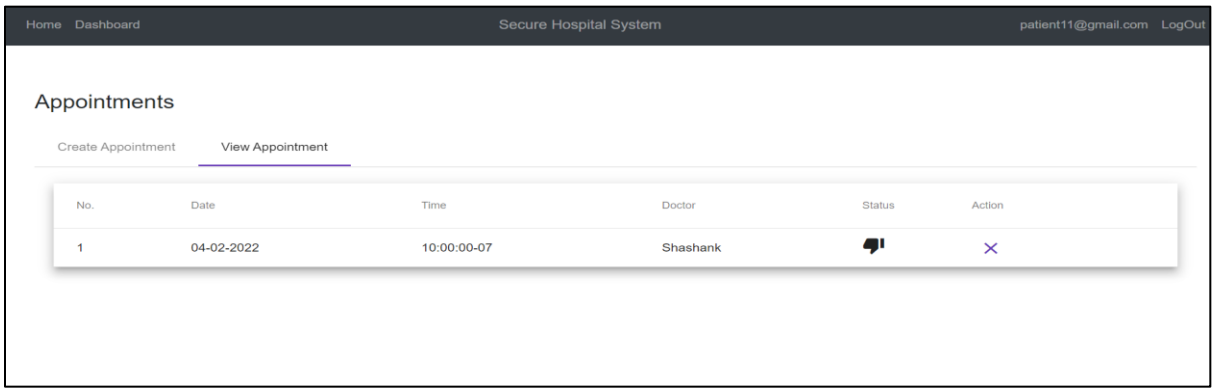
1. After logging in the patient can see the following dashboard for the patient:



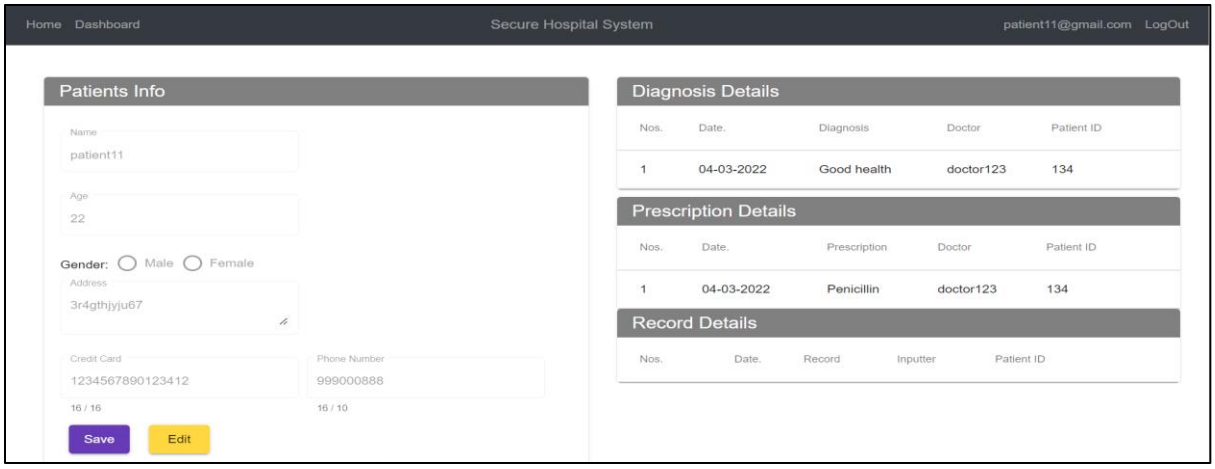
2. Now, by clicking on the appointments tab the patient can book the appointment with the doctor and the next page demonstrates how we can book an appointment with the doctor



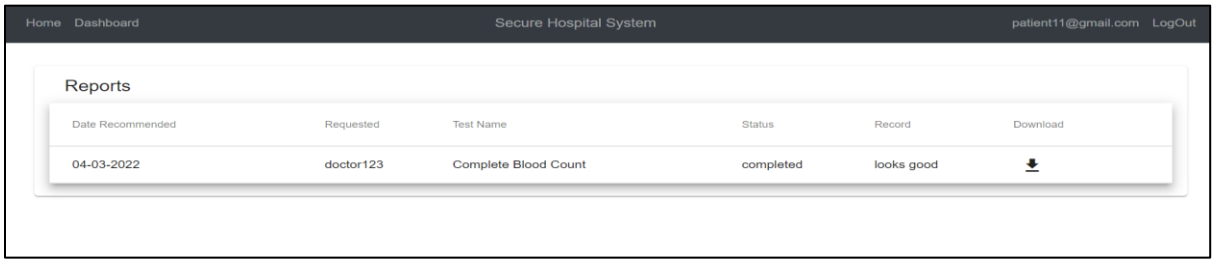
3. Also, after booking the appointment with doctor you can view your booked appointment on this page and also the status of your booked appointment.



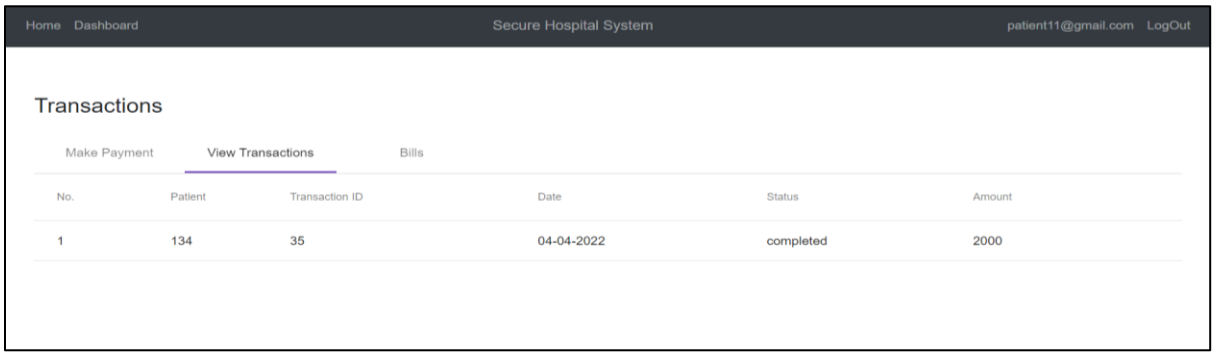
4. Now, the profile of the patient as well as its diagnosis details of any previous appointment as well as the prescription and record of the patient can be found on this page as given below:



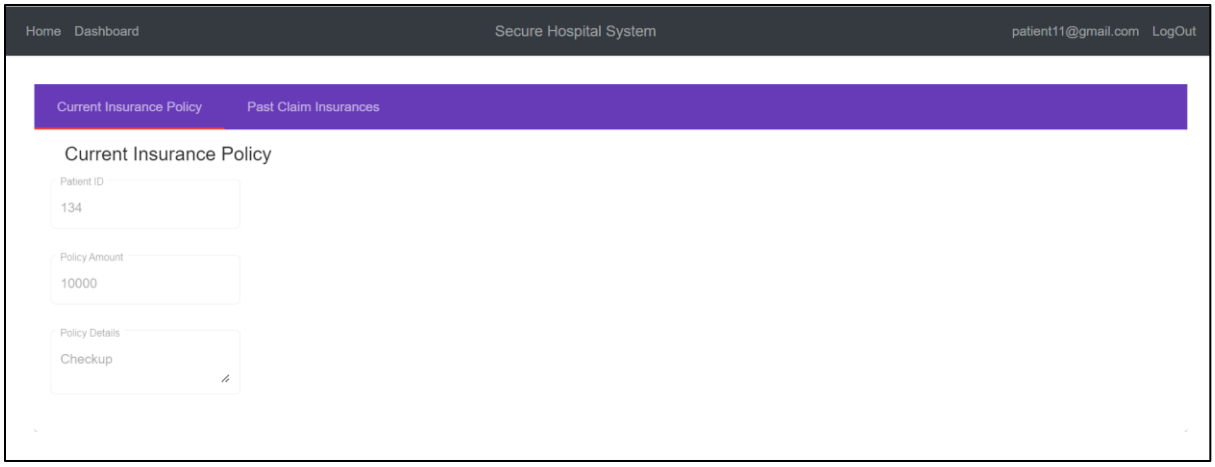
5. Now, the report of the tests for the patient can be found on the page given below. Also, the patient can download the lab report in the form of the pdf as given below:



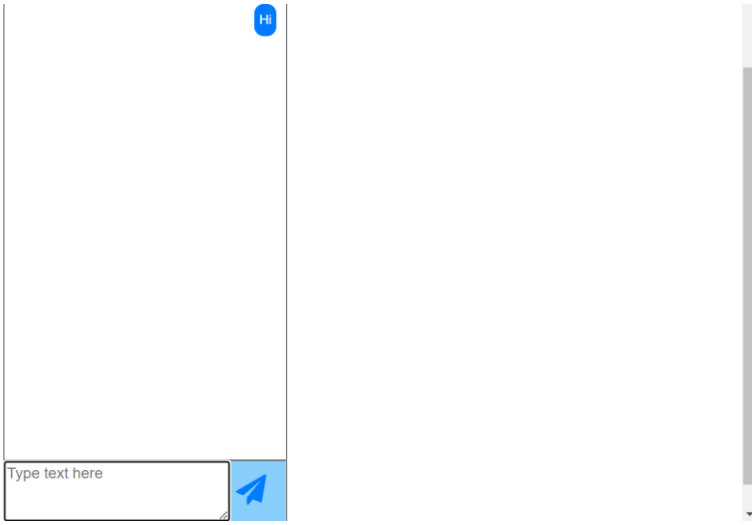
6. Now the bills of the patient can be downloaded on this page as it has tabs to see what transactions are being performed by the patient and also the past payment made by the patient. Also, the status of the transaction and the amount is also given on this page.



7. Now, the given below page shows the insurances claimed by the patient as well as current insurance policy of the patient.



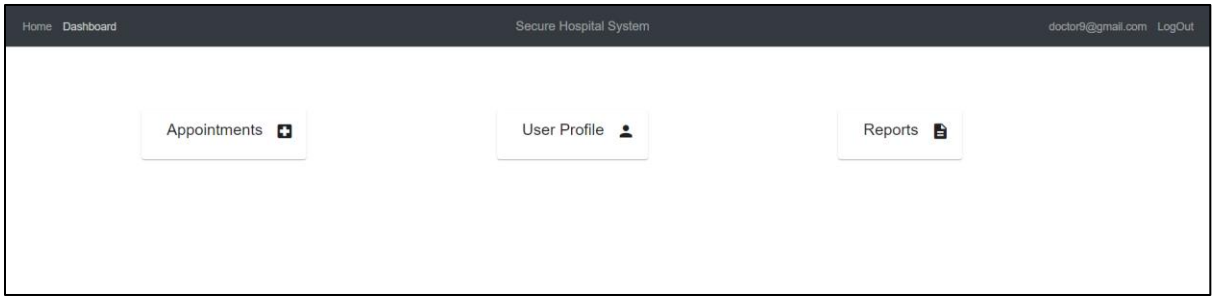
8. Now this page shows the chat bot which would help the user in getting the information it needs from the application.



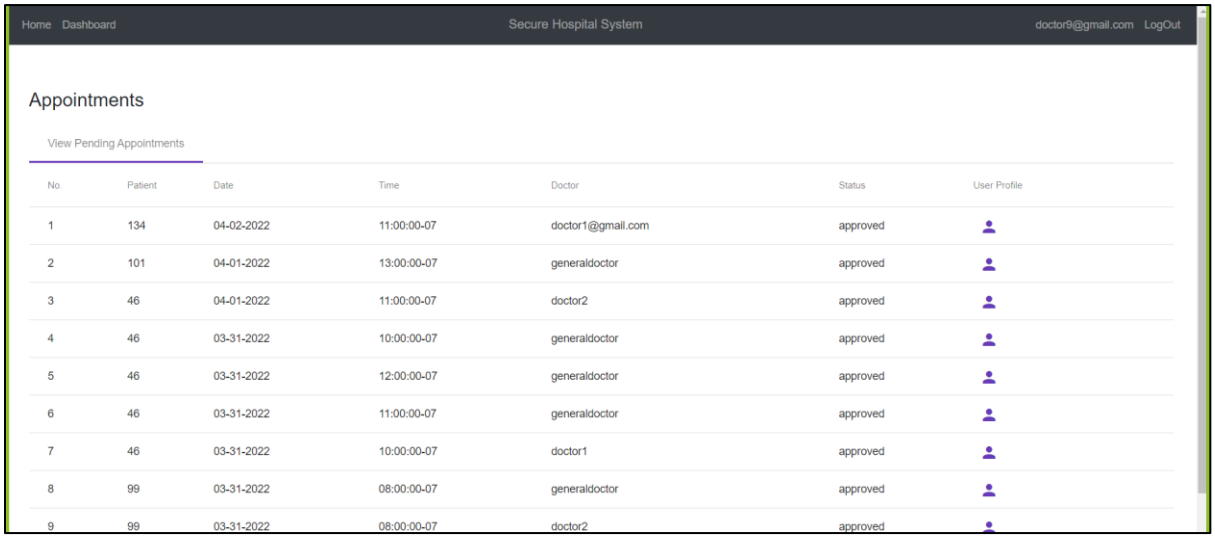
6.2 User: Doctor

Login to the SHS webpage with the user role as doctor. Below is the functionality and navigation through the doctor user.

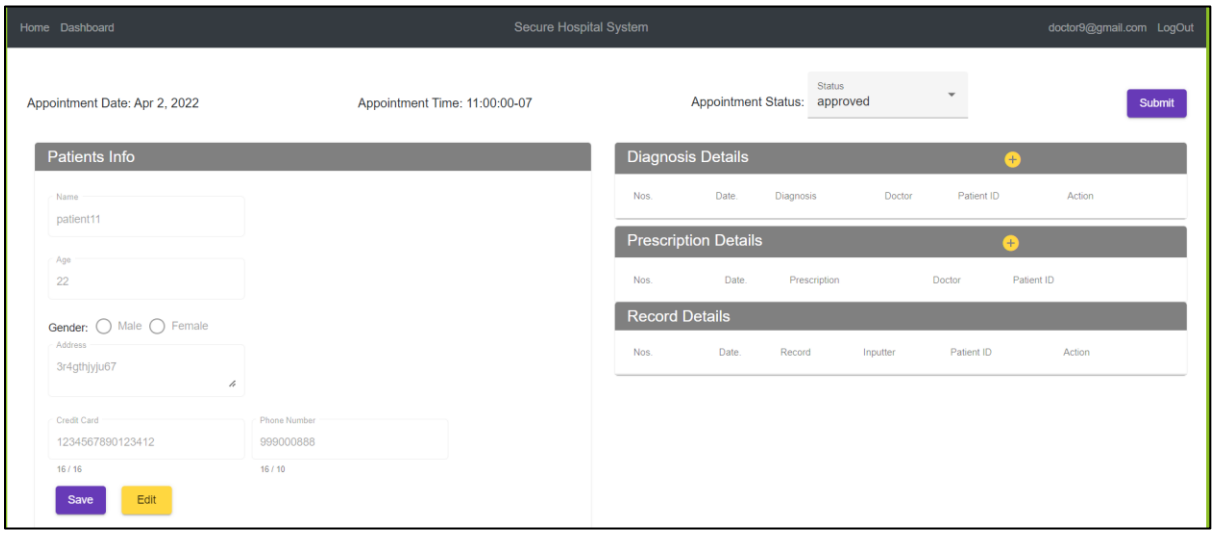
1. After logging in, the doctor will be able to see this dashboard:



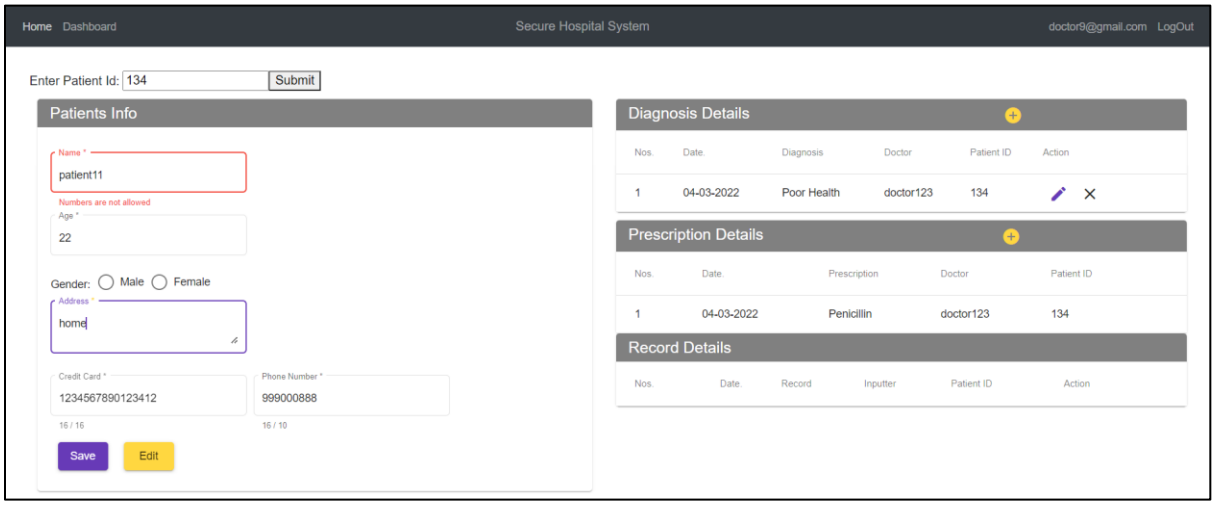
2. Doctor can click on appointments to see their approved appointments.



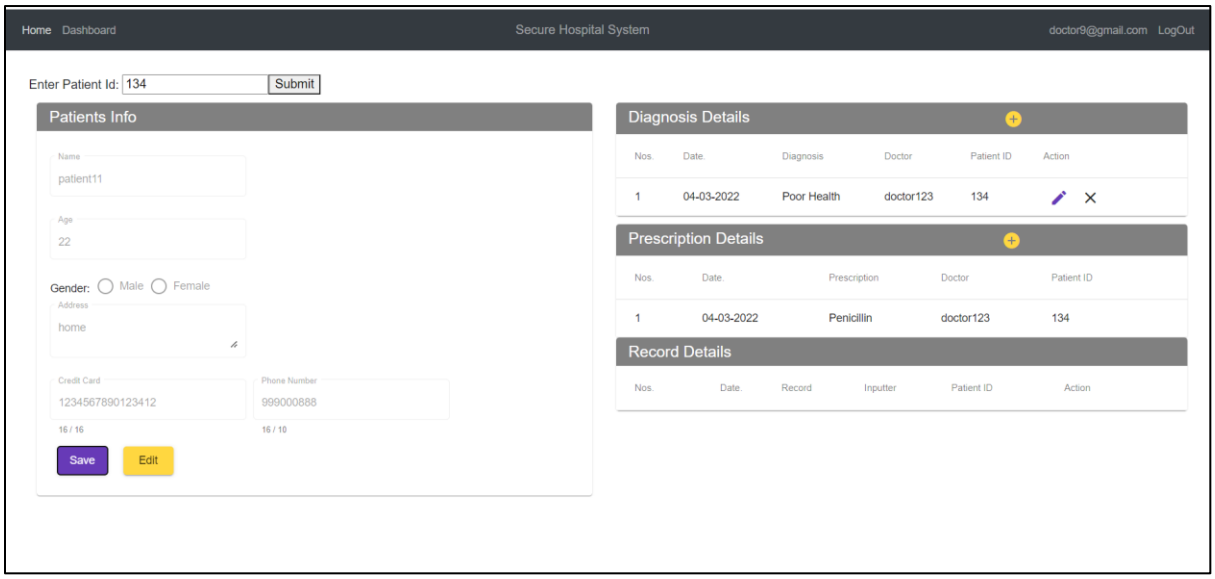
3. They can check the patient data of the patients who have approved appointments.



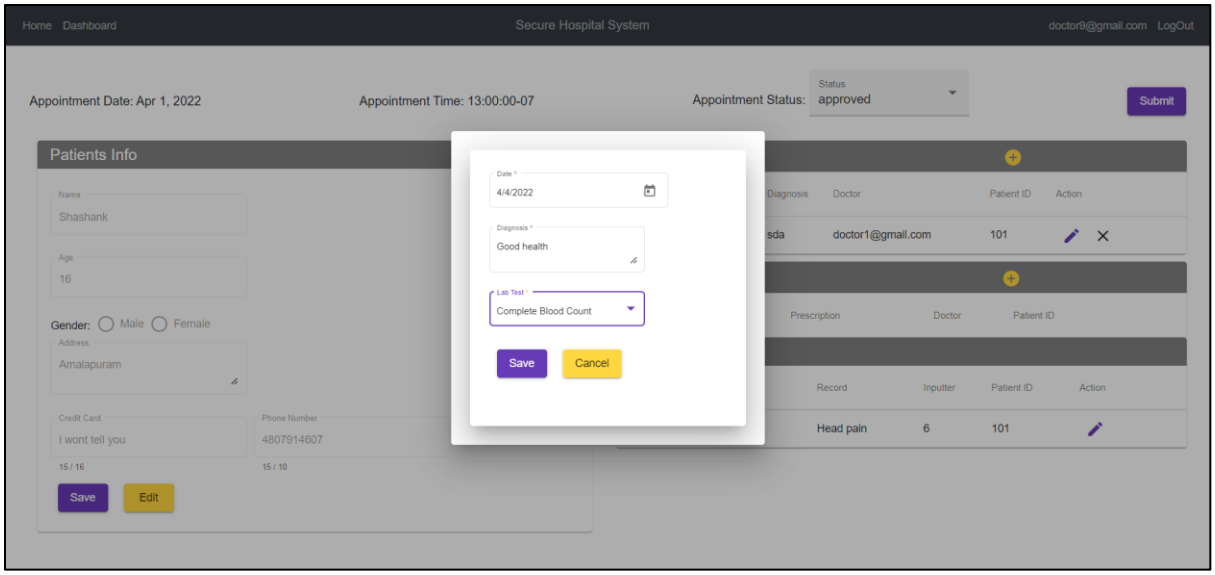
4. Doctor can edit the patient data.



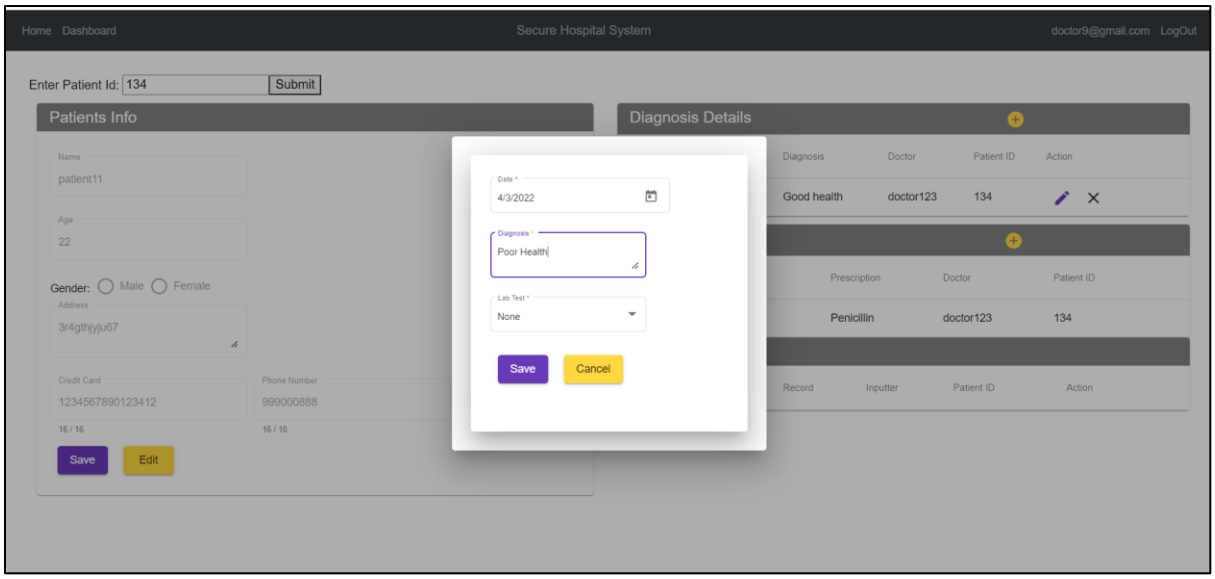
Below is the post update view.

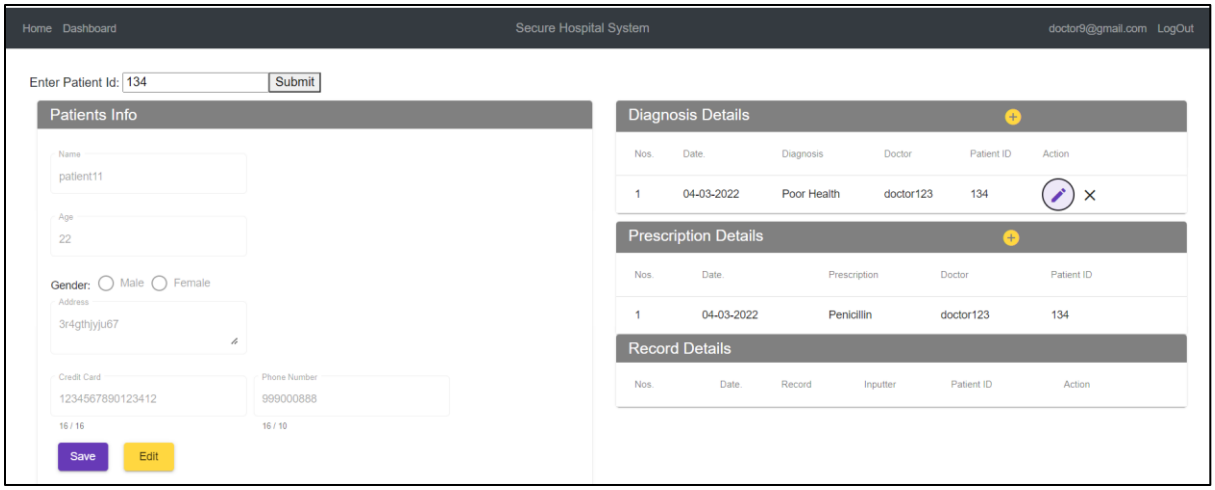


5. They can add diagnoses and request lab tests.

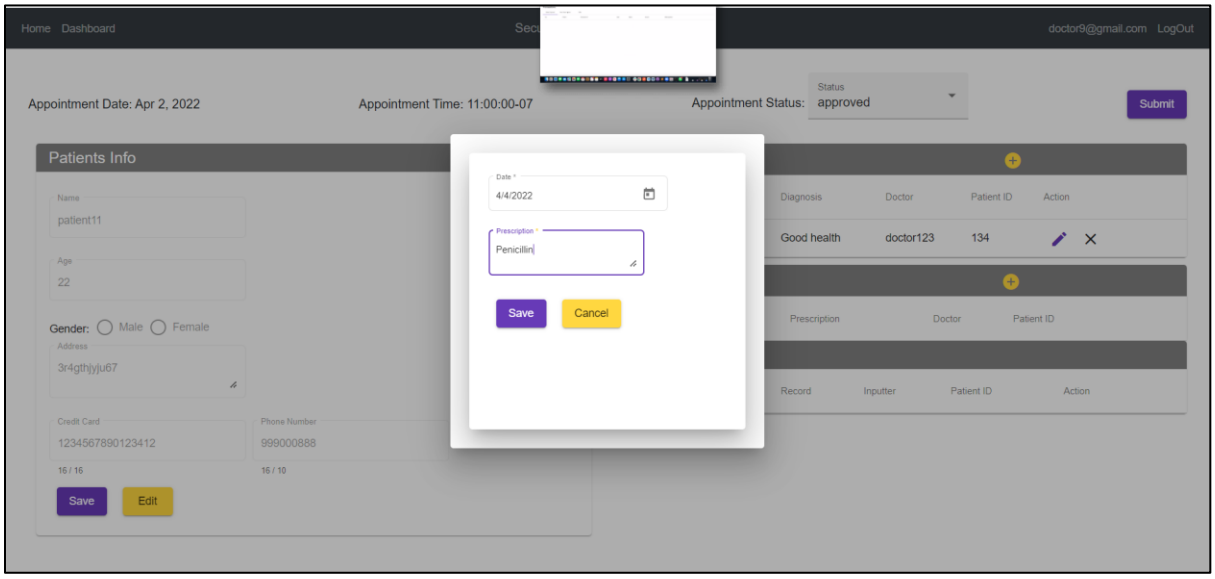


6. Doctor can edit diagnoses.

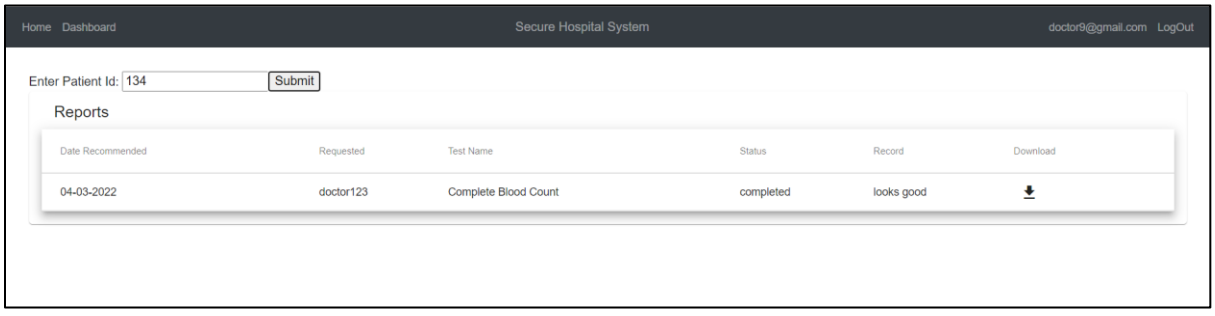




7. They can prescribe medicine.



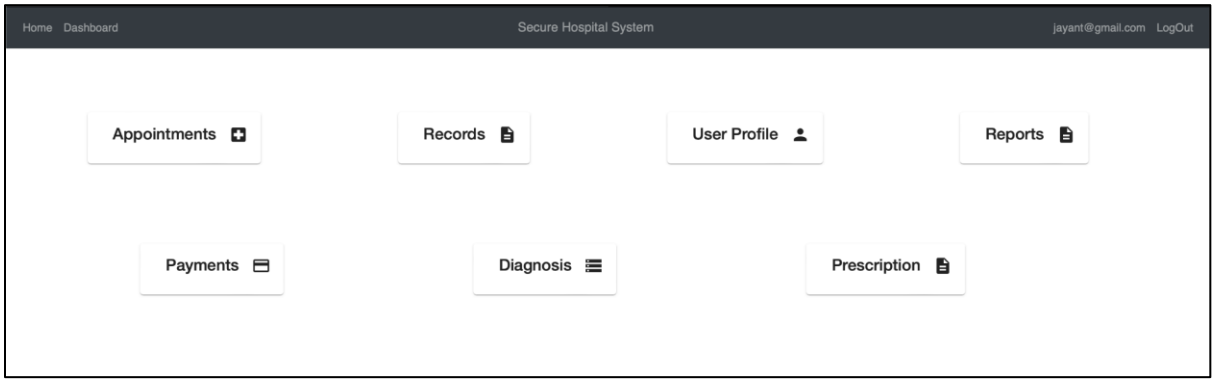
8. They can view lab test reports of the patient with the patient ID.



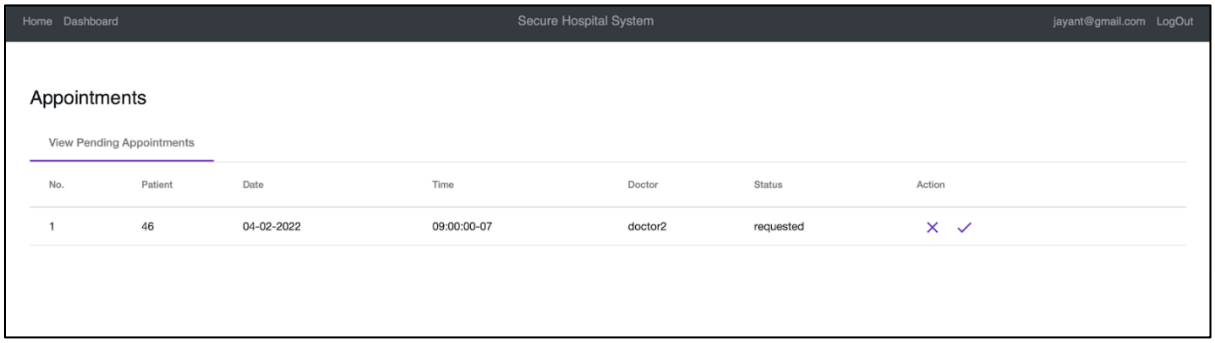
6.3 User: Hospital Staff

Login to the SHS webpage with the user role as hospital staff. Below is the functionality and navigation through the hospital staff user.

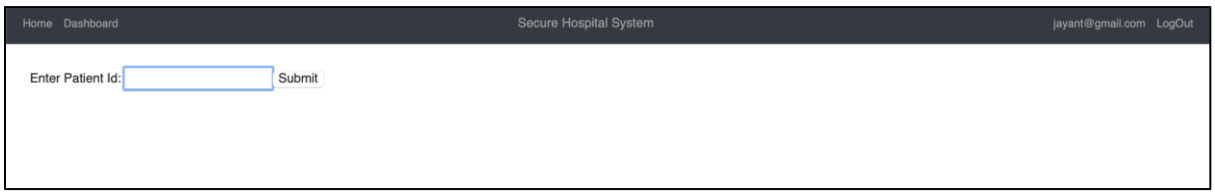
1. After logging in the hospital staff can see the following dashboard:



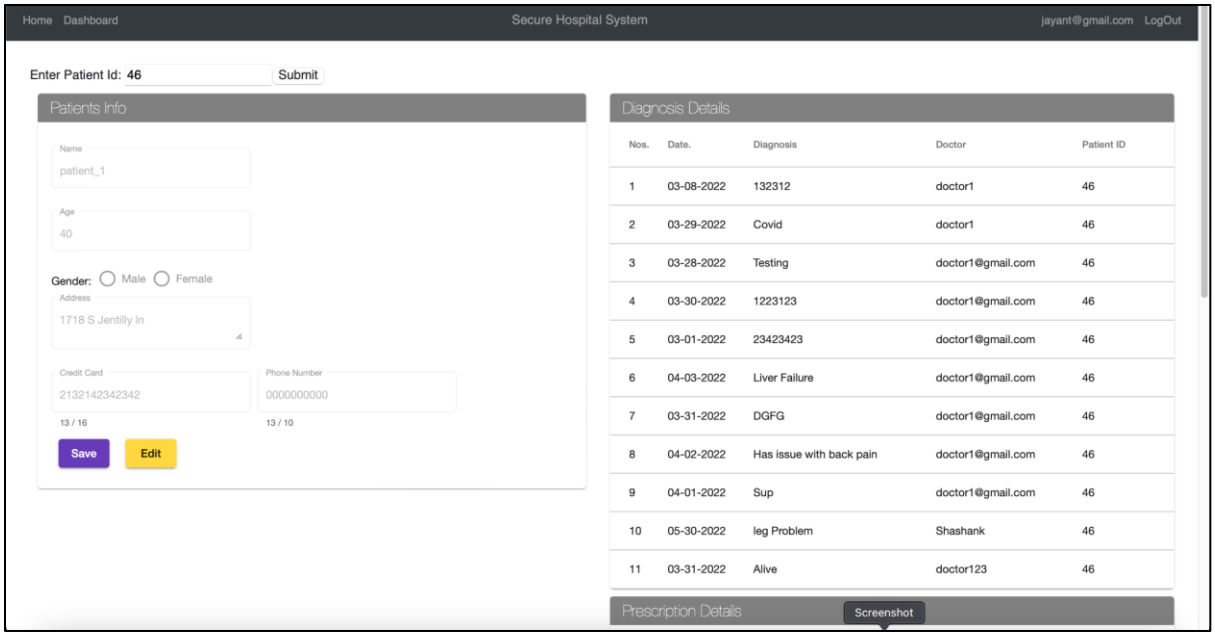
2. Now, by clicking on the appointments tab, hospital staff can review the appointments that were booked by the patients and approve them depending upon the priority and availability of the doctor.



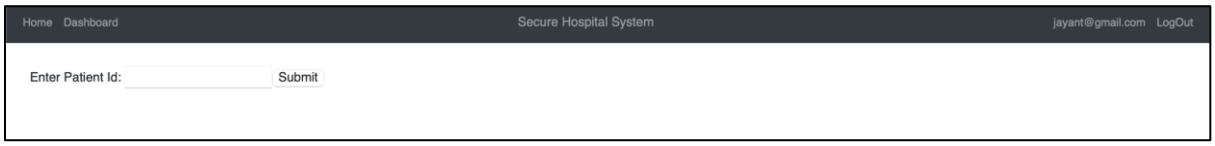
3. Now, we have a records section which requires the patient id to check or add information of a patient.



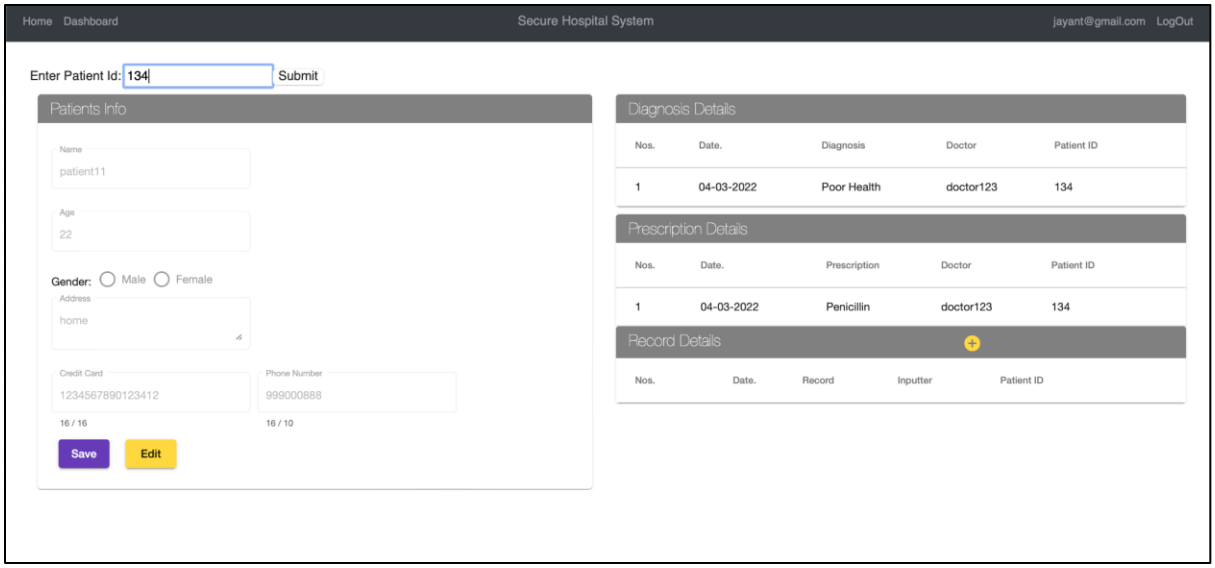
4. After the patient id is given, the hospital staff will have the option to add and update the patient details. They have the option to view the diagnosis, prescription and record details and the hospital staff will have the option to add the record details. All of the above 3 features have the same webpage to make the required changes.



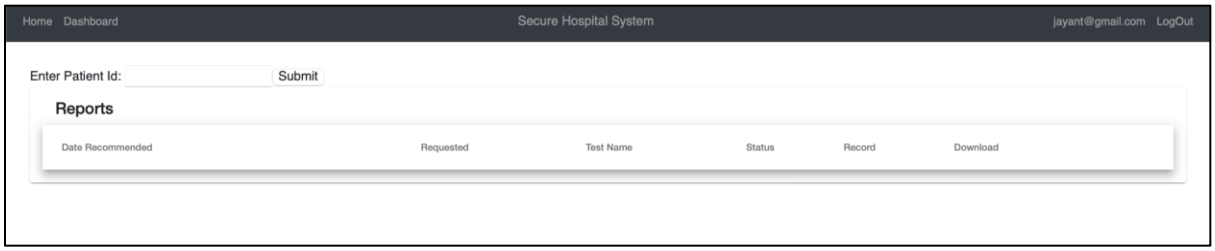
5. In the user profile section of the hospital staff, we will be asked the patient id to have the option to update the patient's personal info upon the request of the patient.



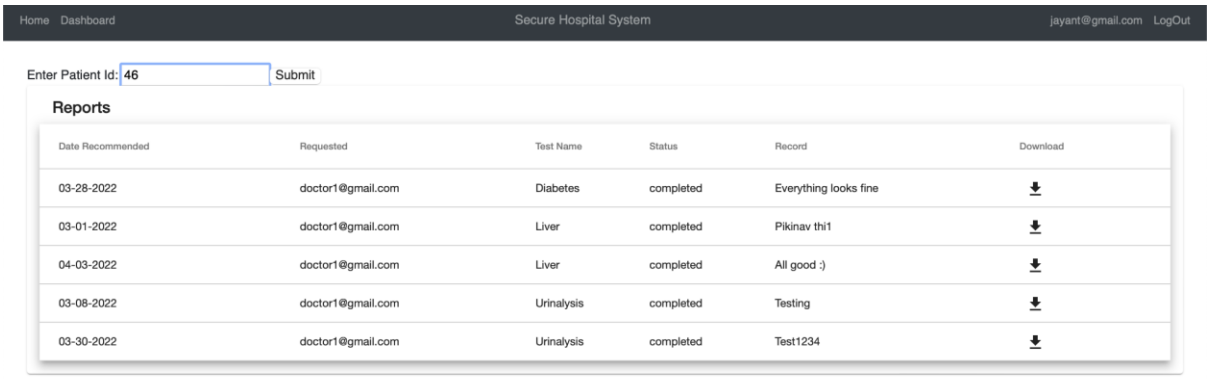
6. This page will help us to update the data of the patient upon their request.



7. Then there is a reports tab which helps to see the reports of the patient after entering the patient id.

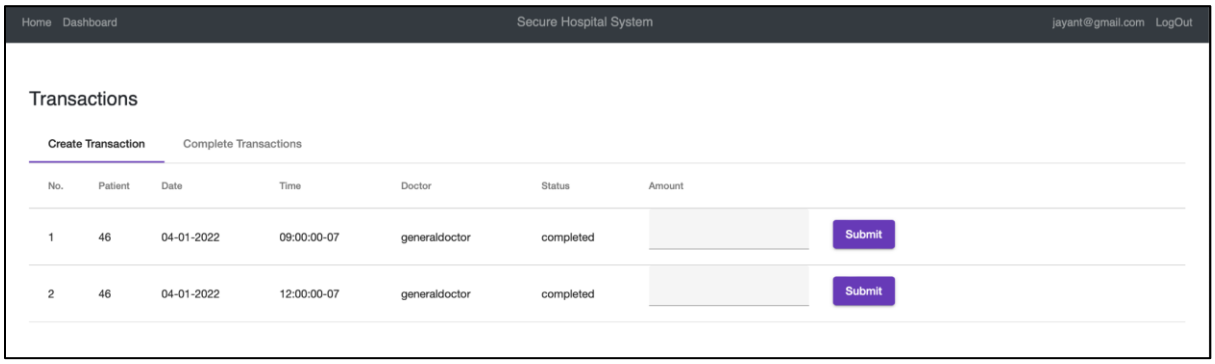


Down below is the screenshot which shows different reports concerning a patient id

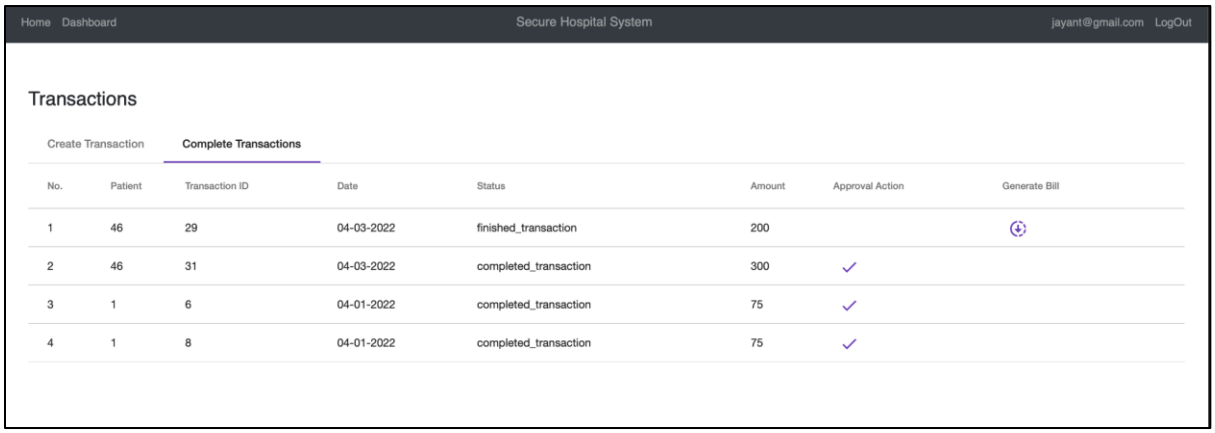


8. Now, we have a payment tab where we have two options: create transaction and complete transaction. First, we can see the create transaction tab. Here, if the

appointment with a doctor is confirmed, we can add the respective fee to consult the doctor and submit the transaction request.



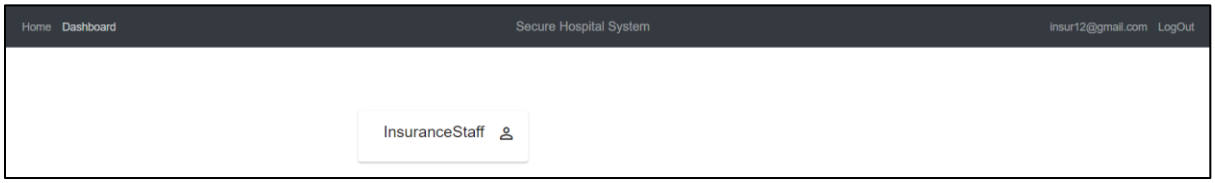
9. Now, we can look at the second tab which is the complete transaction. Here we have the option to complete the transaction upon approval of the appointment with the doctor. The bill is generated, and we will have the option to download and print the bill upon the request of the patient.



6.4 User: Insurance Staff

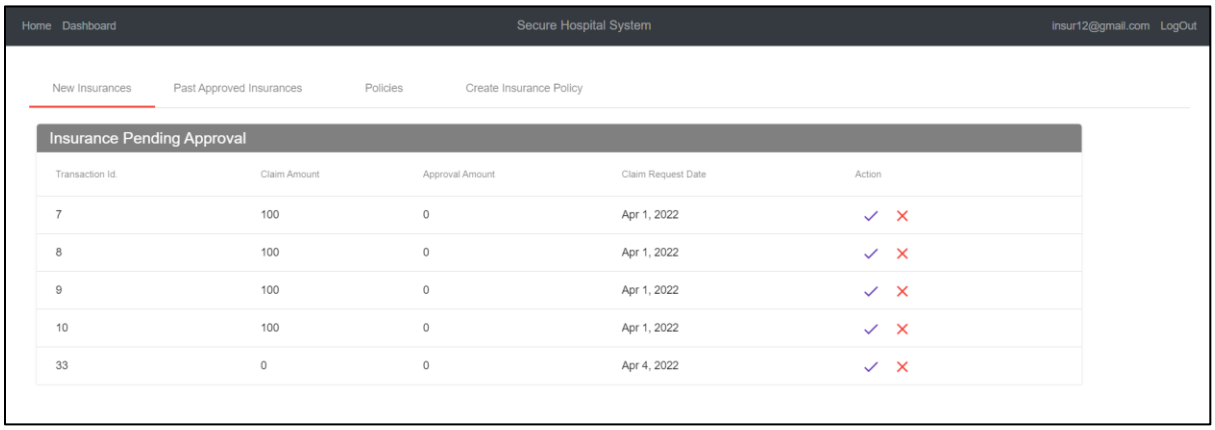
The insurance staff has the privileges to manage the Insurance Claims of the patients and also add individual Insurance Policies to any of the required patients. Insurance staff has the access to view, review and validate claim requests received from the patients. They can create new insurance policies and insurance records of patients whenever necessary. They have the power to review the insurance claims and approve/deny any insurance claim.

1. Once logged in as an Insurance Staff member, the staff member has his individual Staff bar in the dashboard, where he can perform all of his activities related to Insurance Claims and Policies.

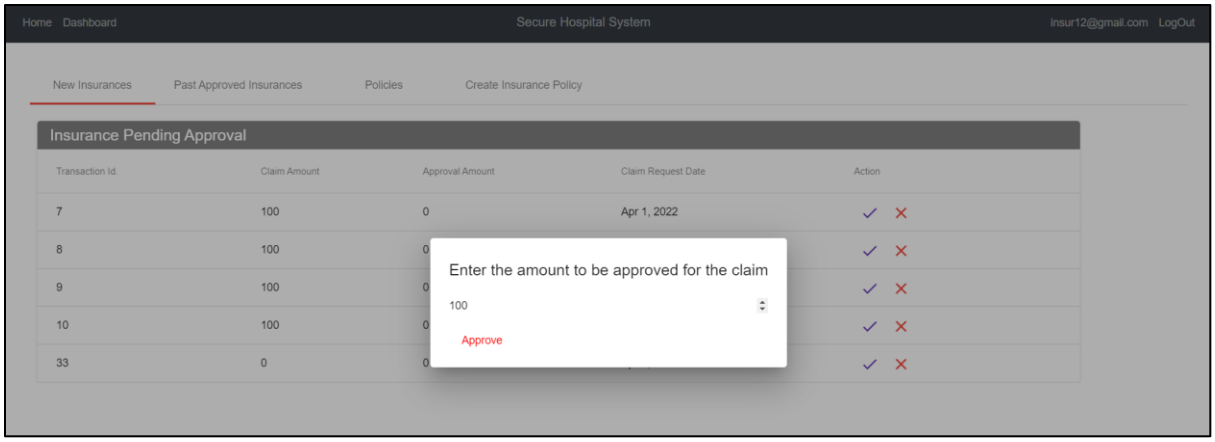


2. Once he selects the Insurance Staff Bar, they get to choose from the list of tasks they want to perform, such as reviewing new Claims, and also the ability to create new insurance policies.

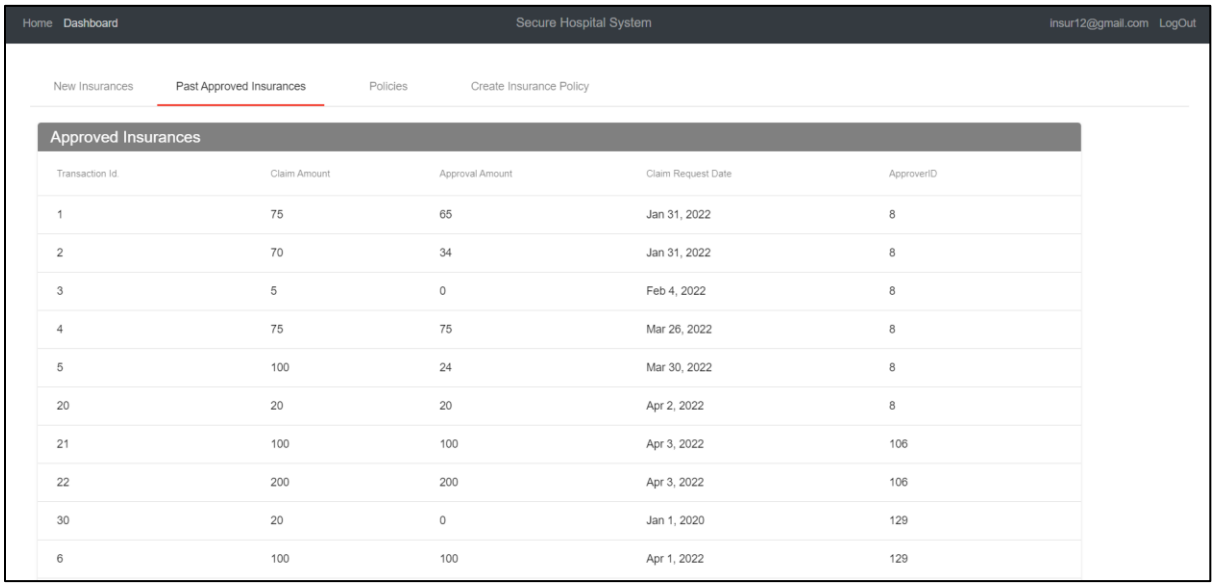
   Under the New Insurances section, they will have all the list of Insurance Claims that have been requested by various patients. The staff will be able to review and verify the Claims requested by the patients. If the staff chooses to deny the request of a particular claim, they can just click on the corresponding 'X' remove it from the list.



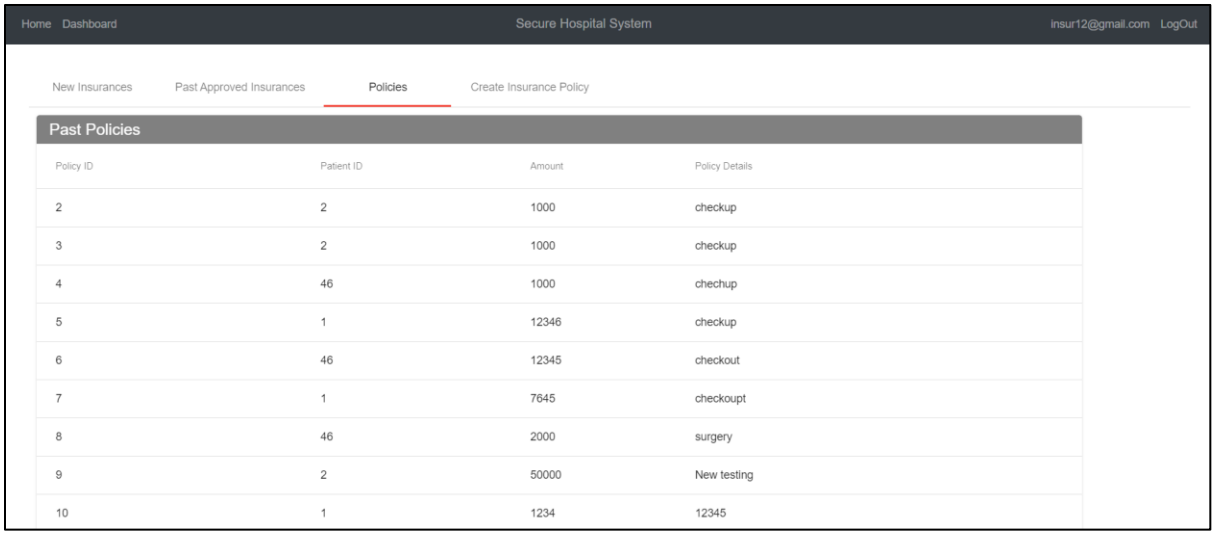
3. If the insurance staff choose to approve a particular request, they just need to click on the 'Tick' and a new window pops up where they can enter the amount they wish to approve.



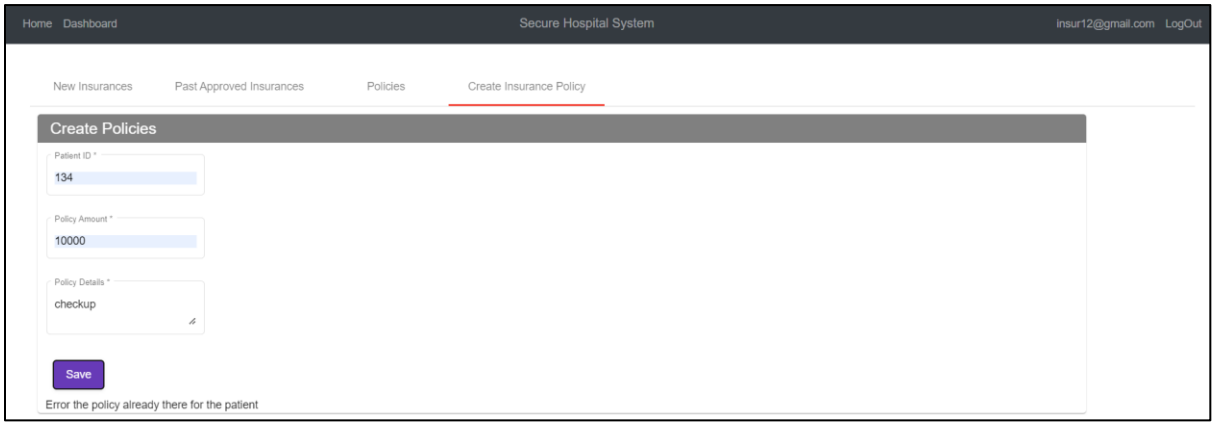
4. On approval of insurance claims, it will be moved to the Past Approved Insurances, where they will be able to access the record of all the approved insurance claims. Once the insurance staff approves/denies an insurance request, it gets reflected in the patient profile as well, so that the patient will be able to view the status of their insurance claims.



5. The Policies section contains all the records of the insurance policies associated with the patients. This lets the insurance staff an estimate of how much amount they can approve for an insurance claim.



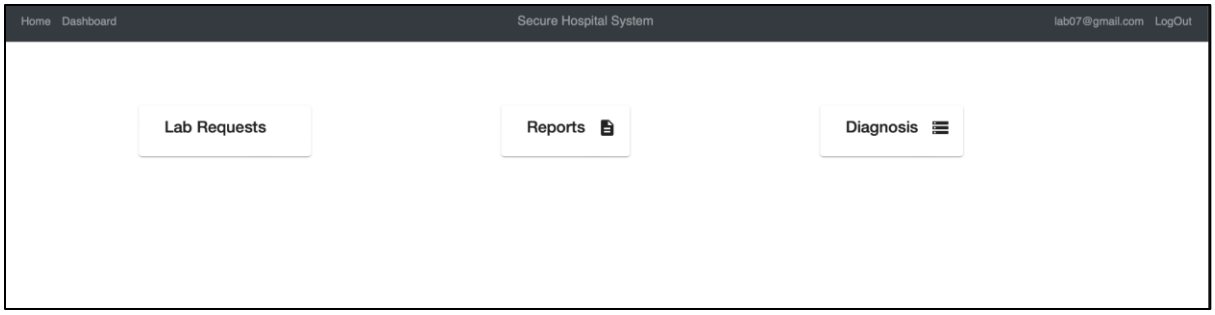
6. Under the Create Insurance Policy section, the staff will be able to create new insurance policies for patients based on their requirements and eligibility.



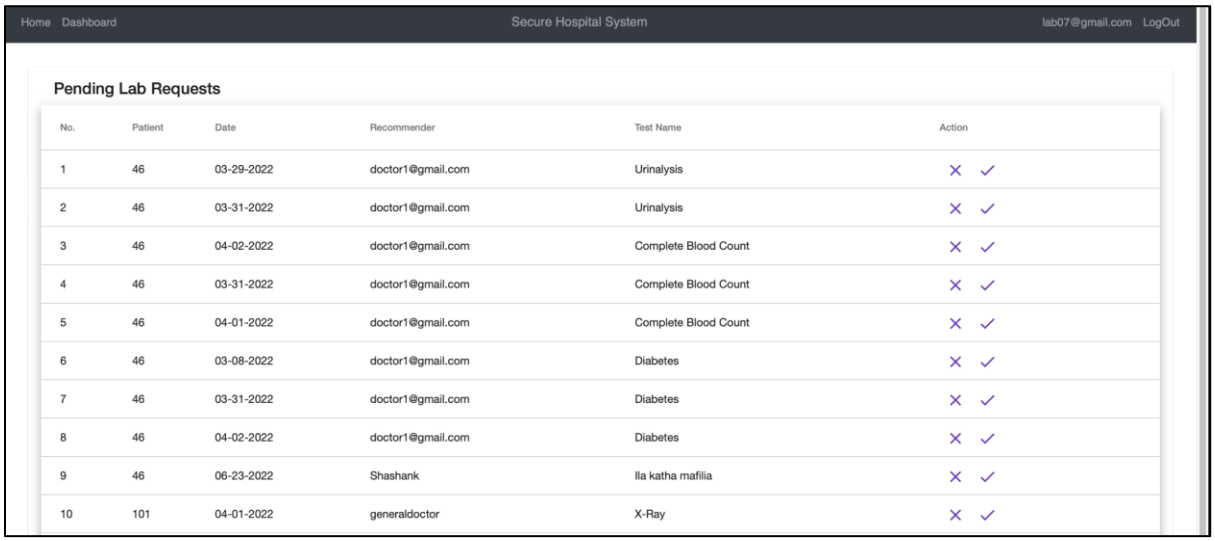
6.5 User: Lab Staff

Login to the SHS webpage with the user role as lab staff. Below is the functionality and navigation through the lab staff user.

1. After logging in as a lab staff, this is the dashboard the lab staff will have access to:

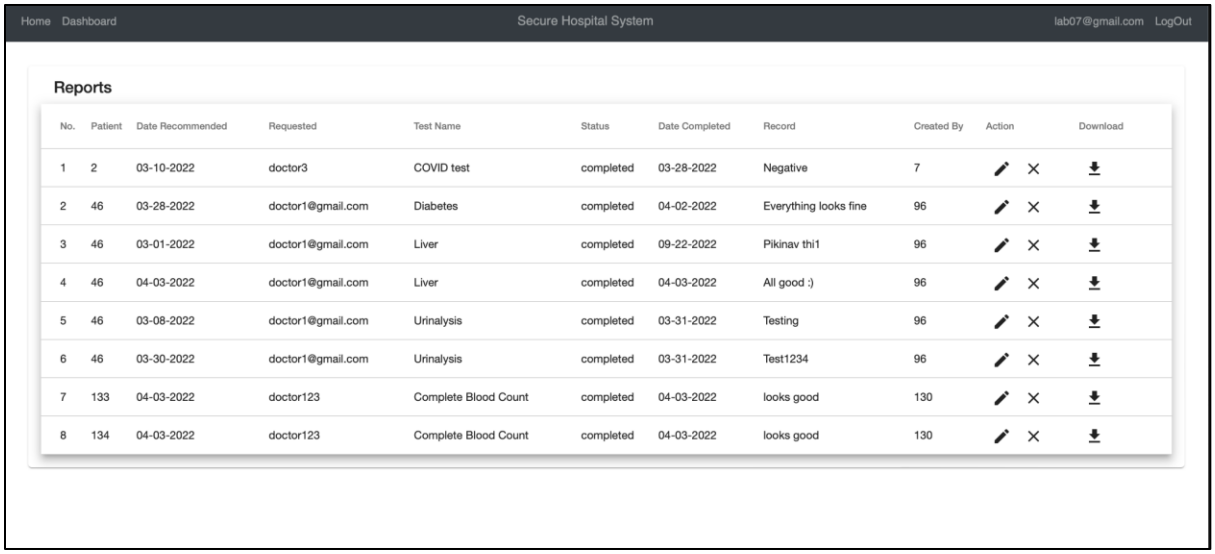


2. After going into the lab requests, we can see all the pending lab requests that were requested by the doctor. Now, the lab staff will have the option to approve those requests they get from the doctors for patients.

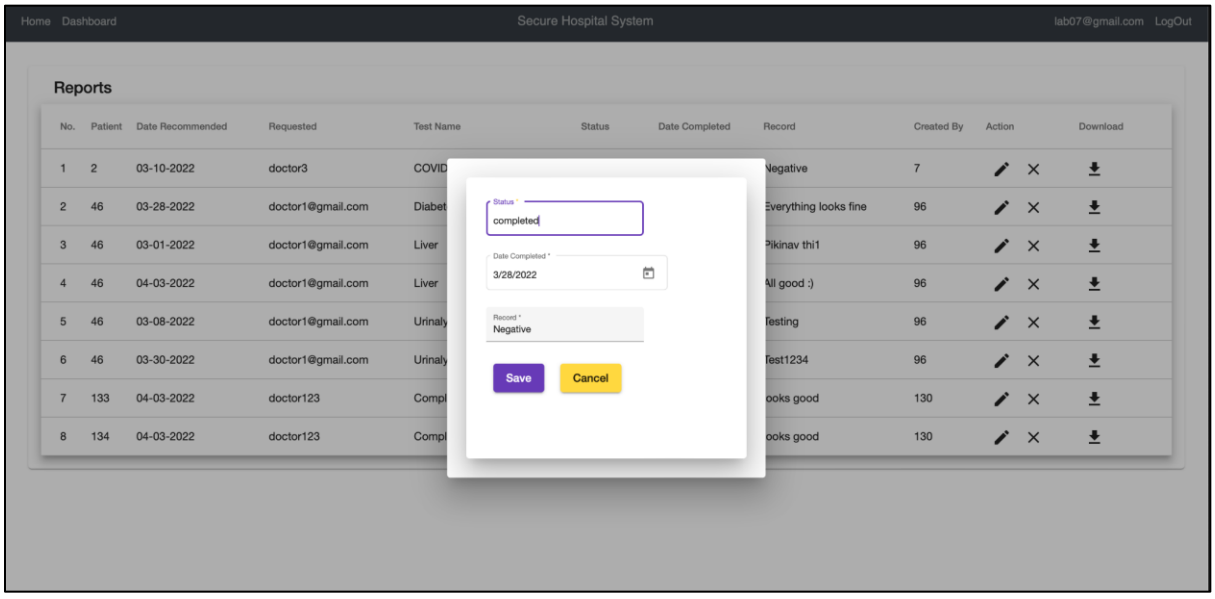


No.	Patient	Date	Recommender	Test Name	Action
1	46	03-29-2022	doctor1@gmail.com	Urinalysis	
2	46	03-31-2022	doctor1@gmail.com	Urinalysis	
3	46	04-02-2022	doctor1@gmail.com	Complete Blood Count	
4	46	03-31-2022	doctor1@gmail.com	Complete Blood Count	
5	46	04-01-2022	doctor1@gmail.com	Complete Blood Count	
6	46	03-08-2022	doctor1@gmail.com	Diabetes	
7	46	03-31-2022	doctor1@gmail.com	Diabetes	
8	46	04-02-2022	doctor1@gmail.com	Diabetes	
9	46	06-23-2022	Shashank	Ila katha mafilia	
10	101	04-01-2022	generaldoctor	X-Ray	

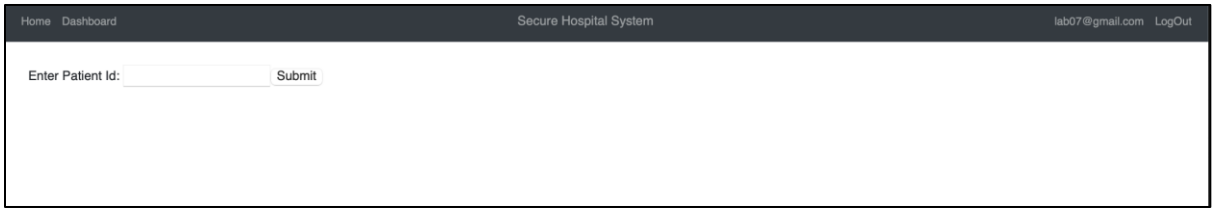
3. After the lab staff approves the pending requests, it is moved to the report section of the lab staff, where the lab staff have to fill in the details depending on the status of the lab test that is performed. Down below is the report section of the lab staff where they can edit and update the status of the test. Once the report is finalized, the lab staff will have the option to download the report.



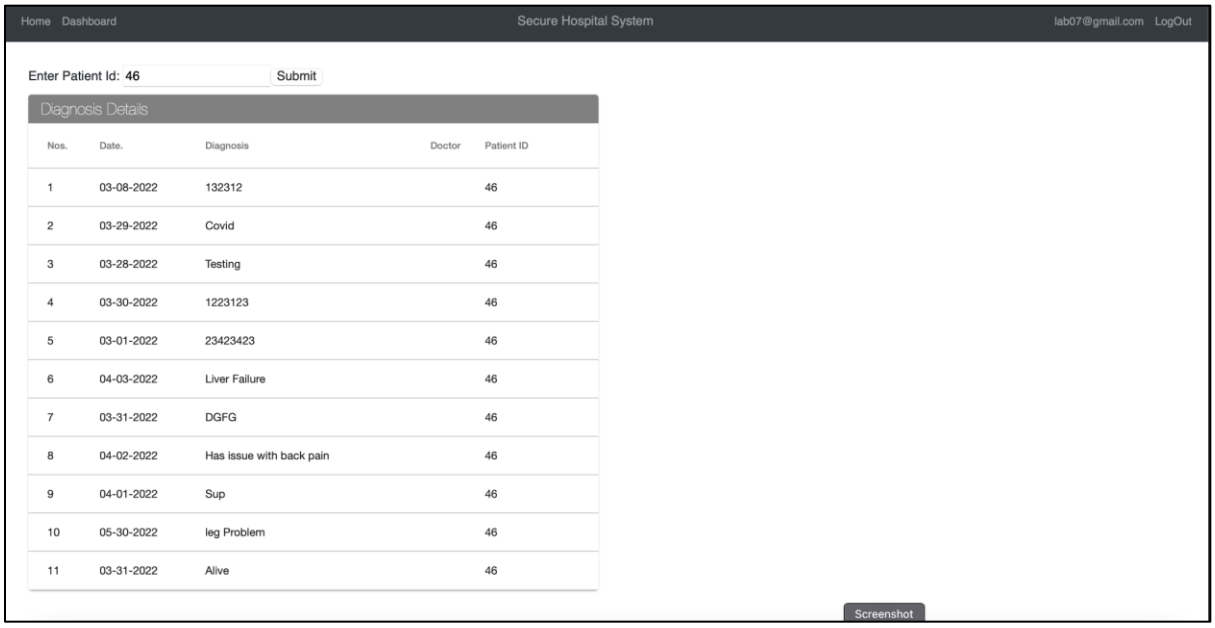
4. In the action bar of the report, we have three sections which is status, date and the record which can be edited by the lab staff whenever possible.



5. Another functionality of the lab staff is diagnosis, where the lab staff has the option to view the patient diagnosis. After, clicking the diagnosis tab in the dashboard we will be asked for a patient id.



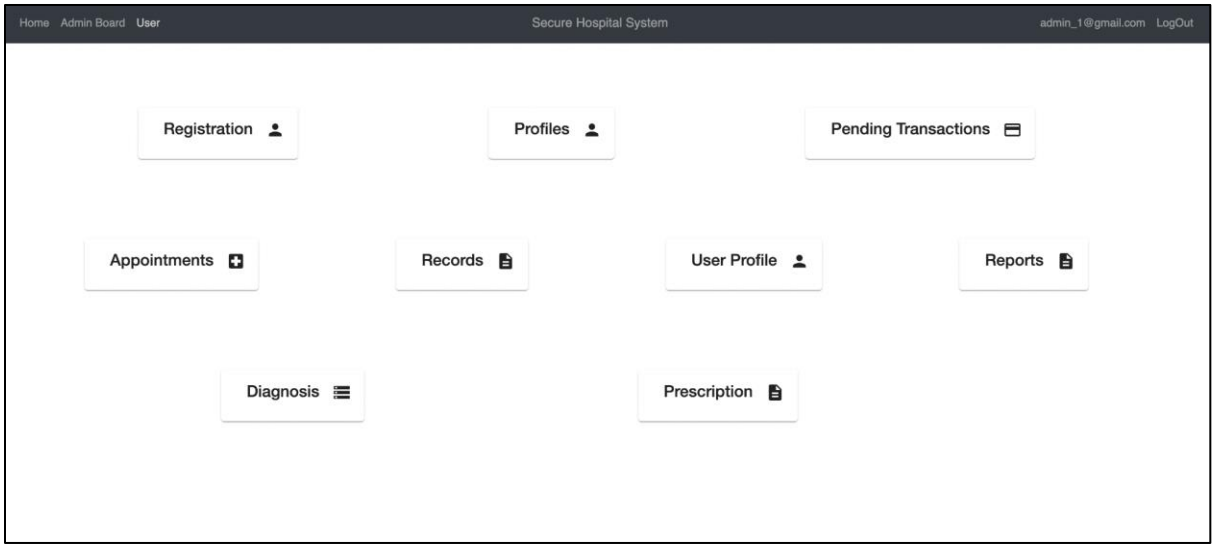
6. After entering the patient id, the diagnosis data of the patient given by the doctor is available for the lab staff for viewing.



6.6 User: Admin

The administrator role has all the highest privileges on the application, as he can create, edit, and delete any records, must authorize transactions, and has access to all log files. The admin is responsible for the smooth functioning of the hospital system.

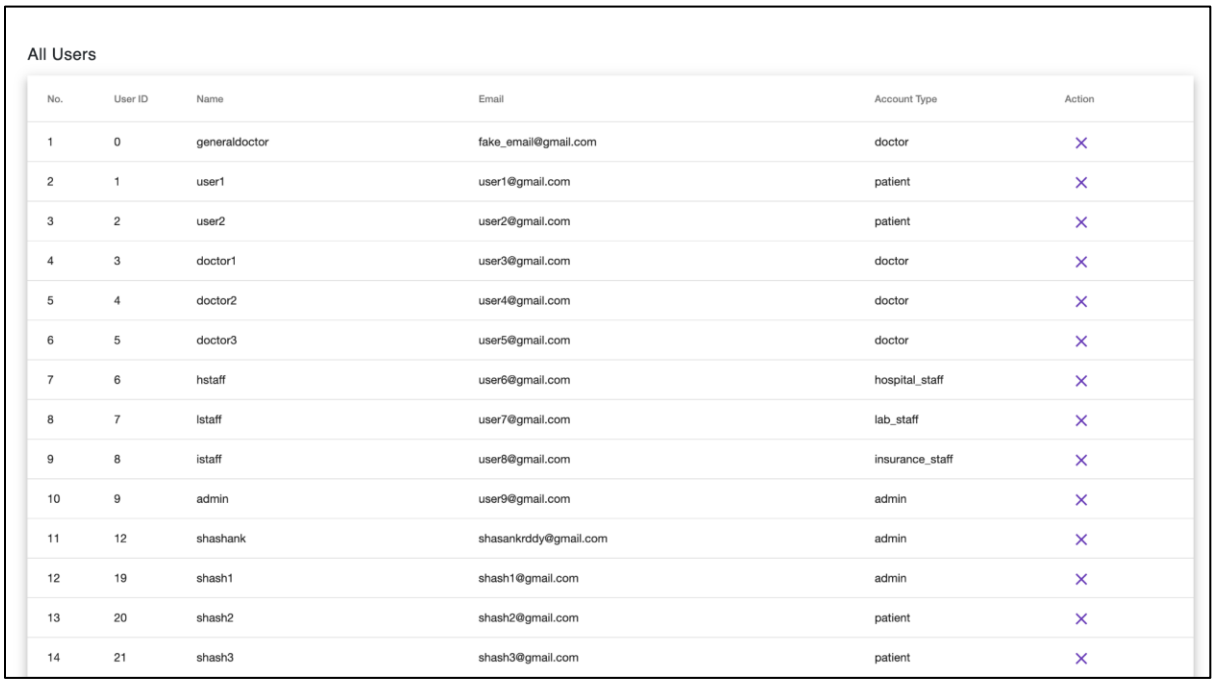
1. After logging in to his profile a dashboard as below is displayed, where he can navigate to the respective pages.



2. When the admin navigates to the “Registration” page, he will be able to register a employees of the hospital like doctor, lab staff, insurance staff, hospital staff by providing their details

3. Admin can edit the profiles of the employees by navigating to the profiles page

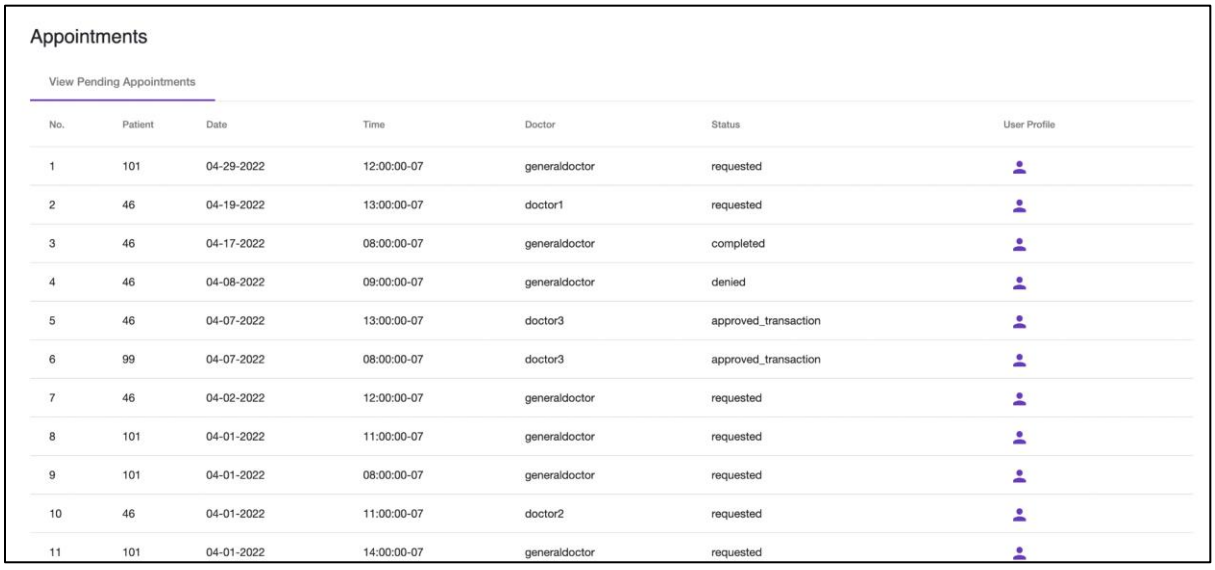
4. Admin should approve transactions by navigating to the transactions page



All Users

No.	User ID	Name	Email	Account Type	Action
1	0	generaldoctor	fake_email@gmail.com	doctor	×
2	1	user1	user1@gmail.com	patient	×
3	2	user2	user2@gmail.com	patient	×
4	3	doctor1	user3@gmail.com	doctor	×
5	4	doctor2	user4@gmail.com	doctor	×
6	5	doctor3	user5@gmail.com	doctor	×
7	6	hstaff	user6@gmail.com	hospital_staff	×
8	7	lstaff	user7@gmail.com	lab_staff	×
9	8	istaff	user8@gmail.com	insurance_staff	×
10	9	admin	user9@gmail.com	admin	×
11	12	shashank	shasankrddy@gmail.com	admin	×
12	19	shash1	shash1@gmail.com	admin	×
13	20	shash2	shash2@gmail.com	patient	×
14	21	shash3	shash3@gmail.com	patient	×

5. The admin can act on appointments by navigating to the “appointments” page

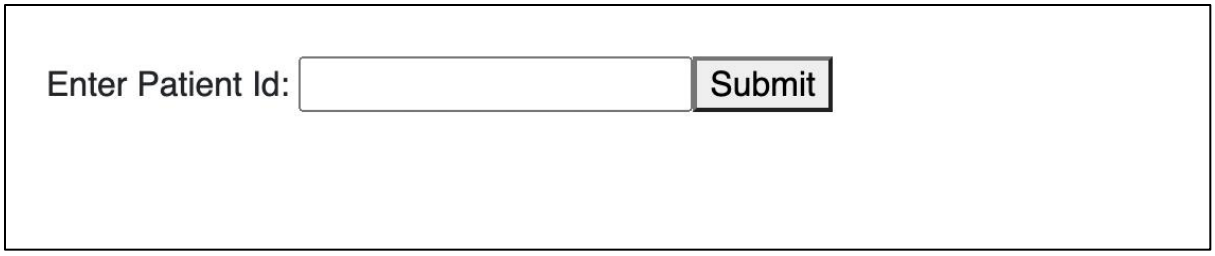


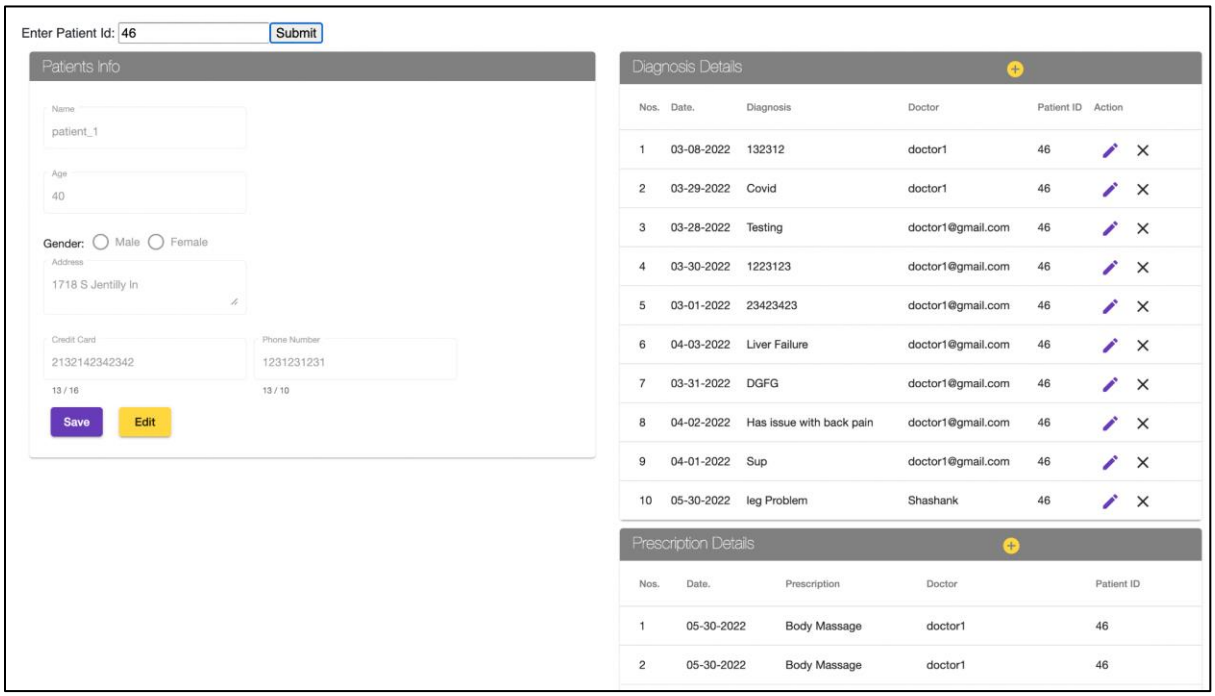
Appointments

View Pending Appointments

No.	Patient	Date	Time	Doctor	Status	User Profile
1	101	04-29-2022	12:00:00-07	generaldoctor	requested	
2	46	04-19-2022	13:00:00-07	doctor1	requested	
3	46	04-17-2022	08:00:00-07	generaldoctor	completed	
4	46	04-08-2022	09:00:00-07	generaldoctor	denied	
5	46	04-07-2022	13:00:00-07	doctor3	approved_transaction	
6	99	04-07-2022	08:00:00-07	doctor3	approved_transaction	
7	46	04-02-2022	12:00:00-07	generaldoctor	requested	
8	101	04-01-2022	11:00:00-07	generaldoctor	requested	
9	101	04-01-2022	08:00:00-07	generaldoctor	requested	
10	46	04-01-2022	11:00:00-07	doctor2	requested	
11	101	04-01-2022	14:00:00-07	generaldoctor	requested	

6. The admin can access patient profiles and update each of their prescriptions, diagnosis, and records by providing the patient’s id





7 Non-Functional Design

7.1 Security Design

7.1.1 Secure Login with token and session management

Secure login for any user is done in the two following ways. First method is the JWT authentication token. As depicted in the figure below, a new user signs up for an account and the database checks for an existing account for validation. Once, the account is created and the user signs in with his/her credentials, we generate an JWT token (string with a secret) and return it. This token must always be sent with a request from the front-end to access the user data. The access to other data/pages is restricted for that particular token.

The second method is two-factor authentication, where we are going to implement Two Factor Authentication functionality with a Soft Token and Spring Security. We are going to be adding the new functionality into an existing, simple login flow and use the Google Authenticator app to generate the tokens. To use Google Authenticator in our app we need to generate a secret key and provide secret key to the user via QR-code and verify token entered by the user using this secret key. And so, users provide an extra “verification token” during authentication – a one-time password verification code based on Time-based One-time Password TOTP algorithm. We will use a simple server-side library to generate/verify one-time password by adding the certain dependencies to the app start-up xml.

2FA is important for web security because it immediately eliminates the risks associated with compromised passwords. If a password is hacked, guessed, or phished, it is no longer sufficient to allow access to an intruder: the password is useless without the permission of the second factor.

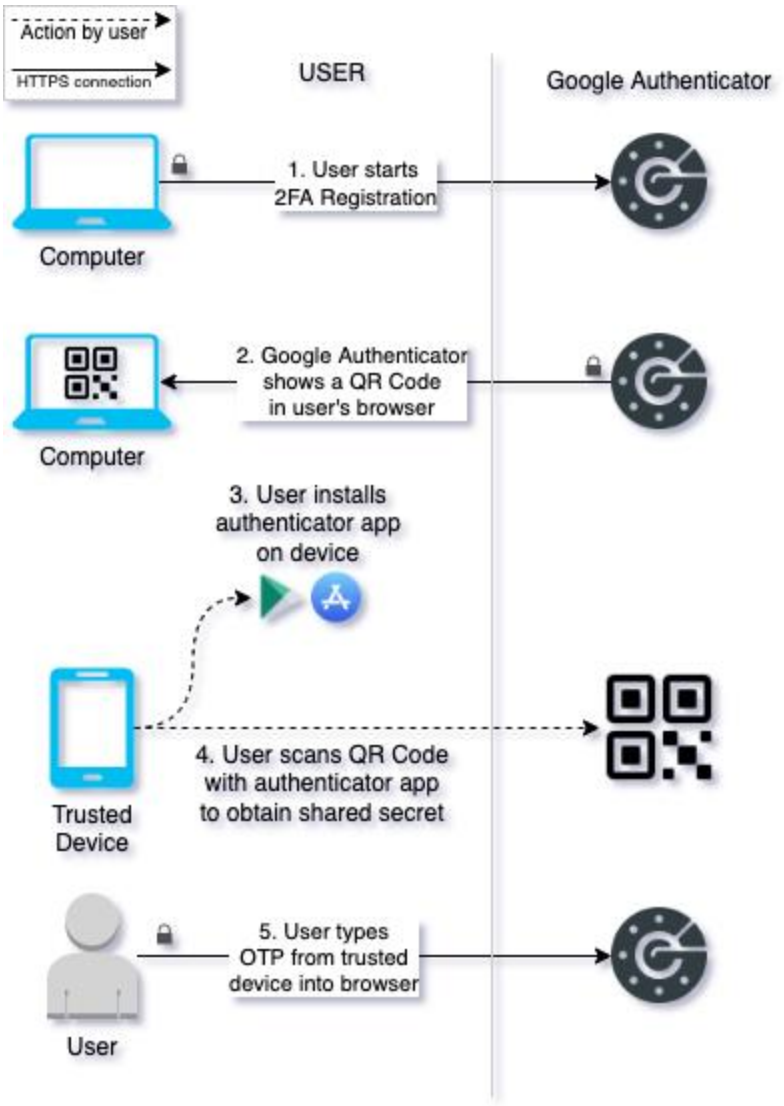


Fig. Two-factor Authentication - Security Workflow

We will be implementing the two-factor authentication using the Google Authenticator app. We hosted the node is server on Amazon Web Services for implementing the two-factor authenticator. We have used speakeasy package in the node is server which helps in validating and verifying the code which we enter on the website with the code generated in the Google Authenticator on our trusted device. In order for the google authenticator on the device to be linked with the user credentials on the website, we use a QR code generated using qrcode package in the node is server which has the token generated for a specific user, thus linking the app to the user credentials on the website.

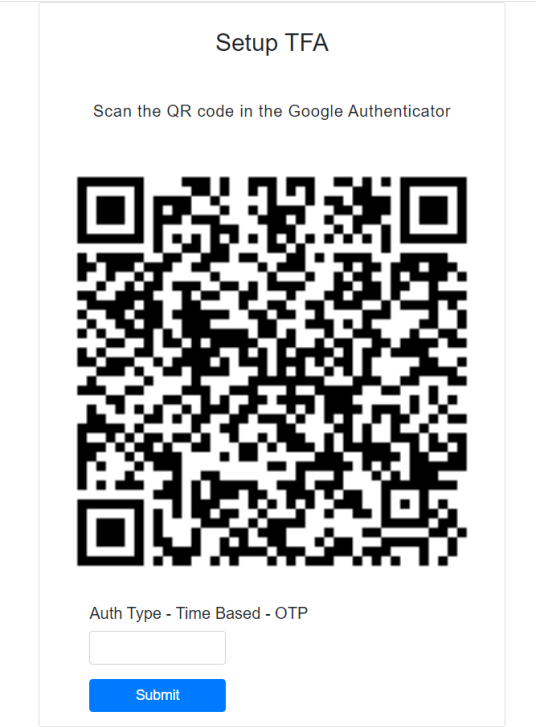


Fig. QR Code - Link Google Authenticator

The Google authenticator generates the new codes every 30 seconds. The user needs to enter the code generated in the authenticator app into the OTP field of the website before the code expires on the authenticator app. The speakeasy package on the node is server validates this 6-digit code and lets the user successfully login.

Fig. Google Authenticator - 6-digit Code

All sign-in history will be logged into the session table. The session expiry is checked for each transaction request from the users.

Once the user clicks on the Sign-In page, it redirects to the Login page where the patient needs to enter the username and password and then solve the captcha. Here we implemented the ‘I am not a robot’ captcha to make sure that bots are not overloading the website’s server and make it more secure.

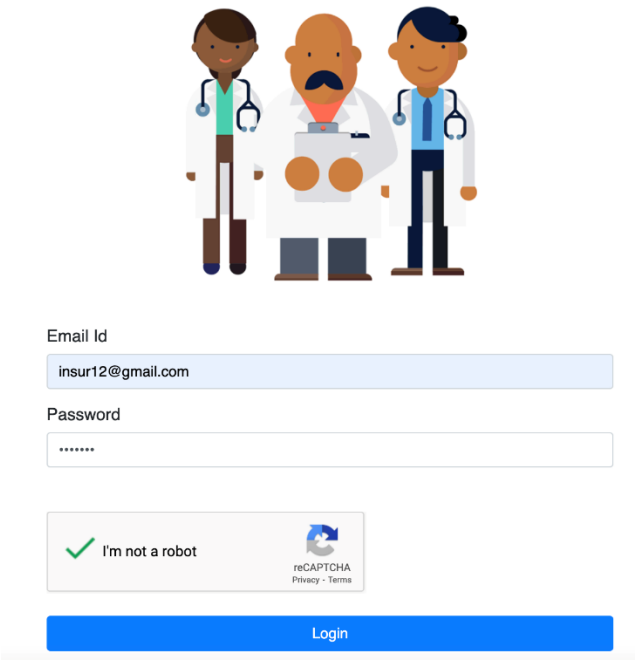


Fig. Login Page - with Captcha

Once the user clicks on the Login button, they will be redirected to a new page where he has to enter the 6-digit code from the Google Authenticator which is valid only for 30 seconds. We have given an option for the user to use the virtual keyboard or the traditional laptop keyboard. We have implemented this functionality because if the user wants a more secure way of handling things, the virtual keyboard will be an efficient tool. It prevents the attackers phishing the key loggings of the user and thus ensuring a secure connection and experience. Using the virtual keyboard, the user can enter the 6-digit code from the authenticator app. If the user enters the correct code within the 30 seconds timeframe, we redirect him to his personal dashboard. And if the code has to fail, we request him to try entering the valid code within the 30 seconds timeframe.

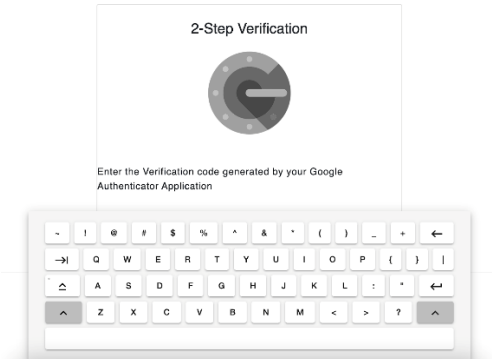


Fig. 2FA - 6-digit code using virtual keyboard

Logged in as ROLE_INSURANCESTAFF.

Fig. Login Successful - Redirected to user dashboard

7.1.2 Malicious Login Control

While many solutions are available for malicious login controls that are attempted by hackers, like Rainbow table attacks, brute force attacks, Dictionary attacks etc., adding ML to the solution takes a different approach. We employ a semi-supervised approach to group together malicious Ips. For each IP, we use the labels we must find features that are helpful in separating the good IPs from the bad ones. We tune the features and the parameters of our clustering algorithm by looking at how effectively the clusters separate the know good labels from the known bad labels. We select about ten features and then use the DBSCAN clustering algorithm (refer to figure below) to find the clusters of IPS.

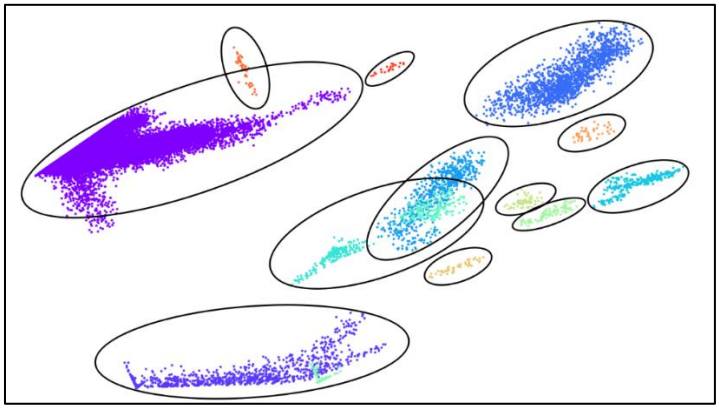
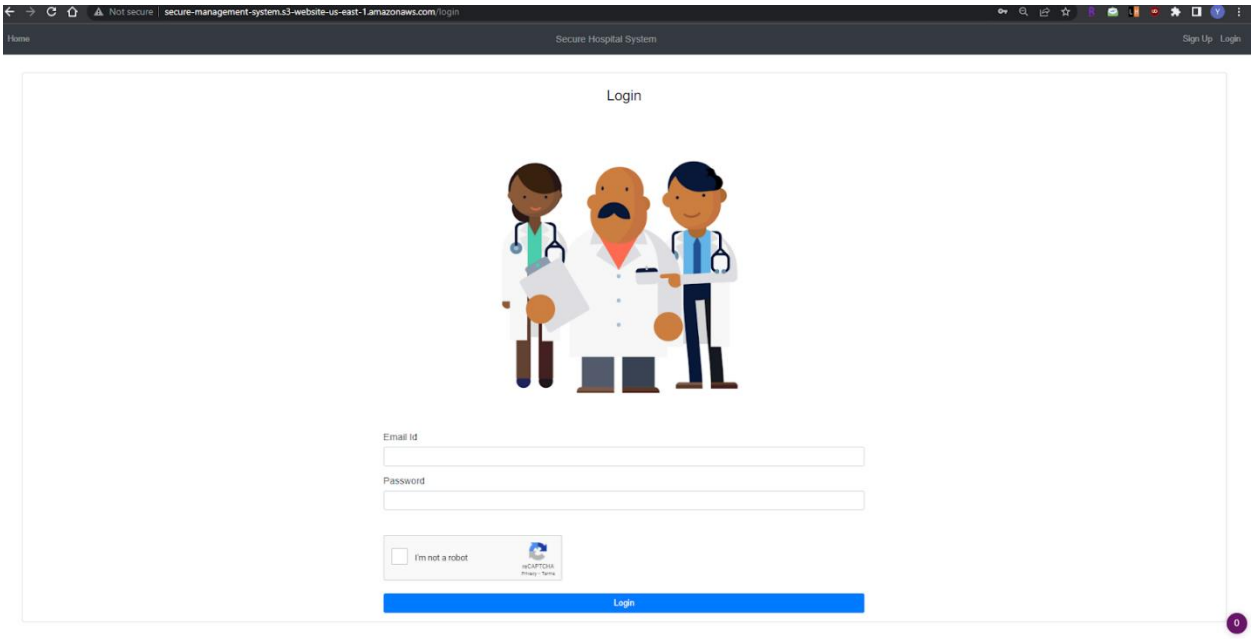
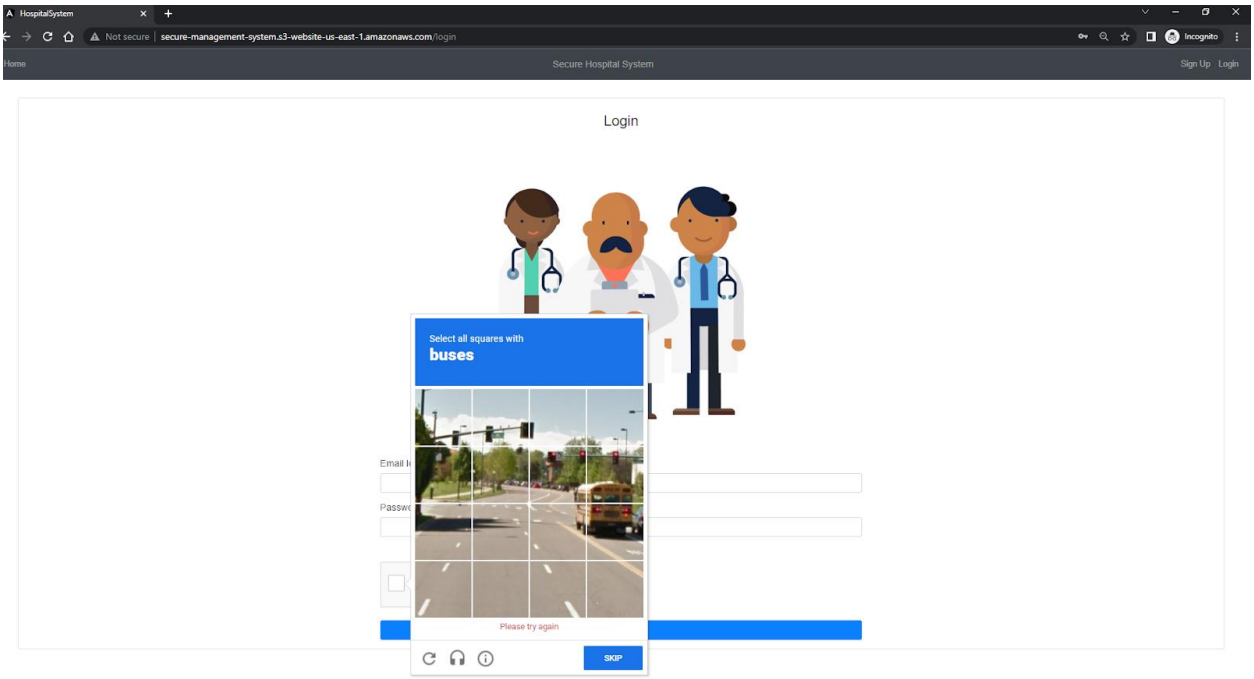


Fig.25 Malicious login pattern recognition

We have also included the reCAPTCHA Verification mechanism for malicious login. The reCAPTCHA Verification mechanism can provide protection against spam or abuse caused by robots. With this mechanism, the user is presented with a web page that contains a simple Turing test provided by the Google reCAPTCHA API. These tests can distinguish a human user from a robot. It helps to prevent robots from accessing your applications.

Below is a screenshot depicting the use of it:

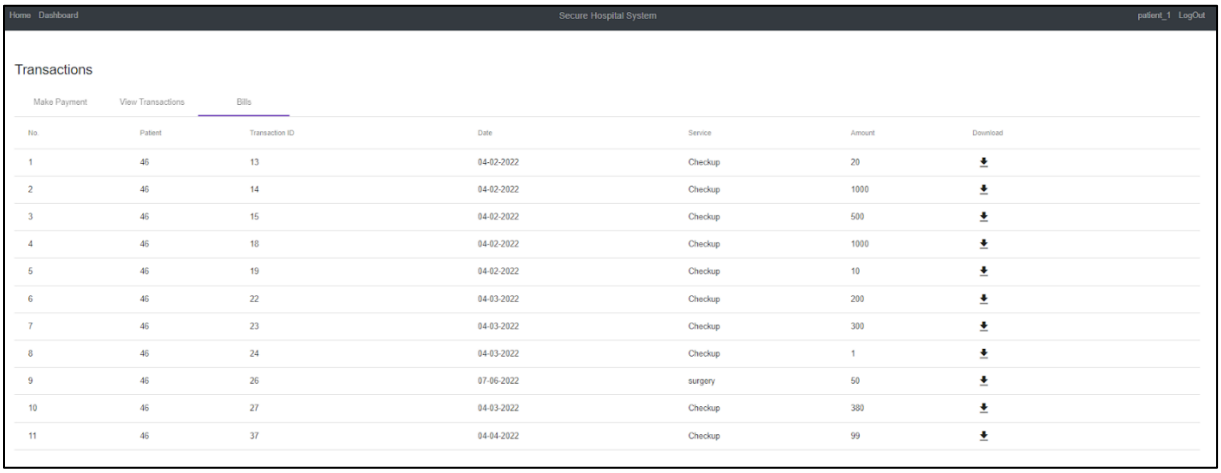


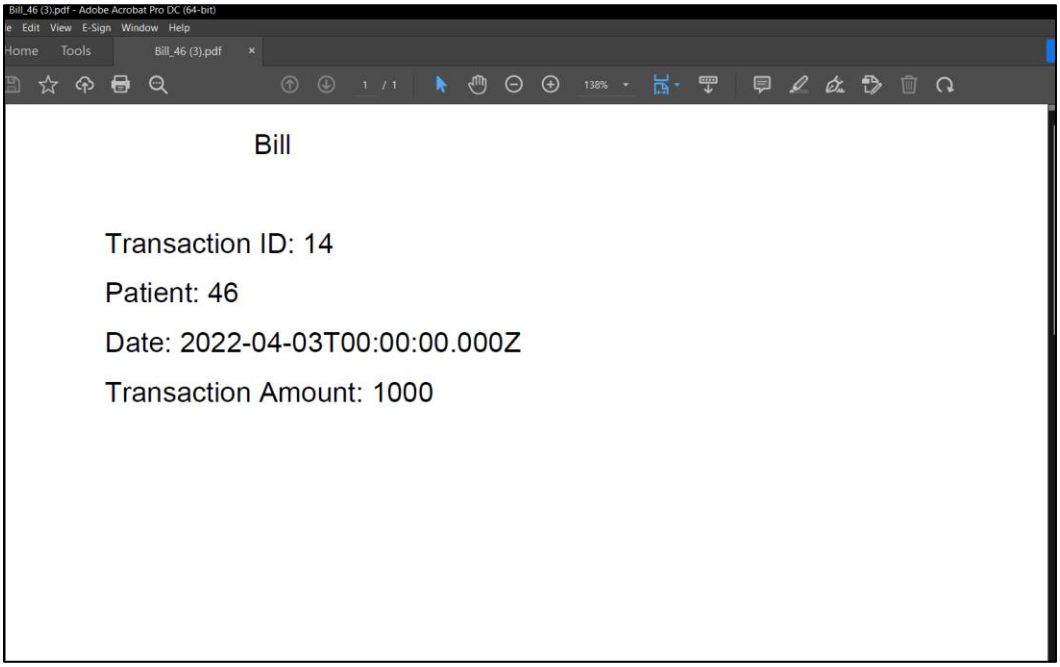


7.1.3 Transaction History Management

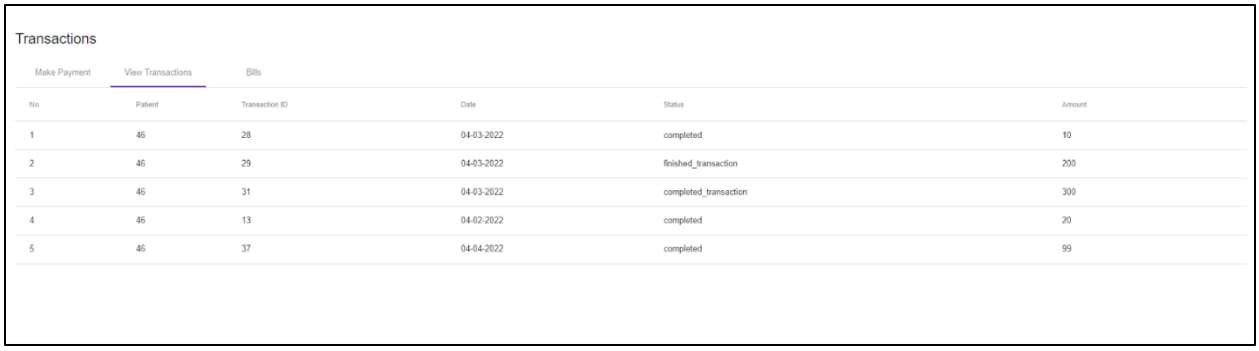
In the database backend, we have created an Enum to indicate the status of a transaction. Transactions are generated from completed appointments by hospital staff, after which they must be approved by administrators before coming to the patient. The patient then has the options to pay by themselves or use their insurance. If the patient makes an insurance claim, then the insurance staff must approve this claim before the transaction completes. Finally, the hospital

staff can generate a bill for this completed transaction. We have created a transaction table with unique transaction id and user id to store the history of the user transaction . The user can also download his bills from the table as shown below.





The user can also view his older transactions along with the status check as shown below:



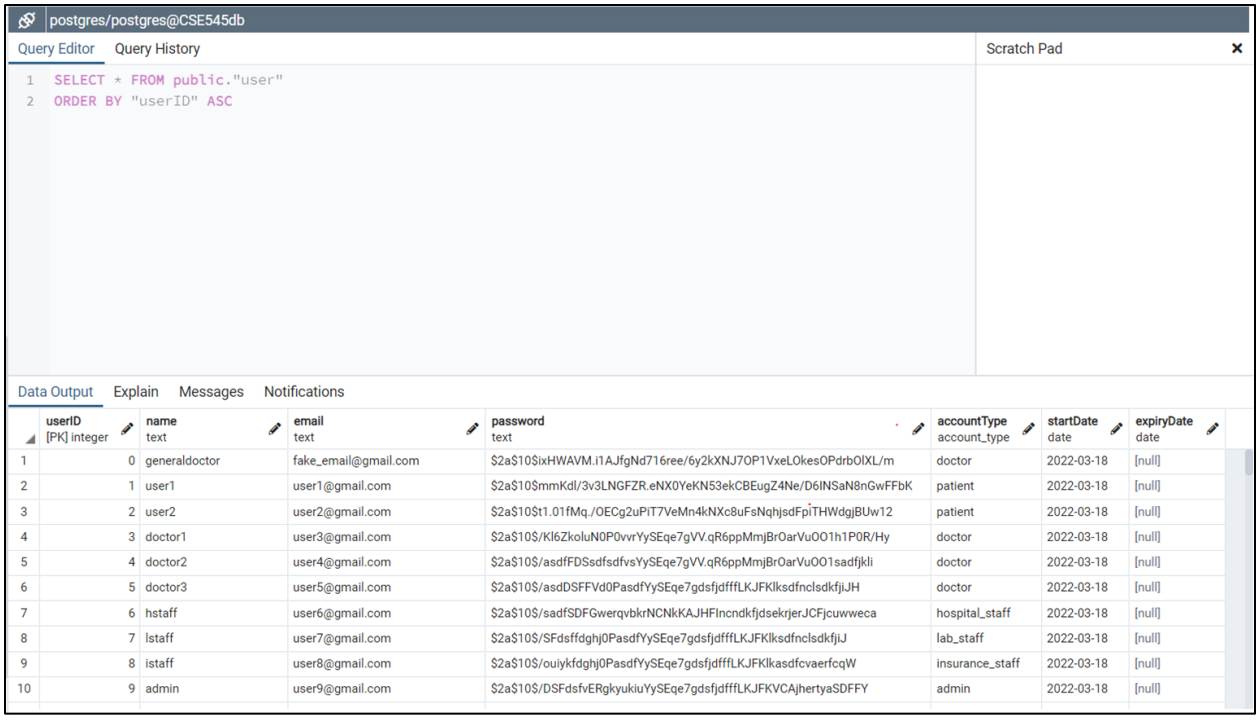
7.1.4 Data Masking/Hashing

To oversee hashing and encryption at rest, we will be using PostgreSQL's pgcrypto module. Effectively, this module provides hashing and encryption functions for use in queries. Data in transit is not encrypted by this module, so communications should be over SSL.

Hashing should be used over encryption when the original value is not necessary and only confirmation that the value to be hashed is equivalent to the original value is necessary - in this case, the hashing algorithm must be slow, to increase the brute forcing work required to crack the hash. The most common case of this is passwords. A second use case is verification - in the case of verification, the hash only serves to verify that the value has not changed from the original value. In this case, the hashing algorithm need not be slow. However, now, we do not have any verification use cases in the database for this project, only in the blockchain.

Encryption, on the other hand, will be used when the original value is necessary. While pgcrypto offers both asymmetric and symmetric encryption functions, we will be using symmetric encryption because all the encrypted data stored in the database must be accessed by the backend, and only the backend - there is no two-party communication involved. Thus, asymmetric encryption does not offer benefits. Meanwhile, symmetric encryption is faster, which is why we will be using it. Patient medical records, diagnoses, prescriptions, and lab test reports will be encrypted, as well as any financial data we store like credit card numbers.

Finally, for data masking, we will be managing it in the backend. Due to the limitations of PostgreSQL's dynamic data masking module, PostgreSQL Anonymizer, it can only mask data on a per-user basis. Since the backend must access all database data, unless a corresponding database user is made for each individual application user, no data can be masked in the database unless the data's complete, original value is not necessary. Thus, we plan to send unmasked data to the backend and have the backend return masked data to the frontend.



	userID [PK] integer	name text	email text	password text	accountType account_type	startDate date	expiryDate date
1	0	generaldoctor	fake_email@gmail.com	$2a$10$ixHWAVM.i1AJfgNd716ree/6y2kXNJ7OP1VxeLOkesOPdrbOlXL/m	doctor	2022-03-18	[null]
2	1	user1	user1@gmail.com	$2a$10$mmKdl/3v3LNGFZR.eNX0YeKN53ekCBEugZ4Ne/D6INSaN8nGwFFbK	patient	2022-03-18	[null]
3	2	user2	user2@gmail.com	$2a$10$t1.01fMq./OECg2uPiT7VeMn4kNXc8uFsNqhjsdFpiTHWdgjBUw12	patient	2022-03-18	[null]
4	3	doctor1	user3@gmail.com	$2a$10$/Kl6ZkoluN0P0vvrYySEqe7gVV.qR6ppMmjBrOarVuOO1h1P0R/Hy	doctor	2022-03-18	[null]
5	4	doctor2	user4@gmail.com	$2a$10$/asdfFDSsdfsdfvsYySEqe7gVV.qR6ppMmjBrOarVuOO1sadfjkli	doctor	2022-03-18	[null]
6	5	doctor3	user5@gmail.com	$2a$10$/asdDSFFVd0PasdfYySEqe7gdsfjdfffLKJFKlksdfnclsdkfjiJH	doctor	2022-03-18	[null]
7	6	hstaff	user6@gmail.com	$2a$10$/sadfSDFGwerqvbkrNCNkKAJHFIncndkfjdsekrjerJCFjcuwweca	hospital_staff	2022-03-18	[null]
8	7	lstaff	user7@gmail.com	$2a$10$/SFdsffdghj0PasdfYySEqe7gdsfjdfffLKJFKlksdfnclsdkfjiJ	lab_staff	2022-03-18	[null]
9	8	istaff	user8@gmail.com	$2a$10$/ouiykfdghj0PasdfYySEqe7gdsfjdfffLKJFKlkasdfcvaerfcqW	insurance_staff	2022-03-18	[null]
10	9	admin	user9@gmail.com	$2a$10$/DSFdsfvERgkyukiuYySEqe7gdsfjdfffLKJFKVCAjhertyaSDFFY	admin	2022-03-18	[null]

7.2 Backup and Archiving

A simple script to export the database as an incremental backup. Incremental backup only captures the change hence require less time for backup.

- Once a week full backup should be taken.
- Production Backup are stored in AWS S3. (Considering limited functionality backups are stored in local machine)

7.3 Performance

Following performance requirement are catered:

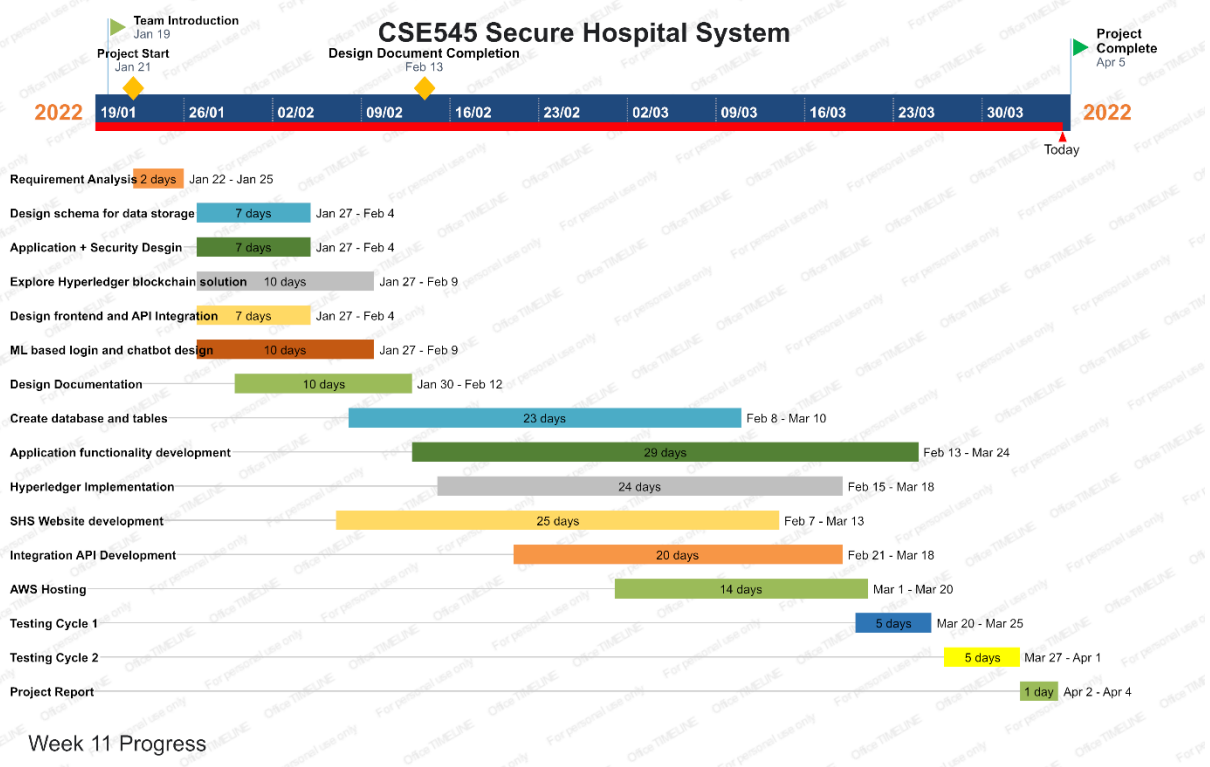
1. Response time of any event should not exceed the standard response time for hospital applications (3 to 5 seconds) – Use of light weight interaction through rest api.
2. System should withstand user load of approximately fifty users per second and operate in 24/7 environment – Setup of multiple EC2 instance with session/thread management

8 Gantt Chart

The Gantt chart provides the overall project plan and progress of the project. All the tasks listed in the Gantt chart have been successfully completed on schedule.

The project was divided into following phases:

1. **Requirement Analysis**
   - Understanding of the requirements provided in the requirement document.
   - Clarification with Professor and TA.
2. **Design**
   - Research on new technologies such as blockchain, malicious login, chat bot, login authentication.
   - Dataflow, class diagram and sequence diagram design for the SHS Functionality.
3. **Design Documentation**
   - Preparation of the design documentation.
4. **Implementation**
   - Implementing Application, database and UI.
   - Integration of all the 3 tiers.
   - Integration of login auth, chat bot, malicious login and blockchain
5. **Deployment**
   - Hosting the components on the AWS
   - Setting up SSL public and private key encryption for communications.
6. **Testing**
   - Functionality testing
   - Security Testing
   - Integration testing
7. **Project report preparation**



9 Bonus Features

All the components of data tier (Postgre DB and Hyperledger fabric) will be setup directly in the AWS EC2 instance. The application and web component will be deployed as an archive into web application server in the respective tiers.

10 References

1. https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html
2. https://www.bezkoder.com/spring-boot-security-postgresql-jwt-authentication/
3. https://medium.com/uber-security-privacy/uber-machine-learning-account-security-3aaadef11e45
4. https://levelup.gitconnected.com/learn-how-to-create-and-deploy-the-angular-application-to-aws-serverless-s3-81f8a838b563
5. https://material.io
6. https://dev.to/rodrigokamada/adding-the-google-recaptcha-v2-to-an-angular-application-1o7o
7. https://github.com/angular/flex-layout