Individual Project Report

Secure Hospital System (SHS)

Course 545 Project

Presented By
Saicharan Papani (1222323895)
Group 3 – Leader

Overview:

Security is highly important for any type of application. So, the main purpose of the project is to develop a hospital system that is highly secure. Following are the Security features that were implemented:

- Session Management: All the cookies that we used to operate the application will be destroyed after 15 minutes of inactivity.
- Prevent malicious login control: Locking account if multiple login failure attempts were seen, activating account after registration for proceeding to login, etc. features we included.
- Allow multiple users to login simultaneously.
- 24/7 accessibility: The site should be available 24/7 for the customers.
- OTP functionality when required.
- Public key certificates.
- DDoS resilient.
- Data masking and hashing algorithms for protecting sensitive data.
- Sign in and sign out logs for monitoring the activities.
- Storing sensitive data in Hyperledger blockchain.

The secure hospital system has six major roles, Administrator, hospital staff, insurance staff, lab staff, doctor, and patient. We have followed agile methodologies while developing the project.

Signup functionality:

All the patient users can do the signup process and register themselves. Admin user will be configured by the DB administrator. And all the other remaining user types must be invited by the admin. After completion of registration, all the accounts must be activated using the link that is sent to their email account.

Login & Forgot password functionality:

All the users (any role) will be able to login into the application using the login UI. If at all any of the users forgets his/her password, forgot password function can be used to reset the password.

Administrator:

As an admin of the application, he/she will have access to every part of the application and can perform all the operations. Following are the operations that an admin can perform:

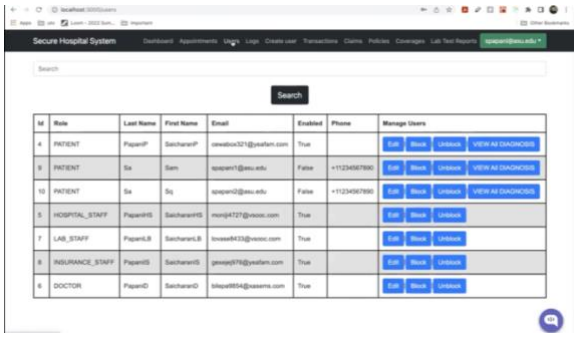

Figure 1: View All Users.

- View all the appointments (both past and current)
- View all the users that are present in the system. Can block or unblock the users. And if the user is a patient, then he/she can view a complete patient record that includes all the appointments, diagnoses, lab tests, transactions, and bills, etc.
- View all users sign-in and sign-out logs.
- Create a user or invite a user (any role).
- View all transactions/claims. And approve or deny transactions/claims.
- Create and view policies.
- Create and view coverages. These coverages can be used to include while creating a policy.
- Can view lab tests reports.

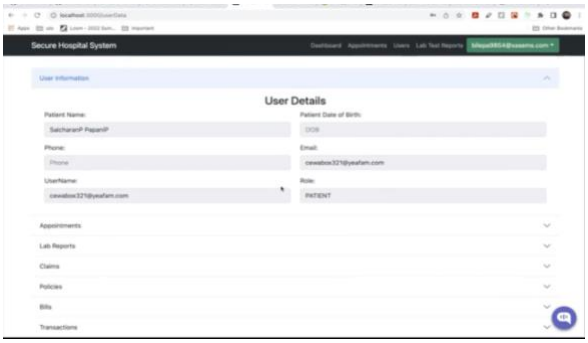

Figure 2: View complete patient record

Lab Staff:

He/Her will be responsible for performing activities regarding lab tests. Lab staff account

must be created by the admin and an invitation is sent for doing the activation of the account for further usage. Following are the functionalities of lab staff:

- View all patient users and diagnoses.
- Create and view all lab tests.
- Create, and update lab reports.

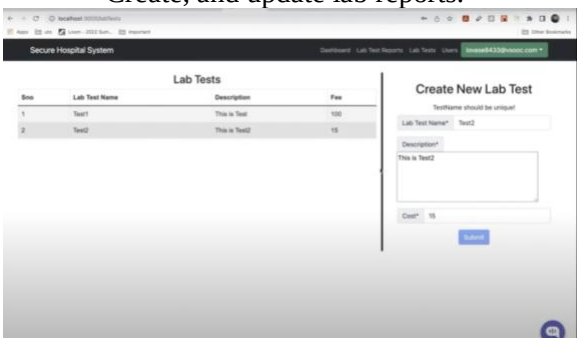

Figure 3: View, Create lab tests

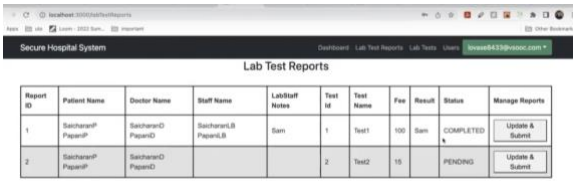

Figure 4: Lab Reports

Insurance Staff:

Insurance staff will take care of insurance and claims-related tasks. The insurance staff account is created by the admin and an invitation is sent for activating the account. Following are the functionalities:

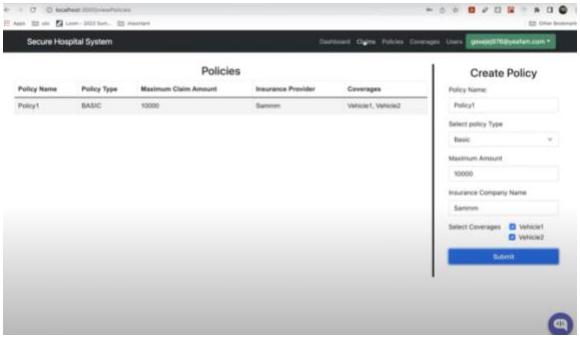

Figure 5: Create and view policies

- View and approve or deny claims from patients.
- Create new insurance policies and coverages.
- Authorize funds.

Patient:

Following are functionalities that the patient can perform:

- Only user role that can do sign-up by himself.
- Appointments:
  - View current appointments
  - View past appointments
  - Book appointment – It either can be specific (it is approved automatically by the system) or general (needs approval and doctor assignment from hospital staff).

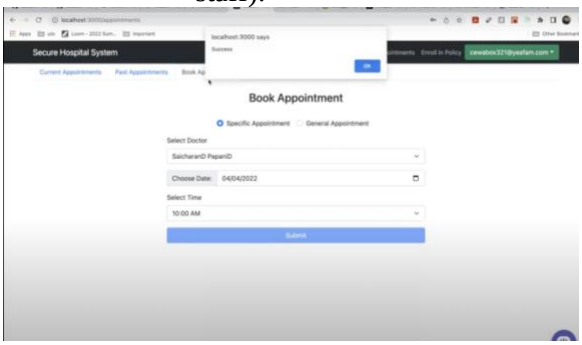

Figure 6: Book Appointment

- Update patient information. (Edit account).
- View complete patient record that includes:
  - All Appointments.
  - All diagnoses and prescriptions.
  - All lab tests.
  - All bills, claims, transactions, and policies. (Refer to Figure 2)
- Enroll in a policy. This is required if the patient wants to raise a claim.
- Raise a claim once the appointment moves to the "COMPLETED" state.
- Make a transaction once a bill is generated.

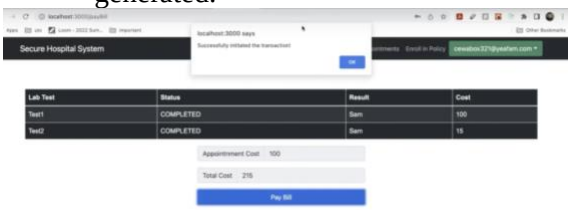

Figure 7: Make a Transaction

Doctor:

Doctor user will be created by the admin and an invitation is sent to activate the account. The

doctor will be responsible for treating patients and the following are the responsibilities:

- View all his current and past appointments.
- Write diagnosis and prescription for the patient (while serving the appointment).
- Recommend lab tests.
- View complete patient records for treating him (Refer to Figure 1).
- View lab reports of the patients.
- Complete the appointment.

Hospital Staff:

This user will be created and invited by the admin as other internal users. Following are the operations that the hospital staff is responsible for:

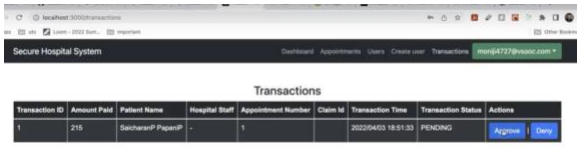

Figure 8: View, accept or reject transactions

- View all appointments.
- Approve or reject general appointments based on the doctor's availability.
- Create, and update patient records.
- View complete patient records.
- Complete or accept the transactions
- Create receipts and bills.

Contributions:

Frontend:

- Implemented the login and sign-up functionalities. Configured the protected and public routing system.
- Developed the system architecture and design in a multitiered fashion.
- Implemented the header in a more configurable manner which goes in sync with the route. Implemented and was responsible for major functionalities of the front-end like UI design and logic building related to Transactions, Policies, coverages, diagnosis, Lab tests, Claims, appointments, and patient records in a more adaptable, generic, and scalable manner.
- I have implemented the complex appointment functionality in both FE and BE.
- Proposed and implemented the state management with redux in react.

Backend:

- Created adaptable and generic APIs for major functionalities of the application which includes 3 layers of the logic building (Controllers, Service, and Repo layers).
- Implemented and was responsible for database schema and models.
- Implemented and configured security features that deal with Authorization and Authenticated by leveraging the spring security and JWT.
- Implemented the role-based restriction to all the APIs.
- Responsible for the design and Architecture of the BE.
- Proper organization of APIs is documented using OpenAPI (Only public APIs are available to hit and all other requires authentication and authorization).

Security features:

- Implemented session management.
- Implemented the public key certificates with SSL.
- Deployed the frontend and backend applications into AWS and Heroku.
- Integrated the Blockchain features into the application.

What have I learned?

- Creating more adaptable, generic, and scalable applications.
- How security plays a crucial role in securing sensitive data and how to implement security features in the application.
- Creating end-to-end system design and data models.
- How to properly manage a team and bring people together (Team building).
- Best practice of agile methodologies.