Project Report

Midpoint Deliverable

Megha Shishodia Pranoy Dev Vaishnavi Shah Yash Kothekar

Github_Repo_Link: https://github.com/pranoy1998/Patient_Tracker_System_CS520
Video_URL : https://drive.google.com/drive/folders/1mBsqjkRvKcfKtoizbiSCQFjEJylhRQ7r?usp=sharing

1. Requirements

1.1. Overview

The Patient Tracker System is an innovative application designed to revolutionize the management of patient information and medical records. The primary goal of this system is to introduce a seamless and efficient approach for both patients and healthcare providers to access, update, and manage patient records in real-time. Embracing digital transformation, this system eradicates the reliance on manual paperwork, offering a robust, secure, and accessible platform for managing patient data.

1.2. Features

A. User Authentication and Access Control:
   a. Secure login for patients and healthcare providers ensured with robust authentication methods, safeguarding patient records and regulating access based on roles and permissions, ensuring data integrity.

B. Comprehensive Patient Database Management:
   a. Centralized storage of a wide array of patient information, including personal details, medical history, prescribed medications, diagnoses, and treatment plans for efficient and organized record-keeping and comprehensive patient care.

C. Intuitive User Interface for Healthcare Professionals:
   a. An intuitive and user-friendly interface designed specifically for healthcare providers, enabling seamless navigation and easy interaction with patient records, optimizing healthcare delivery.

D. Data Security and Compliance Measures:
   a. Implementation of strong data encryption, secure data transmission, and stringent compliance with healthcare data regulations to ensure patient information confidentiality, maintaining trust and legal adherence.

E. Advanced Search and Filtering Capabilities for Efficient Care:
   a. Advanced search and filtering options allowing rapid retrieval of specific patient information, such as medications, previous checkup details, diagnostic reports, and medical history, streamlining healthcare processes for enhanced patient care.

F. Customizable Alerts and Notifications:
   a. Customizable alerts and notifications for healthcare professionals regarding new patient submissions, appointment reminders, and critical updates, ensuring timely and informed decision-making.

G. Collaborative Care Planning and Notes:
   a. Collaborative care planning features allowing healthcare providers to create and share patient notes, ensuring seamless communication and coordinated care among the healthcare team.

1.3. Functional Requirements (Use cases)

- As a Doctor, I want to view concerned patient profiles with medical history to prepare for consultations.
  - Action: Access patient records to review medical history, diagnosis and prescribed medications.
- As a Doctor, I want to update my patient's records, upload any related document and add Doctor Notes.
  - Action: Enter/update diagnosis, treatment plans or prescription updates into the patient's record.
- As a Doctor, I want to receive notifications for any creation/update of any concerned patient records.
  - Action: Receive alerts about new patient submissions or updated records.
- As a Doctor, I want to provide prescription updates or treatment plans for patients
  - Action: Update prescription or treatment plans based on consultation
- As a Doctor, I want to access my Daily Appointment List or dashboard
  - Action: View my appointment List for the day with patient name and appointment slot details
- As an Admin, I want to manage system users and their roles for access control.
  - Action: Administer system user accounts, roles, and permissions for data security.
- As an Admin, I want to ensure data security and compliance with healthcare regulations.
  - Action: Implement and oversee data security measures and compliance with regulations.
- As a Patient, I want to view my personal info, history and records.
  - Action: View personal records with medical history, diagnosis and prescribed medications
- As a Patient, I want to view all my appointments
  - Action: View all my doctor appointments with date, time slot and doctor details.
- As a Patient, I want to add/update profile info and patient notes
  - Action: Having Access to update personal info, upload documents and add patient notes if needed

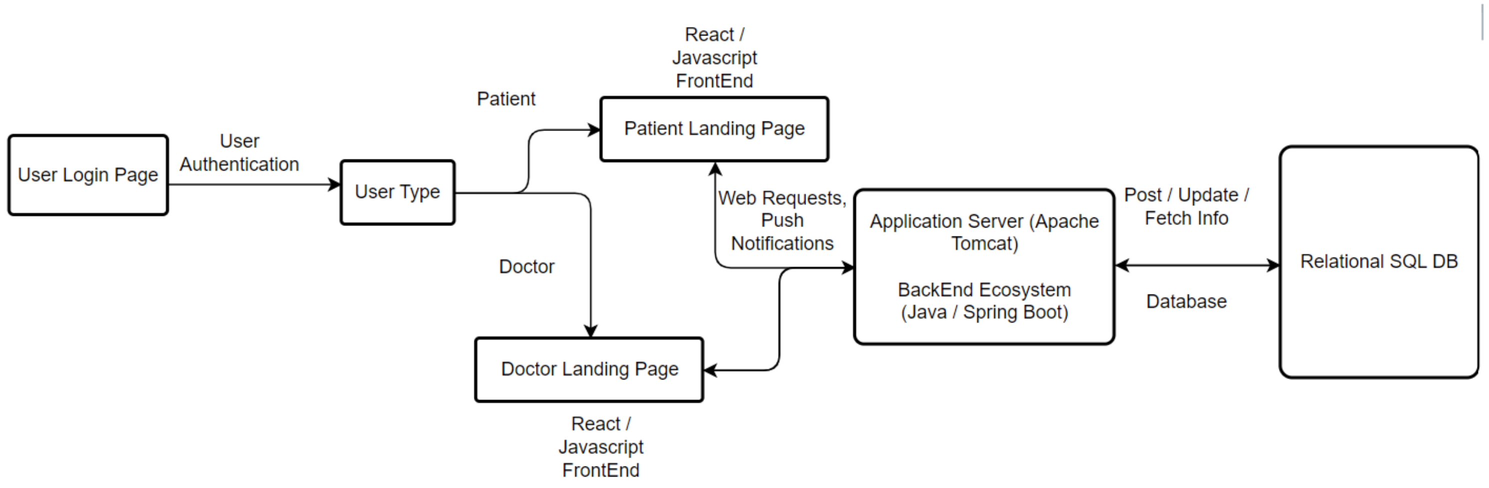
1.4. Non-Functional Requirements

- Performance :
  - To ensure system responsiveness with minimal latency for data retrieval and update actions, ensuring healthcare professionals can quickly access and update patient information during consultations, resulting in efficient patient care.
- Reliability :
  - Maintaining consistent and uninterrupted access to critical healthcare data and system functionality is essential for supporting continuous patient care and ensuring operational efficiency. Healthcare providers rely on this system for accurate, real-time information.
- Security:
  - Implementing robust data encryption, secure data transmission and access control is paramount to safeguard patient records. It ensures the confidentiality and integrity of sensitive patient data, preventing unauthorized access and data breaches.

- Scalability:
    - Designing the system architecture to handle potential future increases in the user base and data volume is crucial for accommodating the growth of the healthcare facility and ensuring that the system remains effective as the patient load increases.
- Usability:
    - Providing an intuitive and user-friendly interface is essential to facilitate easy navigation and use for both doctors and patients. A user-friendly design enhances the overall experience, making it more efficient and reducing the learning curve.
- Compliance:
    - Adhering to healthcare data regulations and standards is critical to ensuring patient data privacy and legal compliance. Compliance guarantees that patient records are handled in a way that meets the necessary legal and ethical requirements, maintaining trust in the system.

2) Design

2.1. Architecture Diagram


React / Javascript FrontEnd
Patient
Patient Landing Page
User Login Page
User Authentication
User Type
Web Requests, Push Notifications
Application Server (Apache Tomcat)
BackEnd Ecosystem (Java / Spring Boot)
Post / Update / Fetch Info
Database
Relational SQL DB
Doctor
Doctor Landing Page
React / Javascript FrontEnd

2.2. Technology Stack with Justification

1) Frontend: JavaScript and React

JavaScript is a widely used programming language for web development supported by all modern web browsers and a vast developer ecosystem for creating interactive and dynamic user interfaces. We can implement classes also as per need to make the code structure object oriented.

React, a JavaScript library for building user interfaces, is an easy-to-use component driven framework. Its component-based architecture makes it easy to create reusable UI elements and manage the frontend efficiently.

Major Advantages:

-> Virtual DOM feature that enhances performance by minimizing direct manipulation of the actual DOM, resulting in faster UI rendering.

-> An active community and good online documentation that contributes towards learning about the libraries, tools and resources to expedite development.

-> React supports server-side rendering, which…generally speaking, is beneficial for initial load times.

-> Its component reusability and a data flow make it easier to maintain and scale your frontend.

2) Backend: Java and Spring Boot

Java is a robust, mature, widely adopted and reliable object-oriented programming language, known for its scalability, security, and extensive libraries. In a healthcare application like the Patient Tracker System, data security and scalability are paramount. Additionally, the extensive Java ecosystem provides various libraries and frameworks for efficient development. Also being an object-oriented language, it allows us to structure our code keeping in important principles like modularity, reusability and other OOPS principles

Spring Boot is also a Java framework that simplifies the development of Java based applications. It provides a range of features that are well-suited for our project, including:

-> Ease of Development: It simplifies the development process by providing defaults and auto-configurations, which reduce the need for extensive setup.

-> Microservices Support: If our project needs to scale or extend its capabilities with microservices in the future, Spring Boot provides excellent support for building microservices.

-> Security: Spring Boot includes robust security features, including authentication and authorization, which are critical for protecting patient data.

-> Database Integration: Spring Boot makes it easy to integrate with various databases, which is essential for storing and managing patient records.

-> Community Support, Documentation: Spring Boot has a large and active community and well documented help websites, so we can find plenty of resources and support as we develop our application.

-> Compliance with Java EE Standards: Spring Boot adheres to Java EE standards, which ensures our application's compatibility with established best practices.
While developing a Java based app, we'll also need a web server or application server to run our Java application. Since you're using the Spring Boot framework for your Java application, we can take advantage of its embedded servers like Tomcat. Spring Boot simplifies the deployment process and offers flexibility in choosing the embedded server that suits our needs.

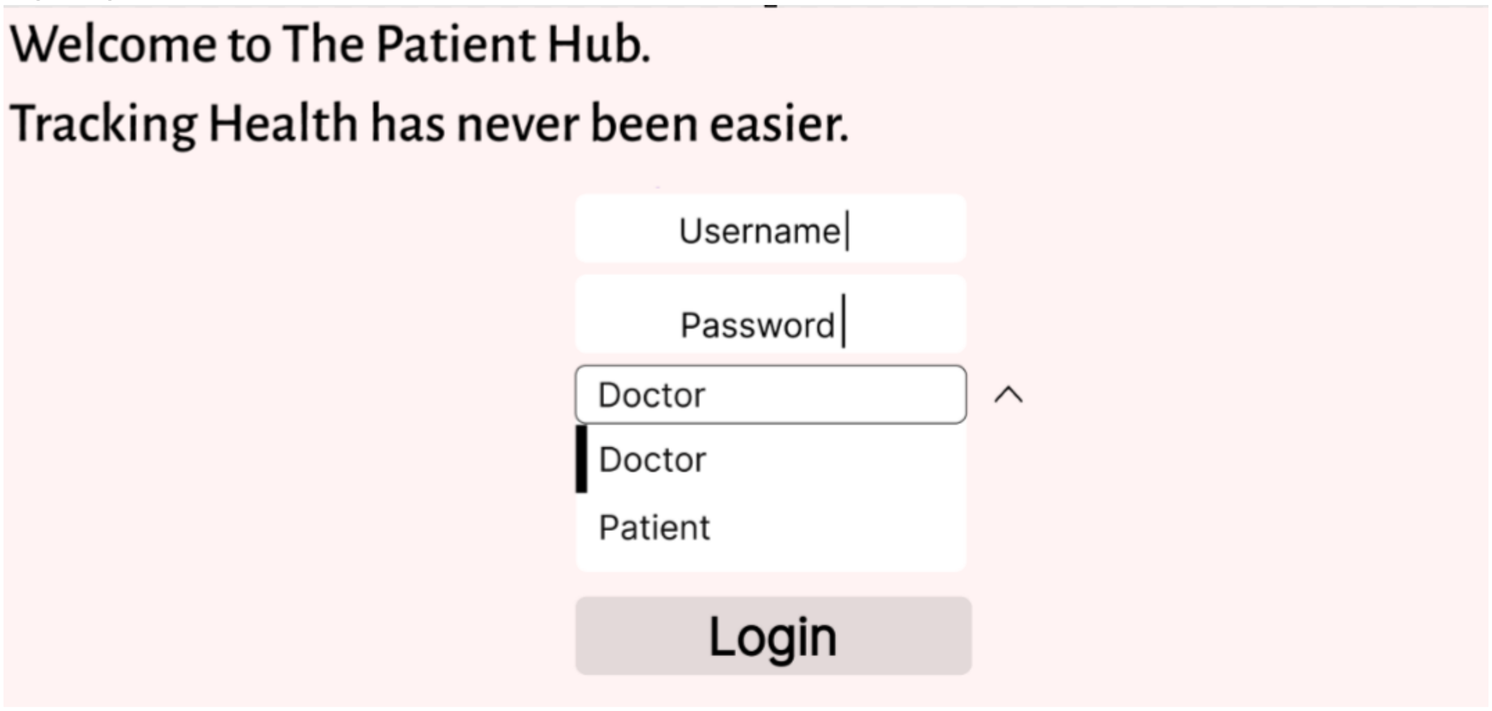
3) Database: RDBMS

Relational Database Management Systems (RDBMS) are known for their data consistency, ACID compliance, and mature query languages. They are suitable for healthcare applications where structured data is important, and relationships between different data points need to be maintained.

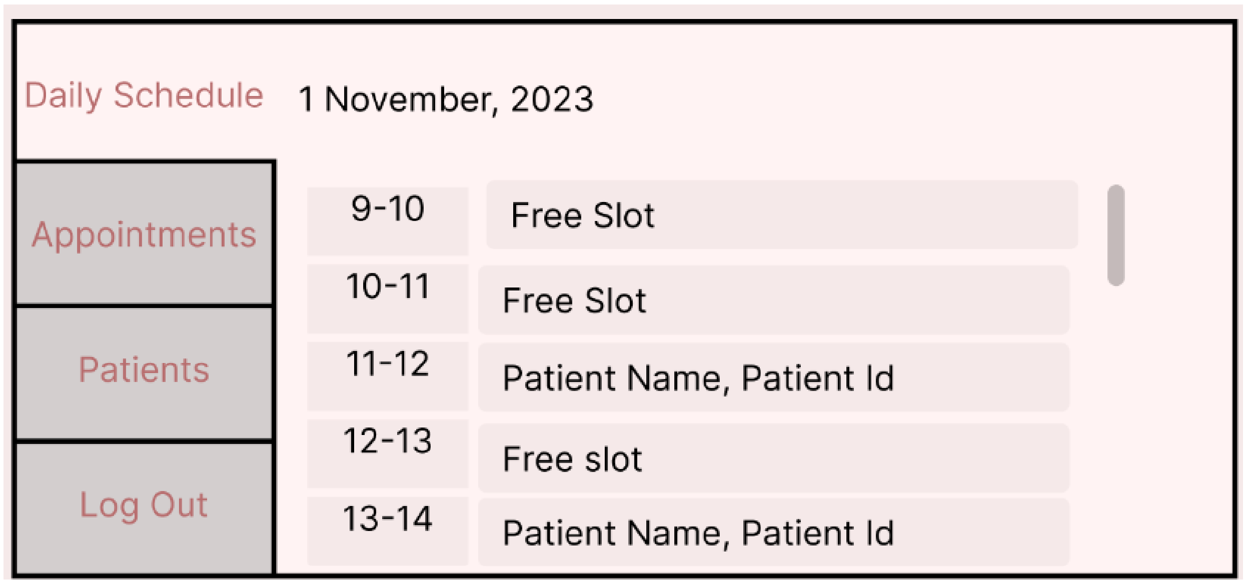
Features:
-> Structured Data: Since our patient data is highly structured and follows a well-defined schema with many relationships (e.g., patients, doctors, appointments, medical history), an RDBMS is a natural fit. SQL databases excel at maintaining data integrity and enforcing relationships between tables.
-> ACID Compliance: RDBMS systems are known for their strong ACID (Atomicity, Consistency, Isolation, Durability) compliance, making them suitable for applications where data consistency and reliability are critical, such as healthcare.
-> Complex Queries: If our application requires complex queries, reporting, and analysis of the patient data, SQL databases provide powerful querying capabilities and join operations.
-> Data Integrity: Healthcare applications demand a high level of data integrity and security. SQL databases provide robust security mechanisms and are well-suited for maintaining patient data privacy.

2.3. Basic UI Mockups

Login Page


Welcome to The Patient Hub.
Tracking Health has never been easier.

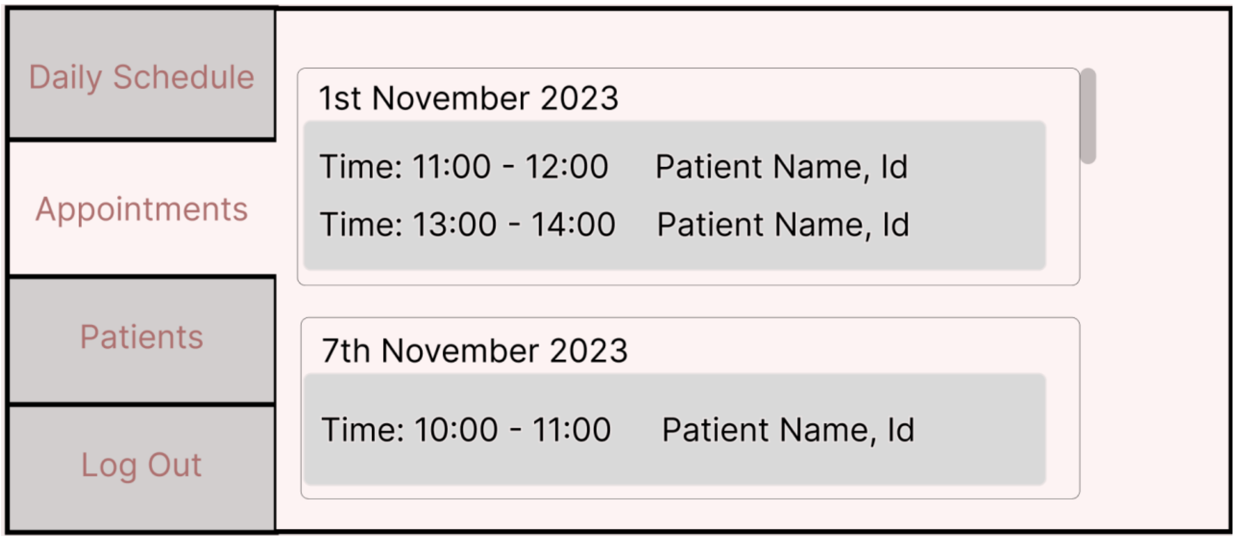
Doctor Daily Schedule View:



1 November, 2023

9-10	Free Slot
10-11	Free Slot
11-12	Patient Name, Patient Id
12-13	Free slot
13-14	Patient Name, Patient Id

Doctor Appointment List View:



1st November 2023

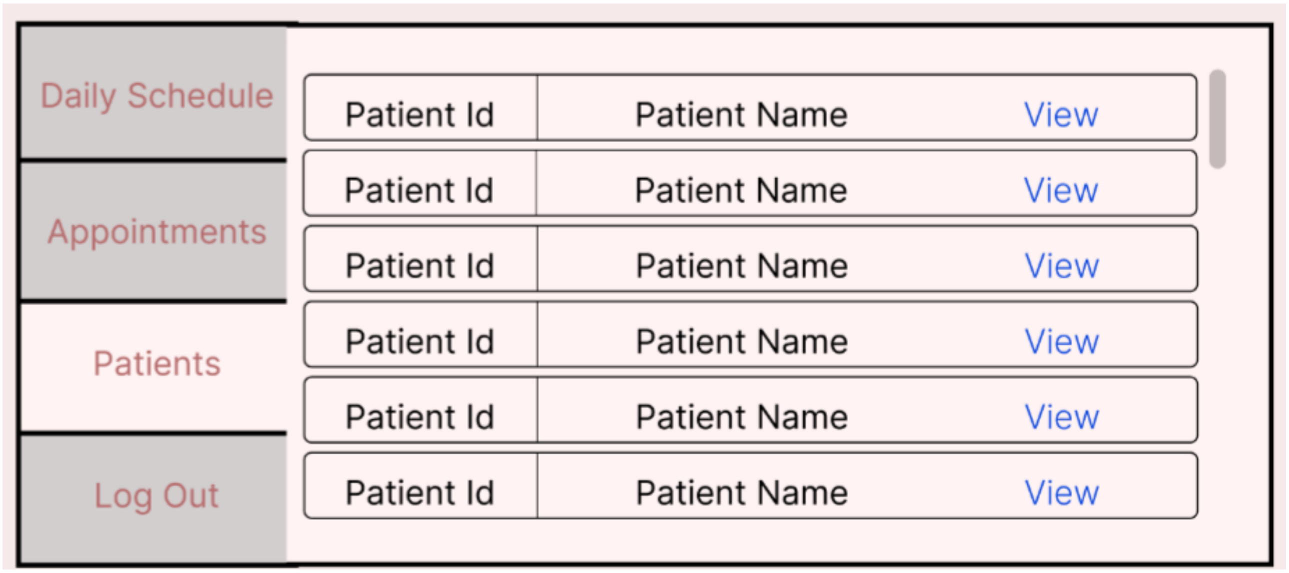
Time: 11:00 - 12:00 Patient Name, Id

Time: 13:00 - 14:00 Patient Name, Id

7th November 2023

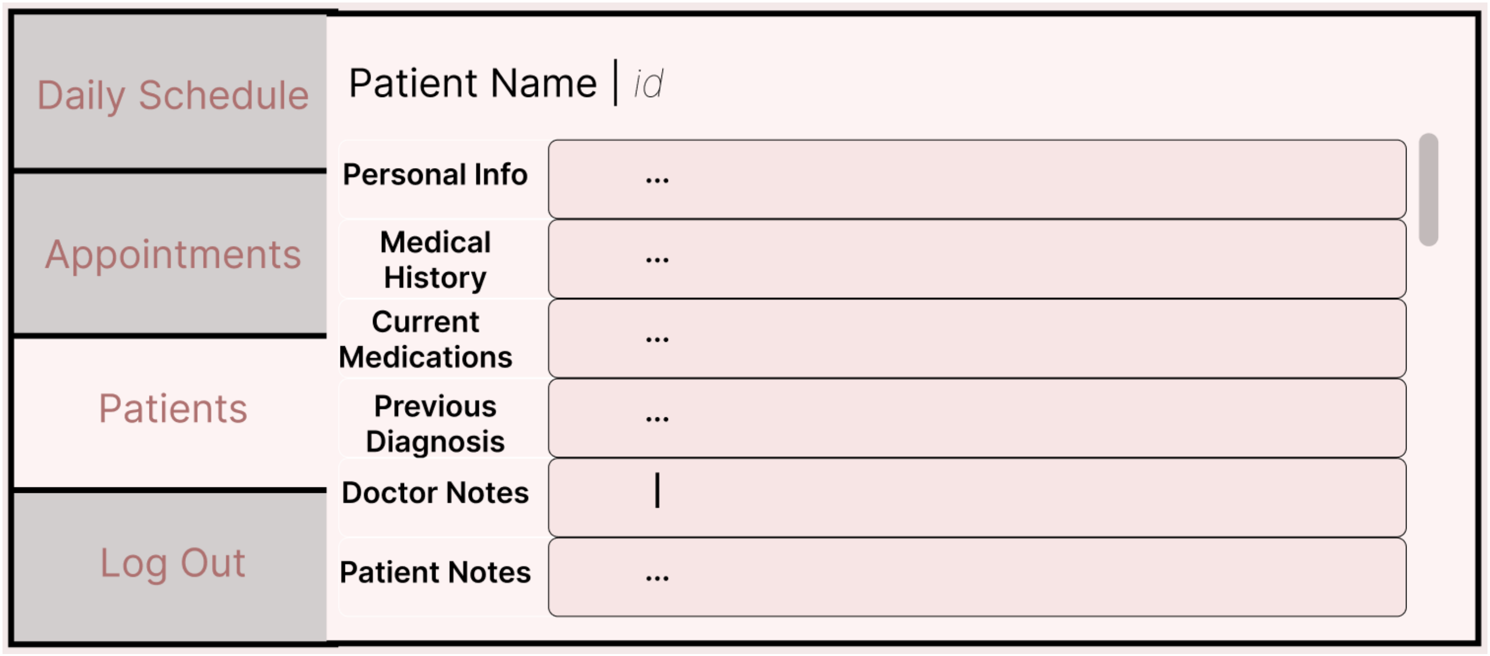
Time: 10:00 - 11:00 Patient Name, Id

Doctor Patient List View:

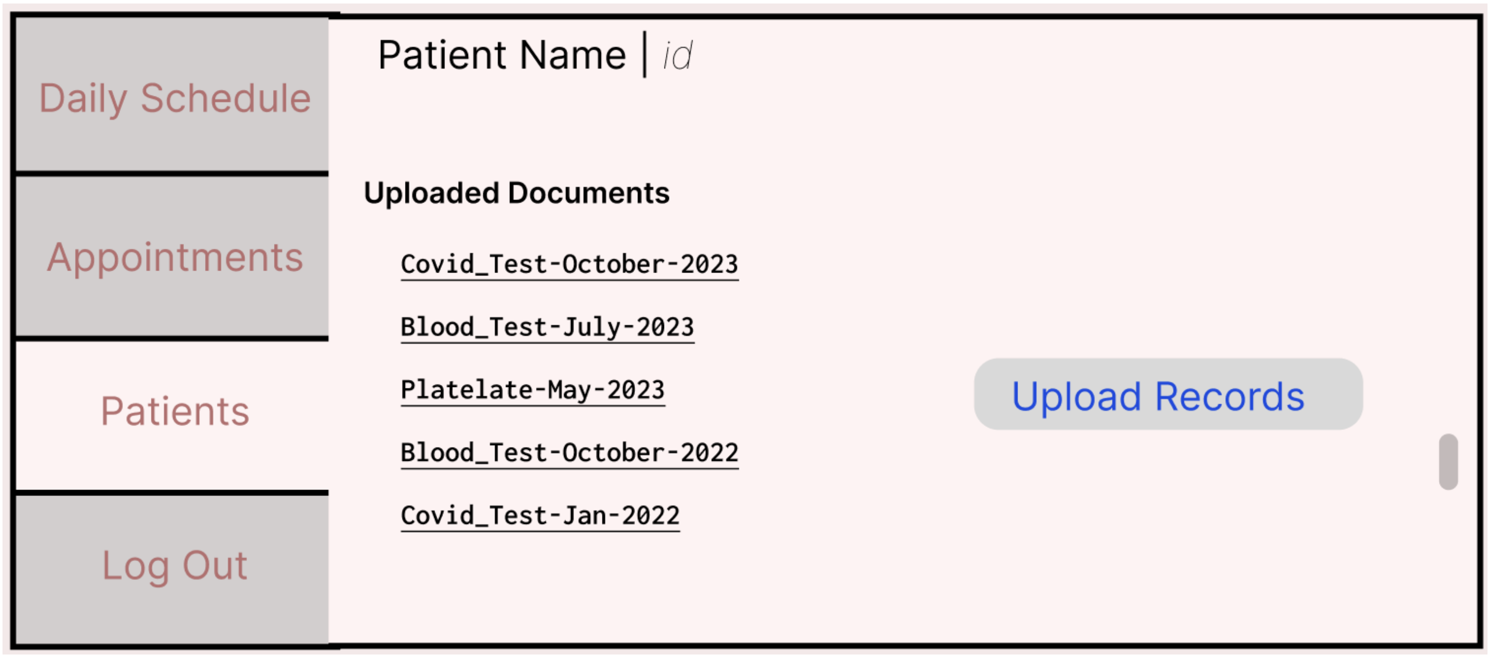


Patient Id	Patient Name	View
Patient Id	Patient Name	View
Patient Id	Patient Name	View
Patient Id	Patient Name	View
Patient Id	Patient Name	View
Patient Id	Patient Name	View

Patient info UI - Doctor View:



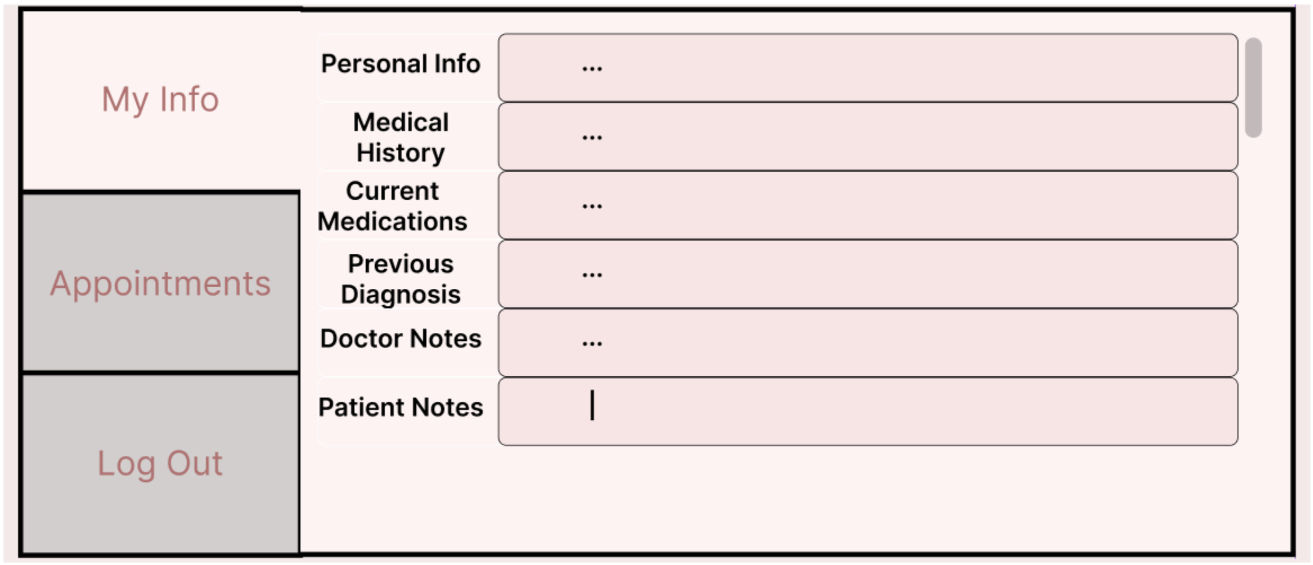
Personal Info	...
Medical History	...
Current Medications	...
Previous Diagnosis	...
Doctor Notes	|
Patient Notes	...



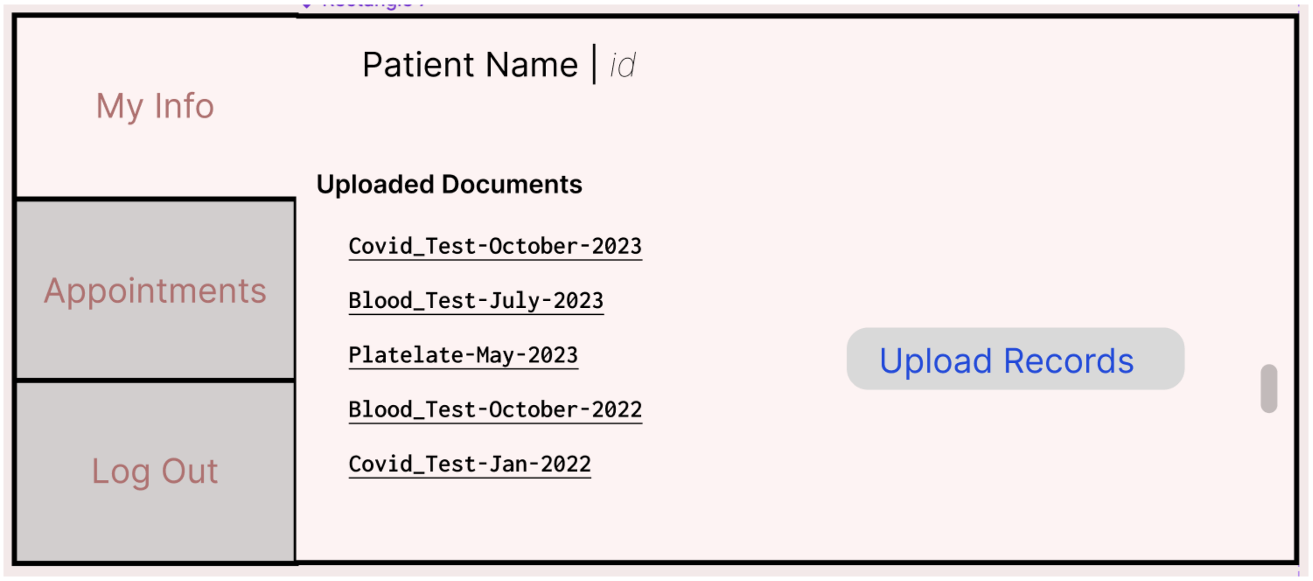
Uploaded Documents

- Covid_Test-October-2023
- Blood_Test-July-2023
- Platelate-May-2023
- Blood_Test-October-2022
- Covid_Test-Jan-2022

Patient View / My Info



Personal Info	...
Medical History	...
Current Medications	...
Previous Diagnosis	...
Doctor Notes	...
Patient Notes	|



Patient Name | *id*

Uploaded Documents

Covid_Test-October-2023

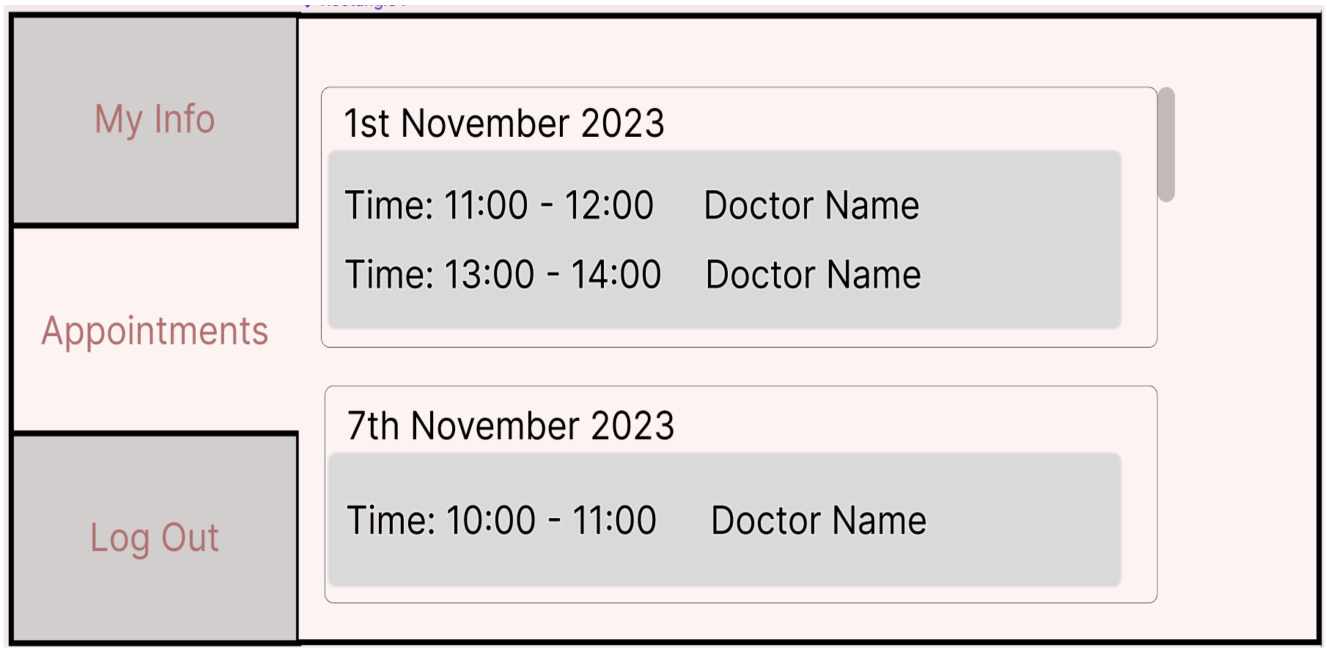
Blood_Test-July-2023

Platelate-May-2023

Blood_Test-October-2022

Covid_Test-Jan-2022

Patient View / Appointments



1st November 2023

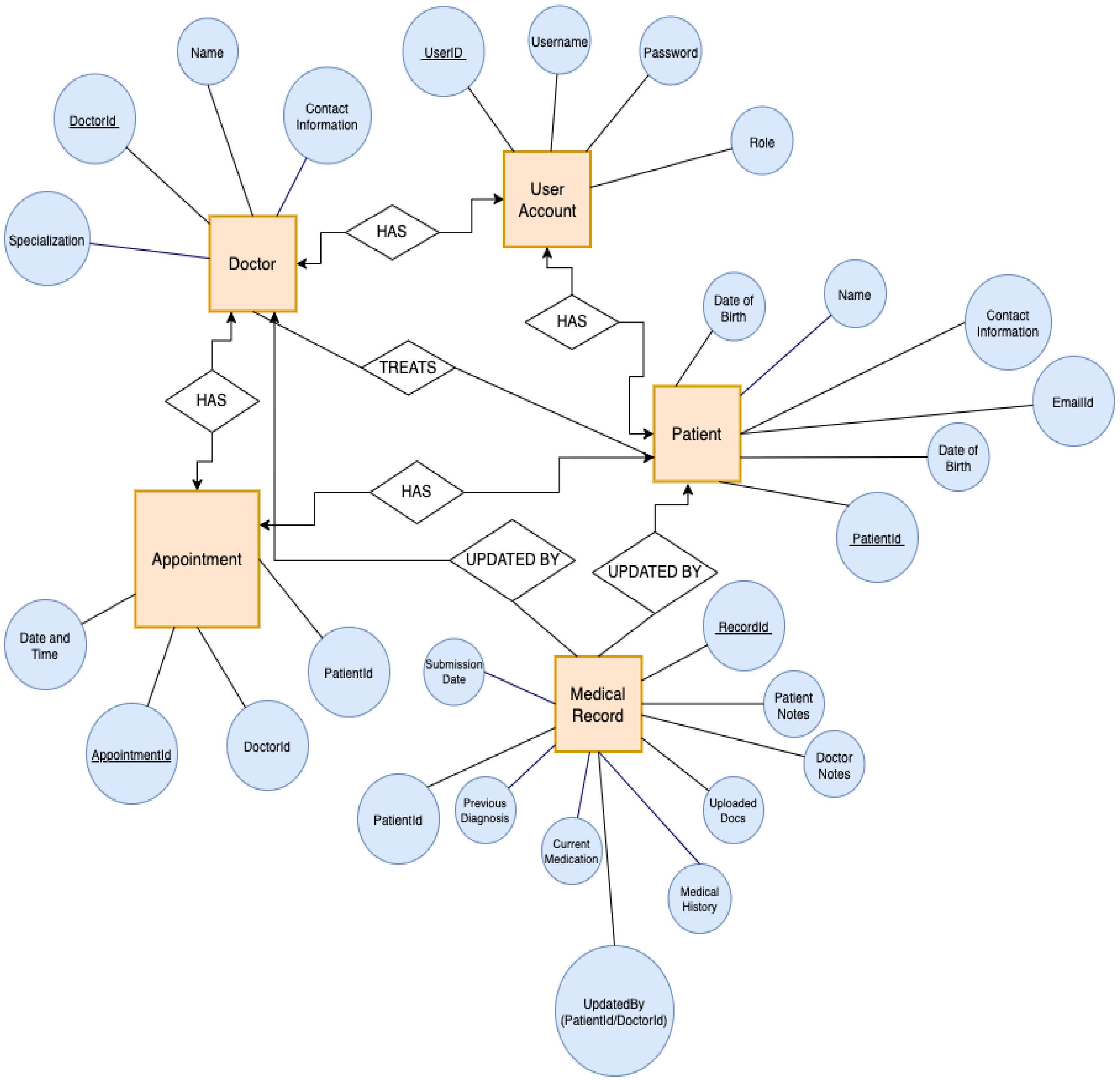
Time: 11:00 - 12:00 Doctor Name

Time: 13:00 - 14:00 Doctor Name

7th November 2023

Time: 10:00 - 11:00 Doctor Name

2.4. Data Model



3. Implementation

Coding Process:
-> Requirements Analysis: Understanding and documenting the requirements of your Patient Tracker System, including user stories, use cases and requirements.
-> Design: Creating a high level system architecture, database schema, UI mockup designs, Flow Diagrams, ER Diagrams etc
-> Development: Implementing the system according to the design using the chosen technologies; following best coding practices, maintaining code quality and conducting code reviews.
-> Testing: Performing unit testing, full flow testing and user acceptance testing to ensure the system functions as expected.
-> Deployment: Deploying the application and is ready to use.

Methodologies:
-> Agile: Using an Agile dev methodology, such as Scrum to promote collaboration, adaptability and incremental changes.
-> Version Control: Using version control systems like Git to manage code and collaborate effectively within our team.
-> Continuous Integration and Continuous Deployment (CI/CD): Implement proper CI/CD pipelines to automate testing and deployment processes, ensuring code changes are integrated smoothly into the production environment.
-> Documentation: Maintaining a comprehensive documentation for code, system architecture and user guides.

Potential Challenges:
-> Data Security: Ensuring the security of patient data via data encryption and access controls.
-> Scalability: As the system grows, we may face challenges related to scalability, hence keeping the design simple and designing the system to be horizontally scalable to handle a larger number of users and patient records.
-> User Training: Training end users on how to use the system effectively. Hence, developing user-friendly and intuitive interfaces and providing comprehensive training materials and documentations.

3.1. Security and Risks

Potential Threats: Threats include unauthorized access and data theft.
Vulnerabilities: Vulnerabilities can arise from weak authentication, insufficient access controls or unencrypted storage.

Prevention Measures:
-> Strong password authentication mechanisms.
-> Encrypt data in transit and at rest using encryption technologies.
-> Apply strict access controls to limit data access to authorized personnel only.
-> Conducting audits and basic penetration testing to identify and mitigate vulnerabilities.
-> Training documents in data security best practices.

HIPAA Compliance:
The Health Insurance Portability and Accountability Act (HIPAA) is a U.S. federal law that establishes national standards for the protection of patients' health information. It sets rules and safeguards to ensure the confidentiality, integrity, and availability of healthcare data.

Relevance: HIPAA compliance is crucial for healthcare-related applications, such as the Patient Tracker System, as it deals with sensitive patient information. Non-compliance can lead to legal consequences and reputational damage.

Steps and Protocols for Compliance:
-> Implementing strong access controls to ensure that only authorized personnel can access patient data.
-> Encrypting patient data both in transit and at rest to protect against unauthorized access.
-> Conducting regular risk assessments and vulnerability assessments to identify and mitigate security risks.
-> Developing and enforcing policies and procedures for handling patient data securely.
-> Training all end users on HIPAA compliance and data security best practices, proper documentation.
-> Having a formal incident response plan in place to address data breaches and security incidents.

3.2. Work Plan

High Level Sprint Schedule:

1) Sprint 1: [Week 1 : Oct 7 - Oct 13 ; Week 2 : Oct 14 - Oct 20]
-> Define and explore project scope, objectives, use cases and high level requirements.
-> Formulate a high level project plan

2) Sprint 2: [Week 3 : Oct 21 - Oct 27 ; Week 4 : Oct 28 - Nov 3]
-> Gathering detailed requirements, create design document and define functional and nonfunctional requirements
-> Creating system architecture and database schema documents
-> Choosing Tech Stack
-> Developing wireframes and prototypes for the user interface
-> Work Distribution and detailed planning before app development begins

3) Sprint 3: [Week 5 : Nov 4 - Nov 10 ; Week 6 : Nov 11 - Nov 17]

-> Initializing development of the application backend.
-> Implementing user authentication and access control fields.
-> Developing database models, relationships and tables.
-> Creating API endpoints for POST, UPDATE and GET.

-> Initializing Frontend development with React.
-> Setting up user interfaces for patient data submission, login screens and doctor interactions.
-> Implement real-time updates for patient records.

-> Continuing server side development with a focus on data storage and retrieval.
-> Peer code reviews within the team and required use case testing

4) Sprint 4: [Week 7 : Nov 18 - Nov 24]
-> Incorporating security features to protect patient data.
-> Extending the frontend to include patient submission forms and access control on note fields
-> Ensuring that user interfaces are user-friendly and responsive.
-> Peer code reviews within the team and required use case testing

5) Sprint 5: [Week 8 : Nov 25 - Dec 1]
-> Integrating the frontend and backend components.
-> Begin testing the application, including unit tests and integration tests.
-> Identifying and resolving any issues or bugs.
-> Ensuring compliance with healthcare data regulations.
-> Peer code reviews within the team and required use case testing

6) Sprint 6: [Week 9 : Dec 2 - Dec 8]
-> Developing user training materials for doctors and patients.
-> Creating comprehensive documentation for system users and administrators, API details and developer documents
-> Performing final testing, including user acceptance testing.
-> Preparing the system for production deployment.
-> Final Peer Code Reviews within the team
-> Deploying the application

High Level Responsibilities:

[Documentation, Peer Code Reviews and Testing shared across per sprint]

-> Megha Shishodia : Server side and database -> setup and development

-> Pranoy Dev : UI interfaces and use cases -> setup and development

-> Vaishnavi Shah : API development, Encryption and Security Review

-> Yash Kothekar : Push Notifications, Integrating components