DBMS End Sem Project Report

Aditya Dhananjay Singh- 22cs02001
Kumar Snehal- 22cs02009
Atharva Penkar- 22cs02011
Meysakthivelan T- 22cs02014

The Database:

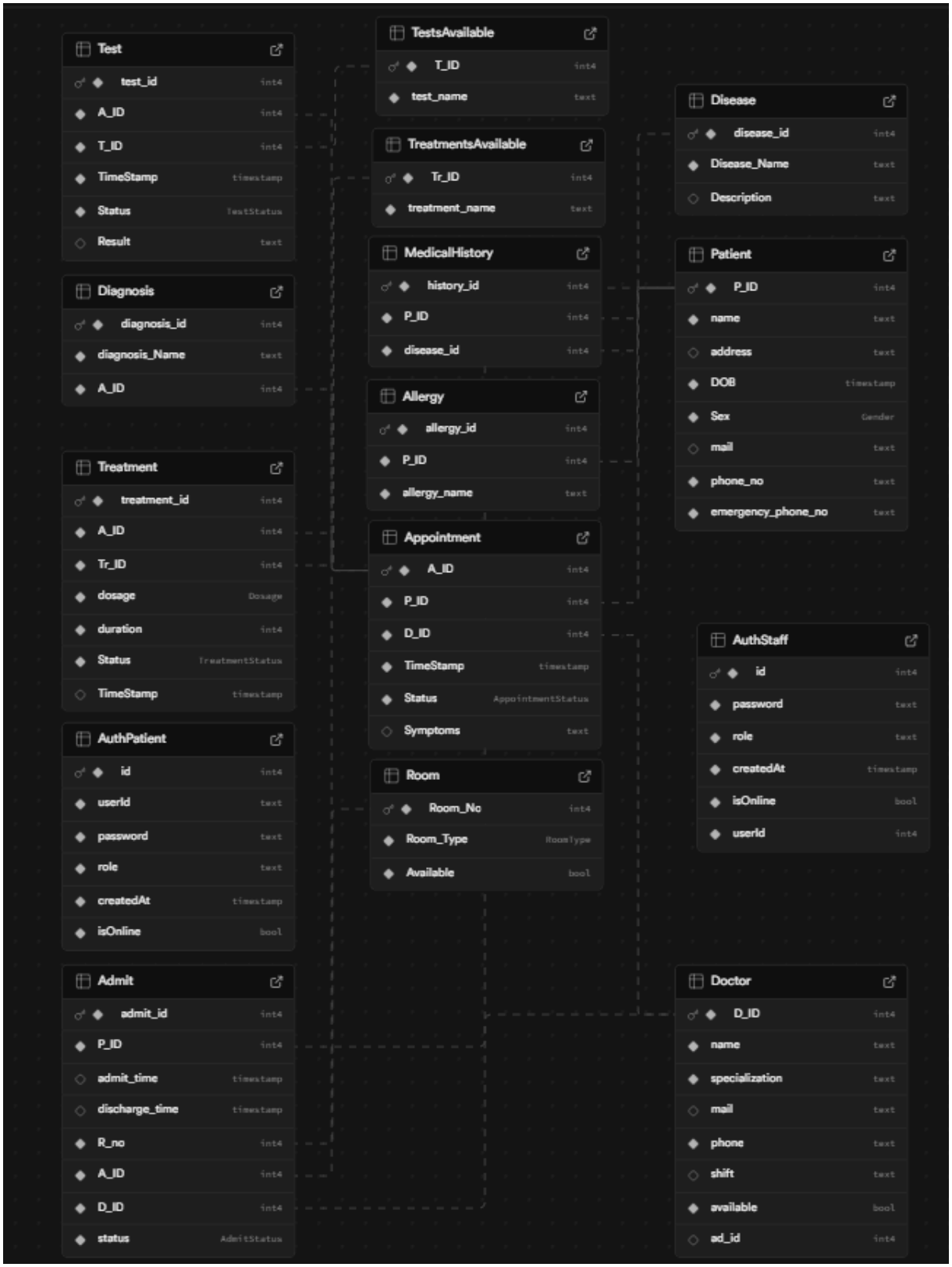
The hospital management system utilizes a well-structured relational database designed to efficiently handle diverse operations ranging from patient care and diagnostics to staff authentication and room allocations. The schema emphasizes normalization, clarity, and extensibility.

Key Features and Strengths

- **Entity-Centric Organization:**
    - The database contains clearly defined entities such as Patient, Doctor, Appointment, Test, Treatment, Room, and Admit, each responsible for a specific domain in the hospital's workflow.
    - Patient-related data is modularly split into relevant tables like Allergy, MedicalHistory, and Diagnosis, reducing redundancy and supporting detailed medical profiling.
- **Strong Referential Integrity:**
    - Foreign key relationships are carefully designed, such as linking Appointment with Patient and Doctor, or Treatment and Test with Appointment. This ensures data consistency and simplifies queries that join multiple tables.
- **Role-Based Authentication:**
    - Separate authentication tables (AuthPatient and AuthStaff) offer controlled access with roles and login states, enhancing security and supporting different system user experiences (e.g., patients vs. doctors vs. admins).

- **Efficient Resource Management:**
    - The Room and Admit tables allow for effective tracking of room availability and patient admissions, with fields for timestamps and status management.
- **Support for Medical Workflows:**
    - Test and treatment modules are dynamically linked to predefined lists (TestsAvailable, TreatmentsAvailable), allowing easy updates and scalability.
    - The inclusion of fields like Status, Result, and Timestamp in Test and Treatment tables enables precise tracking of medical procedures over time.
- **Scalability and Extensibility:**
    - The schema is easily extendable—new departments, procedures, or user roles can be added without disrupting existing structure due to normalized design.
    - Enum-like fields (e.g., Gender, TreatmentStatus, AppointmentStatus) provide clarity while allowing strict validation.
- **Well-Named Attributes:**
    - The attributes are consistently and meaningfully named, improving readability and reducing the chance of misinterpretation during development or reporting.

Database Overview:



Technologies and Tools Used

Our hospital management system is built using a robust and modern tech stack, separated cleanly into frontend and backend components. The system ensures scalability, maintainability, and a smooth user experience.

Frontend

- **React**: Used as the core library for building responsive and component-based user interfaces.
- **TypeScript**: Adds static typing to JavaScript, improving code safety and developer experience.
- **Tailwind CSS**: A utility-first CSS framework that enables rapid UI development with consistent styling.
- **shadcn/ui**: A modern UI component library built on top of Tailwind CSS, used to design clean and accessible components.
- **Role-Based Component Structure**: Components are organized by user roles (e.g., front desk operator) to maintain clarity and separation of concerns.

Backend

- **Express.js**: A lightweight and flexible Node.js framework used to handle routing and REST API logic.
- **TypeScript**: Used throughout the backend for type safety and improved code maintainability.
- **Prisma ORM**: Acts as the bridge between the application and the PostgreSQL database, enabling type-safe database access and schema modeling.
- **PostgreSQL**: A powerful open-source relational database system used to store all hospital-related data, such as patients, appointments, tests, and treatments.

Tools and Utilities

- **Postman**: Used for testing and verifying RESTful API endpoints during development.
- **Supabase**: Employed for storing and managing assets like images or documents related to the system (e.g., medical records or reports), and optionally as an authentication layer for certain modules.

Workflow Overview: From Database to UI

1. **User Interaction (Frontend – `client`)**
   A user (e.g., front desk operator) interacts with a React component (e.g., clicking "Schedule Test" or "View Treatments"), triggering a handler function.

2. **API Call (Frontend → Backend)**
   The frontend uses the **native `fetch` API** to send a RESTful request (e.g., `fetch('/api/appointments')`) to the backend server.

3. **Routing Layer (Backend – `server/src/routes`)**
   The request is routed by Express to the appropriate **route handler**, which maps the endpoint to a controller function.

4. **Controller Logic (Backend – `server/src/controllers`)**
   The controller processes the request, applies any business logic, and uses **Prisma ORM** to interact with the PostgreSQL database.

5. **Database Interaction (Backend – `prisma`)**
   Prisma translates the queries into SQL and communicates with **PostgreSQL** to fetch or update the data as needed.

6. **Response to Frontend**
   The controller sends a JSON response containing the requested data or operation result.

7. **UI Rendering (Frontend – `client/components/pages`)**
   The frontend receives the response, updates React state, and re-renders the component to display the latest data using Tailwind CSS and `shadcn/ui` components.

HTTP Requests Used in our Project:

1. GET

The GET method is used to retrieve data from the server without making any modifications to the database. It is designed to be a read-only operation.
Purpose: Data retrieval
Data transmission: Parameters are typically passed via the URL (query string or path parameters)
Security: Should not be used for transmitting sensitive information, as the data is visible in the URL and may be stored in browser history
Use Case Example: Fetching patient records or viewing appointment details

2. POST

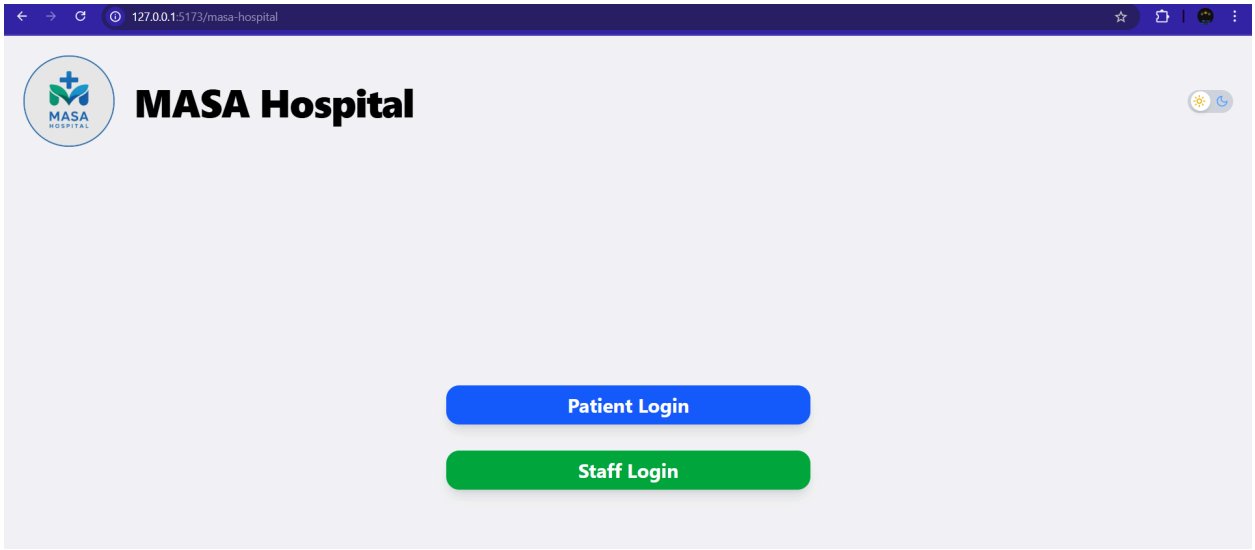
The POST method is used to create new resources on the server. It is also commonly used for user authentication, as it allows the transmission of data securely within the request body.
Purpose: Data creation and secure data submission
Data transmission: Sent via the request body in formats like JSON
Security: Suitable for transmitting sensitive information such as login credentials, as the data is not exposed in the URL
Use Case Example: Adding a new patient record to the database or logging in a staff member using credentials

3. PUT

The PUT method is used to update existing data on the server. It is generally intended for replacing the entire resource with the new data provided in the request.
Purpose: Full update of an existing resource
Data transmission: Sent via the request body, similar to POST
Requirement: The resource being updated (e.g., a specific patient by ID) must already exist
Use Case Example: Updating a patient's details, such as changing the diagnosis or age

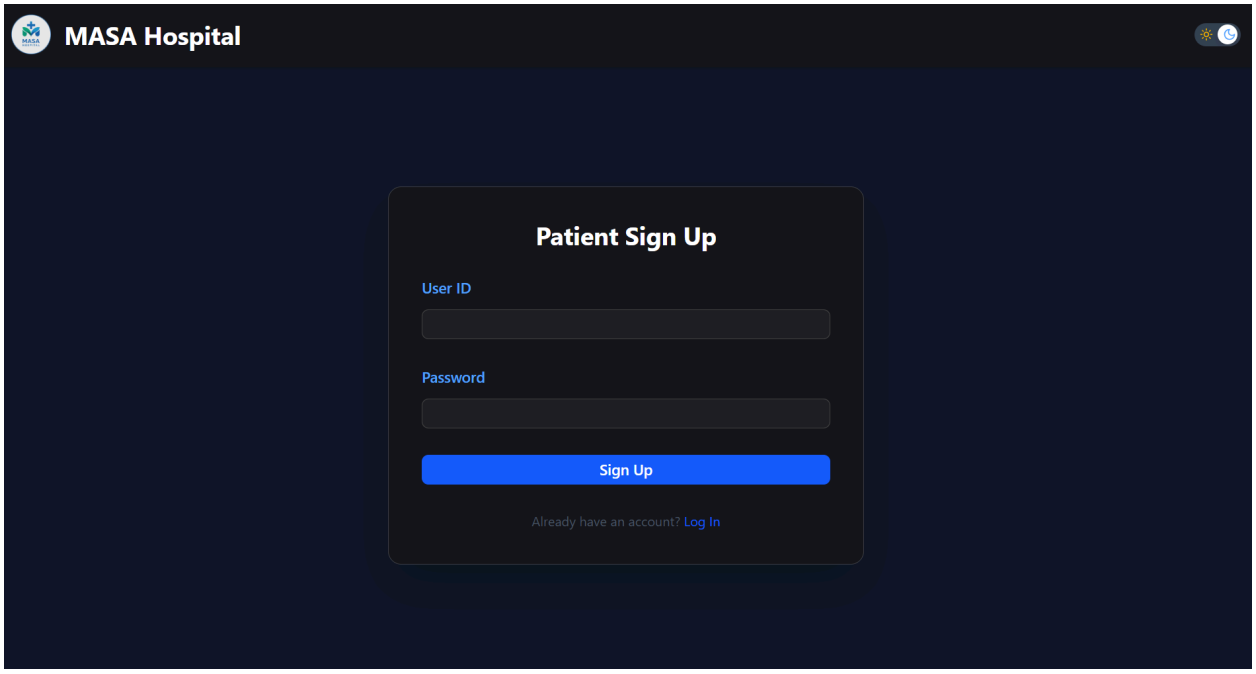
Page-by-Page overview of functionalities available

Masa-hospital: Home Page



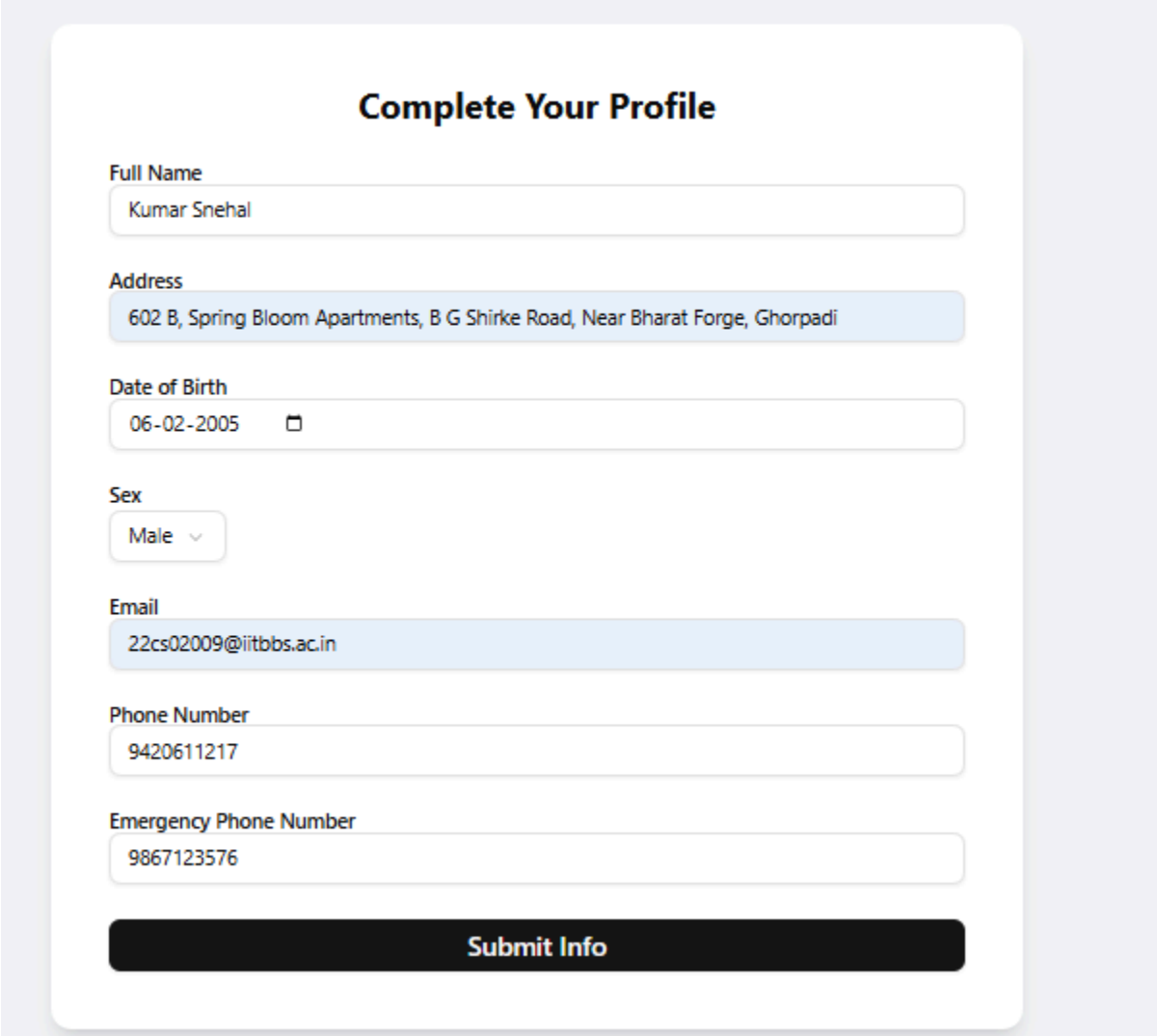
- Allows patients to move to patient signup/login page
- Allows the staff to move to their login pages (staff includes doctors, front desk operators and data entry operators)

Patient Sign-up :



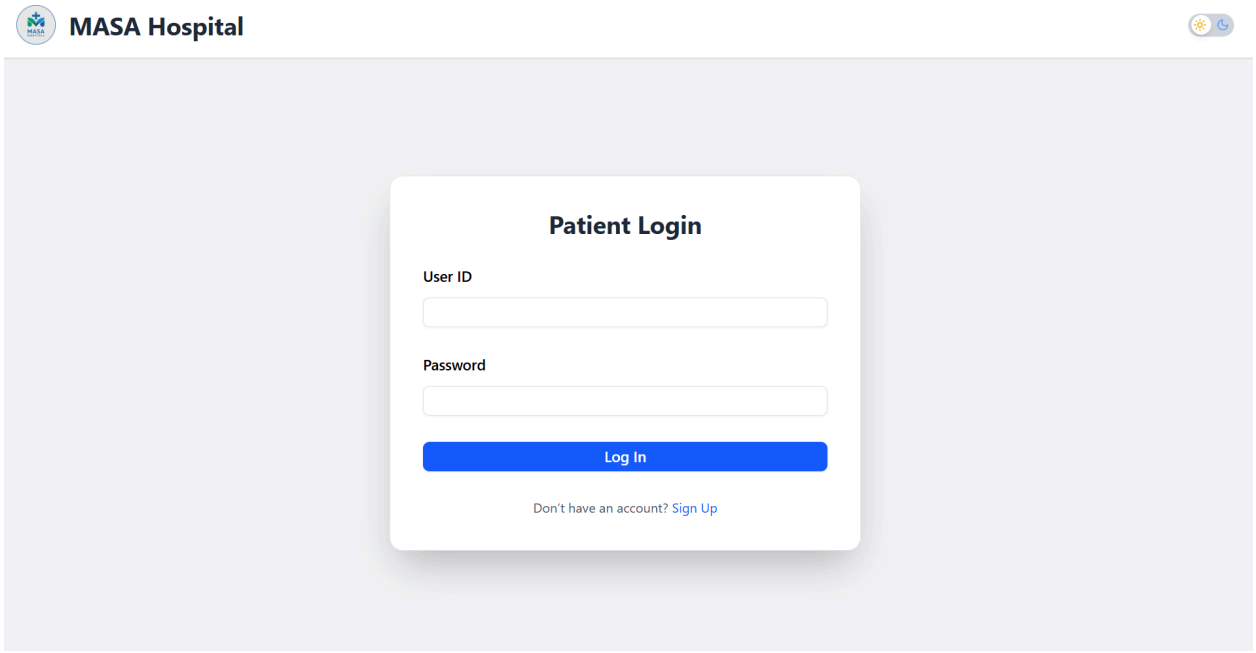
- Helps a new patient to sign-up and move to Sign-up form (where the data of a new patient is collected for the 1st time)
- Allows existing patient to move to Log In page

Patient Sign-up Form:



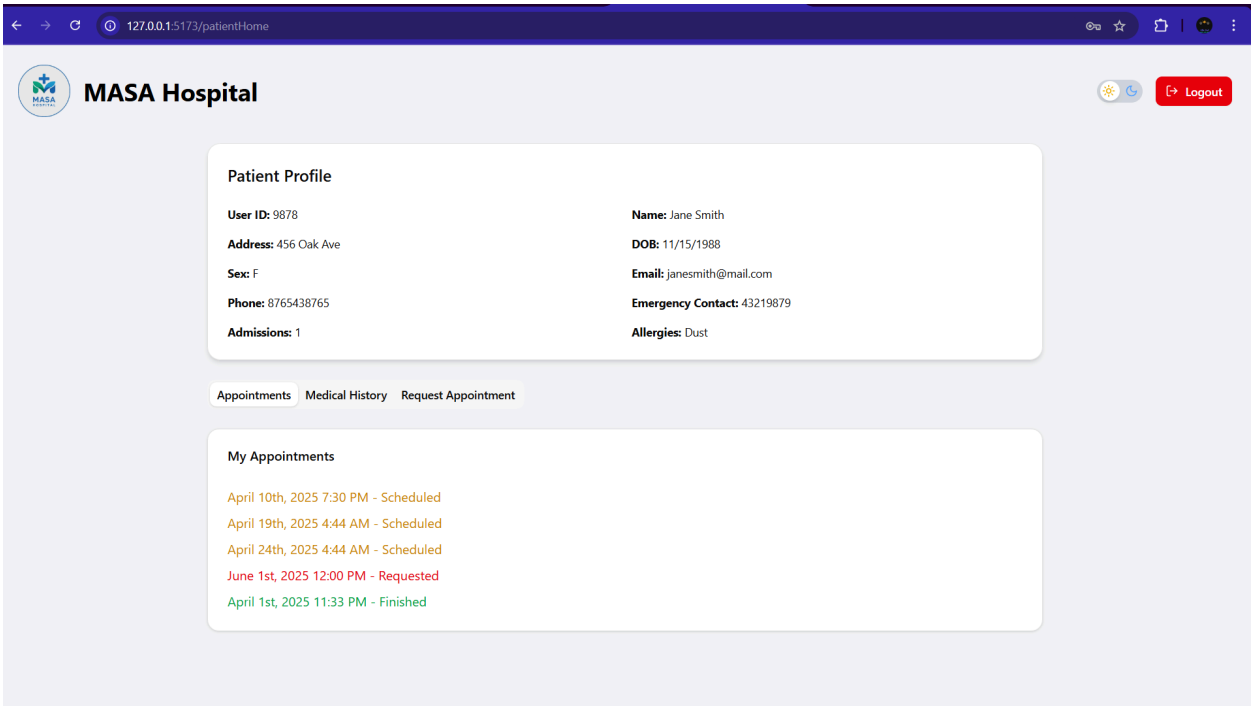
- Allows a new patient to register their details to the hospital database
- Includes interactive calendar to select DOB in required format, ensuring no error.

Patient Login:

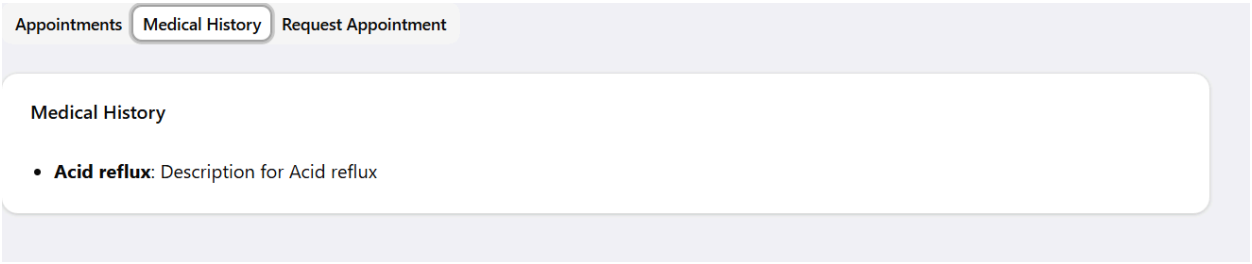


- Allows new users to move to Sign Up
- Navigates current users to their home-page after a successful login.

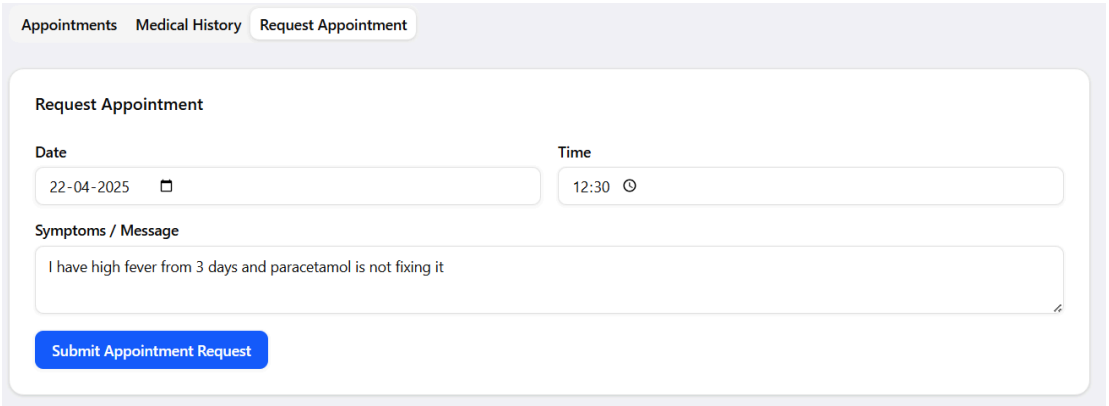
Patient Home Page:



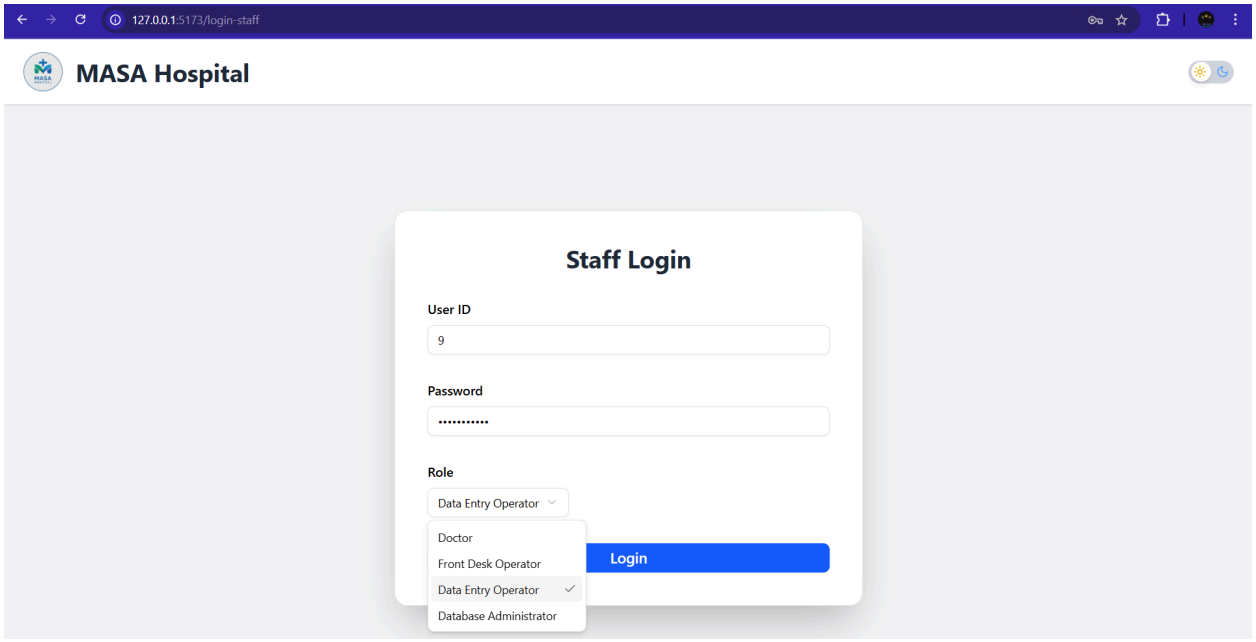
- Displays the general details of a patient (that the patient would have filled while filling in the Sign-up form)
- Shows the status and details of appointments
- Shows the medical history (if any) of the patient



- Allows the patient to request a new appointment



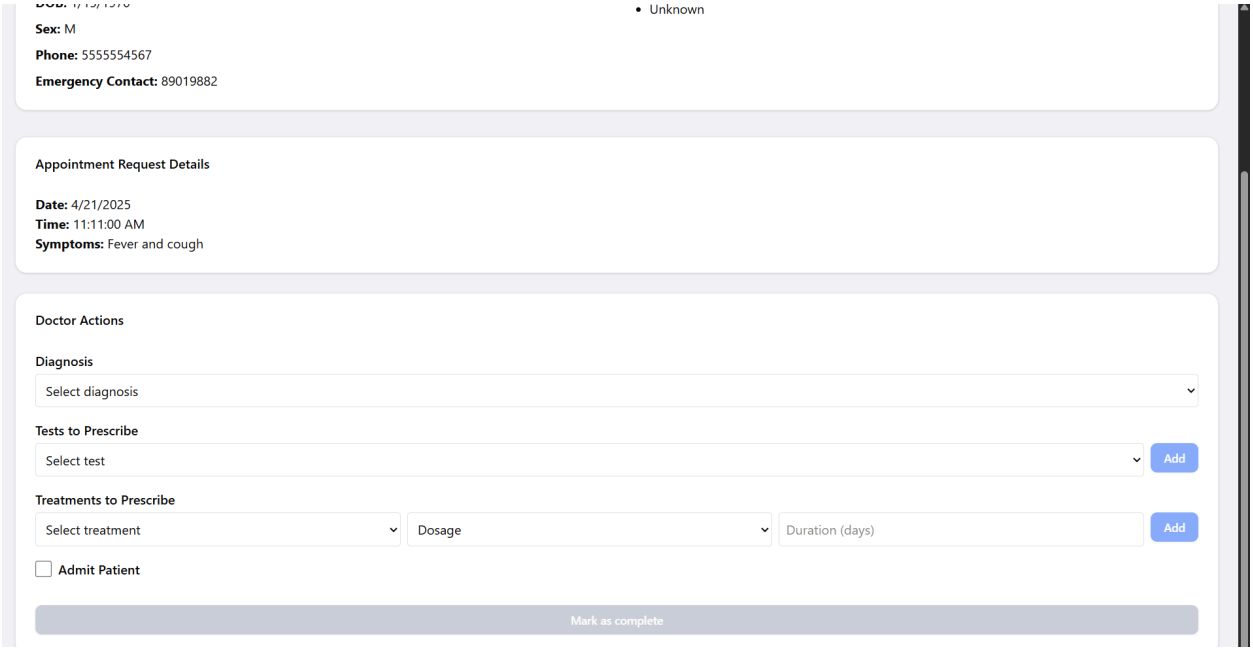
Staff Login:



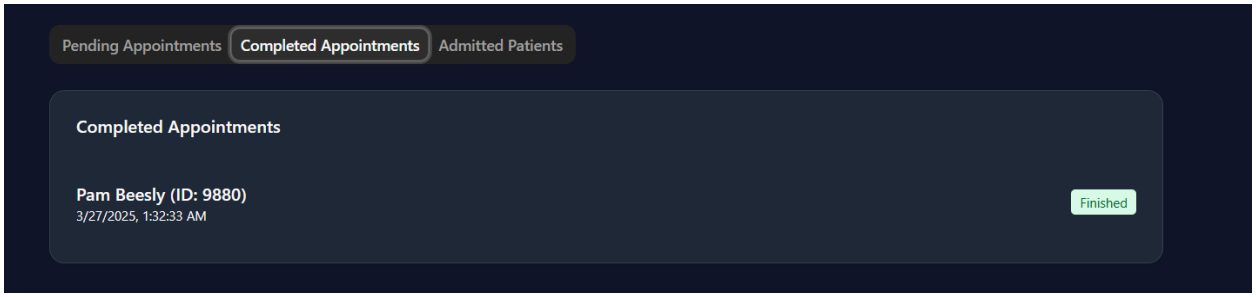
- Allows staff to select their particular roles
- Moves to their dashboard/homepage after successful login

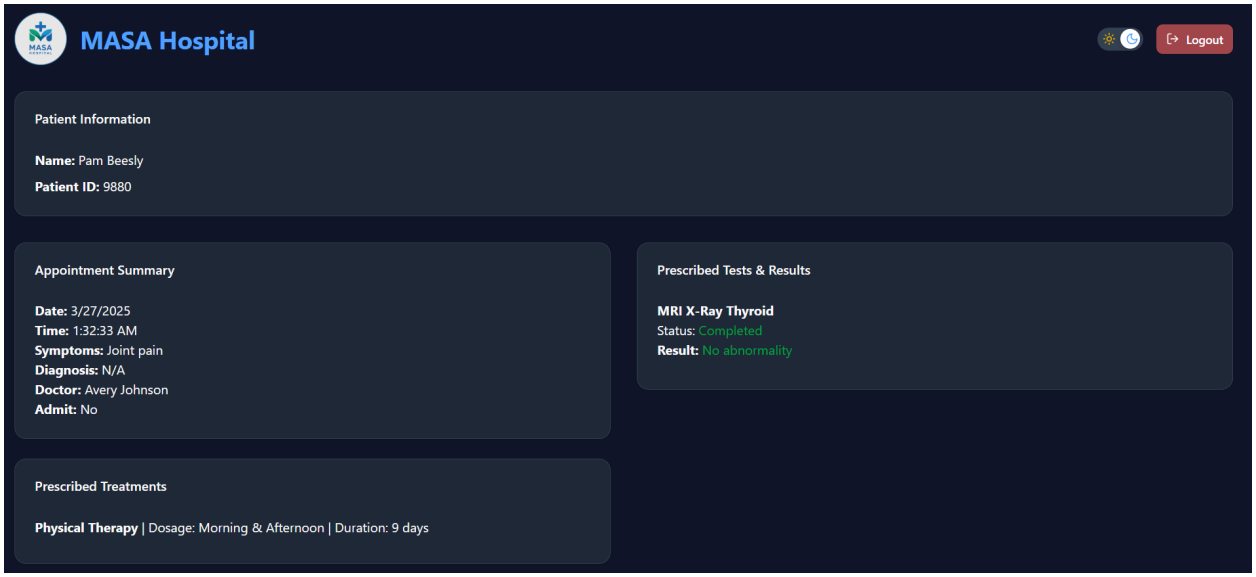
Doctor Homepage:

- Pending appointments: shows him the list of available appointments
- Clicking on that particular appointment allows him to see all details about that patient and also resolve the appointment (give a diagnosis/test/treatment/admit the patient)

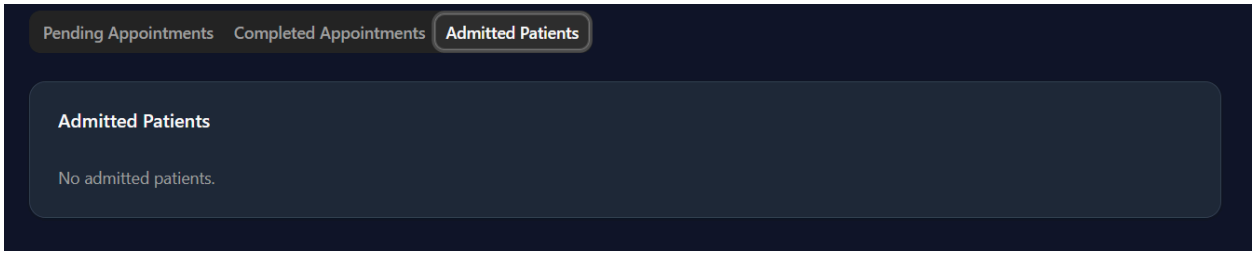


- Completed appointments: shows all completed appointments and everything that was performed on.prescribed to that patient (if clicked)

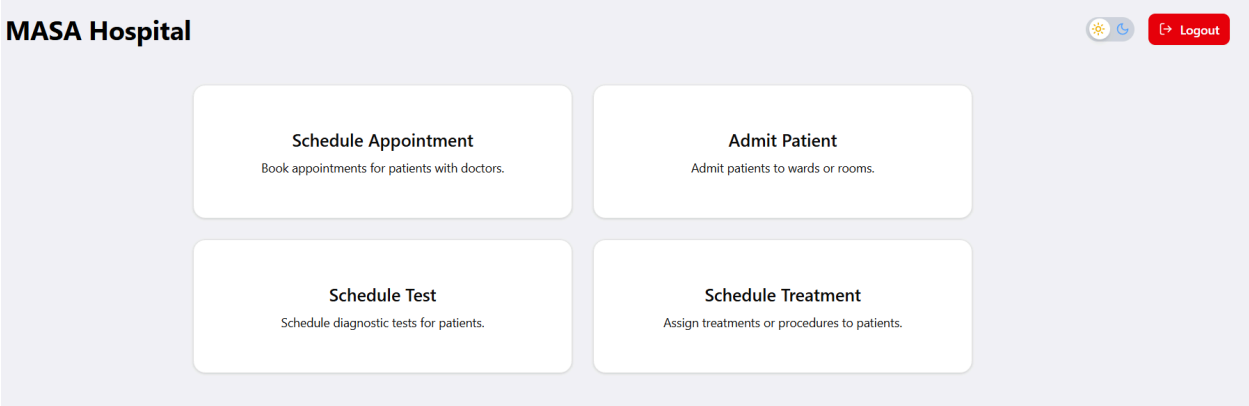




- Admitted patients: allows the doctor to view his admissions and put in a discharge request for any patient



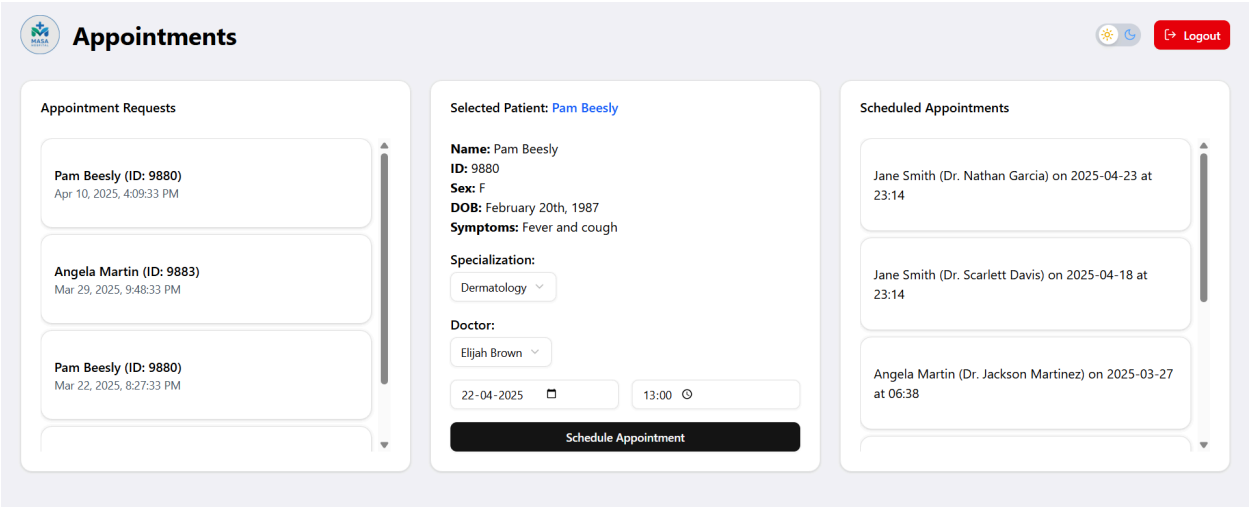
Front-desk Operator homepage:



- Move to scheduling appointments, admitting a patient, scheduling a test, scheduling a treatment

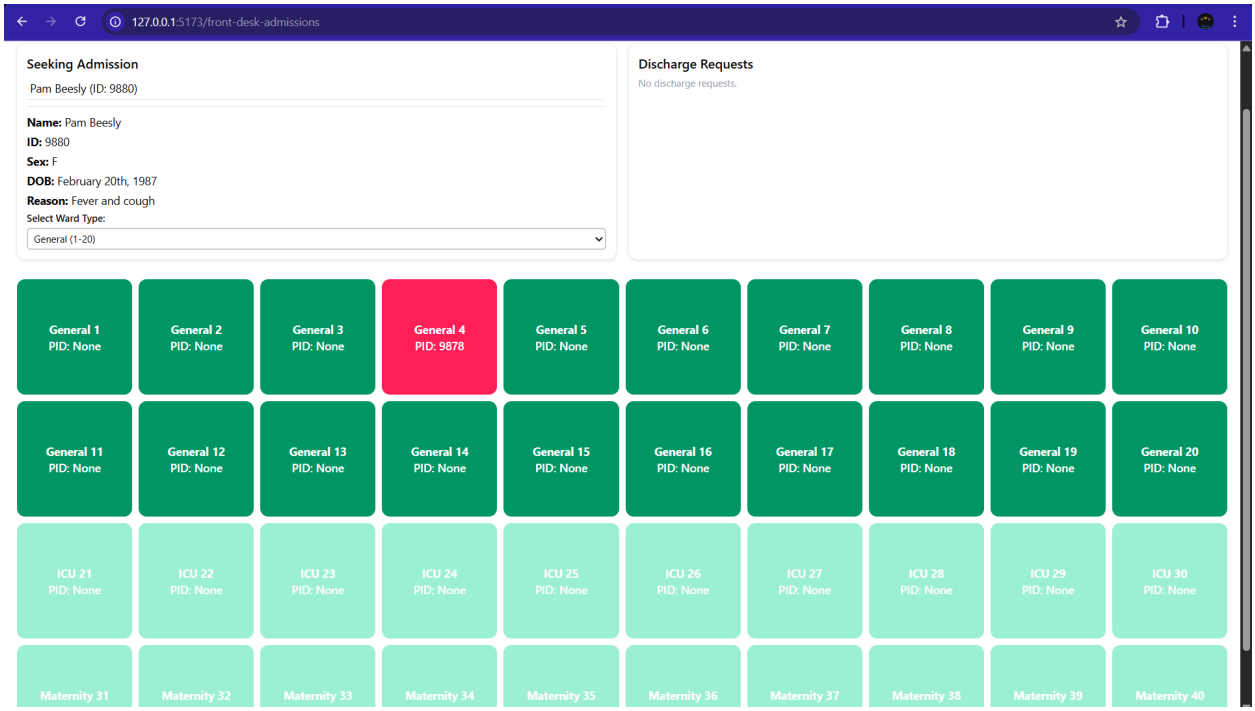
Scheduling appointments:

- Can see requested as well as scheduled appointments, can click on a requested appointment and schedule it as shown.



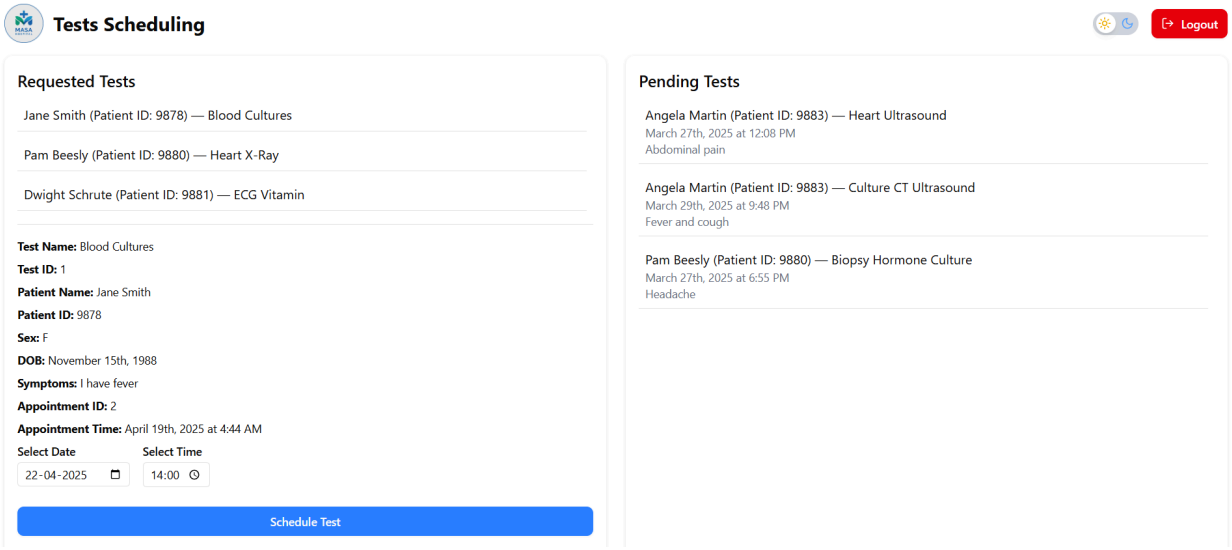
Admit Patient:

- Can see all patients seeking admission/discharge, and can allot a room (if available) and discharge patients



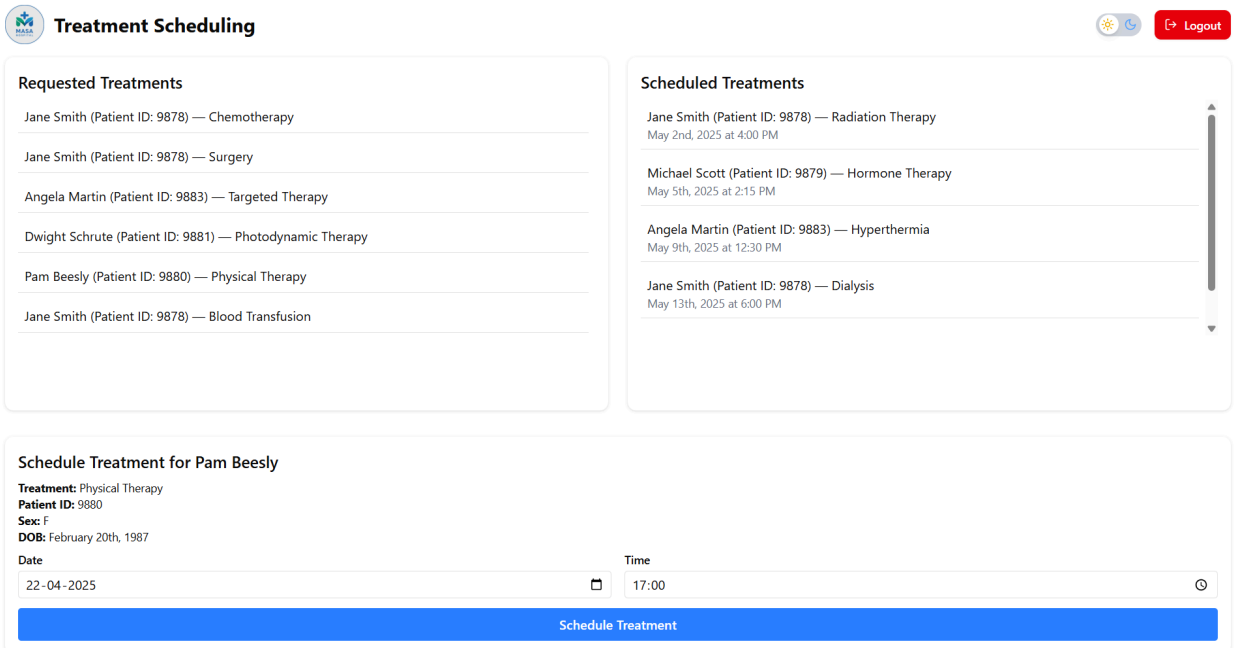
Schedule tests:

- All info about pending and scheduled tests and can schedule tests by selecting time and date

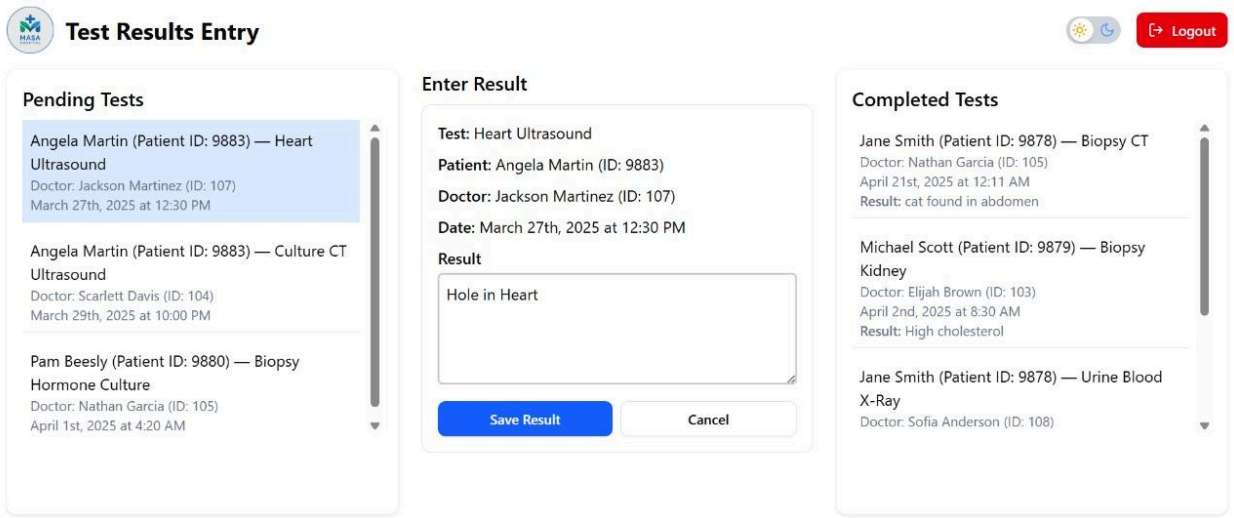


Treatment scheduling :

- Similar to test scheduling page, shows all pending and scheduled ones and can schedule treatments



Data Entry Operator:

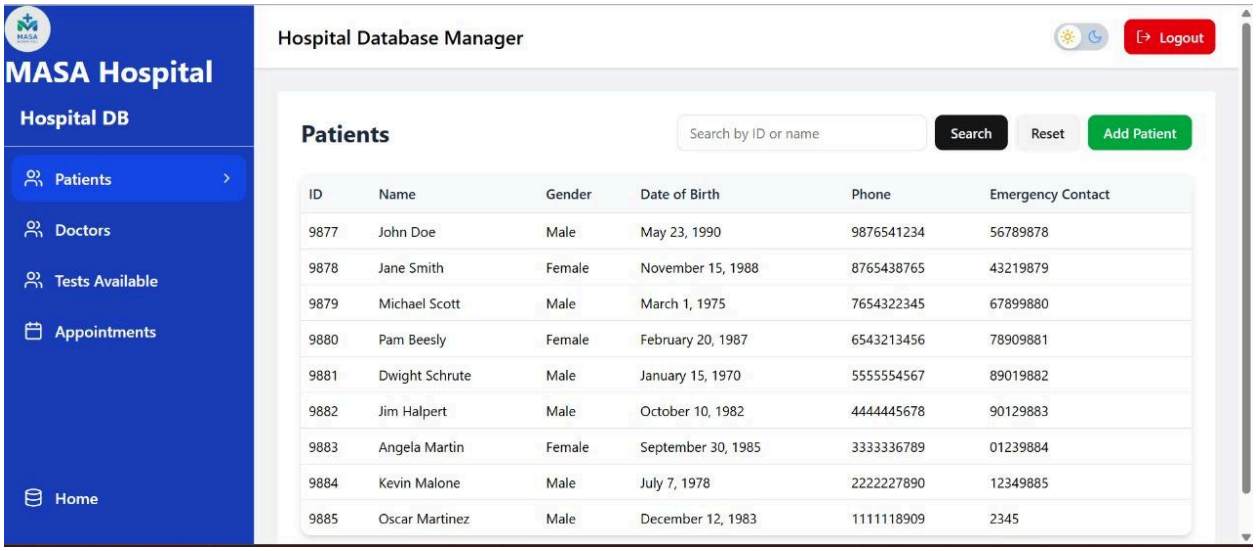


- Can enter the result of a pending test, and can see all data of all completed tests

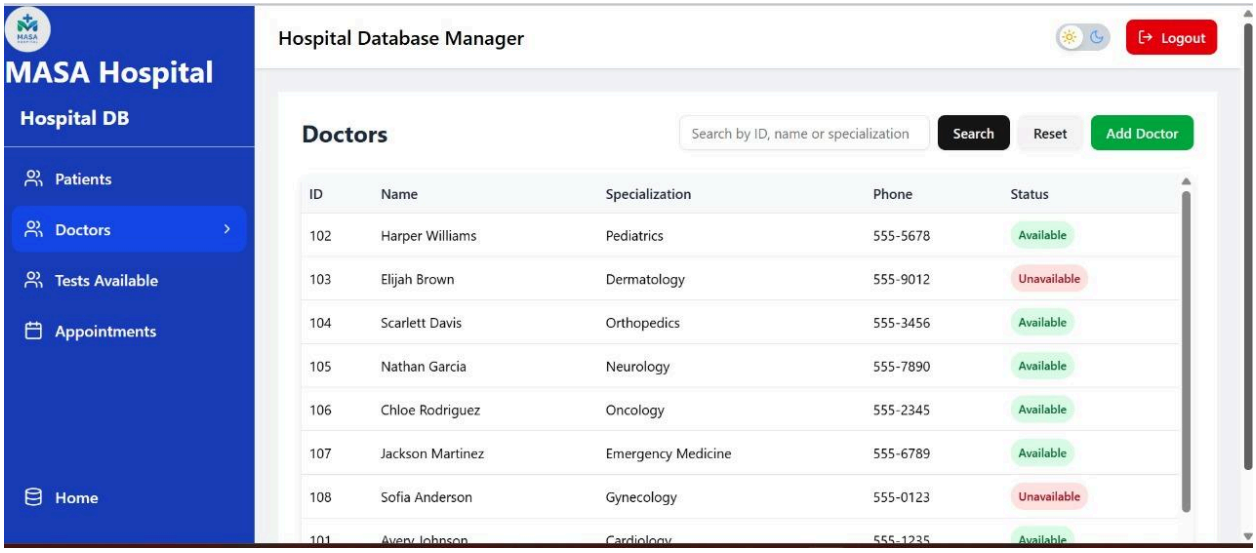
Database Administrator:

Can search for patients/ remove them/ add new ones, and a doctor can only be added by this ID

- Patients Tab:



- Doctor Tab:



- Can add a new test, for example the hospital procured a new equipment with which they can test new diseases