Indian Institute of Technology Hyderabad (IIT)

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Software Completion Report

Medical Access Service System (MASS)

SRGroupXY

Submitted by:
Priyanshu Sharma (CS24BTKMU11004)
Sadikshya Pokharel (CS24BTKMU11002)
Surbhi (CS22BTECH11057)
Mane Pooja Vinod (CS22BTECH11035)

Date: 2nd May 2025

Contents

1 Team Contributions **2**

2 Test Cases Executed **2**

3 Use Cases Implemented **3**

4 Use Cases Not Implemented **4**

5 Total Lines of Code **4**

6 Total Number of Modules **4**

7 List of Modules and Their Lines of Code **4**

8 Software UI Screenshots **5**

1 Team Contributions

Priyanshu Sharma

Worked as a full-stack developer. Designed the backend architecture using secure APIs, handled database schema, integrated modules, and wrote core logic for appointments, notifications, and authentication.

Sadikshya Pokharel

Worked on the UI/UX aspect and frontend for patient, doctor, and hospital portals along with authentication and profiles. Handled form validations, user onboarding pages, and component design with reusable layouts.

Mane Pooja Vinod

Worked on the frontend for the notification and appointment sections. Tested all user flows thoroughly. Verified form validations, appointment booking scenarios, and reset-password functionalities.

Surbhi

Led system testing and maintained correlation between SRS and test plan. Collected all necessary information about the hospital and doctors. Managed test coverage reports and ensured database was populated correctly for the demo.

2 Test Cases Executed

The test plan included a total of 25+ test cases, categorized into unit, integration, and system testing. Most of them were passed successfully during dry runs. One major feature (Google Calendar) was not implemented.

Key test executions:

- Hospital Signup with valid/missing/invalid inputs — **Passed**
- Login with correct and incorrect credentials — **Passed**
- Appointment Booking and slot double-booking prevention — **Passed**
- Notification system to patient/doctor/hospital — **Passed**
- Google Calendar integration — **Failed** (Due to OAuth limitations)

- Password reset and hashing — **Passed**

3 Use Cases Implemented

Hospital Registration and Verification

Hospitals can sign up with proper registration documents. After submission, the system verifies them automatically based on uploaded credentials and allows login only after successful verification.

Patient Signup and Login

Patients can register and log in using a valid email and phone number. Credentials are securely hashed, and access tokens are generated upon login. Validations are added to ensure correctness.

Doctor Registration and Hospital Approval

Doctors can register independently but must link to a valid hospital. Hospital administrators verify the doctor's credentials and mark them as "verified", post which login is enabled.

Appointment Booking by Patient

Patients can view a list of verified hospitals and available doctors. They can select a doctor, date, and time to book an appointment. The system prevents overlapping or invalid bookings.

Appointment Confirmation by Hospital Admin

Hospital admins review pending appointment requests. Upon approval, the appointment status changes to "confirmed". Due to OAuth constraints, calendar syncing was replaced by email notifications.

Notification Handling

The system generates contextual notifications:

- Patients are notified about appointment status.
- Doctors are notified when a new appointment is made.
- Hospital admins receive verification and booking alerts.

Password Reset

Users can request password resets via email. OTP-based verification is used. New passwords are hashed using `bcrypt` before storing in the database.

4 Use Cases Not Implemented

Doctor Information Management by Hospital Admin

Originally planned to let hospital admins manage doctor profiles, this was changed based on TA/client feedback. Doctors now manage their own information directly.

Google Calendar Integration

OAuth restrictions and time limitations prevented the integration of Google Calendar APIs. Email-based notifications are used instead for confirmations and reminders.

Managing Diets and Prescriptions

Due to project complexity and time constraints, this use case could not be implemented. It would have allowed patients to track prescriptions, receive reminders, and view dietary guidelines.

5 Total Lines of Code

The codebase contains approximately **7,480 lines of code**, excluding build files and node dependencies.

6 Total Number of Modules

A total of **12 functional modules** were implemented across frontend and backend components.

7 List of Modules and Their Lines of Code

- **Authentication (Signup/Login)** – 750 LOC
- **Hospital Dashboard** – 800 LOC
- **Patient Dashboard** – 900 LOC

- **Doctor Dashboard** – 600 LOC
- **Appointment Booking System** – 300 LOC
- **Notification System** – 500 LOC
- **Admin Verification Logic** – 300 LOC
- **Token Management (JWT)** – 200 LOC
- **Password Reset and Hashing** – 300 LOC
- **Validation Utils** – 250 LOC
- **Email Service Module** – 480 LOC
- **UI Component Library (Reusable Widgets)** – 1200 LOC

8 Software UI Screenshots

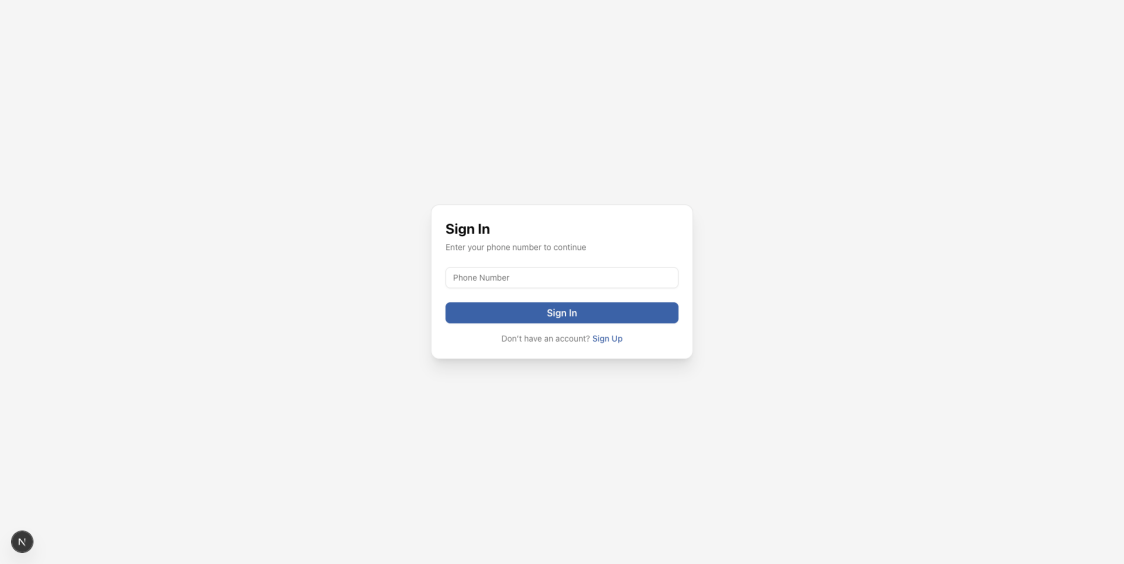

Figure 1: Signup Screen

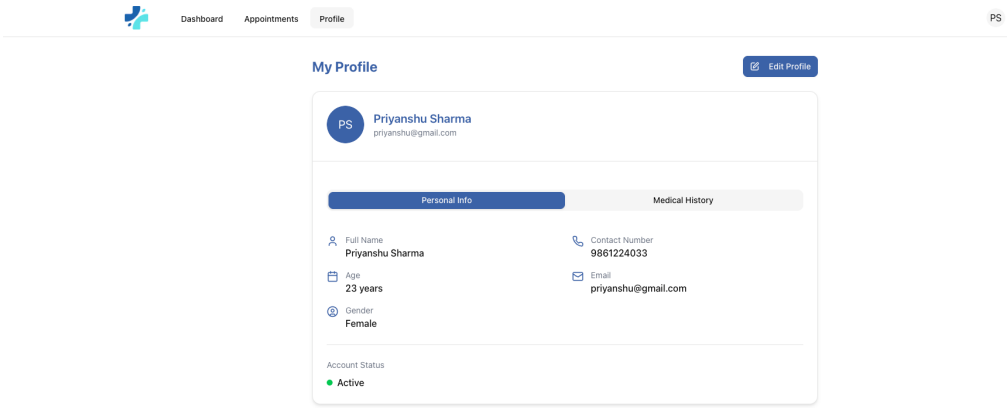

Figure 2: User Profile Screen

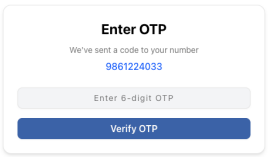

Figure 3: OTP Verification Screen

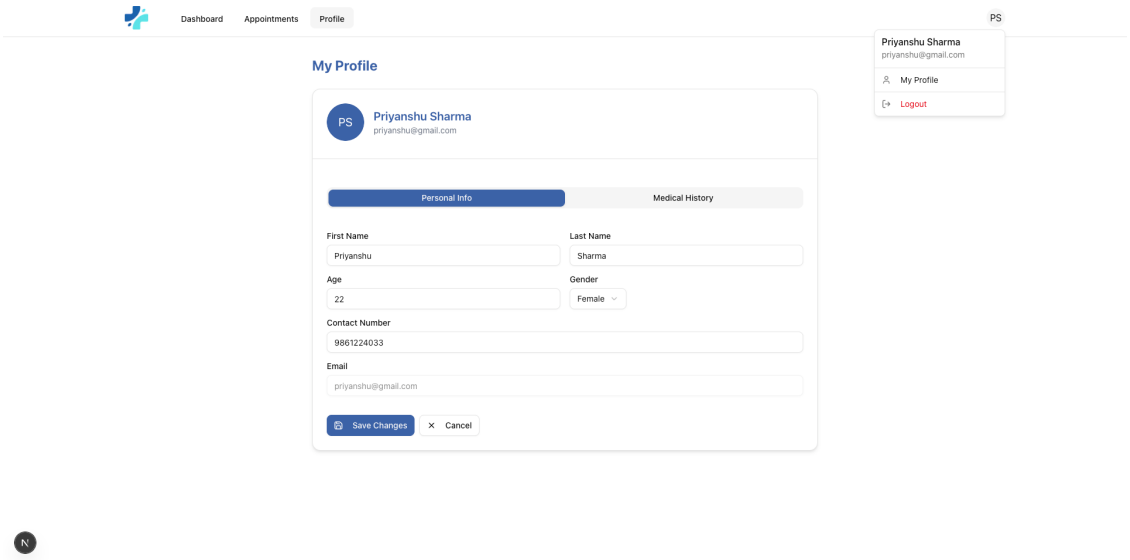

Figure 4: Logout Confirmation

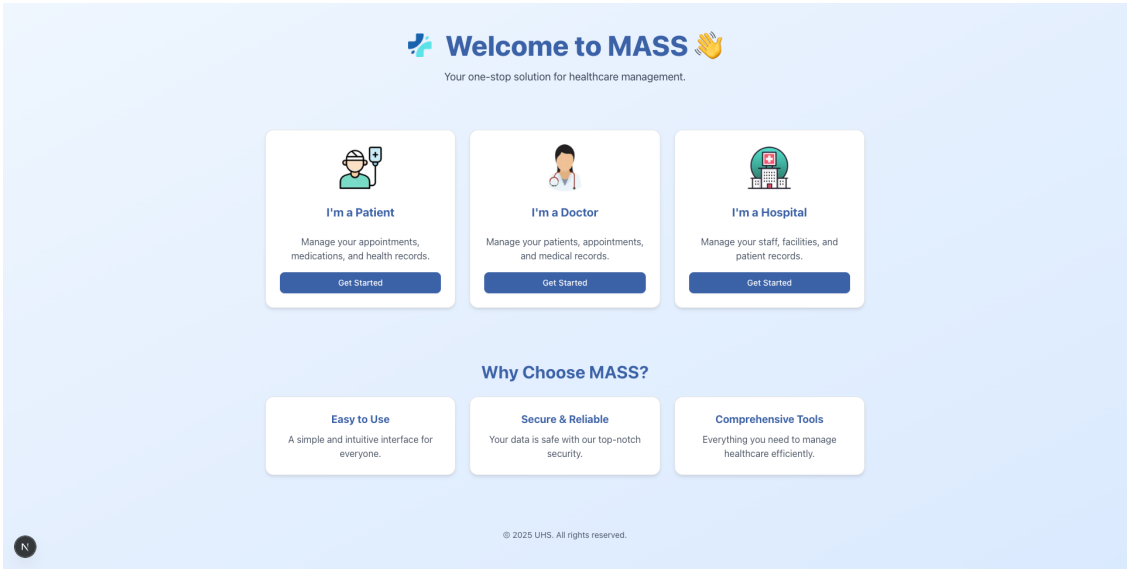

Figure 5: Landing Page

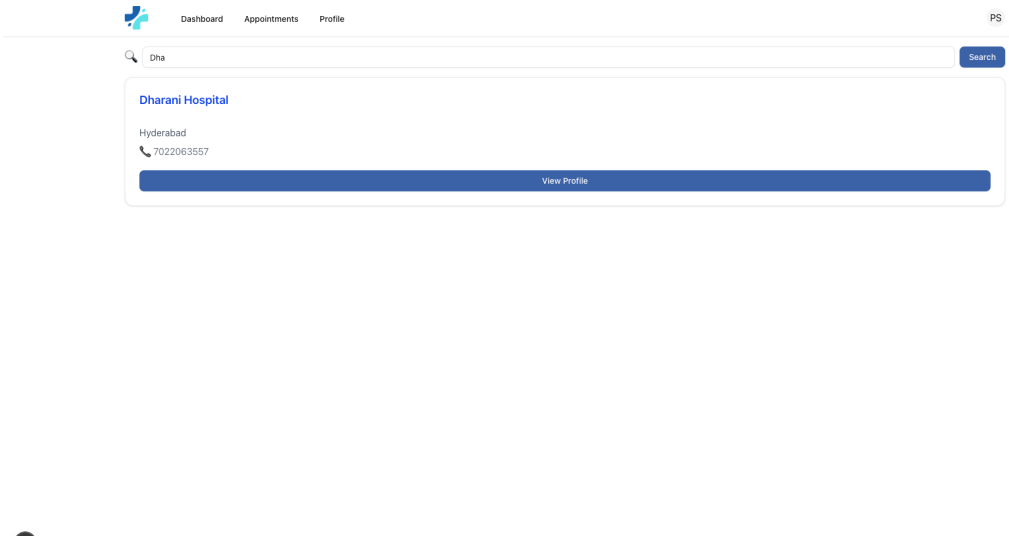

Figure 6: Search Hospital Screen

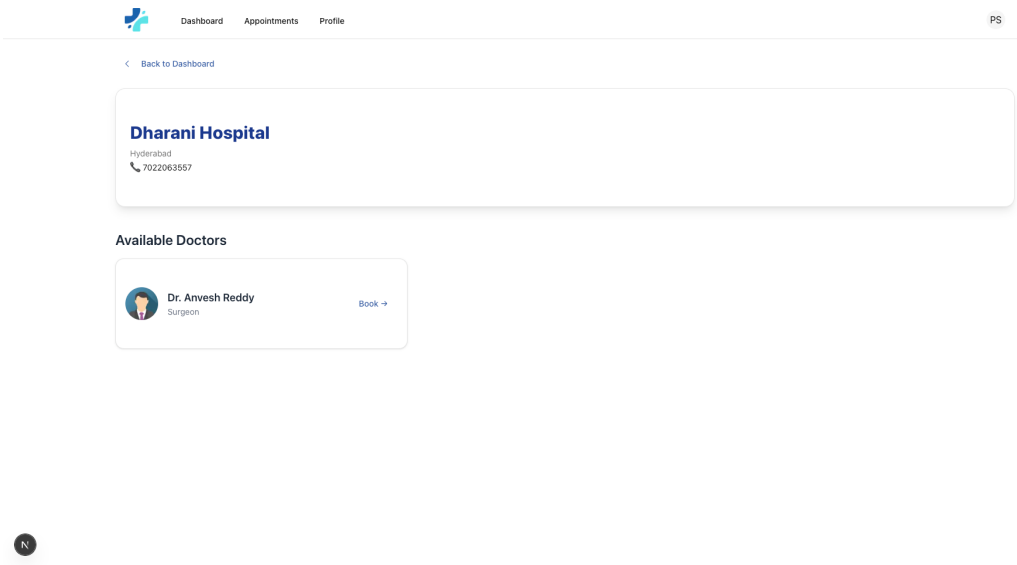

Figure 7: Doctors List in a Hospital

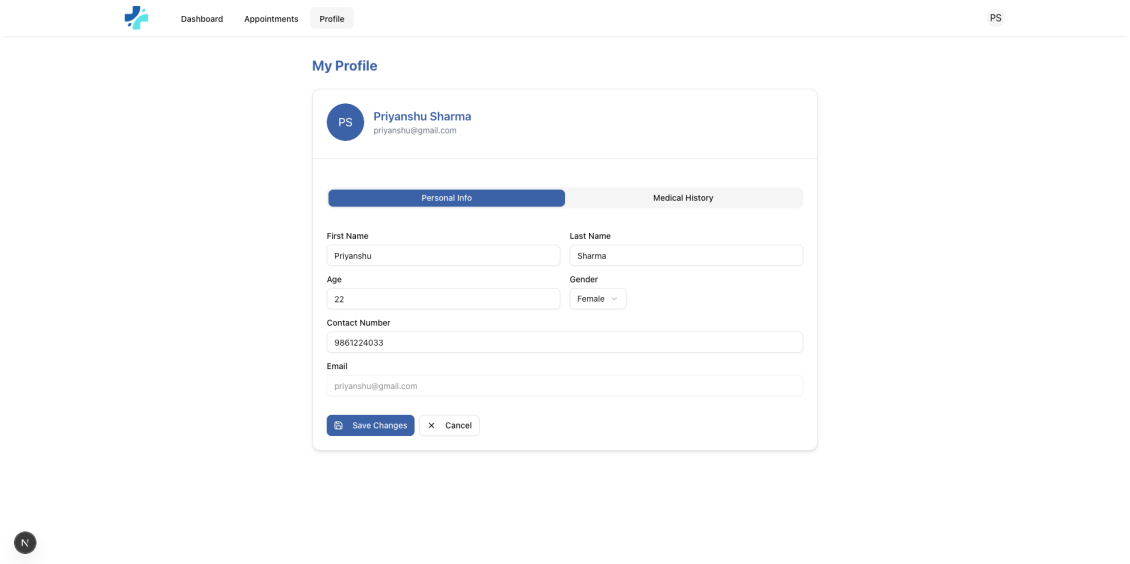

Figure 8: Edit Profile Interface

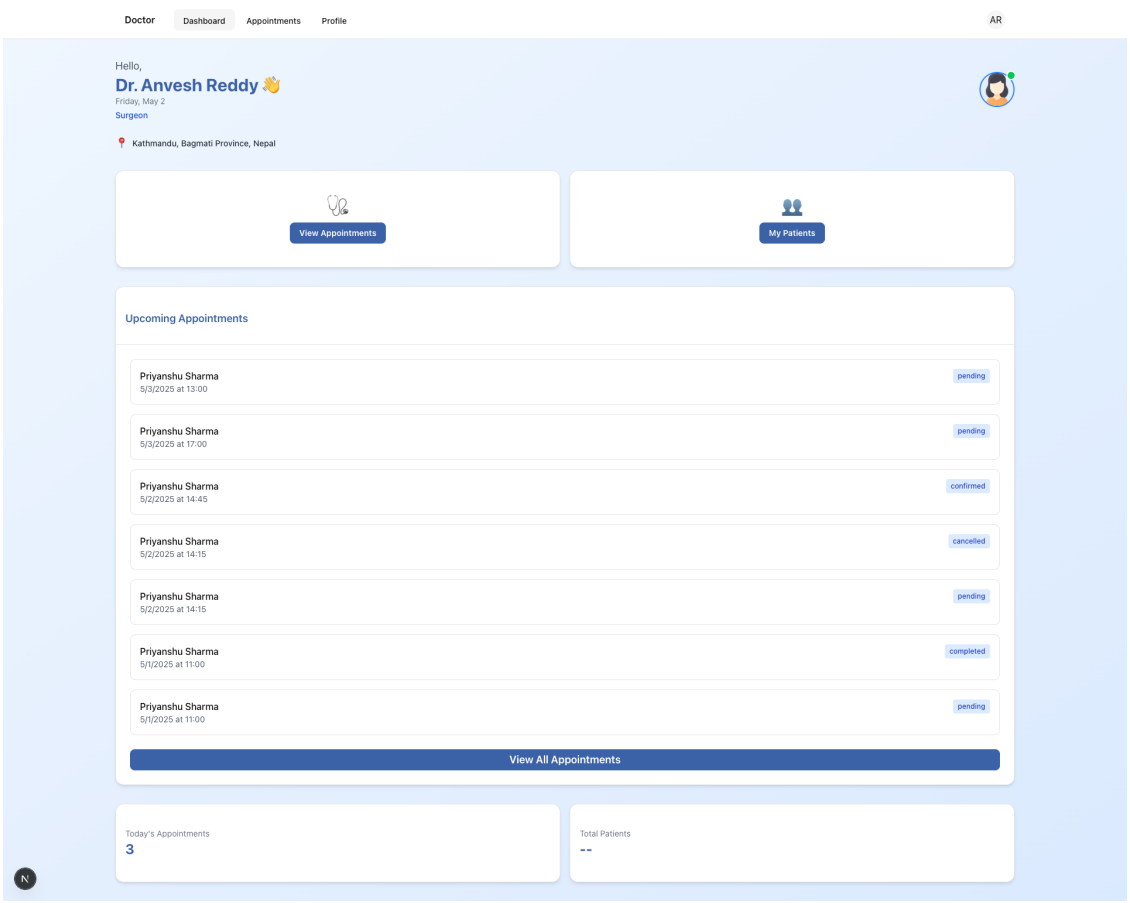

Figure 9: Doctor Dashboard View

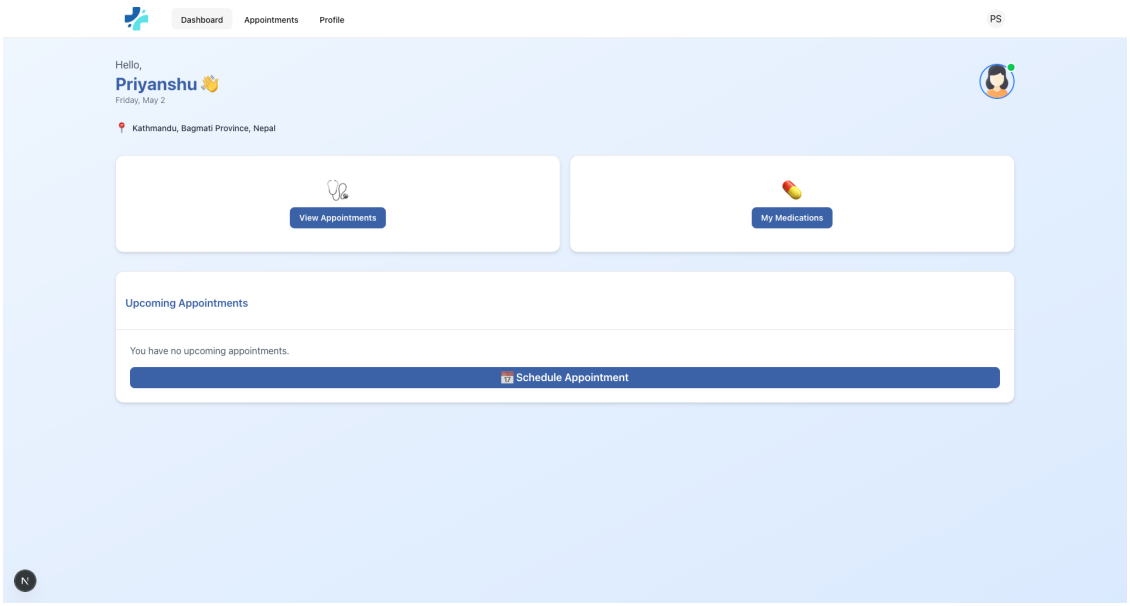

Figure 10: General Dashboard View

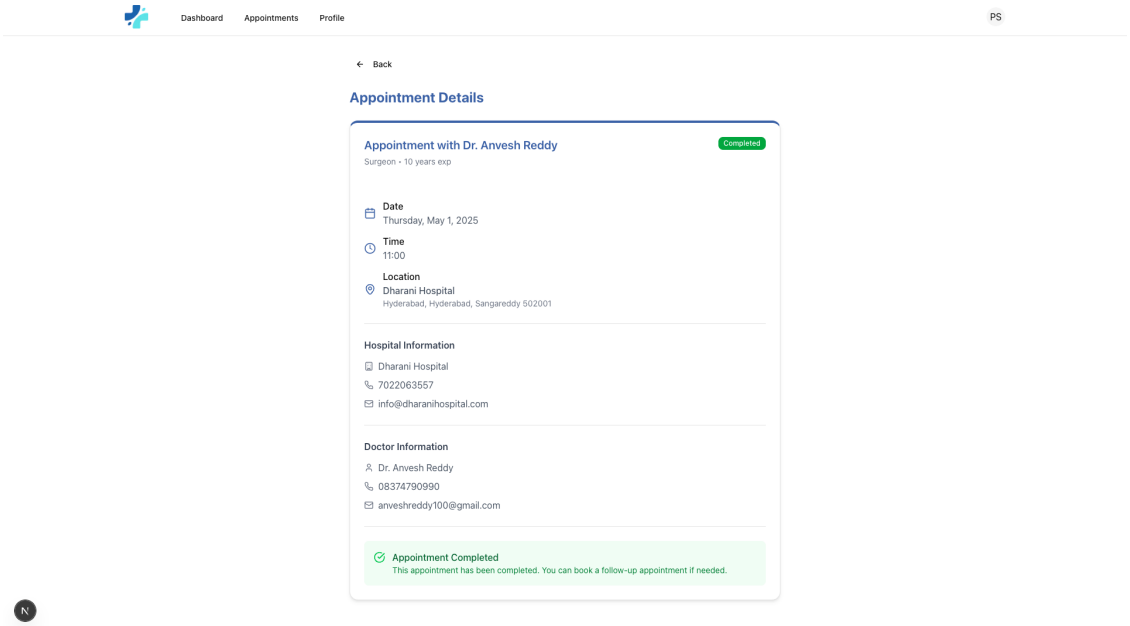

Figure 11: Completed Appointment View

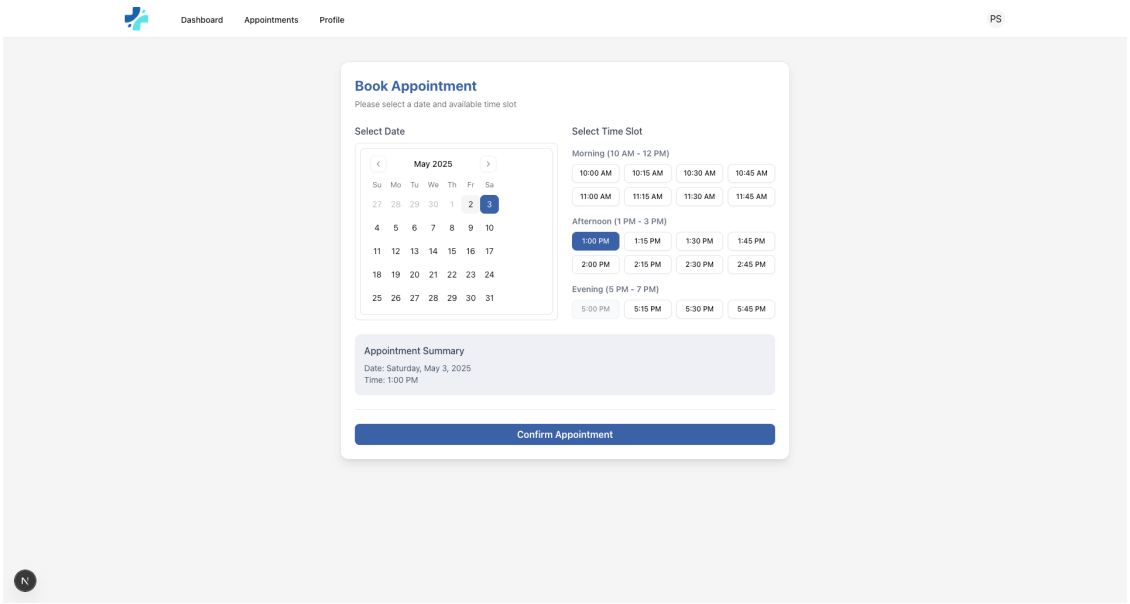

Figure 12: Book Appointment Screen

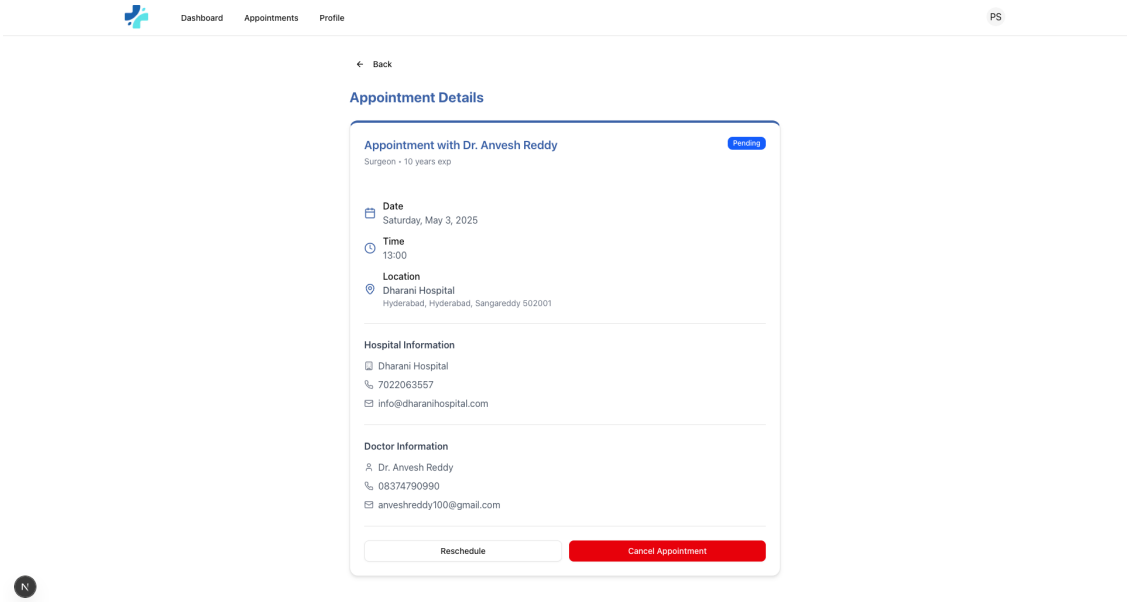

Figure 13: Appointment Details Screen

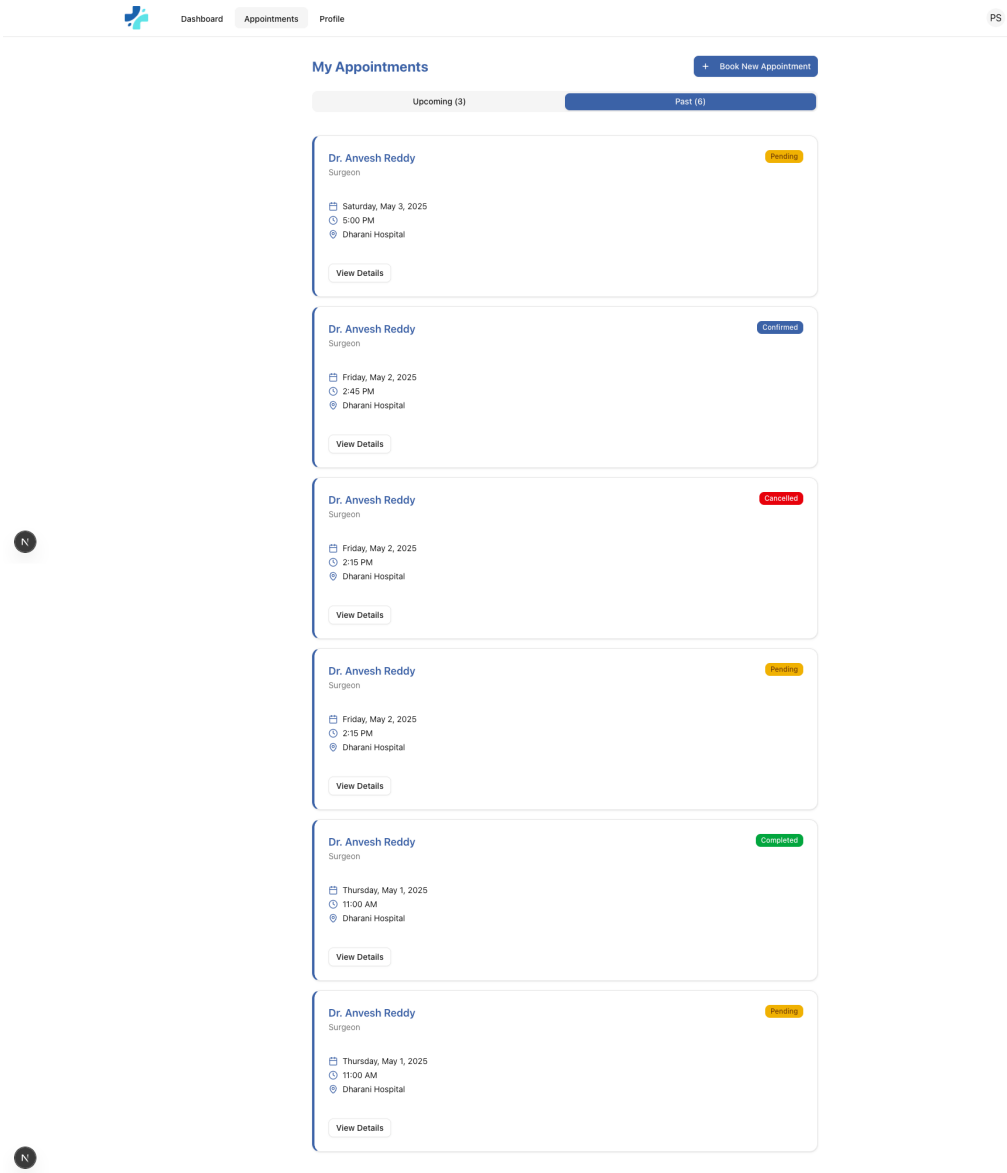

Figure 14: All Appointments List