Health Portal

(Health Care Service Web Application)

Submitted By

Tuhin

221-35-1017

Supervised By

Mr. A.H.M Shahariar Parvez

Associate Professor

Department of Software Engineering, Daffodil International University

This project report has been submitted for fulfilling the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF PROJECT AND COPYRIGHT

Author's Full Name :

Date of Birth :

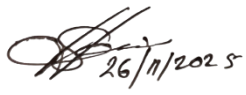
Title :

Academic Session :

I acknowledge that Daffodil International University reserves the following rights:

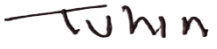
1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:



_____

(Supervisor's Signature)



_____

(Student's Signature)

A.H.M Shahariar Parvez

_____

Name of Supervisor

Date:

221-35-1017

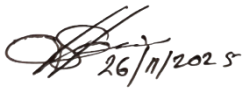
_____

Student ID

Date:

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project thoroughly and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



_____

(Supervisor's Signature)

Full Name : A.H.M Shahariar Parvez

Position : Associate Professor

Date :

Student’s Declaration

I am Tuhin and I declare that the project titled “Health Portal - Health Care Service Web Application” have been completed by me under the guidance of A.H.M Shahariar Parvez. The work is original and it has not been submitted anywhere else. All sources of information have been duly acknowledged.

_____

(Student’s Signature)

Full Name : Tuhin

ID Number : 221-35-1017

Date :

Health Portal
(Health Care Service Web Application)

Tuhin

Project submitted in fulfillment of the requirements for the award of the degree of Bachelor of Science

Department of Software Engineering

Daffodil International University

December 2025

Acknowledgement

First and foremost, I would like to sincerely thank Allah for giving me the strength and ability to complete this project.

I'm also really thankful to my supervisor Mr. A.H.M Shahariar Parvez of the Department of Software Engineering for his excellent guidance, invaluable advice and constant encouragement during the development of this project. His support was instrumental in getting through technical and conceptual challenges. I would also like to thank all those who participated in the testing and validation. Their enthusiastic involvement and insightful feedback were crucial in the development of the Health Portal system and made it practical to use.

Lastly, I'm very grateful for the unconditional love, kindness, support, and encouragement from my parents. Without their dua, guidance and patience, I would not be able to complete this project.

Dedication

I hereby declare that I did this project under the supervision of Mr. A.H.M Shahariar Parvez, Associate Professor, Department of Software Engineering, Daffodil International University. I also confirm that this work is wholly my own, and has not been submitted, either in whole or in part, to any other institution for a degree.

Abstract

The Health Portal is a central online platform that links all healthcare stakeholders in Bangladesh such as hospitals, patients, doctors and pharmacies. I developed it with reliable tools, such as Next.js and MongoDB, so that these groups of people will be able to easily and efficiently communicate with one another. The system gives information about a lot of public and private hospitals, along with secure methods to book appointments, manage doctors and issue online prescriptions. Patients can easily find reliable doctors, schedule visits, view past prescriptions and quickly find emergency contacts. They can do all this from their regular phone or computer browser. For the administrative users, there's a dashboard to manage doctors and view live appointments and create prescriptions that include a QR code that can be scanned. This QR code allows the pharmacies to immediately check the medication instructions by just scanning it. By bringing everyone on one single platform, the Health Portal eliminates paperwork and lengthy wait times. It helps in bringing reliable healthcare to more people, especially those living outside the big cities. Additionally, this project sets us up very nicely for future expansion, such as mobile apps and insurance support.

Table of Content

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Background

The healthcare system in Bangladesh is a mix of public and private hospitals, from giant hospitals in cities to small clinics in rural Bangladesh. This network should theoretically cover the entire country. But patients still find it difficult to figure out who's available when, how to book appointments or follow up on prescriptions. There's also the fact that many facilities are still stuck with fragmented digital tools or handwritten records. It feels frustrating that we are still at this stage. As digitalization spreads, the need for one reliable, easy-to-use, and user-friendly platform becomes crystal clear.

1.1.1 Context and Relevance

Recently, Bangladesh made some real progress in building new health infrastructure and improving services. Still, essential care is isolated in different places and is causing issues for both patients and providers. I've heard of people depending on personal recommendation or going to the hospital just to see if any specialist is available - because the appointment systems at most hospitals are not interoperable. A unified digital platform might solve all that madness as it would give everyone a consistent access to doctor info, scheduling tools and prescription records, and help create a more organized, patient-centric health ecosystem.

1.1.2 Problem Identification

I am truly confused by the disheveled state of healthcare management. Everyone is in a racce to locate a qualified doctor and make appointments without losing his or her head, and most of the time one would find himself in a clinic or would be making several phone calls. Then there is prescription management, there is always risk of losing or misunderstand a written paper script and pharmacies sometimes struggle with the handwritings. At the functional level, most hospitals lack centralized systems to manage appointments and monitor the availability of the doctor and patient records are not kept in proper order. All this issues reveals that we require a good solid centralized structure that will ensure that everything works in order and smoothly.

1.1.3 Purpose and Justification

The Health Portal is virtually the solution to the problems that I have been hearing about day after day. It provides an easy to use and clean digital place of residence to patients, doctors and the personnel staffing it. The platform joins major services features, such as reliable doctor information, a safe booking system and online digital prescription management. The patients are in fact able to browse through the profiles of doctors, book an appointment, and audit past

prescriptions whenever necessary. Admins receive tools that allow managing doctors signups, control clinical tasks and prescribe medications that can be found online with a single QR scan. Besides, the portal provides an immediate assistance option which is beneficial in both regular health visits and emergency cases.

1.1.4 Scope

This is a project on the creation of a user friendly healthcare web site for the public. It combines information about hospitals and convenient patient and employee applications. The user facing sections consist of a home page, doctor search and emergency help. After logging in, the user will be able to update the personal details and see all the prescription he was prescribed to. The hospital administrators remain in a different zone where they are able to check the doctor schedules, track appointments and assign prescriptions. The back end is based on specific APIs whose functions include sign in, profile management, booking appointments, doctor information management and prescription management. I do not deal with mobile applications, Web payments, or AI driven health tips at this time, those will be addressed in the future.

1.2 Project Planning and Initiation

It is important to do a formal planning step so that we can get the Health Portal to be implemented successfully and within a manageable time window. This is the step where the viability of the project is checked, the resources necessary are documented and the viability of the entire project is even verified before any development has begun. The feasibility study provides a methodical examination to ascertain that we will be able to formulate, launch and sustain the system excellently.

1.2.1 Feasibility Study

Operational

Certainly, this system will become an addition to existing workflows in hospitals where there will be no issues. The only thing the patients need is a regular web browser on a smartphone or a pc and adding a new doctor is a mere guided wizard. In essence, everyone is able to join in on this smoothly without requiring expertise in technology.

Technical

The tools that will be used are Next.js, MongoDB, and NextAuth. They are mainstream tools that have enormous communities that support them. So the development of the app is simple, patient data security is in control, and expansion to additional users is certainly part of the plan.

Economic

Initial investments are minimal, requiring cloud hosting and MongoDB Atlas. The number of

users can increase at a slow pace hence the costs will also increase at a slow pace making the whole thing affordable.

Schedule

I want to complete this in 14 weeks timeline dividing into many components. Every chunk is dedicated to a particular feature such as login, doctor management, appointment scheduling, prescription writing etc. So that we are on schedule and bring out the core functionality on time.

Legal & Ethical

The design has already taken into account patient privacy and consent where necessary. We will eventually achieve a full HIPAA level of compliance, but at the moment, the system is handling health data in a responsible manner and adhering to simple data-safety principles.

1.3 User Profile and Tentative Elicitation Process.

It is imperative to understand who uses the portal in order to create something that is really useful. The Health Portal is to be used by a wide range of stakeholders with their expectational needs, level of technological comfort and usage habits. In this case we outline the key user groups and summarize the way we gathered their requirements.

1.3.1 Target User

The site targets a broad community of individuals undertaking daily healthcare tasks. The largest group is patients in search of easy access to doctors and e-prescriptions. The doctors have the system to update their professional records and manage clinical contacts. Admins use the tools that are centralized to onboard a doctor, keep track of schedules, and track the activity of appointments. Pharmacies also have a less important, yet significant role, where they access QR-based access to verify prescription quickly.

1.3.2 User Profile

To make sure that my choices are actually reflecting the needs of the users, the following tables are given with details and attributes of each type of user for whom the project is will be designed.

Table 1: User Profile for Patients

Attribute	Details
User class	Patients and caregivers using Health Portal for day-to-day care
Note on characteristics	Highly motivated during illness, value transparency, and often juggle work and family duties
Type of user	Account-based consumer who needs appointments, prescriptions, and ambulance links
Age range	18–70, with the heaviest usage from caretakers aged 25–55
Frequency of use	Weekly to monthly; spikes during multi-visit treatment plans
Mandatory	Profile completion is required for bookings and prescription retrieval
Computer experience	Basic smartphone/desktop literacy; accustomed to chat apps and online forms
Goal	Discover qualified doctors, secure appointments, review digital prescriptions, and access emergency contacts
Language skills	Bangla-first audience; plain English labels acceptable
Number of users	Largest cohort, scaling into tens of thousands as hospitals onboard

Table 2: User Profile for Doctors

Attribute	Details
User class	Licensed physicians listed on Health Portal
Note on characteristics	Time-constrained professionals who value concise workflows and accuracy
Type of user	Verified practitioner with dedicated dashboard access
Age range	28–65
Frequency of use	Several times per week to adjust availability, confirm appointments, or submit prescriptions

Mandatory	Admin approval and credential verification before profile publication
Computer experience	Moderate; familiar with EMR portals, spreadsheets, and secure messaging
Education	MBBS or equivalent with postgraduate specialization
Goal	Maintain a trustworthy public profile, manage schedules, and record prescription instructions
Language skills	Fluent Bangla; professional proficiency in English for medical terminology
Number of users	Hundreds initially, growing with hospital partnerships
Training	Short video or quick-reference guide covering dashboard actions
Other system use	Hospital EMRs, WhatsApp/Viber groups, hospital rostering tools

Table 3: User Profile for Administrators

Attribute	**Details**
User class	Hospital operations managers, coordinators, and IT support
Note on characteristics	Detail-oriented, data-driven, responsible for compliance and smooth service delivery
Type of user	Elevated account with authority to onboard doctors, approve appointments, and manage prescriptions
Age range	25–55
Frequency of use	Daily. responsible for monitoring dashboards and responding to escalations
Mandatory	Role-based authentication, audit logging, and dual-approval for sensitive actions
Computer experience	High; comfortable with spreadsheets, CRMs, BI tools, and basic scripting

Education	Undergraduate degree in business administration, IT, or healthcare management
Goal	Keep doctor roster up to date, balance appointment load, ensure prescriptions are issued accurately, and resolve incidents
Language skills	Bangla and English; able to interpret compliance notices and system alerts
Number of users	Dozens per major hospital network
Training	Instructor-led onboarding session plus sandbox access for practice
Other system use	Hospital information systems, HR suites, ticketing tools, enterprise email

1.3.3 Elicitation Process

At first, I thought that I would never be able to develop a proper health portal. So I knew I had to get into the world of everyone.

Interviews: I conducted some interviews on a one-on-one basis, going to hospitals and speaking to patients, their families and even employees at the front desk. Immediately, the greatest points of frustration arose, such as the sheer inconvenience of making an appointment, panicking because crucial information is lost, and the way that prescriptions simply disappear once one is discharged. Most importantly, these discussions also focused on the overwhelming emotional burden on families, a factor that informed the supportive tone that I created as part of the user experience.

Surveys: I jumped from one hospital to another to interview doctors and administration personnel. This played a crucial role in obtaining solid information on what profile information cannot be compromised, the frequency of changes in the doctor schedule and the actual information that admins would require to be able to determine quickly which appointment requires urgent attention. The pure data of such surveys served as the real-life base of the default fields of doctor sign-up pages and the required reporting consoles of the administrative team.

Focus groups: This was done through joint meetings with pharmacy reps and IT to map the whole prescription verification process. This identified the necessity of a simple access system based on QR-codes that can be used on shared kiosks and older phones. The practical constraints, such as the slow rural internet were also revealed during the workshops directly informing the data caching and public prescription access strategy.

Comparison: I was not out to have a complete overhaul on its own account. I studied all available sites that are in operation in the hospitals as far as making and taking appointments are concerned and I even had to leaf through the antique manual logbooks. This study allowed us to identify best practice that is industry standard, the pitfalls that most users fall into and the regulatory boxes that I simply could not ignore. It proved to be the ideal contrast to my field work, indicating us what users were putting up with (although probably not willingly) at the moment, and what they had simply turned their backs on.

With all these personal stories and good hard figures, my information-gathering process resulted in a strong and workable plan that actually portrays the stressful, day-to-day healthcare conditions in this part of the world in Bangladesh.

1.4 Project Block Diagram

The diagram below represents the general layout of the Health Portal in that, the patient interface, administrative tools, backend services and database all interact with each other to create a system.

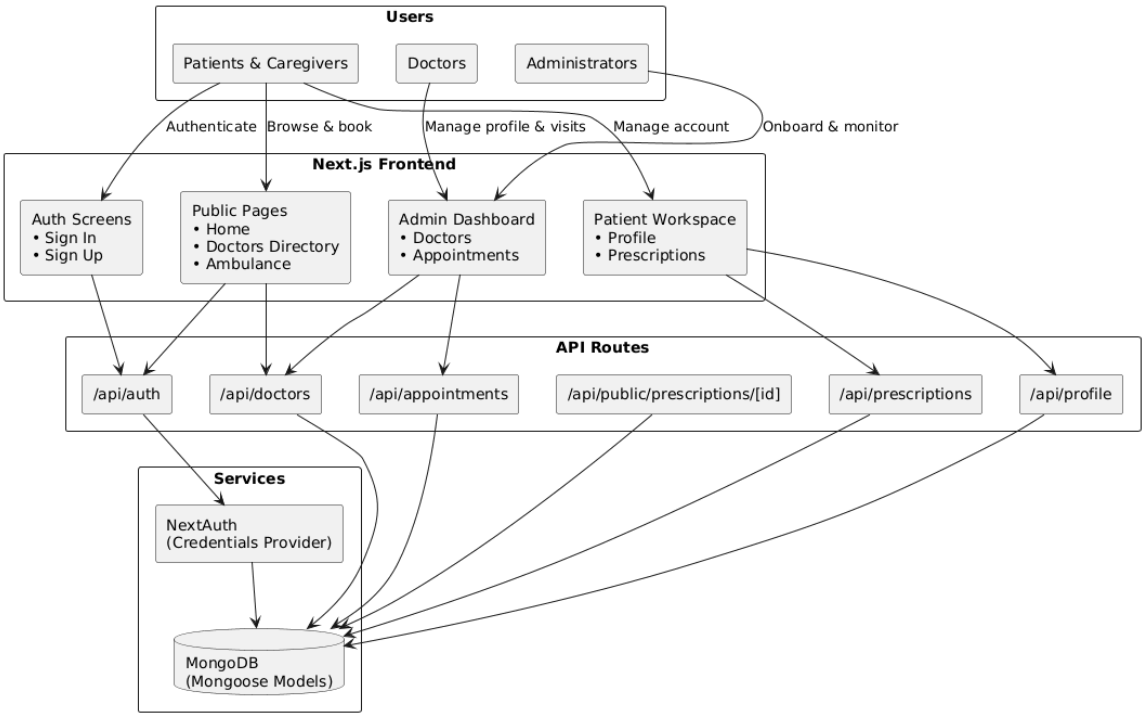


Figure 01: Block Diagram

Users Layer

Users Layer is the highest layer in the system and we have three types of users, they are patients, doctors and administrators. Admins can manage the accounts and onboarding and the doctors can manage their visits, and patients can browse and book appointments. The permissions allow each role to view certain sections of the application.

Next.js Frontend Layer

This layer serves as the user interface component of the system. It is divided into open sections such as Home page and Doctors Directory as well as secure auth screens where one enters to manage his/her dashboards and executive admins and patients have their own sets of spaces.

API Routes Layer

The API Routes Layer acts as the interface to the frontend and database as it interprets all data requests. It has certain endpoints including /api/auth where authentication happens and /api/appointments where business logic gets run before data is either retrieved or saved.

Services Layer

The back-end core which drives the system is the Services Layer. It will have NextAuth to verify credentials and manage sessions and MongoDB, which serves as a long-term database where doctor profiles, appointments, and prescriptions are stored.

1.5 System Requirements

In this section, hardware, software, and operational constraints that are required to develop, deploy, and maintain the Health Portal system are enumerated. Such needs can be used to make sure the platform operates well and remains accessible by all groups of users.

1.5.1 Hardware Requirements

Development Environment

Development and testing requires any typical modern work station. The minimum specifications will be:

- Modern Intel, AMD, Snapdragon or Apple Silicon CPU
- 8 GB RAM or higher
- At least 2 GB of free disk space for project dependencies and build files

Deployment Environment

The site is hosted on cloud infrastructure:

- Hosting Platform: Vercel (serverless environment)
- Database: MongoDB Atlas M0 or higher tier

This configuration removes the necessity for on site hardware. This also has auto scaling feature.

1.5.2 Software Requirements

Operating System

Compatible with:

- Windows
- macOS
- Any Linux distributions

Development Tools

- Node.js 22+ and npm
- MongoDB Atlas or MongoDB Compass
- IDE: Visual Studio Code or any equivalent modern code editor
- Version Control: Git (GitHub, GitLab, or Bitbucket)

Frameworks and Libraries

The platform is developed using the following modern web technologies:

- Next.js 15 (App Router)
- React 19
- TypeScript
- NextAuth (for authentication)
- Tailwind CSS 4 (for styling)
- Mongoose 8 (for database modelling)
- react-qr-code (for QR-based prescription sharing)

1.5.3 Constraints and Dependencies

To develop and deploy the system, the following dependencies and operational constraints were required:

Internet Connectivity

The system depends on a cloud-based MongoDB database and to deploy in Vercel through github a stable internet connection is required. for both development and production use.

Authentication Configuration

NextAuth requires the following correct configuration of:

- NEXTAUTH_SECRET
- JWT signing settings

An incorrect configuration will prevent users from logging in, session handling or prevent the users from entering protected routes.

QR Code Generation

The web application relies on the react-qr-code library to process prescription generation and sharing of prescriptions. But it may not work in the cases when the library is updated or when it is incompatible with certain web browsers on which the QR generation should be done.

Build and Type Checking

Next.js provides the rules of strict TypeScript, ESLint to keep the code intact. Every code should be based on those regulations in order to make the project constructed and implemented.

1.6 Project Scheduling

We will be having each sprint take two weeks, which is also approximately a semester. The incremental development model of the project, that is, each sprint will allow us to have a working set of features which will lead to the final system being constructed.

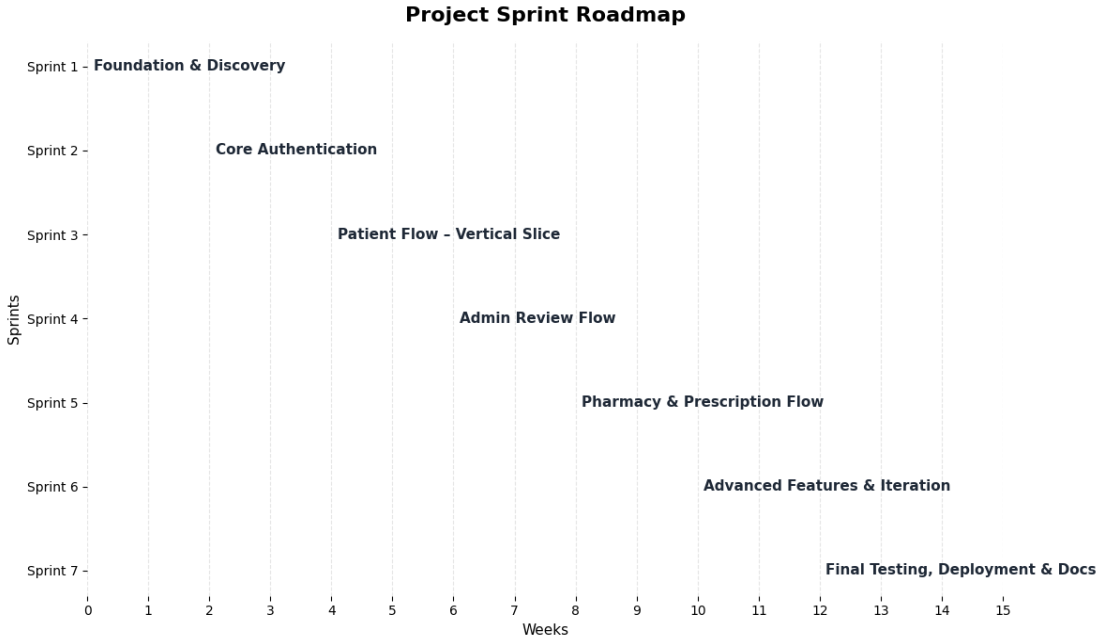


Figure 02: Project Sprint Roadmap

Sprint 1 (Weeks 1–2): Foundation & Discovery
Gather all the necessary information and requirements, set up the github repository and utilize continuous integration continuous development workflows.

Sprint 2 (Weeks 3–4): Core Authentication
Use NextAuth to build the authentication system. Add pages such as sign in, sign up.

Sprint 3 (Weeks 5–6): Patient Flow – Vertical Slice
Develop doctors directory with search and filtering options. Add a modal for booking appointments and integrate basic user profile.

Sprint 4 (Weeks 7–8): Admin Review Flow
Create the admin dashboard with doctor management.

Sprint 5 (Weeks 9–10): Pharmacy & Prescription Flow
The appointment section needs to be developed, assign prescription modal and implement QR rendering.

Sprint 6 (Weeks 11–12): Advanced Features & Iteration
Add details to user profile, add prescription in profile, update the homepage and integrate emergency Contact section.

Sprint 7 (Weeks 13–14): Final Testing, Deployment & Documentation
Before deploying the project into production, test the project thoroughly, verify the builds and prepare full project documentation.

1.7 Summary

This chapter told shared insighst about why I started the Health Portal project and how I planned it. I also talked about the important people involved, what the project features are and the process I followed to make good early choices. My plan uses agile development cycles called sprints. This keeps the project moving smoothly toward a dependable and growing platform for managing healthcare. The next chapters will explain how the system works, how I built it, how I checked it for errors, how I put it online, and how people can use it.

CHAPTER 2: DESIGN AND IMPLEMENTATION

2.1 Introduction

This part of the document explains what the Health Portal needs to do (its features) and how it should work (the non-functional requirements). It also includes the system's design, shown with diagrams called UML diagrams. The main purpose here is to show exactly how the features I planned will become a real, structured system and design.

2.2 Functional Requirements

The following Table shows the functional requirements which define key behaviours of the system.

ID	Requirement	Description	Primary Actors
FR01	User Authentication	The system must allow users to register, sign in with credentials, and maintain authenticated sessions securely using NextAuth's JWT strategy.	Patients, Doctors, Administrators
FR02	Doctor Directory	Patients should be able to browse, search, and filter doctors by name, specialization, experience, hospital, and consultation fee.	Patients
FR03	Appointment Booking	Authenticated patients can schedule appointments with available doctors by choosing date, time, and optional visit reason; admins can book on their behalf.	Patients, Administrators
FR04	Admin Doctor Management	Administrators can create doctor accounts, update profile details, toggle active status, and deactivate doctors when necessary.	Administrators
FR05	Prescription Handling	Admins or authorized doctors can create prescriptions tied to appointments; patients can view them, and pharmacists can verify them via QR/public link.	Doctors, Administrators, Patients

FR06	Patient Profile Management	Patients can view and update personal information such as contact details, address, blood group, emergency contacts, and medical notes.	Patients
FR07	Appointment Oversight	Administrators can review all appointments, filter by doctor or status, update statuses (confirmed, completed, cancelled), and attach prescriptions.	Administrators
FR08	Public Prescription Share	Each prescription must expose a secure public endpoint (and QR code) so pharmacies can read instructions without logging into the system.	Patients, Pharmacists
FR09	Emergency Support	The platform must provide a configurable ambulance page with a tel link so users can quickly dial the emergency hotline from any device.	Patients, Caregivers
FR10	Role-Based Access Control	Navigation and API endpoints must enforce role-specific access (patient, doctor, admin) to ensure sensitive dashboards and data remain protected.	All Authenticated Users
FR11	Dashboard Metrics	The homepage and admin dashboard must surface aggregate metrics (active doctors, upcoming appointments, total prescriptions) sourced from live data.	Administrators, Visitors
FR12	Session Awareness in Navbar	The global navigation must reflect the user's session state, showing relevant links (Profile, Dashboard, Sign out) or guest options when unauthenticated.	All Users

Table 04: Functional Requirement

2.3 Non-Functional Requirements

2.3.1 Performance

The system should respond to dashboard related API calls within 500 milliseconds during normal activity. Public pages such as Home, Doctors, and Ambulance should load within two seconds on standard broadband connections.

2.3.2 Reliability

MongoDB sessions must maintain data integrity when handling appointments and prescriptions, ensuring each document follows ACID properties. Serverless functions should retry operations automatically if a temporary network issue occurs during communication with MongoDB Atlas.

2.3.3 Portability

The web application must run smoothly across common modern browsers including Chrome, Firefox, Edge, and Safari. It should also deploy without major changes on Vercel or any hosting environment supporting Node.js 18 or newer. Environment configuration should remain portable through the use of .env.local files.

2.4 Object-Oriented System Design using UML

2.4.1 Use Case Diagram

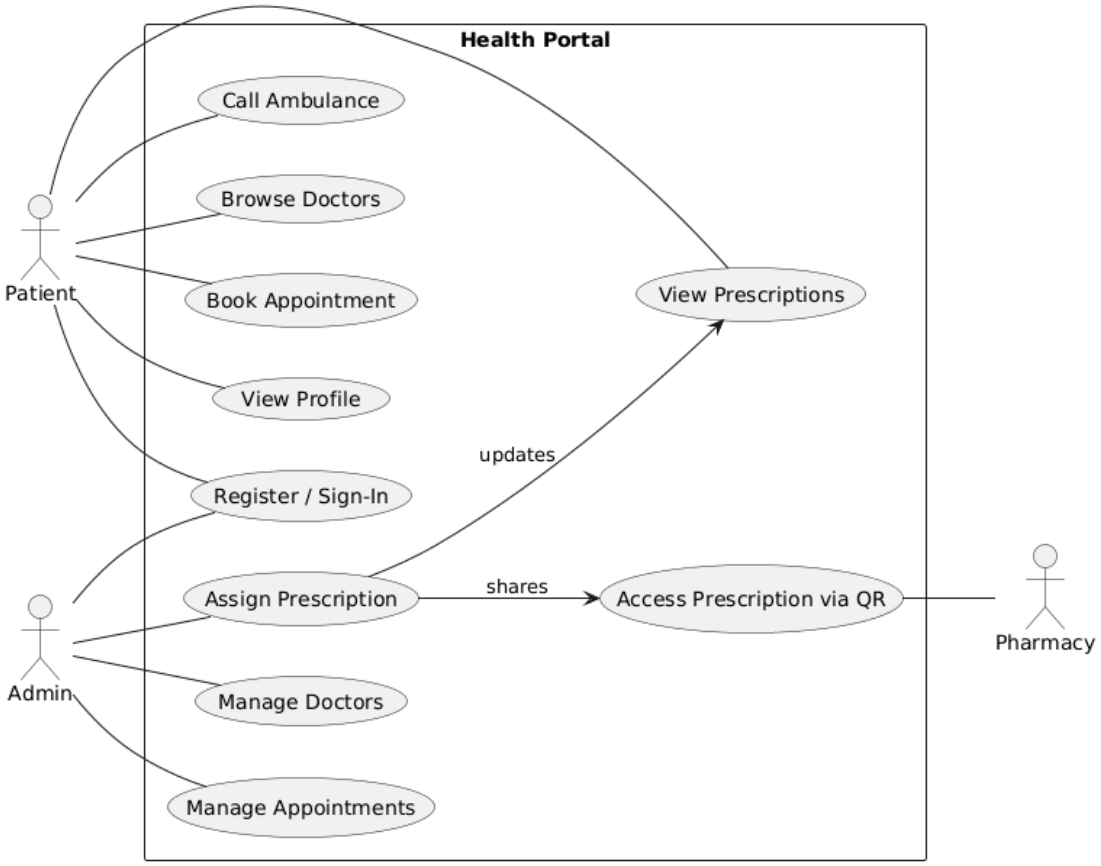


Figure 03: Use Case Diagram

2.4.2 Use Case Descriptions

No.	Use Case	Primary Actors	Preconditions	Summary of Flow & Outcome
	Register / Sign-In	Patient, Admin	User is not authenticated	Enter credentials → validate → establish session; invalid credentials return error.

	Quick Demo Login	Patient, Admin	Demo credentials configured	Click demo button → auto login → redirect to role page; missing credentials show warning.
	Browse Doctors	Patient, Visitor	Active doctors exist	Open directory → apply filters → view cards with hospital/location; empty state if no match.
	Book Appointment	Patient	Patient signed in; doctor active	Pick doctor → enter date/time/reason → submit → appointment created (pending); invalid input blocked.
	Manage Doctors	Admin	Admin signed in	Use admin tab to add/update doctors (hospital, location, fee) and toggle status; validation enforced.
	Manage Appointments	Admin	Admin signed in	Filter by doctor → review appointments → assign prescriptions or cancel as needed.
	Assign Prescription	Admin	Appointment exists (pending/confirmed)	Enter instructions → save → appointment marked completed + QR generated for patient.
	View Profile	Patient	Patient signed in	Edit personal/contact info → save → data updated in MongoDB.

	View Prescriptions	Patient	Patient signed in	Open prescriptions tab → see QR-coded history; empty state if none.
	Access Prescription via QR	Pharmacy, Patient	Valid prescription ID	Scan QR / open link → server returns read-only prescription details; invalid ID shows 404.
	Call Ambulance	Patient, Visitor	Emergency number configured	Tap call button → device dialer opens with configured number.

Table 5: Use Case Description

2.4.3 Activity Diagram

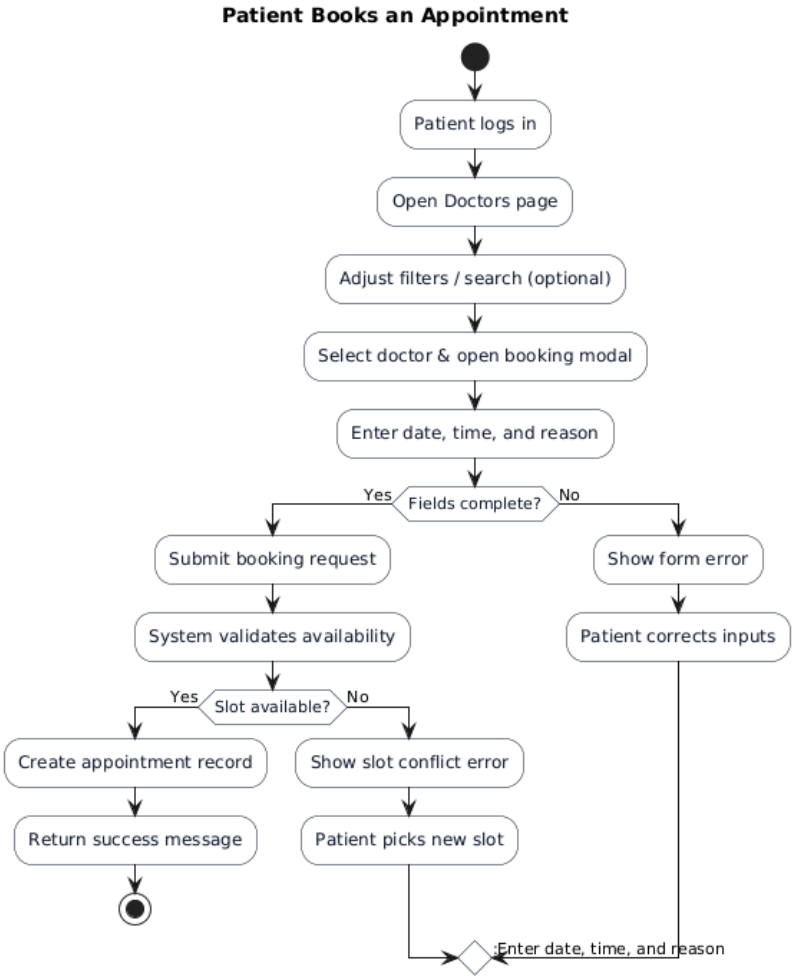


Figure 04: Patients Booking Appoinment Activity Diagram

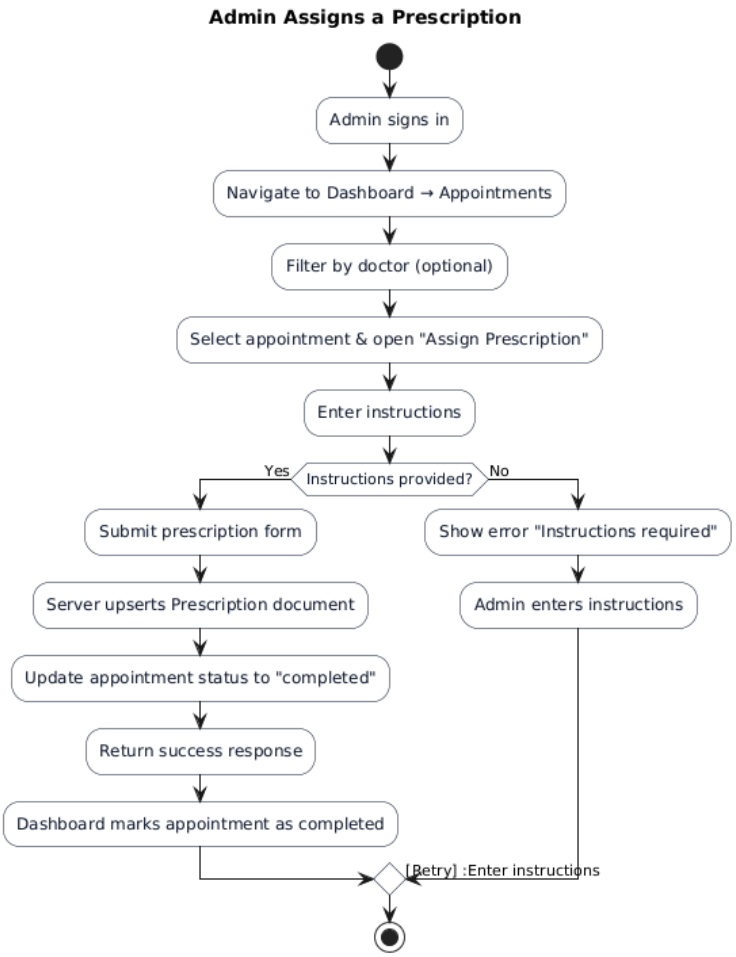


Figure 05: Admin Assigns Prescription

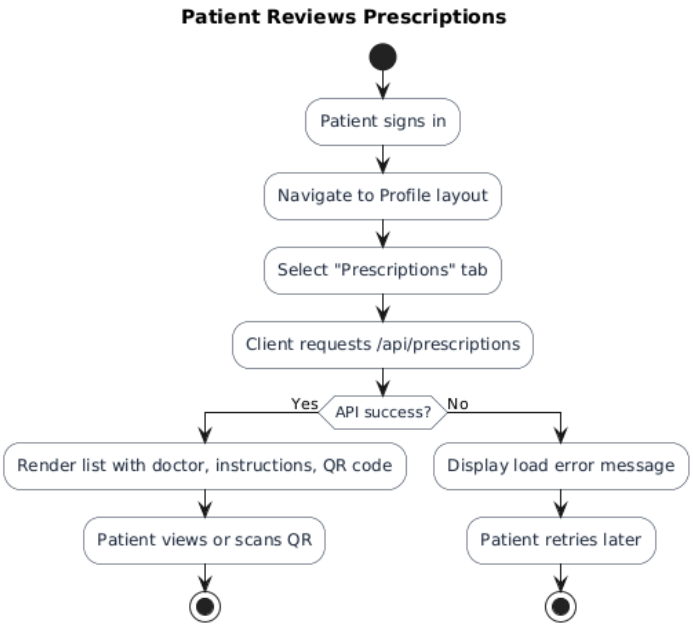


Figure 06: Patient Reviews Prescription

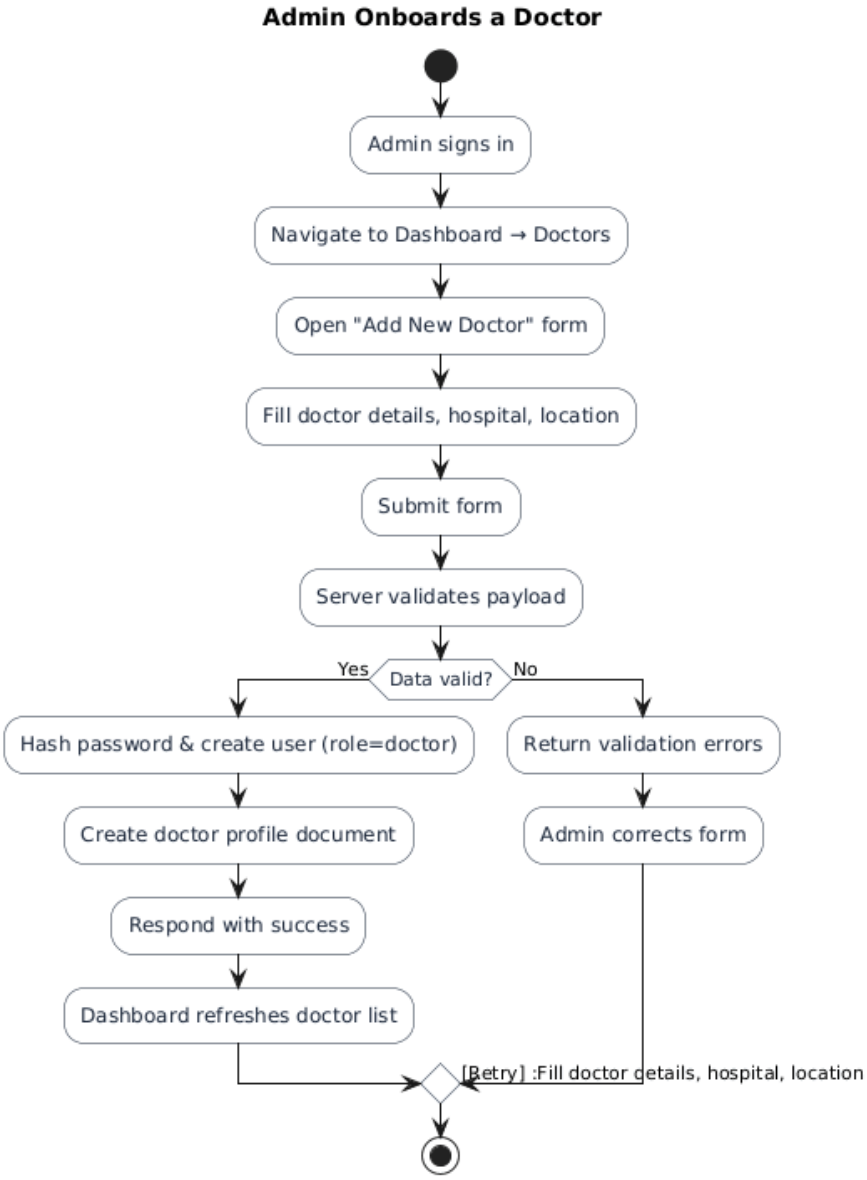


Figure 07: Admin adds a doctor

2.4.4 Sequence Diagram

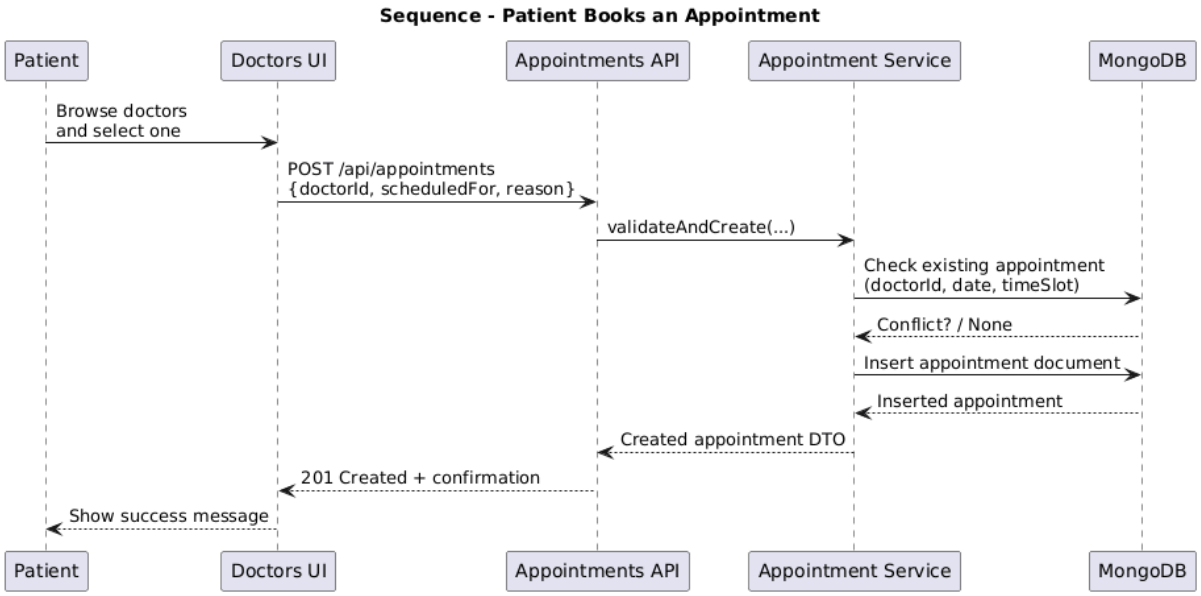


Figure 08: Patient Booking appoinment Sequence Diagram

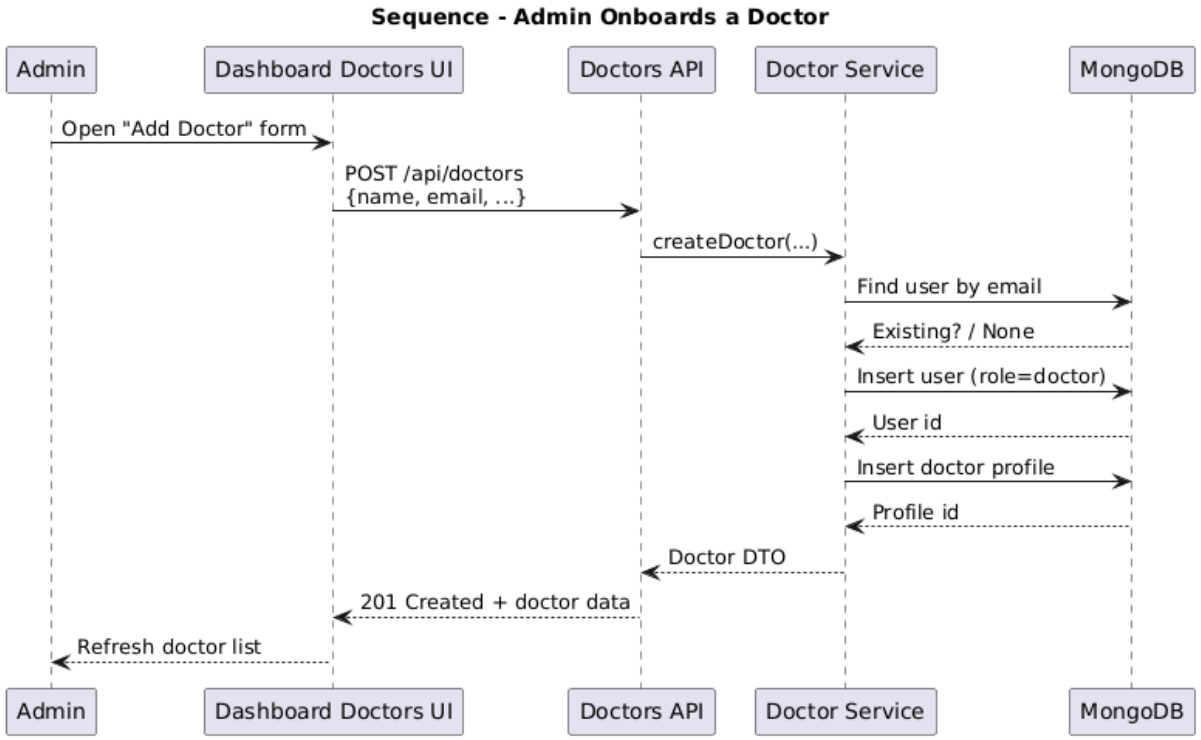


Figure 09: Admin Adds a doctor

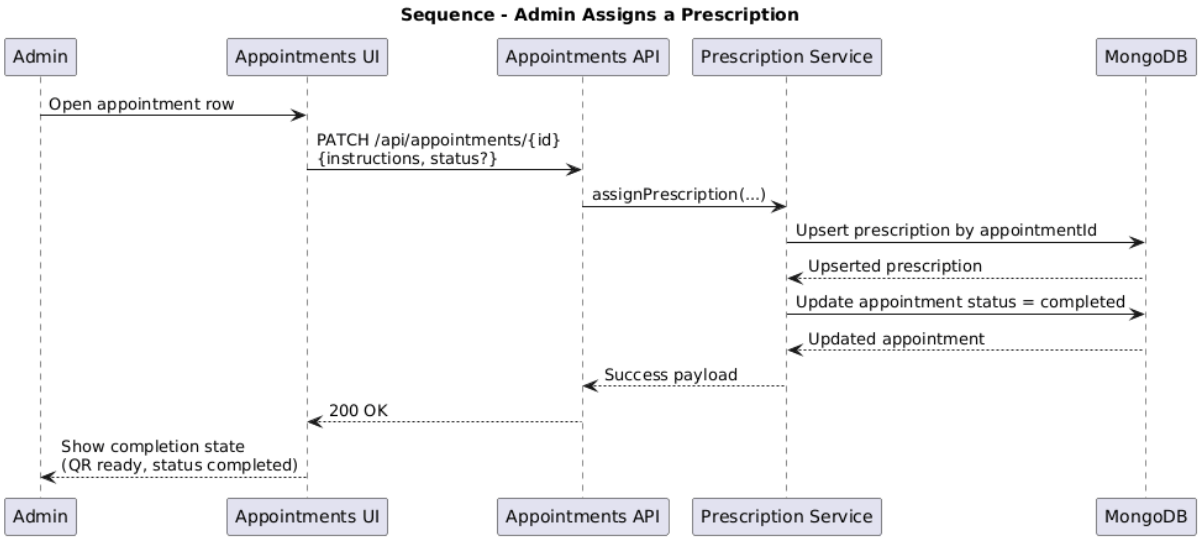


Figure 10: Admin assigns prescription

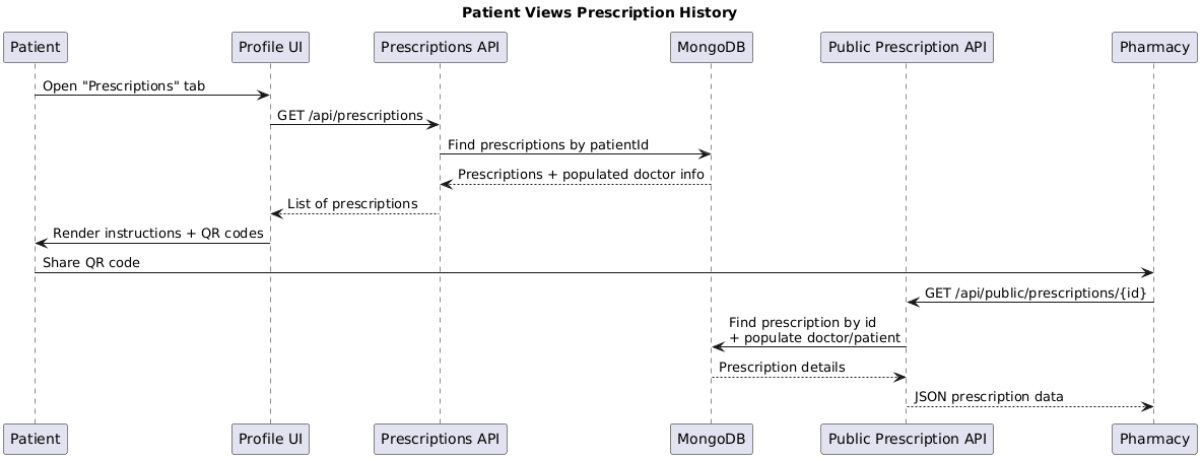


Figure 11: Patient views Prescription

2.4.5 Class Diagram (Core Domain)

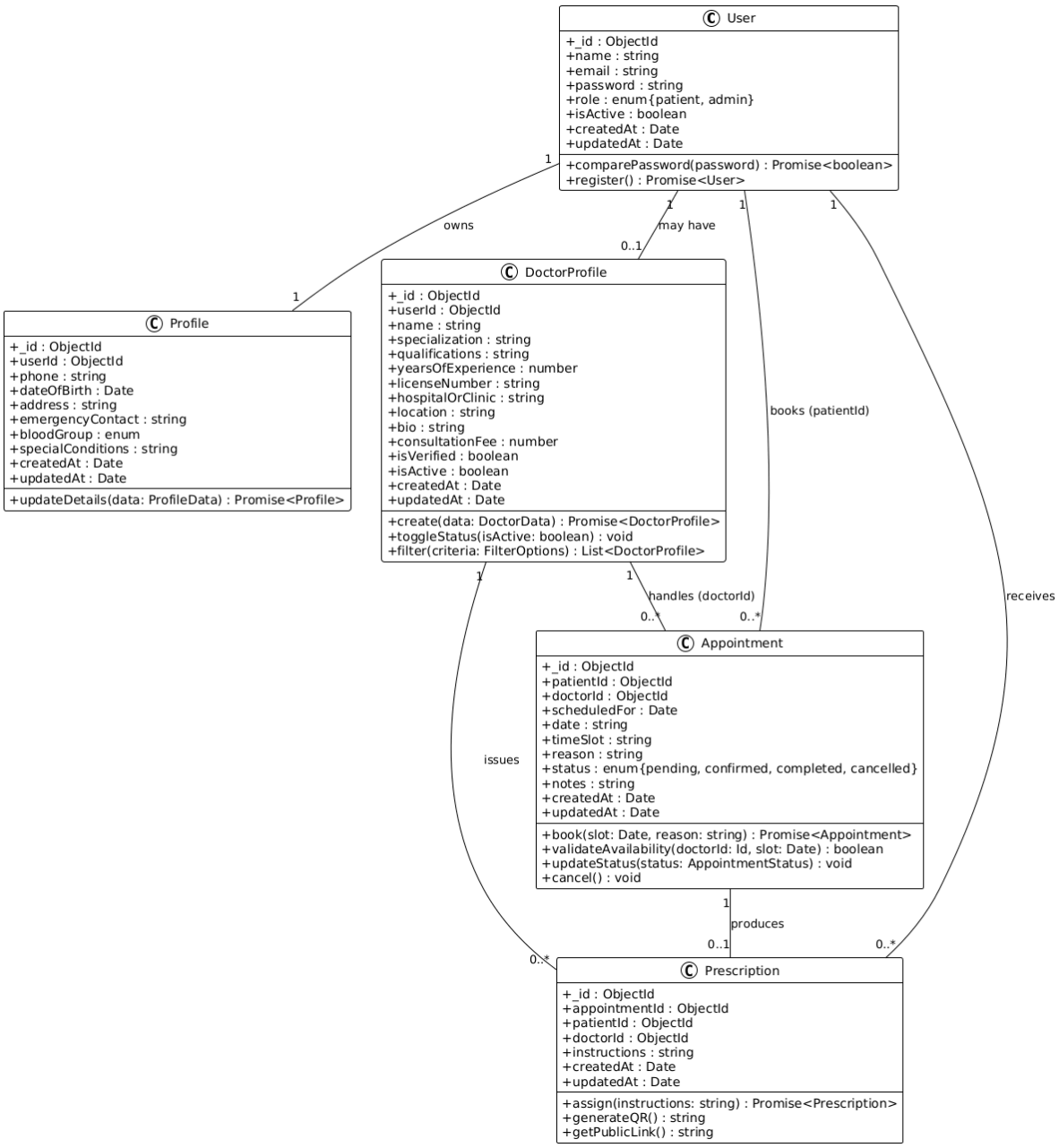


Figure 12: Class Diagram

2.4.6 ER Diagram

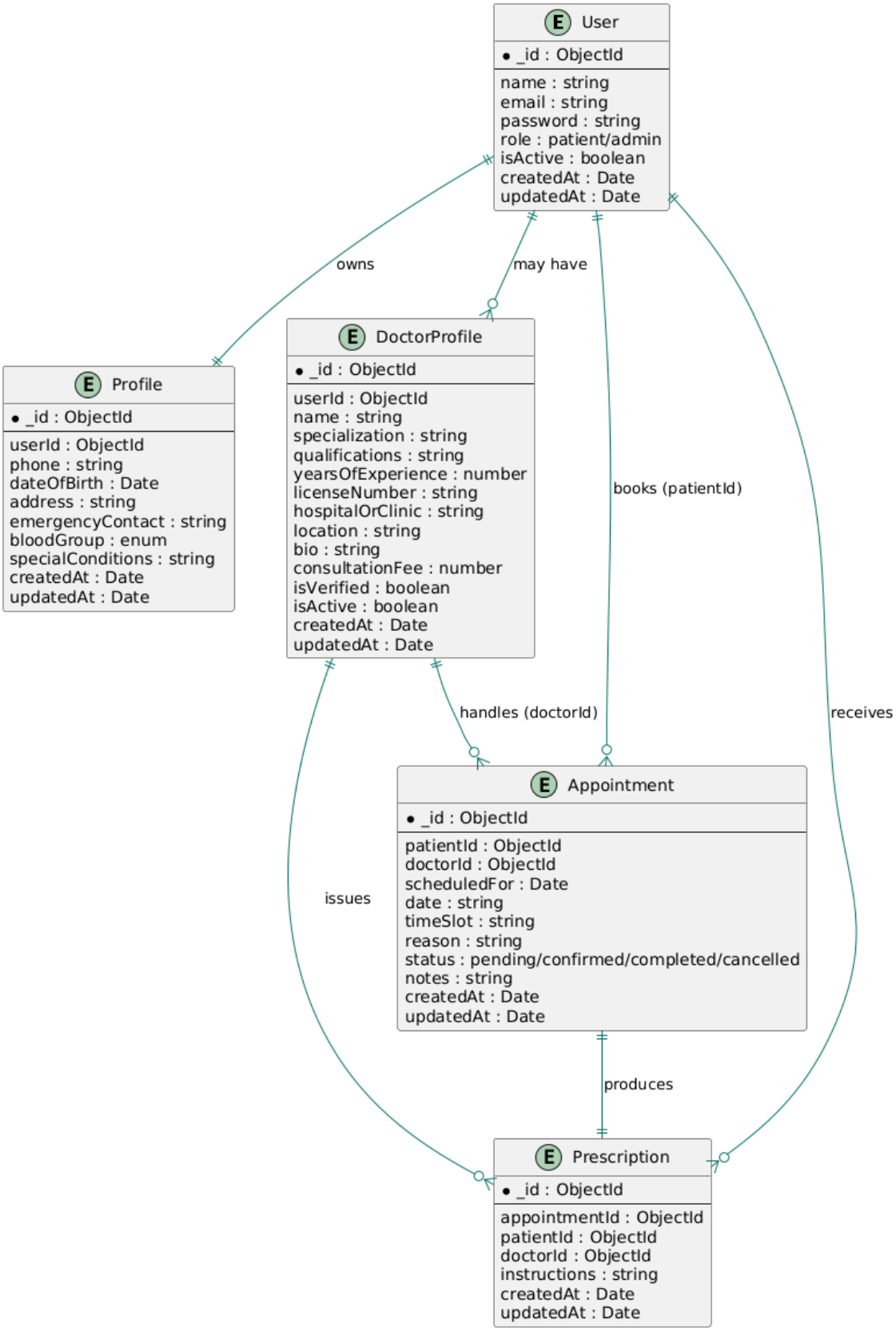


Figure 13: ER Diagram

2.5 Appendix A: Sample Code Snippets

Below are representative excerpts from the Health Portal codebase. They illustrate how authentication, profile updates, doctor management, and appointment handling were implemented.

A1. NextAuth Configuration (src/auth.ts)

```
import "server-only";
import NextAuth from "next-auth";
import Credentials from "next-auth/providers/credentials";
import bcrypt from "bcryptjs";
import User from "@/models/user";
import connectDB from "@/lib/mongodb";
export const { handlers, signIn, signOut, auth } = NextAuth({
  providers: [
    Credentials({
      credentials: {
        email: { label: "Email", type: "email" },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        await connectDB();
        const user = await User.findOne({ email: credentials?.email, isActive:
true });
        if (!user) {
          throw new Error("User not found or inactive");
        }
        const validPassword = await bcrypt.compare(
          credentials?.password ?? "",
          user.password
        );
        if (!validPassword) {
          throw new Error("Invalid password");
        }
        return {
          id: user._id.toString(),
          name: user.name,
          email: user.email,
          role: user.role,
        };
```

```
      },
    }),
  ],
  session: {
    strategy: "jwt",
  },
  callbacks: {
    async jwt({ token, user }) {
      if (user) {
        token.id = user.id;
        token.role = (user as any).role;
      }
      return token;
    },
    async session({ session, token }) {
      if (token?.id) session.user.id = token.id as string;
      if (token?.role) session.user.role = token.role as "admin" | "patient";
      return session;
    },
  },
  pages: { signIn: "/auth/signin" },
});
```

A2. Profile Update API (src/app/api/profile/route.ts)

```
import { NextRequest, NextResponse } from "next/server";
import { auth } from "@/auth";
import connectDB from "@/lib/mongodb";
import User from "@/models/user";
import Profile from "@/models/profile";
import DoctorProfile from "@/models/doctorProfile";
export async function PUT(request: NextRequest) {
  try {
    const session = await auth();
    if (!session?.user?.id) {
      return NextResponse.json({ error: "Unauthorized" }, { status: 401 });
    }
    const body = await request.json();
    const { name, profile } = body;
    await connectDB();
```

```
    if (typeof name === "string" && name.trim()) {
      await User.findByIdAndUpdate(session.user.id, { name });
    }
    if (profile) {
      await Profile.findOneAndUpdate(
        { userId: session.user.id },
        {
          $set: {
            phone: profile.phone ?? "",
            dateOfBirth: profile.dateOfBirth ?? null,
            address: profile.address ?? "",
            emergencyContact: profile.emergencyContact ?? "",
            bloodGroup: profile.bloodGroup ?? "",
            specialConditions: profile.specialConditions ?? "",
          },
          $setOnInsert: { userId: session.user.id },
        },
        { upsert: true }
      );
    }
    // Example: allow doctors to update location info via same endpoint
    if (profile?.hospitalOrClinic || profile?.location) {
      const doctorProfile = await DoctorProfile.findOne({ userId:
session.user.id });
      if (doctorProfile) {
        doctorProfile.hospitalOrClinic = profile.hospitalOrClinic ??
doctorProfile.hospitalOrClinic;
        doctorProfile.location = profile.location ?? doctorProfile.location;
        await doctorProfile.save();
      }
    }
    return NextResponse.json({ message: "Profile updated" });
  } catch (error) {
    console.error("PUT /api/profile error", error);
    return NextResponse.json({ error: "Internal server error" }, { status: 500
});
  }
}
```

A3. Doctor Directory API (src/app/api/doctors/route.ts)

```
import { NextRequest, NextResponse } from "next/server";
import mongoose from "mongoose";
import bcrypt from "bcryptjs";
import { auth } from "@/auth";
import connectDB from "@/lib/mongodb";
import User from "@/models/user";
import DoctorProfile from "@/models/doctorProfile";
export async function GET(request: NextRequest) {
  await connectDB();
  const includeInactive = request.nextUrl.searchParams.get("includeInactive")
=== "true";
  const doctors = await DoctorProfile.find(includeInactive ? {} : { isActive:
true })
    .sort({ name: 1 })
    .lean();
  const userIds = doctors.map((doc) => doc.userId);
  const users = await User.find({ _id: { $in: userIds } }, "email
isActive").lean();
  const userMap = new Map(users.map((u) => [String(u._id), u]));
  return NextResponse.json(
    doctors.map((doctor) => ({
      id: String(doctor._id),
      userId: String(doctor.userId),
      name: doctor.name,
      specialization: doctor.specialization,
      qualifications: doctor.qualifications,
      yearsOfExperience: doctor.yearsOfExperience,
      hospitalOrClinic: doctor.hospitalOrClinic,
      location: doctor.location,
      bio: doctor.bio,
      consultationFee: doctor.consultationFee,
      isVerified: doctor.isVerified,
      isActive: doctor.isActive,
      email: userMap.get(String(doctor.userId))?.email ?? null,
      userIsActive: userMap.get(String(doctor.userId))?.isActive ?? null,
      createdAt: doctor.createdAt,
      updatedAt: doctor.updatedAt,
    }))
  );
}
```

```
export async function POST(request: NextRequest) {
  const session = await auth();
  if (!session?.user || session.user.role !== "admin") {
    return NextResponse.json({ error: "Unauthorized" }, { status: 401 });
  }
  const body = await request.json();
  const {
    name,
    email,
    password,
    specialization,
    qualifications,
    yearsOfExperience,
    licenseNumber,
    hospitalOrClinic,
    location,
    bio,
    consultationFee,
  } = body;
  if (
    !name ||
    !email ||
    !password ||
    !specialization ||
    !qualifications ||
    typeof yearsOfExperience !== "number" ||
    !licenseNumber ||
    !hospitalOrClinic ||
    !location ||
    !bio ||
    typeof consultationFee !== "number"
  ) {
    return NextResponse.json({ error: "Missing required doctor fields" }, {
status: 400 });
  }
  await connectDB();
  const existingUser = await User.findOne({ email });
  if (existingUser) {
    return NextResponse.json({ error: "User with this email already exists"
}, { status: 409 });
```

```
  }
  const hashed = await bcrypt.hash(password, 12);
  const user = await User.create({ name, email, password: hashed, role:
"doctor", isActive: true });
  const doctorProfile = await DoctorProfile.create({
    userId: user._id,
    name,
    specialization,
    qualifications,
    yearsOfExperience,
    licenseNumber,
    hospitalOrClinic,
    location,
    bio,
    consultationFee,
    isVerified: false,
    isActive: true,
  });
  return NextResponse.json(
    {
      id: String(doctorProfile._id),
      userId: String(user._id),
      name,
      email,
      specialization,
      qualifications,
      yearsOfExperience,
      hospitalOrClinic,
      location,
      bio,
      consultationFee,
      isVerified: doctorProfile.isVerified,
      isActive: doctorProfile.isActive,
    },
    { status: 201 }
  );
}
export async function DELETE(request: NextRequest) {
  const session = await auth();
  if (!session?.user || session.user.role !== "admin") {
```

```
    return NextResponse.json({ error: "Unauthorized" }, { status: 401 });
  }
  const id = request.nextUrl.searchParams.get("id");
  if (!id || !mongoose.Types.ObjectId.isValid(id)) {
    return NextResponse.json({ error: "Valid doctor id is required" }, {
status: 400 });
  }
  await connectDB();
  const doctorProfile = await DoctorProfile.findById(id);
  if (!doctorProfile) {
    return NextResponse.json({ error: "Doctor not found" }, { status: 404 });
  }
  doctorProfile.isActive = false;
  await doctorProfile.save();
  await User.findByIdAndUpdate(doctorProfile.userId, { isActive: false });
  return NextResponse.json({ message: "Doctor deactivated" });
```

A4. Patient Booking Component (src/app/doctors/page.tsx)

```
"use client";
import { FormEvent, useEffect, useMemo, useState } from "react";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";
type DoctorCard = {
  id: string;
  name: string;
  specialization: string;
  qualifications: string;
  bio: string;
  consultationFee: number;
  yearsOfExperience: number;
  hospitalOrClinic: string;
  location: string;
  isActive: boolean;
};
type AppointmentForm = {
  scheduledDate: string;
  scheduledTime: string;
  reason: string;
};
const initialForm: AppointmentForm = {
  scheduledDate: "",
  scheduledTime: "",
```

```
  reason: "",
};
export default function DoctorsPage() {
  const { data: session } = useSession();
  const router = useRouter();
  const [doctors, setDoctors] = useState<DoctorCard[]>([]);
  const [filteredDoctors, setFilteredDoctors] = useState<DoctorCard[]>([]);
  const [selectedDoctor, setSelectedDoctor] = useState<DoctorCard | null>(null);
  const [formState, setFormState] = useState(initialForm);
  const [bookingStatus, setBookingStatus] = useState<"idle" | "saving" | "success" | "error">("idle");
  const [bookingError, setBookingError] = useState<string | null>(null);
  const [minDate, setMinDate] = useState("");
  useEffect(() => {
    void (async () => {
      try {
        const response = await fetch("/api/doctors");
        if (!response.ok) throw new Error("Failed to load doctors");
        const data = await response.json();
        const active = data
          .filter((doctor: DoctorCard) => doctor.isActive)
          .map((doctor: DoctorCard) => ({
            ...doctor,
            hospitalOrClinic: doctor.hospitalOrClinic ?? "",
            location: doctor.location ?? "",
          }));
        setDoctors(active);
        setFilteredDoctors(active);
      } catch (error) {
        console.error(error);
      }
    })();
  }, []);
  useEffect(() => {
    setMinDate(new Date().toISOString().slice(0, 10));
  }, []);
  const isAuthenticatedPatient = useMemo(() => session?.user?.role === "patient", [session]);
  const openBookingModal = (doctor: DoctorCard) => {
    if (!session) {
      router.push("/auth/signin");
      return;
    }
    setSelectedDoctor(doctor);
```

```
    setFormState(initialForm);
    setBookingStatus("idle");
    setBookingError(null);
  };
  const handleFormChange = (field: keyof AppointmentForm, value: string) => {
    setFormState((prev) => ({ ...prev, [field]: value }));
  };
  const handleBooking = async (event: FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    if (!selectedDoctor) return;
    if (!formState.scheduledDate || !formState.scheduledTime) {
      setBookingError("Please provide both date and time.");
      return;
    }
    const combinedDate =
`${formState.scheduledDate}T${formState.scheduledTime}`;
    const dateValue = new Date(combinedDate);
    if (Number.isNaN(dateValue.getTime())) {
      setBookingError("Invalid date/time input.");
      return;
    }
    setBookingStatus("saving");
    setBookingError(null);
    try {
      const response = await fetch("/api/appointments", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          doctorId: selectedDoctor.id,
          scheduledFor: dateValue.toISOString(),
          reason: formState.reason || undefined,
        }),
      });
      if (!response.ok) {
        const payload = await response.json().catch(() => ({}));
        throw new Error(payload.error ?? "Booking failed.");
      }
      setBookingStatus("success");
    } catch (error) {
      console.error(error);
      setBookingStatus("error");
      setBookingError(
        error instanceof Error ? error.message : "Booking failed. Please try
again."
      );
```

```
    }
  };
  return (
    // UI
  );
}
```

A5. Prescription Share Page (src/app/prescriptions/[id]/page.tsx)

```
import { notFound } from "next/navigation";
import connectDB from "@/lib/mongodb";
import Prescription from "@/models/prescription";
import "@/models/doctorProfile";
import "@/models/user";
type Params = { id: string };
async function fetchPrescription(id: string) {
  await connectDB();
  const record = await Prescription.findById(id)
    .populate("doctorId", "name specialization hospitalOrClinic location")
    .populate("patientId", "name email")
    .populate("appointmentId", "scheduledFor status")
    .lean();
  if (!record) return null;
  return {
    id: String(record._id),
    instructions: record.instructions,
    createdAt: record.createdAt,
    doctor: record.doctorId
      ? {
          name: (record.doctorId as any).name,
          specialization: (record.doctorId as any).specialization,
          hospitalOrClinic: (record.doctorId as any).hospitalOrClinic,
          location: (record.doctorId as any).location,
        }
      : null,
    patient: record.patientId
      ? {
          name: (record.patientId as any).name,
          email: (record.patientId as any).email,
```

```
        }
      : null,
    appointment: record.appointmentId
      ? {
          scheduledFor: (record.appointmentId as any).scheduledFor,
          status: (record.appointmentId as any).status,
        }
      : null,
  };
}
export default async function PrescriptionSharePage({ params }: { params:
Params }) {
  const details = await fetchPrescription(params.id);
  if (!details) notFound();
  const appointmentDate = details.appointment?.scheduledFor
    ? new Date(details.appointment.scheduledFor).toLocaleString(undefined, {
        dateStyle: "medium",
        timeStyle: "short",
      })
    : "Not recorded";
  return (
    <main className="min-h-screen bg-gray-100 py-16 px-4">
      <div  className="max-w-3xl  mx-auto  bg-white  border  border-teal-100
rounded-2xl shadow">
        <header className="px-8 py-6 border-b border-gray-100">
          <p  className="text-sm  uppercase  tracking-wide  text-teal-500  font-
semibold">
            Health Portal Prescription
          </p>
          <h1 className="mt-2 text-2xl font-bold text-gray-900">
            Medication & Care Instructions
          </h1>
          <p className="mt-2 text-sm text-gray-500">
            Issued on{" "}
            {new Date(details.createdAt).toLocaleDateString(undefined, {
              dateStyle: "medium",
            })}
          </p>
        </header>
        <div className="px-8 py-6 space-y-6 text-gray-700">
```

```
          <section>
            <h2 className="text-sm font-semibold text-gray-500 uppercase
tracking-wide">
              Patient
            </h2>
            <p className="mt-1 text-lg font-medium text-gray-900">
              {details.patient?.name ?? "Patient"}
            </p>
            {details.patient?.email && (
              <p                 className="text-sm                 text-gray-
500">{details.patient.email}</p>
            )}
          </section>
          <section>
            <h2 className="text-sm font-semibold text-gray-500 uppercase
tracking-wide">
              Prescribing Doctor
            </h2>
            <p className="mt-1 text-lg font-medium text-gray-900">
              {details.doctor?.name ?? "Treating Physician"}
            </p>
            <p className="text-sm text-gray-500">
              {details.doctor?.specialization ?? "Medical Practitioner"}
            </p>
            {(details.doctor?.hospitalOrClinic || details.doctor?.location)
&& (
              <p className="text-sm text-gray-500">
                {details.doctor?.hospitalOrClinic}
                {details.doctor?.location ? ` • ${details.doctor.location}` :
""}
              </p>
            )}
          </section>
          <section>
            <h2 className="text-sm font-semibold text-gray-500 uppercase
tracking-wide">
              Appointment
            </h2>
            <p className="mt-1 text-sm text-gray-600">{appointmentDate}</p>
            <p className="text-xs text-gray-500 uppercase tracking-wide">
              Status: {details.appointment?.status ?? "N/A"}
```

```
            </p>
          </section>
          <section>
            <h2 className="text-sm font-semibold text-gray-500 uppercase
tracking-wide">
              Instructions
            </h2>
            <pre className="mt-3 whitespace-pre-line text-gray-800 bg-teal-50
border border-teal-100 rounded-xl px-6 py-4">
              {details.instructions}
            </pre>
          </section>
        </div>
      </div>
    </main>
  );
}
```

2.6 Summary

This chapter captured all functional/non-functional requirements and demonstrated how the design materializes via UML artifacts. The combination of plantuml diagrams and model definitions forms the blueprint guiding implementation. Upcoming chapters detail testing strategies, deployment considerations, and user guidance.

CHAPTER 3: SOFTWARE TESTING

3.1 Introduction

This chapter documents the testing approach used to validate the Health Portal, covering the features under test, strategies employed, test environments, and representative test cases. The goal is to ensure that the system functions correctly for patients, administrators and third-party pharmacies under typical usage scenarios.

3.2 Testing Features

During testing, major features will be:

- **Authentication:** Check by authenticating, registering, demos and role-based routing.
- **Doctor Management:** Administration of doctor creation and doctor deactivation and metadata.
- **Appointment Scheduling:** Workflow to book patients, check availability, filtering by the administrator and cancellation.
- **Prescription Management:** Appointments can be made automatically, prescription links are available to patients and the general public, generating QRs, and assigning prescriptions.
- **Patient Profile Tabs:** Modification of personal data and navigation of the history of past prescriptions.
- **Public Pages:** Filtering of doctor directory, content on the home page and emergency/ambulance page.

3.3 Testing Strategies

3.3.1 Test Approach

- **Unit/Component Testing:** Manipulations of concepts (React Testing Library) in the form of informal component validations.
- **Integration Testing:** API MongoDB integrated testing using Postman and scripted testing (appointments, prescription, profile).

- **End-to-End Testing:** Flow simulation of demo credentials by manual testing to replicate common patient, administrator, and pharmacists functions.
- **Regression Testing:** Re-running core paths following significant restructuring of a code to verify stability.

3.3.2 Pass/Fail Criteria

- The test is successful in case the result is equal to the expected behaviour, e.g. data is stored correctly, the HTTP status is correct or the UI state is correct.
- A test fails if incorrect data is stored, errors appear unexpectedly, or UI becomes inconsistent.
- Critical failures (authentication issues, booking errors, prescription retrieval errors) must be resolved immediately before future development.

3.4 System Testing (Representative Test Cases)

TC ID	Scenario	Steps	Expected Result	Status
TC-01	Patient sign-up & redirect	1. Visit /auth/signup → 2. Enter valid data → 3. Submit	User created; redirect to /profile	Pass
TC-02	Quick sign-in (patient)	1. Visit /auth/signin → 2. Click “Sign in as Patient”	Demo credentials used; redirect to /profile	Pass
TC-03	Admin adds doctor	1. Login as admin → 2. Go to /dashboard/doctors → 3. Fill form (hospital, location, fee) → 4. Save	Doctor appears in admin table and public directory	Pass
TC-04	Patient filters doctors	1. Open /doctors → 2. Filter specialization “Cardiology” → 3. Experience “10+ years”	Only matching doctors visible; others hidden	Pass
TC-05	Book appointment	1. Patient selects doctor → 2. Opens modal → 3. Chooses date/time → 4. Submits	Appointment created (status = pending); success toast	Pass
TC-06	Assign prescription	1. Admin opens /dashboard/appointments → 2. Filter by doctor → 3. Assign instructions → 4. Submit	Prescription saved, appointment marked completed, QR available in patient tab	Pass

TC-07	Cancel appointment	1. Admin presses “Cancel” → 2. Confirm dialog	Appointment status updates to “cancelled,” actions disabled	Pass
TC-08	Patient tabs	1. Navigate to /profile → 2. Switch to “Prescriptions” tab	Profile tab shows editable form; prescription tab lists history or empty state	Pass
TC-09	Public prescription link	1. Copy QR URL → 2. Open in incognito	Prescription details (doctor, instructions) visible without auth	Pass
TC-10	Ambulance hotline	1. Visit /ambulance → 2. Click “Call Ambulance” (mobile simulator)	Device dialer opens with number configured in NEXT_PUBLIC_AMBULANCE_NUMBER	Pass

Table 06: Test Case

3.5 Summary

After thorough testing It confirmed that Health Portal performs as per the requirement in both patient and admin experiences. The flow of signup/signin, adding doctor, appointment booking cycle, prescription sharing, emergency contact button functions as expected. The hidden issues were taken as notes for future optimization. Overall, the application is ready for deployment.

CHAPTER 4: DEPLOYMENT AND MAINTENANCE

4.1 Introduction

I am thrilled to tell you how I managed to put the Health Portal on the rail and keep it running. We will go through the areas that the system really works, what I did to implement the system without hiccups, and how I maintain it on a daily basis. I will also discuss how to update it, how I deal it when things go wrong and how I have overcome the bugs so that the patients, doctor, administrator and the pharmacies can always count on the service. Software Release Life Cycle (SRLC) is a methodology created to facilitate knowledge sharing which enables users to generate a list of tasks and deliverables that must be completed and satisfied by project participants.

4.2 Software Release Life Cycle (SRLC)

A Software Release Life Cycle (SRLC) is a methodology designed to share knowledge, so that project participants can create a list of tasks and deliverables that have to be fulfilled.

4.2.1 Environments

The Health Portal is developed and tested in 3 key locations. As we continue to code, I and the team go through the process of coding with an npm run dev that includes an.env.local file, which either points to an online MongoDB Atlas database or a local MongoDB server. When an individual requests a change, such as a new feature or a fix, we will automatically build a version on Vercel. That is our secret playground that we can fool around before anything is deposited on the live site. The last and most accessible version is on Vercel as well and its secret preferences remain securely stored in the control panel of Vercel. Nothing remains mixed up in order to ensure that experiments do not creep into the live product or leak confidential information.

4.2.2 Build & Deployment Workflow

When a tweak is done to the code, I always ensure that the code remains solid, stable and easy to read. When one of the teammates diverges to introduce a feature, the other team members view the work, and the CI pipeline executes npm run lint that finds bugs and provides style. I also conduct manual and automated tests on core flows such as the process of logging in, booking appointments, managing doctors and the use of QR codes. The build step (npm run build) is used to prepare the app, run any mishaps and make sure that the main pages load

within a reasonable amount of time. To take the last shove, Vercel packages the server code, diversifies the assets and chucks out a fast experience on a global scale. Immediately after launch, I conduct fast sanity tests to ensure that all things, such as sign-in and prescription QR scans are functioning as anticipated.

4.2.3 Maintenance Activities

Maintaining the portal is a hustle all the time. I also monitor the MongoDB Atlas database by pinging it periodically to ensure that queries remain fast, and that the number of connections, search time, and the use of indices are monitored. To ensure security, I ensure that secrets such as the NEXTAUTH_SECRET are rotated on a regular basis as well as demo passwords. Another thing that I maintain is that middleware and libraries are regularly updated by monitoring dependencies and bumping them accordingly. Once the entire stack has been updated I re-test it to ensure that the entire stack has not broken. Logging is immediate and hence when something goes wrong we know immediately and can sprint a fix. The use of HTTPS is required everywhere and I am already brainstorming about the multi-factor authentication in the future. MongoDB Atlas auto-back ups ensure data security and I have a backup plan with well defined administrative duties. Regarding performance, I am concerned with minimizing the time spent on loading the site at the start and ensuring that the site and APIs can execute the work of a large number of users at the same time.

4.2.4 Upgrade Guidelines

When upgrading the system, I do it slowly to avoid the system glitching. The change of databases will be an option, and the older features will be saved, with all the steps being recorded. The feature flags will allow me to launch new capabilities gradually, reducing the risk. In case a bug appears, I can roll back to a previous version automatically using Vercel without any manual effort. The changes are all noted down to be referred to in future.

4.2.5 Support & Issue Handling

GitHub Issues or Jira support tickets are recorded, and the description of what and why it happened, its severity and urgency is clear. I receive feedback on the users of the service, which is in the form of surveys or helpdesk forms of the hospitals, patients and pharmacies. On a critical blocker appearing such as a failed log-in, an appointment problem I stop the flow, create a patch, test it, and deploy it asap. No messy history keeps any updates regarding what changed, the environment changes, or any migrations of data.

4.3 Summary

The Health Portal is based on modern technologies such as Vercel hosting and MongoDB Atlas database. Due to automated build, extensive testing, and frequent check-ins, the platform is reliable, secure, and fast. Through this SRLC plan, the system will be able to embrace new characteristics, integrate with other services and continue to enhance. The result? A reliable, secure and fast patient, doctors, and service provider portal.

CHAPTER 5: USER MANUAL

5.1 Introduction

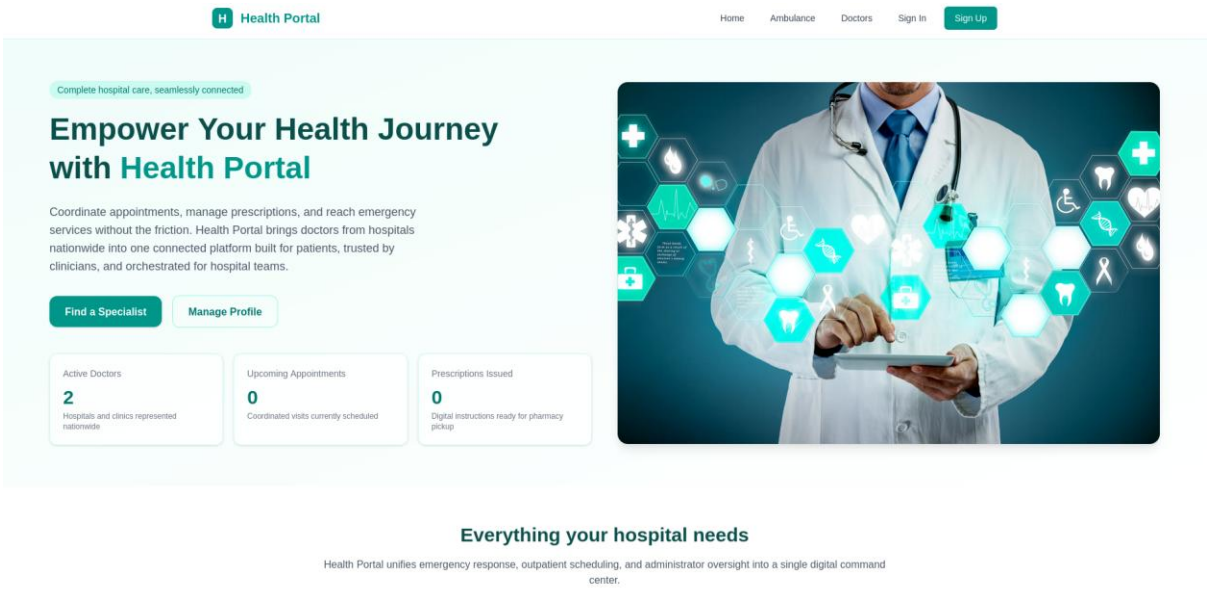
This chapter is a practical guide for users of Health Portal which include patients, admins and pharmacies. It gives details about the main features, navigation paths, and marked for specific interactions within both public pages and role based authenticated pages. It helps enable users to navigate through the system without getting lost.

5.2 Project Functionalities

5.2.1 Getting Started & Authentication

Accessing the Application

Open a modern browser such as Chrome, Firefox, Edge, or Safari and navigate to the deployed URL of Health Portal. The system is fully responsive and works on desktops, tablets, and mobile devices.

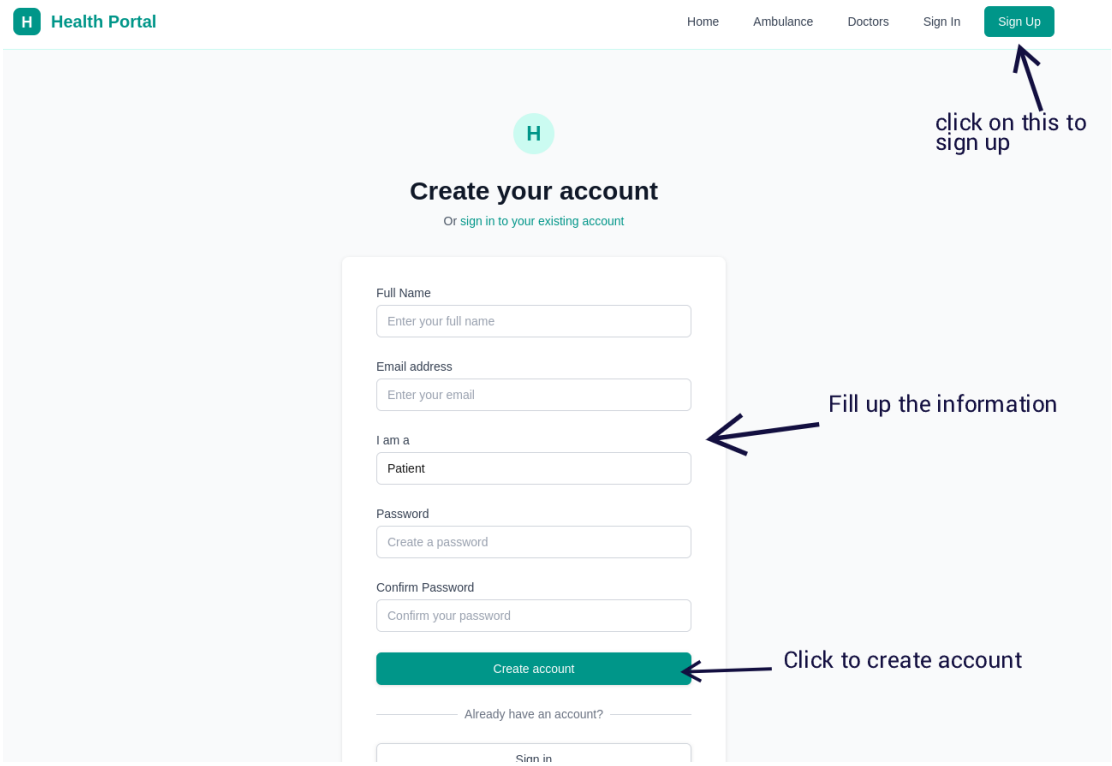


Sign-Up / Sign-In

Patients can create an account through the “Create Account” option on /auth/signup by providing basic details like name, email, password, and role.

Steps to create account:

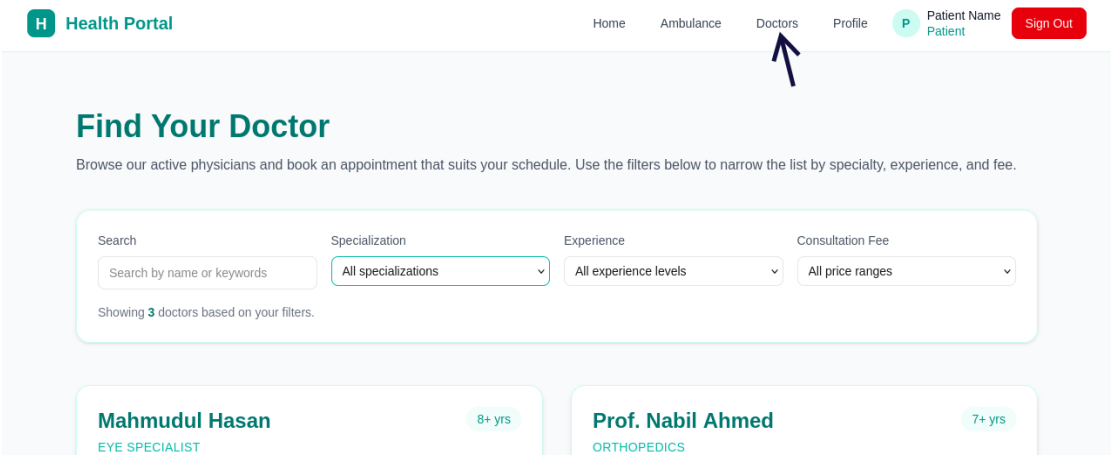
1. Click on Sign Up on navbar
2. Fill up your information in the form
3. Click on Create account button to create your account



5.2.2 Public Site Navigation

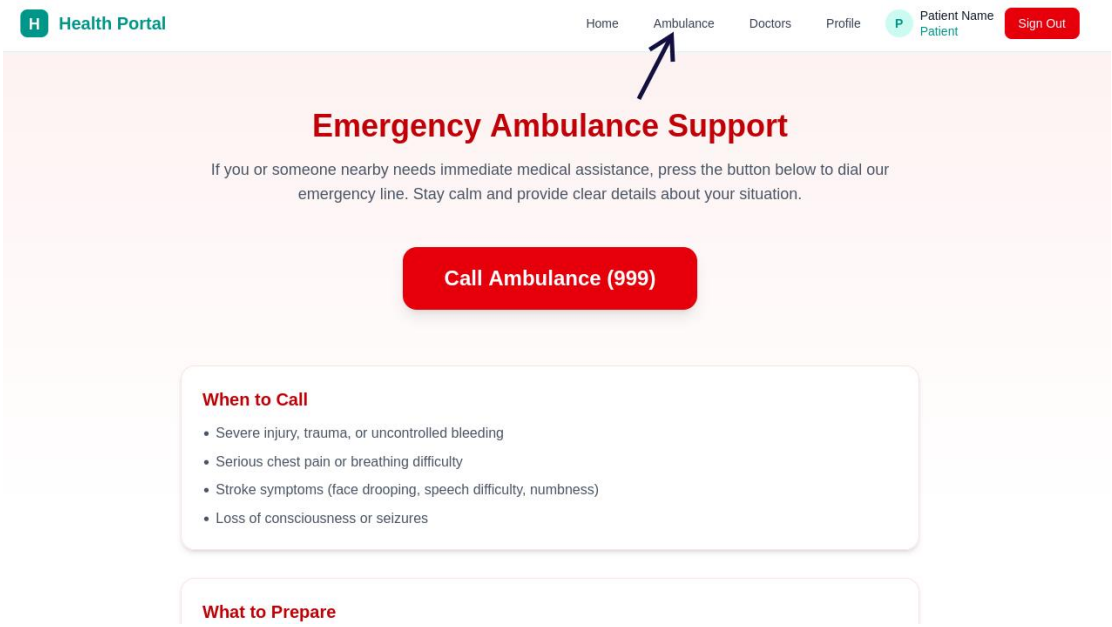
Doctors Directory (/doctors)

Allows users to filter doctors by specialization, years of experience, and consultation fees.



Ambulance **(/ambulance)**

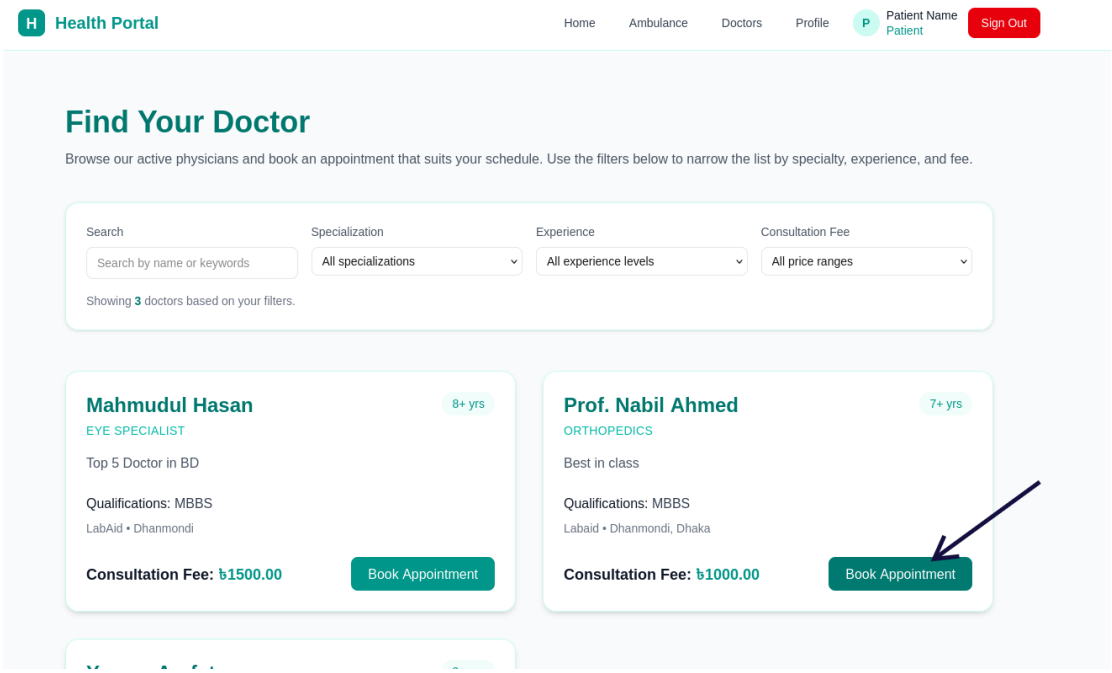
Provides a one-tap emergency contact button. On mobile devices, clicking the “Call Ambulance” button automatically opens the device dialer.



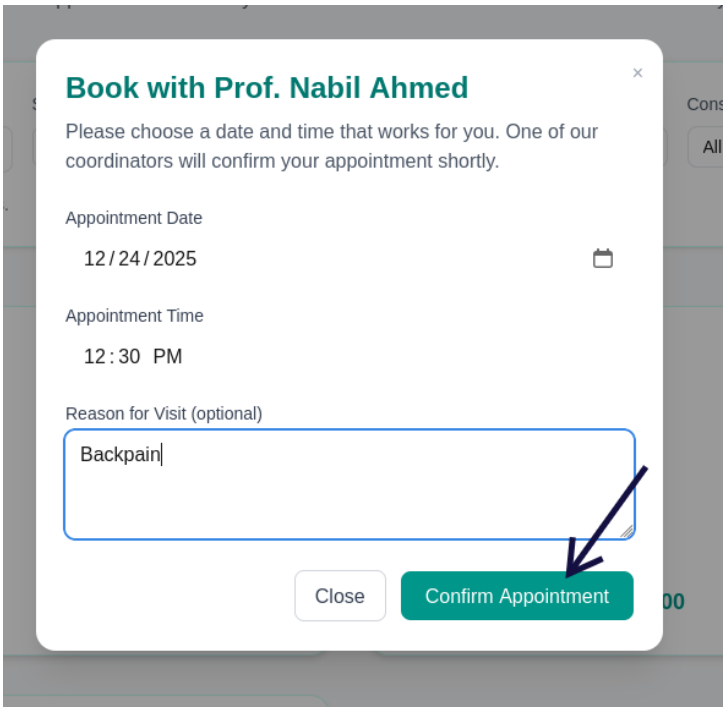
5.2.3 Patient Portal

Booking Appointments

1. Go to Doctors directory
2. Select the doctor to book his appointment

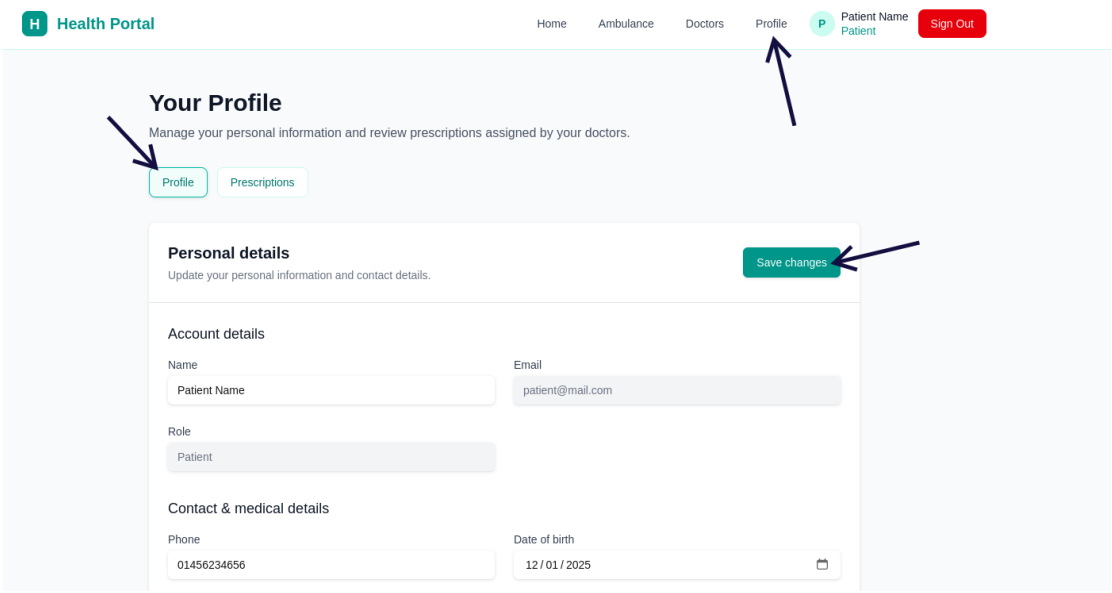


3. Fill up the required information
4. Click on Confirm Appoinment button to book



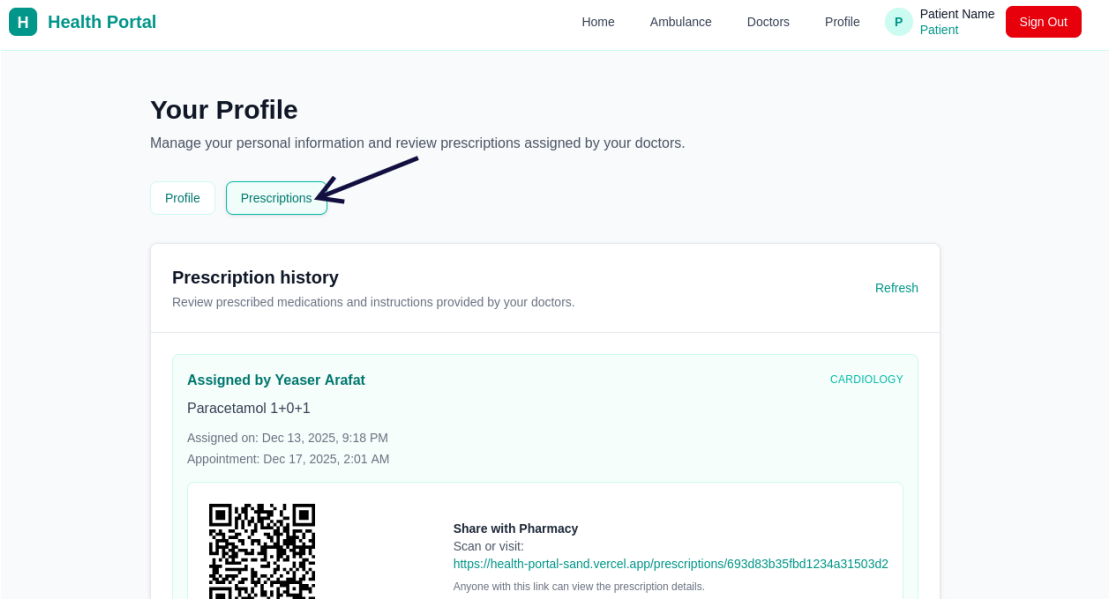
Profile Tabs (/profile)

Patients can view and modify personal information and click Save changes to update the information.



Prescription (/profile/prescriptions)

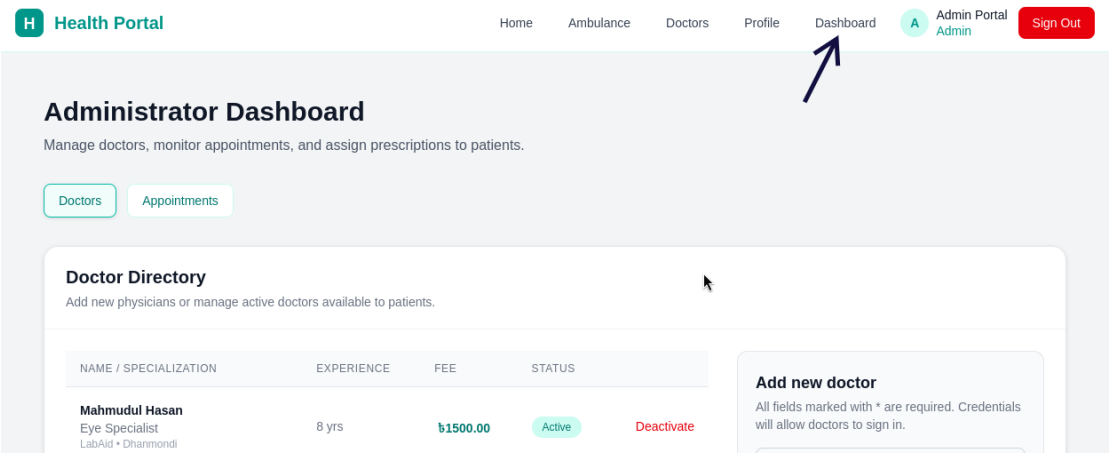
Here you can see all the prescription assigned to you



5.2.4 Admin Dashboard

Access

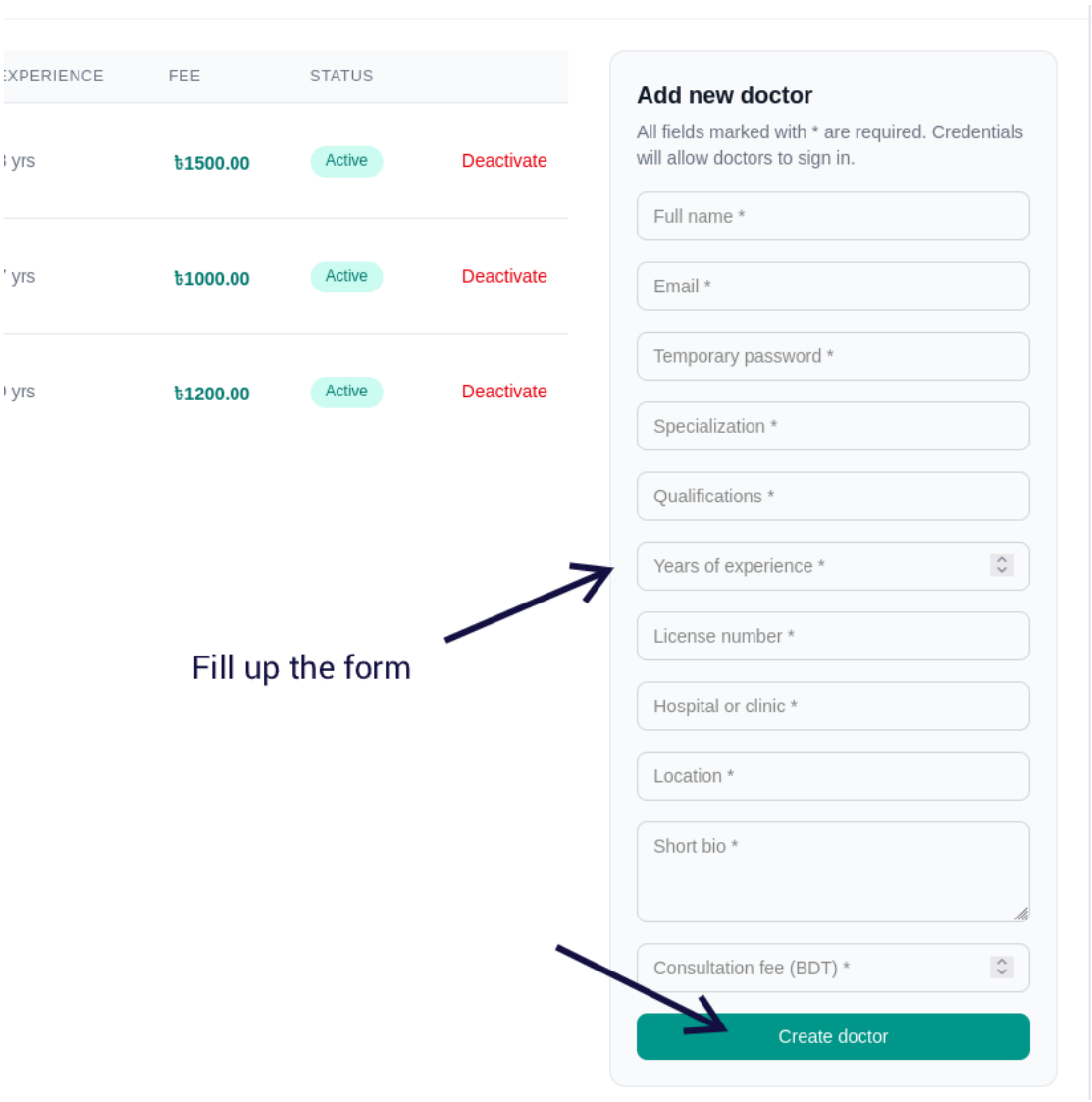
After loging in as admin navigate to /dashboard to access the administrative interface. Admins are redirected to /dashboard/doctors by default.



Doctors **Tab**

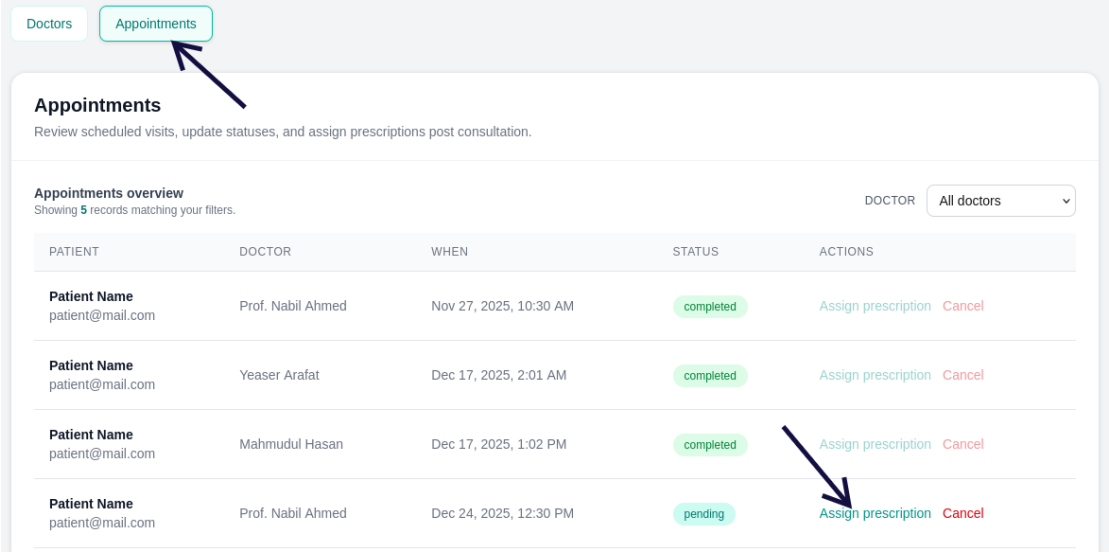
Admins can create new doctor profiles

1. Fill up the form with doctors information
2. Click on Create Doctor button to add

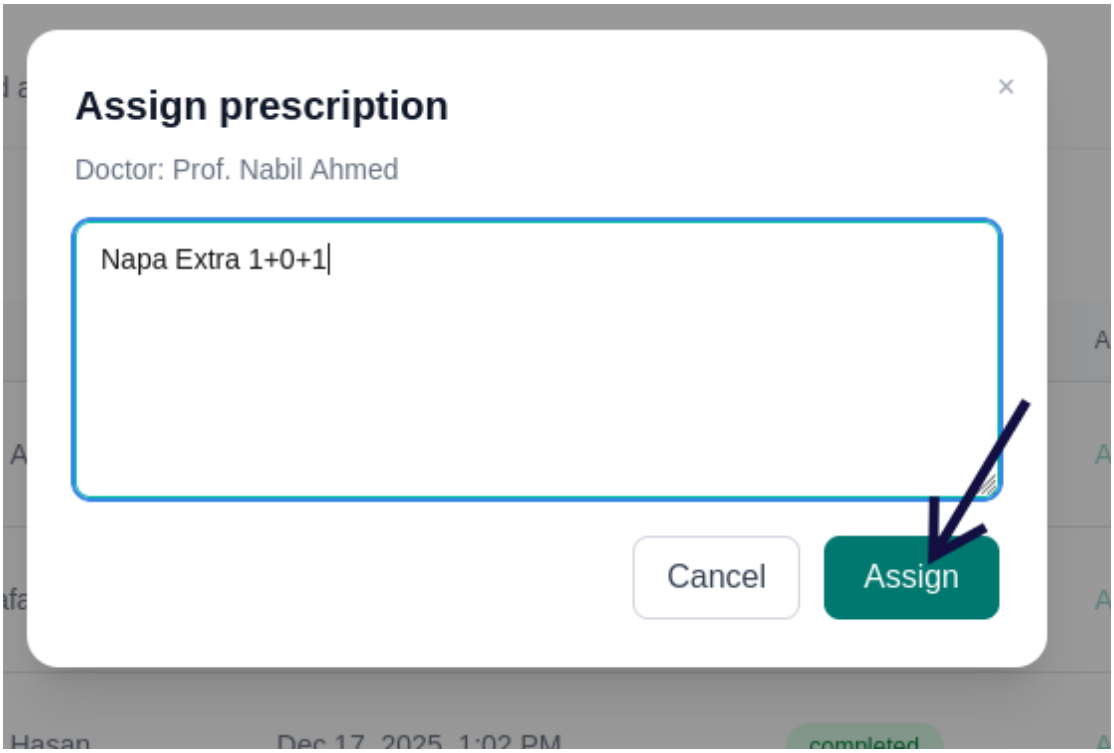


Appointments Tab

1. Click on appointments to view all the appoinments
2. Click on assign prescription



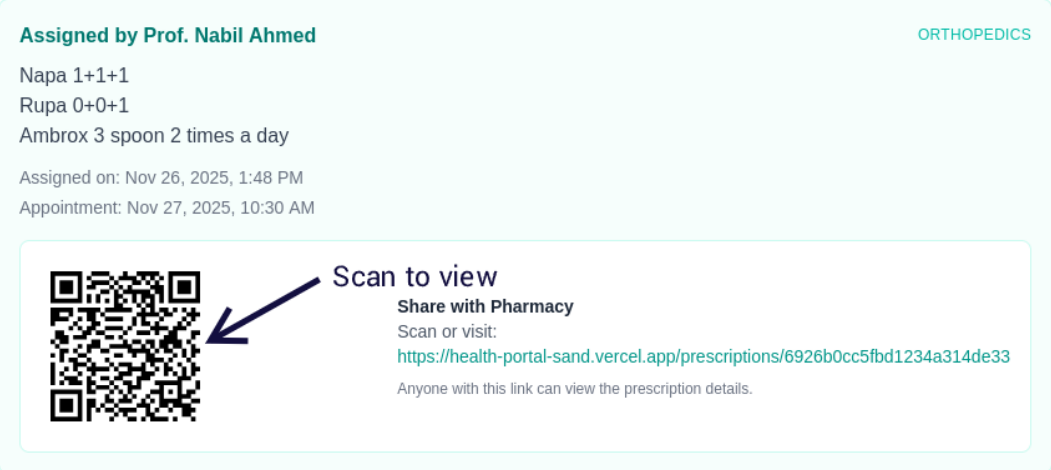
3. Then write the prescription and click on assign button to assign it to the patient



5.2.5 Pharmacy / Third-Party Access

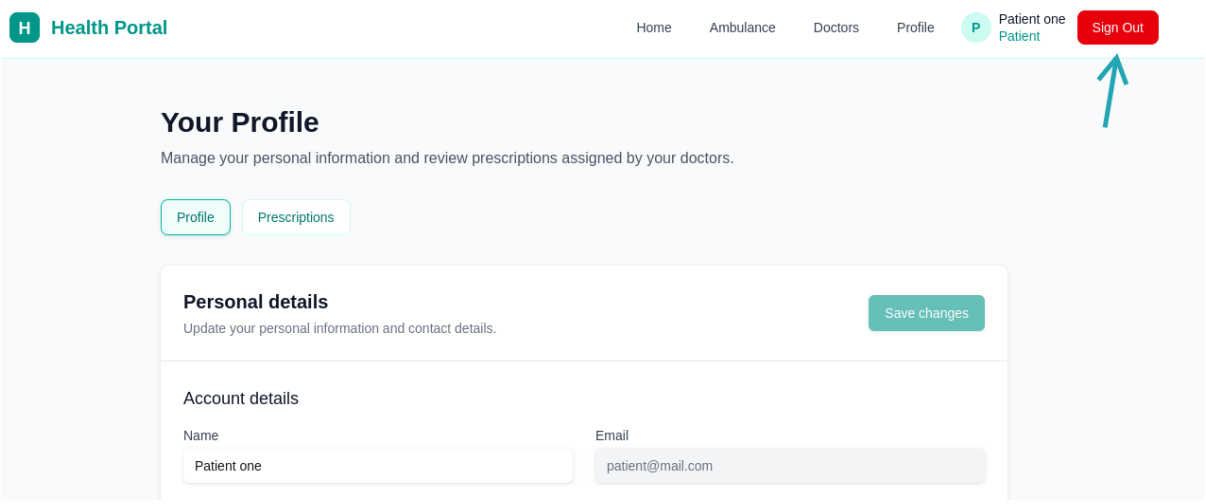
Prescription Share Links

QR codes generated for each prescription allow pharmacists or authorized third parties to view details without authentication. Visiting the link displays the instructions, prescribing doctor, appointment date, and timestamp in a read-only format.



5.2.6 Logging Out

Users can log out via the “Sign out” button in the top navigation menu. Sessions are managed using NextAuth JWT tokens, ensuring secure termination and automatic redirection to the homepage.



5.3 Summary

This guide just showed you how to use the Health Portal. Think of it as your personal online health spot. It clearly explained the easy way to move around the system and everything important you can do there. What you can do really depends on who you are. If you are just a regular person, you are a Public User. You can easily search for doctors and any health service that is available anywhere in the whole country. This platform is a centralized health platform for all to use. A patient gets their own space to book an appointment for their desired doctor and check their prescription history. An Admin is the overseer of the platform. He manages doctors' data and appointment schedules to ensure that everything is working as expected. For pharmacy staff it's just a quick QR code scan. In conclusion, it was designed to be simple and easy for all stakeholders to sign up and adapt to the features that are made for them.

CHAPTER 6: PROJECT SUMMARY

6.1 Introduction

This chapter gives a concise understanding of the Health Portal application. It reflects on the achieveent, limitations, scope and potential future developments. It also highlights the key result of the architecture design, implementation, testing phases and concludes with insights for future improvement.

6.2 Project Limitations

This application successfully integrates different healthcare components together to function. But several limitations remains. Quick login demo credentials are hard-coded placeholders requiring manual setup before production. Doctors currently cannot edit their own profiles; updates must be routed through administrators. Analytical features on the homepage are basic, offering counts but no trend visualization or dashboards. The system is currently designed for English-speaking users on modern browsers, with localization and accessibility enhancements pending. Offline functionality is not supported, as continuous connectivity is assumed for API and database interactions. Advanced security features such as multi-factor authentication, password reset flows, and audit logging are also out of scope in this version.

6.3 Scope (Achieved)

The project successfully delivered a web-based platform providing:

- A unified nationwide doctor directory with filtering options and location metadata.
- Patient-centered appointment booking with a tabbed profile interface for personal information and prescription history.
- An administrator dashboard that separates doctor management from appointment management, including prescription assignment.
- QR-coded prescriptions enabling secure access by pharmacies without authentication.
- An ambulance hotline and a homepage that updates dynamically showing some activity.
- API routes are implemented which cover authentication, profile management, doctors, appointments, prescriptions and public prescription access using QR code.

6.4 Future Work

Before and after improvements intended to increase functionality and usability:

Doctor Portal Improvements: Support self-service profile updates and availability management as well as telehealth system integration.

Automation of Pharmacy Workflow: Implement prescriptions fulfillment system and notification system to educate patients and administrators.

Advanced Analytics and Reporting: Build dashboards and visualization to show the trends in appointments, prescriptions and hospital performance measures.

Mobile Optimization and PWA Support: Add the features of Progressive Web Apps to access any site offline, send notifications on the phone, and be more mobile-friendly.

Localization & Accessibility: Bengali language, screen readers and more form validation should be supported to allow a wider level of use.

I**nsurance and payment integration:** available modules are billing, insurance verification and online payment gateways.

Active Health Warnings: Issue follow up appointment reminders and medication reminders.

6.5 Conclusion

Health Portal manages to create a unified online system that links patients, doctors and administrators of the hospital chain in Bangladesh. The fundamental functionalities such as role-based access, doctor onboarding, appointment lifecycle management, QR based prescription sharing and emergency support have been adopted and tested. Stability testing is done to ensure that the system is stable, and finally the documentation containing the report and the user manual makes the platform deployable. Health Portal can be optimally developed into an all-encompassing healthcare coordination tool with planned improvements, one that can be used to support routine healthcare and emergency response throughout Bangladesh.

CHAPTER 7: CONCLUSION

7.1 Reflective Summary

This chapter gives some conclusive thoughts on the project of Health Portal and summarizes its main contributions. Health Portal was created to bring hospitals, patients and pharmacies in Bangladesh with each other with the help of one and integrated platform. The project achieved the subsequent main features in several sprints of development:

- Role-aware authentication, including demo sign-in shortcuts for quick access.
- A directory of doctors nationwide containing hospital affinity and location data.
- Booking and patient personal profile and tabbed prescription history functionality, which is patient-centered.
- An administrator dashboard for onboarding doctors, filtering appointments, and generating QR-coded prescriptions.
- Public emergency support and shareable prescription URLs for third-party access.

7.2 Lessons Learned

A few lessons were learned in the process of development and testing:

- The ability to divide dashboard and patient space into two separate layouts and tabbed interfaces will lead to lower maintenance costs in the future and will simplify the addition of new features.
- MongoDB schema design using well-defined indexes can be used to avoid concurrency problems, but constant monitoring of duplicate indexs warnings is still required.
- Usability is also being enhanced like the use of quick login buttons as well as the use of tabs in navigation, which is as important as the backend functionality in terms of adoption.

7.3 Immediate Focus

The short term post-deployment priorities are:

- Tracking builds on errors or performance problems at production.
- recording demo credentials and conventional maintenance processes.
- Recruitment of the hospital pilot users to provide feedback to guide the roadmap of the future improvements.

Having the core objectives delivered successfully and a strong foundation built, Health Portal will be prepared to be tested on pilot basis, production maintenance, and improvement through repetition throughout the ecosystem of Bangladesh hospitals.

References

Next.js Documentation. (n.d.). Build full-stack web apps with React and the App Router. Retrieved https://nextjs.org/docs/app/getting-started

React Documentation. (n.d.). React: The library for web and native user interfaces. Retrieved from https://react.dev/learn

Node.js Documentation. (n.d.). Node.js v22.21.0 Documentation. Retrieved from https://nodejs.org/docs/latest-v22.x/api/index.html

Express.js. (n.d.). Fast, unopinionated, minimalist web framework for Node.js. Retrieved from https://expressjs.com/en/starter/installing.html

MongoDB Documentation. (n.d.). MongoDB Manual. Retrieved from https://www.mongodb.com/docs/manual/

Mongoose ODM. (n.d.). Mongoose v8.0.3 Documentation. Retrieved from https://mongoosejs.com/docs/

Tailwind CSS. (n.d.). Rapidly build modern websites without ever leaving your HTML. Retrieved from https://tailwindcss.com/docs/

NextAuth.js Documentation. (n.d.). Authentication for Next.js applications. Retrieved from https://authjs.dev/getting-started

React-QR-Code. (n.d.). Lightweight QR code component for React. Retrieved from https://www.npmjs.com/package/react-qr-code

PlantUML. (n.d.). Open-source tool to draw UML diagrams. Retrieved from https://plantuml.com/

Vercel Documentation. (n.d.). Deploy Next.js apps with zero configuration. Retrieved from https://vercel.com/docs

221-35-1017

ORIGINALITY REPORT

14%	11%	2%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

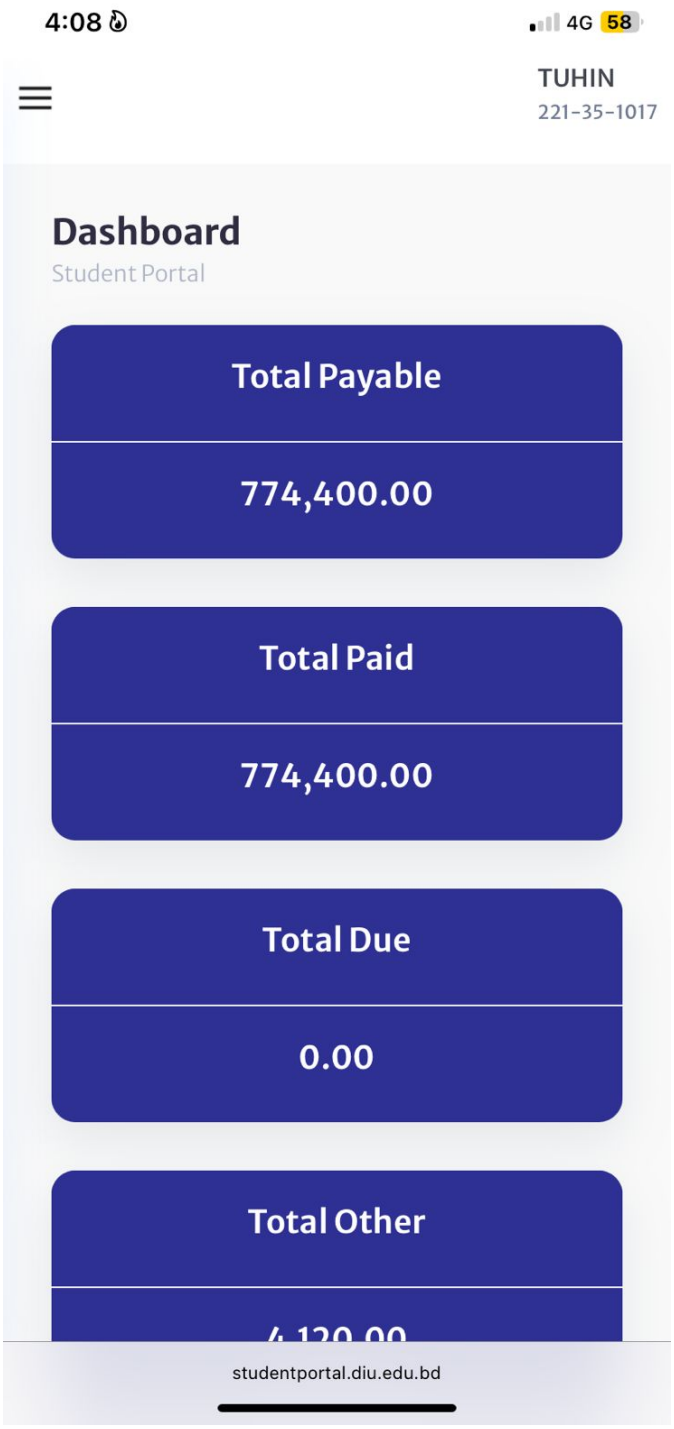
PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	2%
2	Submitted to University of Northampton Student Paper	1%
3	Submitted to NCC Education Student Paper	1%
4	Submitted to University of Hertfordshire Student Paper	1%
5	Submitted to Indiana Tech Univeristy Student Paper	1%
6	stackademic.com Internet Source	<1%
7	Submitted to University of Greenwich Student Paper	<1%
8	stackoverflow.com Internet Source	<1%
9	umpir.ump.edu.my Internet Source	<1%
10	Submitted to Technological University Dublin Student Paper	<1%
11	Submitted to Oberoi International School (JVLR) Student Paper	<1%
12	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%

13	Submitted to University of New York in Tirana Student Paper	<1 %
14	Submitted to Chester College of Higher Education Student Paper	<1 %
15	Submitted to Middlesex University Student Paper	<1 %
16	community.vercel.com Internet Source	<1 %
17	Submitted to American University in the Emirates Student Paper	<1 %
18	Submitted to Coventry University Student Paper	<1 %
19	dr.csbc.edu.ua Internet Source	<1 %
20	Submitted to Canadian International School Bangalore Student Paper	<1 %
21	Submitted to Universitat Politècnica de València Student Paper	<1 %
22	www.better-saas.org Internet Source	<1 %
23	Submitted to De Montfort University Student Paper	<1 %
24	Submitted to Murdoch University Student Paper	<1 %
25	Submitted to University of Piraeus Student Paper	<1 %

13	Submitted to University of New York in Tirana Student Paper	<1 %
14	Submitted to Chester College of Higher Education Student Paper	<1 %
15	Submitted to Middlesex University Student Paper	<1 %
16	community.vercel.com Internet Source	<1 %
17	Submitted to American University in the Emirates Student Paper	<1 %
18	Submitted to Coventry University Student Paper	<1 %
19	dr.csbc.edu.ua Internet Source	<1 %
20	Submitted to Canadian International School Bangalore Student Paper	<1 %
21	Submitted to Universitat Politècnica de València Student Paper	<1 %
22	www.better-saas.org Internet Source	<1 %
23	Submitted to De Montfort University Student Paper	<1 %
24	Submitted to Murdoch University Student Paper	<1 %
25	Submitted to University of Piraeus Student Paper	<1 %

ACCOUNT CLEARANCE



LIBRARY CLEARANCE