The 5-Day Development Plan

Here is a day-by-day breakdown to bring your vision to life.

Day 1: Project Setup & Backend Foundation

The goal today is to set up the entire project structure and define how our data will be stored. This is the bedrock of the application.

Project Initialization:

- Set up a project folder with two sub-folders: server (for backend) and client (for frontend).
- Initialize a Node.js project in the server directory (npm init -y).
- Initialize a Next.js project in the client directory (npx create-next-app@latest .).

Database Setup:

- Create a free cluster on MongoDB Atlas.
- Get your database connection string.
- Set up a .env file in your server to securely store this string and other secrets (like a JWT secret key).

Backend Server Setup:

- Install core dependencies: express, mongoose, cors, bcryptjs, jsonwebtoken, dotenv, nodemon.
- Create a basic Express server that connects to your MongoDB Atlas database.

Data Modeling (Mongoose Schemas):

userSchema:
- name (String, required)
- email (String, required, unique)
- password (String, required)
- role (String, enum: ['patient', 'doctor', 'admin'], default: 'patient')
- doctorProfile (ObjectId, ref: 'Doctor') - This will link to the doctor's specific details.

doctorSchema:
- user (ObjectId, ref: 'User') - A link back to the main user model.
- speciality (String, required)
- photoUrl (String)
- degreeUrl (String)
- verificationStatus (String, enum: ['pending', 'verified', 'rejected'], default: 'pending')

Day 2: Authentication & Doctor Verification API

Today is all about building the logic for users to sign up, log in, and for the admin to verify doctors.

File Upload Logic:

- Set up a free Cloudinary account.
- Install multer and cloudinary packages in the server.
- Create a middleware to handle file uploads from the registration form and send them to Cloudinary.

API Endpoint Development (in Express):

- POST /api/auth/register-patient: Simple registration for patients.
- POST /api/auth/register-doctor: Multi-part form for doctor registration with photo & degree upload.
- POST /api/auth/login: Authenticates users and returns a JWT (doctors must be verified).

Admin API Endpoints:
- GET /api/admin/pending-doctors: Fetch all pending doctors.
- PUT /api/admin/verify-doctor/:id: Verify or reject a doctor.

Day 3: Frontend UI - Registration & Login

Now we build the user-facing part of the app so people can interact with the backend we just built.

Setup Frontend Environment:

- Install Tailwind CSS.
- Install Axios.

Component Building:

- Create a Navbar and Footer.
- Design a landing page.

Page & Form Development:

- Registration Page (toggle between Patient & Doctor).
- Patient form: name, email, password.
- Doctor form: name, email, password, speciality, photo, degree.
- Login Page: email, password.

Connecting Frontend to Backend:

- Use Axios to call APIs.
- Implement state management (React Context or Zustand).
- Provide user feedback (toasts, messages).

Day 4: Doctor Listing & Admin Verification Panel

Let's make the verified doctors visible to the public and build the simple interface for the admin to do their job.

Public Doctors Page (/doctors):
- Backend: GET /api/doctors/verified.
- Frontend: /doctors page shows all verified doctors in a grid/list with clickable profiles.

Dynamic Doctor Profile Page (/doctors/[id]):
- Fetch and display public doctor info.

Admin Dashboard UI:
- Protected route: /admin/dashboard.
- Show pending doctors with details, photo, and degree.
- Approve/Reject buttons connected to backend.

Day 5: Polishing, Unique Features & Deployment Prep
Today is about refining the user experience, adding a unique feature, and getting ready to go live.

UI/UX Polish:
- Use Framer Motion for animations.
- Add loading spinners and toast notifications.

Unique Feature: Appointment Booking System
Backend:
- appointmentSchema (doctor, patient, date, time, status).
- POST /api/appointments/book and GET /api/appointments/my-appointments.

Frontend:
- Book Appointment button on doctor profile.
- Modal with calendar & slots.
- My Appointments page for patients.

Deployment Preparation:
- Frontend on Vercel (Next.js).
- Backend on Render/Heroku.
- Configure environment variables.
- Test full flow.

Ideas for Other & Unique Features (Future Enhancements)
- Doctor Dashboard for managing appointments.
- Interactive 3D Anatomy Model using Three.js.
- Secure real-time chat with Socket.IO.
- AI Symptom Checker.
- E-Prescriptions.