Pharmaceutical Store + Telemedicine App Design Plan

1. Product Summary & Core Flows

App goals

- End users: buy medicines, upload prescriptions, consult doctors online.

- Doctors: provide teleconsultations, issue prescriptions.

- Admins: manage medicines, orders, prescriptions, and doctors.

User flows

1. User: Google login → browse medicines → add to cart → checkout via Razorpay → if prescription-required → upload prescription → admin verifies → order fulfilled.

2. Telemedicine: book doctor → pay via Razorpay → video consult → doctor issues digital prescription.

3. Admin: login (email/password) → manage inventory, orders, and prescriptions.

4. Doctor: login (email/password) → manage profile, conduct consultations, issue prescriptions.

2. High-Level Architecture

- Mobile App (Users): Expo + React Native.

- Web App (Admin & Doctors): Next.js (separate platform).

- Backend API: Node.js (NestJS or Express) + TypeScript.

- Database: PostgreSQL (core data), Redis (caching + sessions).

- File Storage: Local server storage or DigitalOcean Spaces.

- Video / Telemedicine: Agora SDK (or Jitsi self-hosted if needed).

- Payments: Razorpay only.

- Messaging:

- Email: SendGrid / Mailgun

- WhatsApp: Twilio WhatsApp API / Meta Cloud API

- Deployment: Docker containers on VPS or managed services.

3. Tech Stack (Finalized)

Frontend

- Expo (React Native) for mobile app.

- Next.js for admin/doctor portal.

Backend

- Node.js (NestJS/Express) + TypeScript.

- PostgreSQL, Redis, Nginx.

Authentication

- Users: Google OAuth.
- Admins & Doctors: Email/password + JWT.

Payments

- Razorpay.

Messaging

- Email: SendGrid or Mailgun.
- WhatsApp: Twilio / Meta API.

File storage

- DigitalOcean Spaces or MinIO.

Telemedicine

- Agora.io SDK.

Infra

- VPS hosting (DigitalOcean, Linode, Hetzner).
- Dockerized deployment.
- CI/CD: GitHub Actions.

4. Data Models

User

- id, name, email, google_id, role=user, created_at, updated_at

Admin / Doctor

- id, email, password_hash, role, speciality (for doctors), fees, availability

Medicine

- id, name, brand, description, price, stock_quantity, prescription_required, images

Prescription

- id, user_id, file_url, status, verified_by, verified_at

Order

- id, user_id, items, total_amount, prescription_id, payment_status, order_status

Appointment

- id, user_id, doctor_id, scheduled_at, fee, status, video_session_id, prescription_id

5. APIs (Simplified)

- POST /auth/google (users)

- POST /auth/login (admins/doctors)
- GET /medicines
- POST /admin/medicines
- POST /orders
- POST /orders/:id/payment-callback
- POST /prescriptions
- PUT /admin/prescriptions/:id/verify
- POST /appointments
- POST /appointments/:id/start

6. Messaging Flow

- Order confirmation: Email + WhatsApp
- Prescription approval: Email + WhatsApp
- Appointment reminders: WhatsApp
- Doctor notes / digital prescriptions: Email link to PDF

7. Deployment & Infrastructure

- Hosting: DigitalOcean VPS
- CI/CD: GitHub Actions
- Database: Managed PostgreSQL
- File storage: DigitalOcean Spaces
- Video: Agora

8. Security & Compliance

- HTTPS with Let’s Encrypt
- JWT with refresh tokens
- RBAC (role-based access)
- Encrypt prescriptions
- Limited retention for prescriptions
- Audit logs for admin/doctor actions

9. Development Milestones

1. Setup: Repo, CI/CD, staging
2. Auth (Google + email/password)
3. Medicines module

4. Orders + Razorpay payments
5. Prescription upload + verification
6. Telemedicine booking + video
7. Notifications (email + WhatsApp)
8. Admin/doctor Next.js dashboard
9. Testing, security, launch

10. Future Extensions

- Medicine reminders
- Subscriptions
- AI prescription OCR assist
- Loyalty points / wallet
- Integration with delivery partners