High Level Design (HLD)

Healthcare Appointment Platform (MERN Stack)

1 Architecture Overview

Summary

Type: Modular, Layered, Microservices-ready

Stack: MERN (MongoDB, Express.js, React.js, Node.js), Stripe, JWT, Tailwind CSS, Multer, Cloudinary, Gemini API

2 System Components

2.1 Frontend (React + Tailwind)

- Component-based UI with Context API
- Role dashboards: Admin, Doctor, Patient
- Appointment booking, payments, profile image upload
- AI-powered prescription summaries
- Responsive, accessible UI

2.2 Backend (Node.js + Express)

- Controllers, Models, Routes, Middleware (auth, validation, errors)
- JWT auth, RBAC, CRUD for all entities
- Stripe, Cloudinary, Gemini AI integrations, webhooks

2.3 Database (MongoDB Atlas)

- Collections: Users, Appointments, Payments, Prescriptions
- Flexible schema, global replication, backups

2.4 Third-Party Integrations

- Stripe: secure payments
- Cloudinary: media storage & CDN
- Gemini API: OCR & summarization

3 Visual Architecture Diagram

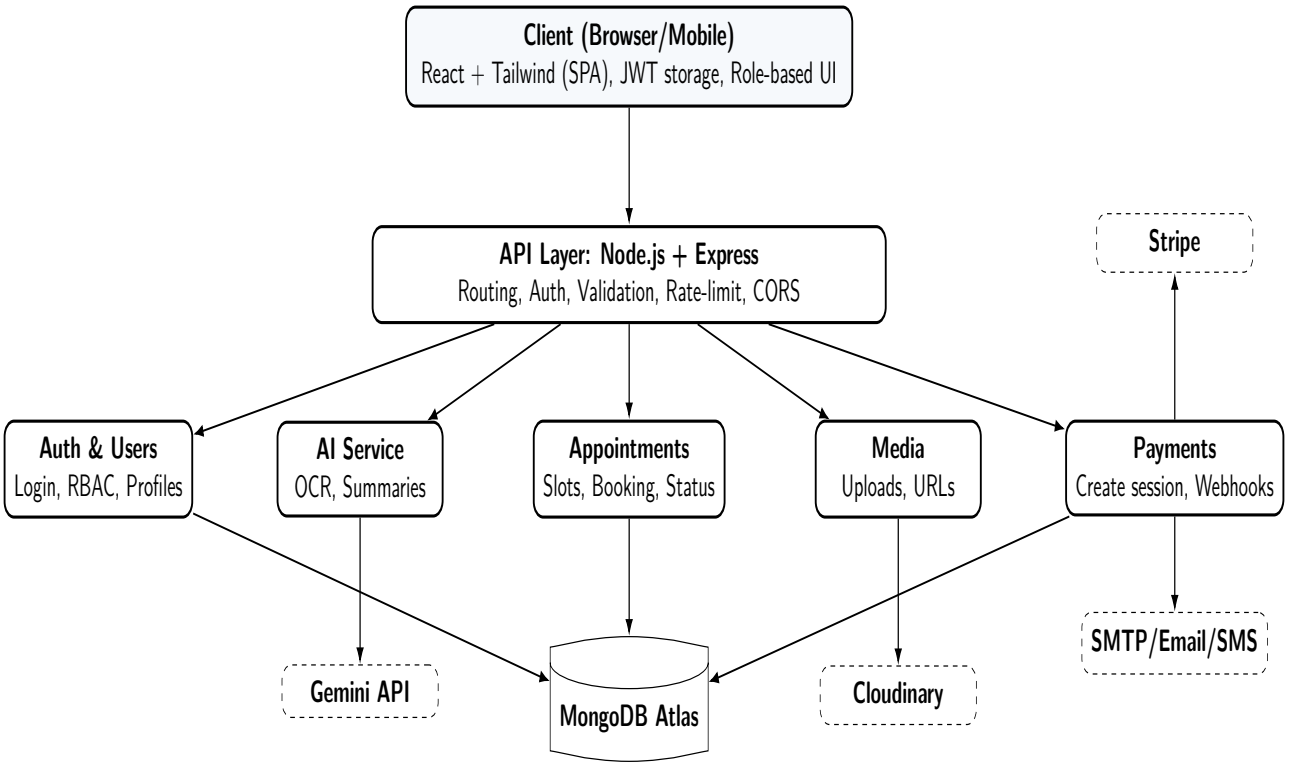

Figure 1: End-to-end Architecture with Core Services, Integrations, and Scale Enablers

4 Visual Architecture Flow

1. **Auth:** User logs in → JWT issued → stored client-side.
2. **Booking:** Patient selects slot → Appointment service → MongoDB saves record.
3. **Payments:** Stripe session created → payment confirmed → webhook updates DB.
4. **Profiles/Media:** File upload → Cloudinary → CDN URL saved.
5. **AI Prescriptions:** File upload → Gemini API → summarized output.

5 Tech Justification

5.1 Why MERN

- Unified JavaScript → same language across frontend and backend.
- React → reusable UI, great developer experience.
- Node.js + Express → scalable, non-blocking APIs.
- Community + ecosystem support → faster delivery.

5.2 MongoDB Atlas vs SQL

- Flexible document model → maps well to healthcare entities.
- Managed features: sharding, replication, backups.
- Scales write-heavy workloads (appointments, logs).

5.3 Other Key Choices

- JWT + RBAC → secure, stateless scaling.
- Tailwind CSS → fast UI iteration, consistent design.
- Stripe, Cloudinary, Gemini API → secure, production-ready services.

6 Future Roadmap

Scale & Reliability Plan

Phase 1: Foundation → Containerize with Docker, enable validation, rate-limiting.
Phase 2: Performance → Redis caching for sessions/availability, query optimization, pagination.
Phase 3: Features & UX → Doctor search, appointment reminders, analytics dashboard.
Phase 4: Resilience → Automated backups, retry logic, idempotent bookings.
Phase 5: Microservices → Split services, Kubernetes deployment, service mesh for resilience.

7 Appendix: Service Responsibilities

Service	Primary Responsibilities	Data Stores / Integrations
Auth & Users	Login, JWT, RBAC, Profiles	MongoDB
Appointments	Slots, Booking, Notifications	MongoDB
Payments	Sessions, Webhooks, Receipts	Stripe, MongoDB, SMTP
Media	Uploads, CDN URLs	Multer, Cloudinary, MongoDB
AI	OCR, Summaries	Gemini API, MongoDB
Admin & Analytics	Dashboard, Reports	MongoDB

This HLD ensures the platform is robust, secure, scalable, and industry-ready.