**Technical Architecture (Hotello – MERN + Clerk + PayPal + Gemini)**

**1. Frontend (React + Clerk)**

* **Tech**: React, Vite/CRA or Next.js, Tailwind CSS for styling, Axios/Fetch for API calls.
* **Responsibilities**:
  + User Interface: home page, hotel browsing, booking flows, admin dashboard.
  + **Authentication UI**: integrated via **Clerk SDK** (@clerk/clerk-react).
  + **AI-powered search & chatbot**: UI components interact with backend endpoints that proxy requests to **Gemini API**.
  + Payment UI: Confirm bookings and redirect to PayPal Checkout.

**2. Backend (Node.js + Express)**

* **Tech**: Node.js, Express.js, MongoDB (via Mongoose), Clerk webhooks/middleware, PayPal REST SDK, Gemini API.
* **Responsibilities**:
  + **Authentication Middleware**: Validate Clerk JWTs to protect routes.
  + **Booking & Hotel Management APIs**:
    - CRUD for hotels (admin/hotel owners).
    - Booking creation, modification, and history retrieval.
  + **Payment Processing**: Handle PayPal sandbox order creation and confirmation.
  + **AI Integration**: Proxy secure calls to Gemini API for chatbot and search suggestions.
  + **Role-based Access**: Assign roles (user, admin, hotelOwner) via Clerk metadata and enforce with middleware.

**3. Database (MongoDB + Mongoose)**

Collections:

| **Collection** | **Fields** | **Notes** |
| --- | --- | --- |
| **users** | \_id, clerkId, name, email, role(user/admin/hotelOwner), preferences (location, budget) | Clerk handles auth, store extras |
| **hotels** | \_id, name, description, location, amenities, images, pricePerNight, ownerId | Owned by hotelOwner or admin |
| **bookings** | \_id, userId, hotelId, checkIn, checkOut, totalAmount, status(pending/confirmed/cancelled) | Linked to PayPal payment records |
| **payments** | \_id, bookingId, paypalOrderId, amount, status(created/completed/failed), createdAt | For payment reconciliation |
| **chats** | \_id, userId, messages([{role:`user | ai`, content, timestamp}]) |

**4. Third-Party Services**

* **Clerk**: Authentication, user management, session tokens, roles.
* **PayPal Sandbox**: Secure online payment processing for bookings.
* **Gemini API**: AI chatbot for travel assistance and smart hotel search.

**5. Deployment**

* **Frontend**: Vercel or Netlify.
* **Backend**: Render, Railway, or Heroku.
* **Database**: MongoDB Atlas.
* **Environment Variables**: Use .env files or provider secrets for Clerk, PayPal, Gemini credentials.

**🧭 Development Order (Step-by-Step)**

**Phase 1 – Setup**

1. **Initialize repo & workspace** → Create folders for frontend and backend, init Git.
2. **Backend scaffolding** → Express server + MongoDB connection.
3. **Clerk integration** → Configure Clerk app + backend token validation. Test protected routes.

**Phase 2 – Core Features**

1. **Database models** → Users, Hotels, Bookings, Payments.
2. **Hotel management APIs** → CRUD for hotels (admin/hotelOwner).
3. **Booking flow APIs** → Booking creation and retrieval.
4. **Frontend pages** → Home, Sign In/Up, Browse Hotels, Booking Details.
5. **Role-based dashboards** → Admin panel & Hotel Owner dashboard.

**Phase 3 – Advanced Features**

1. **PayPal sandbox integration** → Add payment flow for confirmed bookings.
2. **Gemini AI chatbot** → Backend proxy + frontend chat UI for AI assistant and hotel search.
3. **Search optimization** → Use Gemini for semantic search suggestions.

**Phase 4 – Polish & Deploy**

1. **Testing & QA** → Unit tests for APIs, integration tests for auth & payments.
2. **Styling & UX improvements** → Tailwind or Material UI.
3. **Deployment** → Frontend (Vercel/Netlify), Backend (Render), MongoDB Atlas.
4. **Documentation & Screenshots** → Prepare for internship submission.

:

**📂 Project Root**

hotello/

│

├── README.md

├── .gitignore

├── package.json # For shared scripts if using a monorepo

├── .env.example # Sample environment variables

│

├── backend/ # Express + MongoDB API

│ ├── package.json

│ ├── .env

│ ├── server.js # Entry point

│ ├── config/ # App-level configs

│ │ ├── db.js # MongoDB connection

│ │ └── clerk.js # Clerk middleware setup

│ ├── controllers/ # Business logic

│ │ ├── authController.js

│ │ ├── hotelController.js

│ │ ├── bookingController.js

│ │ ├── paymentController.js

│ │ └── chatController.js

│ ├── middleware/ # Custom middleware

│ │ ├── authMiddleware.js # Role-based checks

│ │ └── errorHandler.js

│ ├── models/ # Mongoose schemas

│ │ ├── User.js

│ │ ├── Hotel.js

│ │ ├── Booking.js

│ │ ├── Payment.js

│ │ └── Chat.js

│ ├── routes/ # API routes

│ │ ├── authRoutes.js

│ │ ├── hotelRoutes.js

│ │ ├── bookingRoutes.js

│ │ ├── paymentRoutes.js

│ │ └── chatRoutes.js

│ ├── services/ # External API integration

│ │ ├── paypalService.js # PayPal REST SDK handling

│ │ └── geminiService.js # Gemini API calls

│ ├── utils/ # Utility functions

│ │ └── validators.js

│ └── tests/ # Jest or Mocha tests

│ └── booking.test.js

│

└── frontend/ # React app

├── package.json

├── vite.config.js # or next.config.js if using Next.js

├── .env

├── public/

│ └── index.html

└── src/

├── main.jsx # App entry point

├── App.jsx

├── api/ # Axios API calls

│ ├── authApi.js

│ ├── hotelApi.js

│ ├── bookingApi.js

│ └── paymentApi.js

├── assets/ # Images, icons

├── components/ # Reusable components

│ ├── Navbar.jsx

│ ├── Footer.jsx

│ ├── HotelCard.jsx

│ ├── Chatbot.jsx

│ └── PayButton.jsx

├── context/ # Context API or Zustand/Redux store

│ └── AuthContext.jsx

├── hooks/ # Custom hooks

│ └── useAuth.js

├── pages/ # Page-level components

│ ├── Home.jsx

│ ├── Hotels.jsx

│ ├── Booking.jsx

│ ├── Dashboard.jsx

│ ├── OwnerDashboard.jsx

│ └── PaymentSuccess.jsx

└── styles/ # Tailwind or custom styles

└── globals.css

**🔑 Environment Variables (.env samples)**

**Backend .env**

MONGO\_URI=mongodb+srv://...

CLERK\_SECRET\_KEY=your\_clerk\_secret

PAYPAL\_CLIENT\_ID=your\_paypal\_client\_id

PAYPAL\_SECRET=your\_paypal\_secret

GEMINI\_API\_KEY=your\_gemini\_key

PORT=5000

**Frontend .env**

VITE\_CLERK\_PUBLISHABLE\_KEY=your\_clerk\_public\_key

VITE\_BACKEND\_URL=http://localhost:5000

VITE\_PAYPAL\_CLIENT\_ID=your\_paypal\_client\_id

**🧭 Workflow Tips**

1. **Start Backend First**
   * Build and test your MongoDB models, routes, and Clerk token validation.
   * Stub PayPal and Gemini endpoints.
2. **Frontend Integration**
   * Add Clerk React SDK for login flows.
   * Build pages progressively: Home → Browse Hotels → Booking → Dashboard → Payment → AI Chatbot.
3. **Payment & AI Features Last**
   * After core booking works, wire PayPal sandbox payments.
   * Finally, integrate Gemini API for smart search and chatbot.