Project Name: Hotello - MERN + Clerk + Stripe + Gemini

# Updated Technical Architecture

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

* **Tech Stack**: React, Vite, Tailwind CSS, Axios, Clerk SDK, Stripe Embedded Checkout
* **Key Responsibilities**:
  + User Interface: Home page, hotel browsing, booking flows, account management
  + Authentication: Integrated via Clerk SDK (@clerk/clerk-react)
  + Payment Processing: Stripe Embedded Checkout integration
  + Advanced Features: Hotel filtering, AI search, booking history
  + Responsive Design: Mobile-first approach with custom UI/UX

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

* **Tech Stack**: Node.js, Express, MongoDB/Mongoose, Clerk webhooks, Stripe SDK, Gemini API
* **Key Responsibilities**:
  + Authentication Middleware: Validate Clerk JWTs for protected routes
  + Booking & Hotel Management: Complete CRUD operations
  + Payment Processing: Stripe integration with webhook handling
  + AI Integration: Gemini API for search and chatbot functionality
  + Advanced Filtering: Location, price, and custom filtering endpoints
  + Role-based Access: User, admin, and hotel owner permissions

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

**Updated Collections:**

| Collection | Key Fields | Description |
| --- | --- | --- |
| users | \_id, clerkId, name, email, role, preferences | User profiles with Clerk integration |
| hotels | \_id, name, description, location, pricePerNight, amenities, images, stripePriceId | Hotel data with Stripe product linking |
| bookings | \_id, userId, hotelId, checkIn, checkOut, totalAmount, paymentStatus, roomNumber | Booking records with payment status |
| payments | \_id, bookingId, stripeSessionId, amount, status, createdAt | Payment transaction records |
| chats | \_id, userId, messages[{role, content, timestamp}] | AI chat session history |

**4. Third-Party Services Integration**

* **Clerk**: Authentication, user management, session handling
* **Stripe**: Payment processing, product catalog, webhook handling
* **Gemini API**: AI-powered search and chatbot functionality
* **MongoDB Atlas**: Cloud database hosting
* **Render**: Backend deployment platform
* **Netlify**: Frontend deployment platform

# 🔄 Updated Development Phases

**Phase 1: Core Booking & Payment System + Deployment (Task 1)**

**Priority**: Complete existing implementation from reference repositories

* ✅ Implement Clerk authentication integration
* ✅ Set up hotel booking creation with PENDING status
* ✅ Integrate Stripe Embedded Checkout
* ✅ Configure webhook handling for payment status updates
* ✅ Deploy to production (Netlify + Render)

**Phase 2: Account Management & Booking History (Task 2)**

**New Implementation**:

* **My Account Page**: User profile display and management
* **Booking History**: Chronological display of user bookings
* **Booking Details**: Hotel info, dates, payment status, room numbers
* **Filtering Options**: By status, date range, etc.

**API Endpoints to Add**:

jsx

GET /api/bookings/user/:userId - Get user's booking history

GET /api/users/profile - Get user profile information

**Phase 3: UI/UX Customization & Brand Identity (Task 3)**

**Design Enhancement**:

* Custom color palette and typography
* Unique component library and design system
* Advanced animations and micro-interactions
* Dark/light mode support (bonus)
* Accessibility improvements (WCAG compliance)

**Phase 4: Advanced Hotel Listing & Filtering (Task 4)**

**New Features**:

* Dedicated /hotels listing page
* Location-based filtering with searchable dropdown
* Price range filtering with interactive slider
* Sorting options (price, rating, alphabetical)
* Grid/list view toggle
* Pagination for large datasets

**API Endpoints to Enhance**:

jsx

GET /api/hotels?location=...&minPrice=...&maxPrice=...&sortBy=...&page=...

GET /api/locations - Get unique locations for filters

**Phase 5: AI Search Clear & Reset Functionality (Task 5)**

**UX Improvement**:

* Clear search functionality to return to main hotel view
* Search state management (search mode vs browse mode)
* Visual feedback for current state
* Multiple clear options (button, navigation, keyboard)

# 📁 Updated Project Structure

text

hotello/

├── backend/

│ ├── src/

│ │ ├── application/

│ │ │ ├── hotel.ts # Enhanced with filtering logic

│ │ │ ├── booking.ts # Enhanced with user history

│ │ │ ├── payment.ts # Stripe integration (existing)

│ │ │ ├── user.ts # NEW: User profile management

│ │ │ └── search.ts # NEW: AI search functionality

│ │ ├── routes/

│ │ │ ├── hotelRoutes.js # Enhanced with filter parameters

│ │ │ ├── bookingRoutes.js # Enhanced with user history

│ │ │ ├── userRoutes.js # NEW: User profile routes

│ │ │ └── searchRoutes.js # NEW: AI search routes

│ │ └── models/

│ │ └── Enhanced schemas with new fields

├── frontend/

│ ├── src/

│ │ ├── pages/

│ │ │ ├── MyAccount.jsx # NEW: Account management

│ │ │ ├── Hotels.jsx # NEW: Hotel listing with filters

│ │ │ ├── BookingHistory.jsx # NEW: User booking history

│ │ │ └── Enhanced existing pages

│ │ ├── components/

│ │ │ ├── FilterSidebar.jsx # NEW: Advanced filtering

│ │ │ ├── HotelGrid.jsx # NEW: Hotel listing layout

│ │ │ ├── BookingCard.jsx # NEW: Booking history items

│ │ │ └── Enhanced UI components

│ │ └── hooks/

│ │ ├── useFilters.js # NEW: Filter state management

│ │ └── useSearch.js # NEW: Search functionality

# 🚀 Implementation Strategy

**Immediate Actions (Based on Existing Codebase)**

1. **Complete Phase 1**: Ensure Stripe integration works end-to-end
2. **Deployment Verification**: Test production deployment thoroughly
3. **Reference Implementation**: Study provided repositories for patterns

**Sequential Development Approach**

1. **Start with Task 2** (My Account): Build on existing user authentication
2. **Proceed to Task 4** (Hotel Listing): Create foundation for filtering
3. **Implement Task 5** (Search Clear): Enhance existing search functionality
4. **Complete with Task 3** (UI/UX): Polish and brand the application

# Key Technical Considerations

**State Management for Filters**:

jsx

// Example filter state structure

const [filters, setFilters] = useState({

locations: [],

priceRange: [0, 1000],

sortBy: 'featured',

searchQuery: '',

searchMode: false

});

**Booking History Data Flow**:

jsx

// Enhanced booking response with populated hotel data

{

\_id: bookingId,

checkIn, checkOut, totalAmount, paymentStatus,

hotel: {

name, location, images, amenities

}

}

**Search State Management**:

jsx

const [searchState, setSearchState] = useState({

isSearching: false,

query: '',

results: [],

originalData: [] // Preserve original hotel list

});

const clearSearch = () => {

setSearchState(prev => ({

...prev,

isSearching: false,

query: '',

results: []

}));

// Restore filters and original data

};

# 🔧 Enhanced Environment Variables

**Backend (.env)** - Additional variables for new features:

text

# Existing variables

MONGODB\_URL=

CLERK\_SECRET\_KEY=

STRIPE\_SECRET\_KEY=

STRIPE\_WEBHOOK\_SECRET=

OPENAI\_API\_KEY=

# New variables for enhanced features

FRONTEND\_URL=https://your-frontend-url.netlify.app

**Frontend (.env)** - Enhanced configuration:

text

VITE\_CLERK\_PUBLISHABLE\_KEY=

VITE\_BACKEND\_URL=https://your-backend-url.onrender.com

VITE\_STRIPE\_PUBLISHABLE\_KEY=

**📋 Success Metrics**

Each task has clear deliverables:

* **Task 1**: Live URLs, functional payment flow, proper error handling
* **Task 2**: Complete account page with booking history and filtering
* **Task 3**: Unique design system, improved UX, accessibility compliance
* **Task 4**: Advanced filtering, sorting, pagination, responsive design
* **Task 5**: Seamless search clear functionality, intuitive state management