TravelMate – Simplified Technical Architecture (MERN + Clerk + PayPal)

1. System Overview

TravelMate is a MERN web platform where travelers can sign up, browse properties and vehicles, book stays, make secure PayPal payments, and leave reviews.

Core stack:

- **Frontend**: React (Clerk React SDK for authentication + PayPal JS SDK for checkout).
- **Backend**: Node.js + Express (REST API).
- **Database**: MongoDB (for all app data: users, properties, bookings, reviews, vehicles).
- **Authentication**: Clerk (user registration, login, OAuth, MFA, sessions).
- **Payments**: PayPal Sandbox Orders API (order creation, capture, refund, webhooks).

---

2. High-level Architecture

```
[React + Clerk SDK + PayPal JS]  <---->  [Node.js + Express API]  <---->
[MongoDB]
          |                                        |
          |---- Clerk (identity & auth)            |---- PayPal Orders API
```

---

3. Data Model (MongoDB)

users

```
{
  "_id": ObjectId,
  "clerkUserId": "string",
  "role": "traveler" | "owner" | "admin",
  "displayName": "string",
  "phone": "string",
  "preferences": { "language": "en", "currency": "USD" },
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

properties

```
{
  "_id": ObjectId,
  "ownerId": ObjectId,
```

```
  "title": "string",
  "description": "string",
  "city": "string",
  "address": "string",
  "images": ["string"],  // store image URLs or base64 for MVP
  "amenities": ["wifi", "ac"],
  "propertyType": "villa" | "flat" | "room",
  "capacity": { "adults": 2, "children": 1 },
  "pricing": { "basePrice": 100, "currency": "USD" },
  "availability": [{ "date": "Date", "status": "available"|"blocked" }],
  "ratings": { "avg": 4.5, "count": 12 },
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

bookings

```
{
  "_id": ObjectId,
  "propertyId": ObjectId,
  "travelerId": ObjectId,
  "ownerId": ObjectId,
  "status": "pending" | "confirmed" | "cancelled" | "completed",
  "checkIn": "Date",
  "checkOut": "Date",
  "guests": { "adults": 2, "children": 0 },
  "priceBreakdown": { "nights": 3, "base": 300, "total": 300 },
  "payment": {
    "method": "paypal",
    "paypalOrderId": "string",
    "paypalCaptureId": "string",
    "status": "pending" | "paid" | "refunded" | "failed",
    "amount": 300,
    "currency": "USD"
  },
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

reviews

```
{
  "_id": ObjectId,
  "propertyId": ObjectId,
  "bookingId": ObjectId,
  "reviewerId": ObjectId,
  "score": 4,
  "comment": "Nice stay!",
  "createdAt": "Date"
}
```

vehicles

```
{
```

```
  "_id": ObjectId,
  "ownerId": ObjectId,
  "title": "Toyota Prius",
  "vehicleType": "car",
  "capacity": 4,
  "pricingPerDay": 40,
  "city": "string",
  "images": ["string"],
  "availableDates": ["Date"],
  "createdAt": "Date"
}
```

---

4. Backend API Routes

User (Clerk-managed authentication)

- `GET /api/v1/users/me` → Return user profile (merge Clerk data + MongoDB profile).
- `PUT /api/v1/users/me` → Update profile/preferences.

Sign-up, login, logout, and password reset are handled by Clerk's SDK/UI, not by custom routes.

Properties

- `POST /api/v1/properties` → Create property (owner only).
- `GET /api/v1/properties` → Search/filter list.
- `GET /api/v1/properties/:id` → Property details.
- `PUT /api/v1/properties/:id` → Update listing.
- `DELETE /api/v1/properties/:id` → Delete listing.

Bookings

- `POST /api/v1/properties/:id/book` → Create booking (status: pending).
- `GET /api/v1/bookings` → Traveler/owner list.
- `GET /api/v1/bookings/:id` → Booking details.
- `PUT /api/v1/bookings/:id/cancel` → Cancel booking.

Payments (PayPal)

- `POST /api/v1/payments/create-order` → Create PayPal order for booking.
- `POST /api/v1/payments/capture` → Capture order (optional if done client-side).
- `POST /api/v1/payments/webhook` → Handle PayPal webhooks.
- `POST /api/v1/payments/:id/refund` → Refund booking.

Reviews

- `POST /api/v1/properties/:id/reviews` → Add review (post-stay only).

- `GET /api/v1/properties/:id/reviews` → List reviews.

Vehicles

- `POST /api/v1/vehicles` → Add vehicle (owner).
- `GET /api/v1/vehicles` → Search/list vehicles.
- `POST /api/v1/vehicles/:id/book` → Book a vehicle.

Admin

- `GET /api/v1/admin/users` → List/manage users.
- `GET /api/v1/admin/bookings` → System-wide booking view.
- `GET /api/v1/admin/properties` → Manage property listings.

---

5. Booking & Payment Workflow

1. Traveler creates a booking (`status = pending`).
2. Backend calls PayPal Orders API → returns `orderID`.
3. Traveler pays with PayPal button (PayPal JS SDK).
4. PayPal notifies backend via webhook.
5. Backend verifies and updates booking → `status = confirmed`.
6. If refunded → backend calls PayPal Refund API → update booking `status = refunded`.

---

6. Security

- Clerk JWT verification for all protected routes.
- Role-based access control (`traveler`, `owner`, `admin`).
- No password storage in MongoDB (Clerk handles identity).
- Input validation and sanitization to prevent NoSQL injection.
- Webhook signatures verified against PayPal before processing.

---

7. Deployment & Environment

- **Frontend**: React app served via Netlify, Vercel, or static hosting.
- **Backend**: Node/Express deployed to Heroku, Render, or Railway (simple container hosting).
- **Database**: MongoDB Atlas.

- **Environment variables**:
  - `CLERK_API_KEY`
  - `CLERK_JWT_KEY`
  - `PAYPAL_CLIENT_ID`
  - `PAYPAL_SECRET`
  - `MONGO_URI`