TravelsTREM — Roadmap & MERN Architecture

> Document created for planning and implementation. Contains a Jira-style roadmap (Epics, Features, Tasks, Priorities, Timelines) followed by a high-level MERN stack architecture and deployment/operational notes.

Table of Contents

---

1. Executive Summary

TravelsTREM — Travel, Reservation, Experience & Management — is a full-featured travel booking platform. This document organizes the product into implementable epics and maps them to a scalable MERN architecture suitable for launch and growth.

---

2. Roadmap (Epics → Milestones → Tasks)

Phase 1 — MVP (Core, High Priority)

Goal: Launch a minimal yet bookable product that supports packages, flight/hotel booking basics, payments, user accounts, and an admin panel for package management.

Epics & Key Features (MVP): - EPIC-MVP-USER: User Accounts & Dashboard (High) - Tasks: Auth (email/OTP), profile, saved trips, bookings history, document upload for visa - EPIC-MVP-PACKAGES: Packages Catalogue & Booking (High) - Tasks: Package CRUD (admin), public listing, filters (destination, budget, duration), calendar availability, booking flow (select, confirm, pay) - EPIC-MVP-FLIGHT-HOTEL: Basic Flight & Hotel Search + Booking (High) - Tasks: Integrate one flight aggregator API (or mock for MVP),

hotel search with basic details and images - EPIC-MVP-PAYMENTS: Payment Gateway Integration (High) - Tasks: Integrate primary gateway (UPI + Cards), handle webhooks, payment confirmation, partial payment support (optional) - EPIC-MVP-ADMIN: Admin Panel (High) - Tasks: User management, package management, bookings overview, basic reports export (CSV) - EPIC-MVP-OPS: Infra + Security + Observability (High) - Tasks: HTTPS, JWT sessions, rate limiting, basic logging, error tracking

Deliverables (MVP): - Public site with package search and booking - Flight/hotel placeholder with at least one data source - Payment processing and booking confirmation - Admin CRUD and CSV export - User dashboard with bookings

---

Phase 2 — Growth (Medium Priority)

Goal: Improve UX, personalization, add content, and build AI-assisted features.

Epics: - EPIC-2-AI: AI Trip Suggestor (Medium) - Tasks: Questionnaire UI, backend service to map preferences → package suggestions, integrate simple ML rules or an AI API - EPIC-2-BLOG: Travel Guides & Blog (Medium) - Tasks: CMS for blog posts, SEO, tagging, author profiles, comments/ratings - EPIC-2-UX: Advanced Booking Features (Medium) - Tasks: Seat selection (flights), baggage info, multi-city flows, group booking flows - EPIC-2-ENGAGE: Offers, Coupons & Loyalty (Medium) - Tasks: Coupons engine, referral program, basic loyalty points - EPIC-2-INT: Weather & Map Integrations (Medium) - Tasks: Weather widget, maps embedding, 'near you' suggestions

---

Phase 3 — Expansion (Low → Medium Priority)

Goal: Monetize and scale: mobile apps/PWA, partnerships, advanced analytics.

Epics: - EPIC-3-SCALE: Mobile App / PWA (Medium) - EPIC-3-PARTNER: Collaborations & Vendor Portal (Medium) - Vendor onboarding, vendor dashboard - EPIC-3-ADV: Advanced Admin (Low) — Dynamic Package Builder, bulk upload, dashboards with charts - EPIC-3-COMM: Community & Reviews (Low) - User stories, verified traveler badges, sustainability tags

---

Sample Jira/Trello Card Format

Title: [EPIC] — Feature short title **Description:** What & why **Labels:** epic / backend / frontend / admin / high-priority **Assignee:** **Estimates:** story points / dev days **Acceptance Criteria:** Bulleted **Dependencies:** e.g., payments → cannot launch bookings without payment webhook

---

3. Suggested Timelines & Priorities (Sprint-based)

Assuming 2-week sprints and a small cross-functional team (1 FE, 1 BE, 1 PM/BA, 1 Designer, 1 QA):

- **Sprint 0 (Prep, 1 week):** Requirements, wireframes, infra provisioning, repo setup, CI skeleton.
- **Sprint 1 (Weeks 1–2):** Auth, user model, basic UI shell, list page for packages (mock data).
- **Sprint 2 (Weeks 3–4):** Package CRUD (admin), package listing UI wired to backend, filters.

- **Sprint 3 (Weeks 5–6):** Booking flow (select package → checkout placeholder), DB models for bookings.
- **Sprint 4 (Weeks 7–8):** Payment gateway integration, confirmation flows, webhook handling.
- **Sprint 5 (Weeks 9–10):** Flight/hotel MVP integration (1 provider), user dashboard with bookings.
- **Sprint 6 (Weeks 11–12):** Polish, QA, analytics, basic SEO, deploy to staging and launch MVP.

Priorities key: - High: User auth, packages, payments, admin CRUD, booking confirmation. - Medium: Flight/hotel full features, AI suggestor, blog. - Low: Loyalty, mobile app, vendor portal.

---

4. Sample User Stories & Acceptance Criteria

User Story 1: As a traveler, I want to search packages by destination, budget, and duration so that I can find trips that fit my plan. - AC1: Search returns matching packages sorted by relevance. - AC2: Filters persist in URL.

User Story 2: As a user, I want to book a package and pay online so I receive a booking confirmation and invoice. - AC1: Payment is processed via gateway; booking created on success. - AC2: Customer receives email confirmation and booking is visible in dashboard.

User Story 3: As an admin, I want to create/edit packages so that the site reflects live inventory. - AC1: Admin CRUD for package includes title, images, price tiers, availability calendar.

---

5. High-level MERN Architecture

Overview

- **MERN:** MongoDB (data), Express (API), React (frontend), Node.js (server)
- **Auth:** JWT + Refresh tokens + optional OTP (for UPI / contact verification)
- **Hosting:** Cloud provider (AWS / GCP / DigitalOcean); use managed DB service (MongoDB Atlas)

Components

Frontend (React) - Routes: Home, Packages, Package Detail, Booking Flow, Flights, Hotels, Dashboard, Admin. - State: Redux / Zustand for global state (user, cart/booking draft). Persist auth tokens securely. - UI Kit: Reusable components (Button, Modal, Form, DatePicker, Image Gallery/Slider).

Backend (Node + Express) - Service Layers: AuthService, PackageService, BookingService, PaymentService, SearchService - Middleware: Auth middleware, input validation (Joi/zod), logging, rate-limit

Database (MongoDB) - Collections: users, packages, bookings, payments, flights_cache, hotels_cache, coupons, reviews

Cache / Search - Redis for session/cache; optionally ElasticSearch for advanced package search later.

Core Data Models (short)

- **User**: { _id, name, email, phone, passwordHash, roles, documents: [{type, url}], loyaltyPoints }

- **Package**: { _id, title, description, priceTiers: [{name, price}], media: [urls], destination, durationDays, themes: [], availability: {dates}, inclusions, exclusions }
- **Booking**: { _id, userId, packageId, travelers: [{name,dob,passport}], totalAmount, paymentStatus, bookingStatus, createdAt }
- **Payment**: { _id, bookingId, gatewayRef, status, amount, paidAt }

Example API Endpoints

- `POST /api/auth/register`
- `POST /api/auth/login`
- `GET /api/packages?destination=&minBudget=&maxBudget=&duration=`
- `GET /api/packages/:id`
- `POST /api/bookings` (create booking)
- `POST /api/payments/webhook` (gateway webhook)
- `GET /api/user/bookings`
- `POST /api/admin/packages` (protected)

Integrations & Third-party

- **Payments:** Razorpay / Stripe / PayU (IN); support UPI and cards
- **Flights & Hotels:** Skyscanner API, Amadeus, or RapidAPI aggregators for MVP (or partner with an aggregator for B2B)
- **Weather:** OpenWeatherMap / WeatherAPI
- **Maps:** Google Maps / Mapbox for map & geolocation
- **Chat:** Twilio/WhatsApp API or Intercom / Freshchat for live chat
- **AI/Recommendations:** OpenAI or in-house rule-based suggestion engine

---

6. Non-functional Requirements

- **Security:** HTTPS + HSTS, input sanitization, prepared statements (avoid injection vectors), secure file storage for docs (S3 with pre-signed URLs), encrypt sensitive fields.
- **Performance:** Use CDN for assets, lazy-loading images, pagination for lists, server-side caching for heavy endpoints.
- **Availability:** Multi-AZ DB for production, automated backups daily.
- **Compliance:** GDPR-like considerations for user data, PCI-DSS scope for payments (avoid storing raw card data).

---

7. DevOps / CI-CD / Hosting Recommendations

- **Repo:** Mono-repo or separate repos (frontend, backend) — either works. Mono-repo helps coordinated releases.
- **CI:** GitHub Actions / GitLab CI — run lint, tests, build and deployment.
- **CD:** Deploy frontend to Vercel/Netlify; backend to AWS ECS / Elastic Beanstalk / DigitalOcean App Platform or container-based k8s (for scale).
- **DB:** MongoDB Atlas (managed)
- **Storage:** AWS S3 (or equivalent) for images/docs
- **Secrets:** Vault / AWS Secrets Manager / GitHub Secrets
- **Monitoring:** Sentry for errors, Prometheus + Grafana for metrics, Datadog optional.

---

8. Next steps & Recommended first 3 sprints

Sprint 0 (Prep - 1 week): - Finalize requirements for MVP, create wireframes for key flows - Setup repo, linters, CI skeleton, staging infra (basic) - Choose providers (payments, flights/hotels if any)

Sprint 1 (2 weeks): - Implement Auth & User model (register/login, JWT) - Basic UI shell and router - Package listing page with mock data

Sprint 2 (2 weeks): - Backend package CRUD & connect listing to real backend - Admin panel basic CRUD for packages - Basic booking schema and create-booking endpoint

Sprint 3 (2 weeks): - Integrate payment gateway sandbox - Complete booking flow with payment and confirmation - User dashboard: show bookings

---

Appendix — Quick Risk Register

- **Payments delay or chargeback risk** → mitigate with clear refund policy, hold inventory until confirmed
- **Third-party API reliability (flights/hotels)** → implement caching, fallback to alternatives
- **Scope creep** → focus MVP strictly on booking & payment

If you want, I can convert this into: - A Trello board CSV import file (cards + lists) OR - A set of Jira epics & stories ready to import (CSV) OR - A visual architecture diagram (SVG/PNG)

Tell me which output you'd like next and I'll prepare it.