TravelCam - Product Requirement Document (PRD)

Product Name: TravelCam Enterprise Platform
Developed By: GEP Protech
Document Version: 1.3.0
Status: Approved Architecture Specification
Date: August 26, 2026
Target Platform: Multi-Platform Enterprise Travel Solution (Laravel, Blade, React Native, React JS, MySQL/PSQL)
Document Formats: TravelCam_Product_Requirement_Document.pdf | TravelCam_Product_Requirement_Document.docx

1. Executive Summary & Vision

TravelCam is an enterprise-grade, multi-tenant travel agency management and passenger ticketing platform engineered by **GEP Protech**. It is designed to streamline transport agency operations across multiple regions and branch terminals. The platform provides complete operational autonomy to travel agencies while offering passengers a seamless, real-time booking experience.

Core Objectives

1. **Multi-Tenant Autonomy:** Isolated tenant data (`agency_id`), custom roles, fleet management, and localized branch pricing.
2. **Dynamic Scope-Based Management:** Allow managers to oversee specific scopes (`AGENCY` , `REGION` , `BRANCH`) with granular action permissions without account duplication.
3. **Branch-to-Branch Routing:** Support consolidated trip scheduling where a single bus picks up passengers across multiple intermediate branch stops with transparent departure timetables.
4. **Frictionless Ticketing:** Enable both authenticated passengers and unauthenticated guest (walk-in) passengers to search, select seats, book, and receive digital passenger QR e-tickets.
5. **Polymorphic Infrastructure:** Utilize dynamic polymorphic engines for centralized notifications, enterprise activity audit logs, document attachments, financial ledgers, and service reviews.

2. Project Engineering Team & Developer Roles (GEP Protech)

> The development, architecture, deployment, and quality assurance of the **TravelCam** platform are executed by the specialized engineering team at **GEP Protech**.

2.1 Key Project Personnel Roster

Below is the list of key project members and personnel involved in the engineering lifecycle of TravelCam:

- **Project Director & Lead Solution Architect:** Terence Ngulefac (GEP Protech Leadership)
- **Lead Backend Systems Engineer:** Senior Backend Specialist (GEP Protech Core Team)
- **Mobile Application Engineer:** Mobile Developer Lead (GEP Protech Mobile Unit)
- **Frontend Web Engineer:** Frontend UI Developer (GEP Protech Web Unit)
- **Payment Systems & Integrations Engineer:** Financial Gateway Specialist (GEP Protech Integration Team)
- **QA & DevOps Infrastructure Engineer:** Cloud Infrastructure Specialist (GEP Protech DevOps Unit)

2.2 Developer Roles & Assigned Personnel Matrix

Developer Role / Title	Assigned Personnel	Key Responsibilities & Module Focus	Primary Technology Stack
Lead Solution Architect / Project Manager	Terence Ngulefac	System architecture design, database entity modeling, API security standards, multi-tenant scoping policies, milestone tracking, and stakeholder reporting.	Laravel Architecture, DB Modeling, System Design
Lead Backend Engineer	Senior Backend Developer	Core RESTful API development, Eloquent tenant scoping, dynamic scope-based RBAC engine, polymorphic service engines, and payment gateway integration.	Laravel (PHP 8.x), MySQL / PSQL, REST APIs, Sanctum
Mobile App Developer (iOS & Android)	Mobile Engineer Lead	Cross-platform mobile app development, interactive bus seat matrix component, offline ticket storage, digital ticket wallet, QR code rendering, and push notifications.	React Native, Redux Toolkit / Context, Mobile SDKs
Frontend Web Developer	Frontend Web Lead	Admin panel interface implementation (Phase 1) and passenger single page application (Phase 4), responsive dashboard layouts, station POS UI, and dispatcher manifests.	Laravel Blade, React JS, CSS3, JavaScript
Payment & Integrations Engineer	Financial Gateway Specialist	Direct integration of local mobile money APIs (**CamPay** & **Fapshi**), automated webhook callbacks, transaction ledger reconciliation, and payment security.	REST Webhooks, CamPay SDK, Fapshi API, TLS 1.3
QA & DevOps Engineer	DevOps Infrastructure Lead	Cloud server provisioning on **Hostinger**, Nginx web server configuration, CI/CD pipeline automation, automated testing, app store deployment (Google Play & Apple App Store).	Hostinger VPS, Nginx, Redis, Git, App Store Connect

3. Stakeholders & Actor Profiles

Actor Role	Scope Level	Primary Responsibilities	Authentication Guard
System SuperAdmin	Platform	Global system overseer; approves new agency tenants, manages subscription tiers, audits platform health, and maintains the global permission catalog.	`auth:system_admin`
Agency Manager	Agency	Agency owner/executive; manages agency structure (Regions & Branches), creates custom agency roles, registers fleet buses, and sets global agency policies.	`auth:staff` (Scope: AGENCY)
Regional Manager	Region	Oversees a cluster of branch terminals within a geographic region; monitors regional trip schedules and regional staff performance.	`auth:staff` (Scope: REGION)
Branch Manager	Branch	Manages local terminal operations, branch staff assignments, daily counter cash accounts, and local branch departure timetables.	`auth:staff` (Scope: BRANCH)
Branch Staff	Branch	Counter Cashiers selling walk-in tickets, Dispatchers managing bus boarding/QR scanning, and Luggage Clerks weighing and registering baggage tags.	`auth:staff` (Scope: BRANCH)
Registered Passenger	Customer	Authenticated customer with account history, saved profile, digital ticket wallet, and service review capabilities.	`auth:web` / `auth:api`
Guest Passenger	Guest	Unauthenticated online or walk-in passenger booking tickets without creating an account (stored directly on `Booking`).	Guest (None)

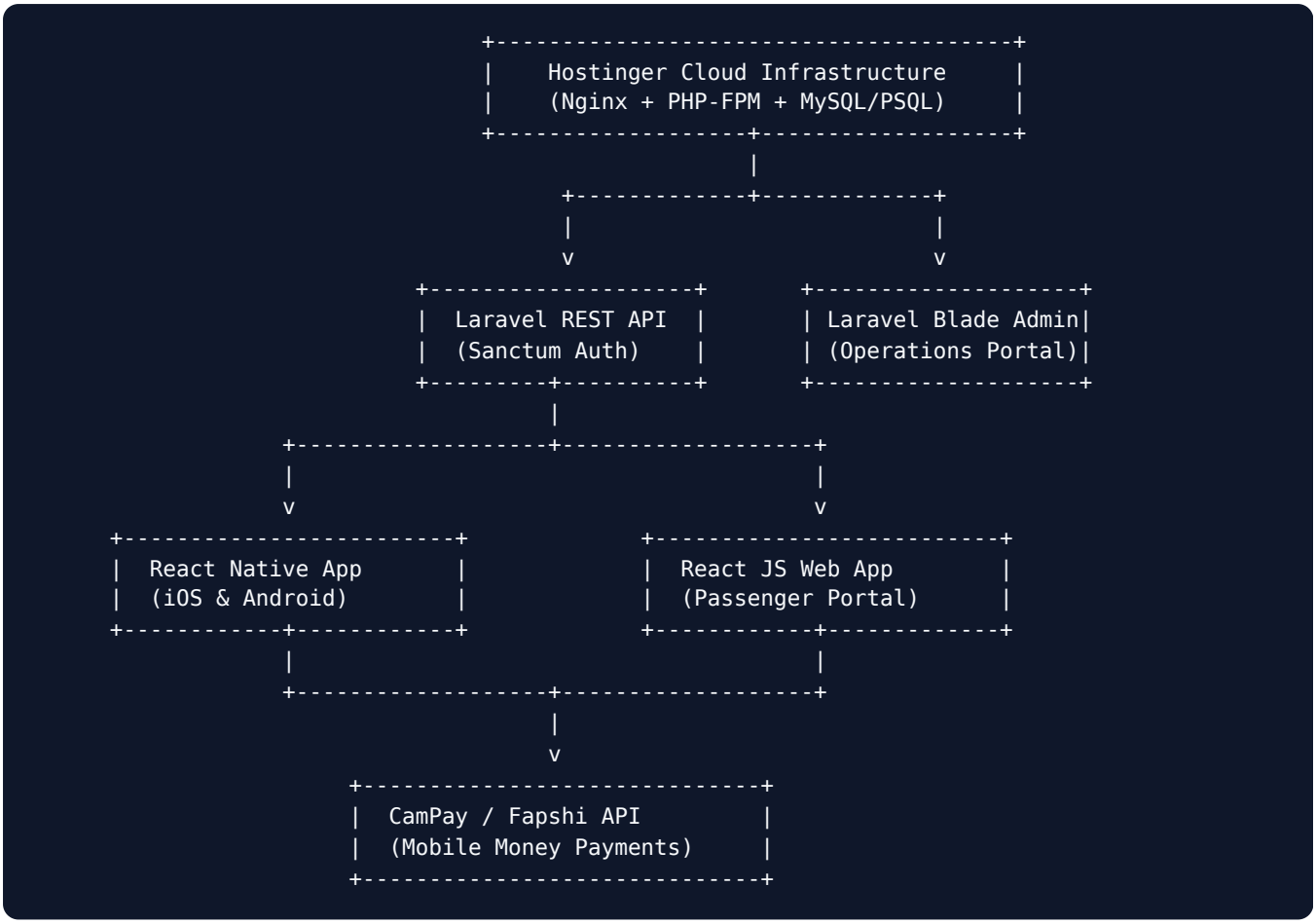
4. Technical Architecture & Developer Tech Stack

This section outlines the technical stack, infrastructure, and architectural choices mandated by GEP Protech for the engineering team developing the TravelCam platform.

4.1 Technology Stack Matrix

Component	Technology / Framework	Details & Justification
Mobile Application	**React Native**	Cross-platform native mobile application for **Android** and **iOS**. Delivers real-time trip booking, interactive seat selection, digital wallet, and QR ticket presentation.
Admin Panel Frontend	**Laravel Blade**	Server-side rendered administration portal integrated with Laravel backend. Used by SuperAdmins, Agency Managers, Branch Managers, Cashiers, and Dispatchers for high-speed counter operations.
Full Web Application	**React JS**	Modern Single Page Application (SPA) for web passengers, delivering identical user experience and feature parity with the mobile app for browser-based booking.
Backend & REST API	**Laravel (PHP 8.x)**	Modular RESTful API and core business logic engine, utilizing Laravel Sanctum for API token authentication and Eloquent ORM with global tenant scopes.
Database Engine	**MySQL** / **PostgreSQL (PSQL)**	Relational database storage with ACID compliance, JSON column support, and spatial capabilities. Uses database-level indexes and foreign key constraints for multi-tenant isolation.
Mobile Money & Payments	**CamPay** / **Fapshi**	Integrated local mobile money payment gateways (MTN Mobile Money, Orange Money) alongside card processing for automated payment verification.
Cloud Hosting & Deployment	**Hostinger**	VPS / Dedicated Cloud Hosting environment running Nginx web server, PHP-FPM, Redis caching, and automated SSL certificate management.

4.2 Developer Integration Architecture



5. Functional Requirements (FRs)

5.1 Module 1: Multi-Tenant Governance & Administration

- **FR-1.1 Agency Onboarding:** System SuperAdmin can review, approve, suspend, or reactivate agency tenant accounts.
- **FR-1.2 Tenant Data Isolation:** All tenant data (Branches, Routes, Staff, Fleet, Bookings) must be isolated per `agency_id`.
- **FR-1.3 Subscription Management:** System Admin can assign subscription plans (e.g., maximum allowed branches, buses, or monthly booking volume).
- **FR-1.4 Global Feature Catalog:** Platform master permissions (`Permission`) are defined globally by SuperAdmins and mapped to backend policy gates.

5.2 Module 2: Unified Staff & Dynamic Scope-Based RBAC

- **FR-2.1 Unified Staff Table:** Managers and operational staff are consolidated into a single `Staff` table with `employee_type` (`STAFF` or `MGR`).
- **FR-2.2 Scope Assignments:** Staff members are dynamically assigned management scopes via `StaffScopeAssignment` referencing `scopeable_type` (`Agency`, `Region`, `Branch`) and `scopeable_id`.

- **FR-2.3 Tenant Role Definition:** Agencies can create custom job titles (`Role`) attached to their `agency_id` and map them to global permissions via `RolePermission` .
- **FR-2.4 Multi-Role Support:** Operational staff can hold multiple active roles simultaneously (e.g. Cashier + Luggage Clerk) via `StaffRole` .

5.2.1 Non-Manager Operational Staff Management (Cashiers, Drivers, Dispatchers, Clerks)

> Non-manager staff (such as **Counter Cashiers**, **Bus Drivers**, **Conductors**, **Gate Dispatchers**, and **Luggage Clerks**) operate within a streamlined, action-oriented workflow:
> * **Home Terminal Binding:** Non-manager staff are directly assigned to their home terminal station via `branch_id` on the `Staff` table.
> * **Operational Roles & Permissions (`StaffRole`):** Agencies define operational roles (e.g. *"Station Cashier"*, *"Bus Driver"*, *"Luggage Clerk"*) linked to global action permissions (`booking.create` , `ticket.scan_qr` , `luggage.register`). Staff members receive one or more roles via the `StaffRole` junction.
> * **Trip Crew Assignments (`Trip`):** When scheduling a `Trip` , managers assign specific staff members to operational roles:
> * `driver_id [FK]` → points to `Staff.id` (Driver)
> * `conductor_id [FK]` → points to `Staff.id` (Conductor)
> * **Transaction Accountability:** Every walk-in ticket sale (`Booking.staff_id`), counter payment (`BranchCashAccount`), and luggage item (`Luggage.staff_id`) records the exact staff member who processed the transaction.

5.3 Module 3: Fleet Management & Branch-to-Branch Routing

- **FR-3.1 Vehicle Registry:** Track bus registration numbers, total seat capacity, bus model, amenities (Wi-Fi, AC, VIP), and operational status.
- **FR-3.2 Bus Maintenance:** Log maintenance work orders, repair costs, servicing dates, and service status (`BusMaintenance`).
- **FR-3.3 Branch-to-Branch Routes:** Routes are explicitly defined between origin and destination branch terminals (`Route` & `RouteBranchStop`).
- **FR-3.4 Multi-Branch Pickup Scheduling:** Schedule trips (`Trip` & `TripStop`) with sequential pickup/dropoff branch stops, estimated arrival times, and stop-specific pricing.

5.4 Module 4: Booking & E-Ticketing Engine

- **FR-4.1 Trip Search:** Passengers can search trips by selecting exact origin branch, destination branch, travel date, and preferred time.
- **FR-4.2 Real-Time Seat Matrix:** Interactive bus seat layout display (`Seat`) allowing passengers or cashiers to pick specific available seats.
- **FR-4.3 Guest & Authenticated Booking:** Bookings support both registered users (`user_id`) and guest passengers (`passenger_name` , `passenger_email` , `passenger_phone`).
- **FR-4.4 Mobile Money & Card Payments:** Process payments seamlessly via **CamPay** or **Fapshi** APIs (MTN Mobile Money, Orange Money) as well as counter cash payments.

- **FR-4.5 Digital Ticket QR Code:** Automatically generate a digital `Ticket` containing a unique passenger verification QR code upon booking confirmation.
- **FR-4.6 Refund & Cancellation:** Process ticket cancellations according to agency refund policies, creating ledger transactions.
- **FR-4.7 Onsite Walk-In Counter Booking:** Passengers visiting a physical station counter can book tickets directly through an on-duty Counter Cashier (`Staff`). The POS interface filters trips departing or picking up at that branch on that day (`Trip` + `TripStop`).
- **FR-4.8 Cashier Shift & Register Accounting:** Every counter booking stores `booking_channel = 'COUNTER'` , `booked_by_staff_id = cashier.id` , and `pickup_branch_id = station.id` . Cash collected is tied to the Cashier's daily shift account (`BranchCashAccount.cashier_staff_id`) for end-of-shift reconciliation.

5.5 Module 5: Terminal Operations & Luggage Registration

- **FR-5.1 Luggage Registration & Tagging:** Counter staff weigh and register passenger luggage, generating baggage tracking numbers and physical baggage tags (`Luggage`) with excess weight fee calculation. *(Note: Luggage tracking utilizes sequential baggage tag IDs and physical receipts without QR codes).*
- **FR-5.2 Passenger Ticket QR Boarding Scan:** Gate Dispatchers scan passenger ticket QR codes at terminal boarding gates to validate ticket authenticity, passenger identity, and boarding branch match.
- **FR-5.3 Boarding Manifest:** Real-time passenger boarding confirmation updating trip manifest status (`DEPARTED` , `IN_TRANSIT` , `COMPLETED`).
- **FR-5.4 Daily Cash Reconciliation:** Branch Managers balance counter cash register collections with `BranchCashAccount` ledger entries.

5.6 Module 6: Polymorphic Shared Engines

- **FR-6.1 Polymorphic Notifications:** Single `Notification` entity dispatching alerts to `SystemAdmin` , `Staff` , or `User` via `notifiable_type` + `notifiable_id` .
- **FR-6.2 Polymorphic Audit Log:** Log administrative and operational actions in `ActivityLog` linking `causer` (actor) and `subject` (entity modified).
- **FR-6.3 Polymorphic Attachments:** Store receipts, national ID copies, maintenance invoices in `Attachment` (`attachable_type` + `attachable_id`).
- **FR-6.4 Polymorphic Financial Ledger:** Record double-entry financial transactions in `LedgerTransaction` (`transactionable_type` + `transactionable_id`).
- **FR-6.5 Polymorphic Reviews:** Allow passengers to review trips, buses, or agency services via `Review` (`reviewable_type` + `reviewable_id`).

---

6. Non-Functional Requirements (NFRs)

6.1 Performance & Scalability

- **API Latency:** Trip search and availability requests respond under 200ms across mobile and web clients.
- **Seat Lock Concurrency:** 5-minute atomic database locks during seat selection to prevent double-booking.
- **Database Indexing:** Composite indexes on `(agency_id, status)`, `(trip_id, seat_number)`, and polymorphic columns `(notifiable_type, notifiable_id)`.

6.2 Security & Data Isolation

- **Multi-Tenant Scoping:** Enforce `agency_id` global scopes on all Eloquent models.
- **RBAC Policy Enforcement:** Every action verified against `StaffScopeAssignment` gates.
- **Encryption:** Passwords hashed with bcrypt; API tokens authenticated via Sanctum; TLS 1.3 enforced.

6.3 Availability & Reliability

- **Uptime Target:** 99.9% availability hosted on Hostinger Cloud infrastructure.
- **Automated Backups:** Daily database snapshots with point-in-time recovery up to 30 days.

---

7. Phased Deliverables & Implementation Roadmap

The development and deployment of TravelCam by GEP Protech will follow a strict 4-phase sequential delivery roadmap:

Phase 1: Admin Panel Web App (Laravel & Blade) — *Primary Priority*

- **Core Foundation:** Database schema migrations (MySQL/PSQL) and multi-tenant scoping.
- **Management Portals:** System SuperAdmin, Agency Manager, Regional Manager, and Branch Manager administrative dashboards.
- **Operational POS:** Station Counter Cashier POS interface for walk-in booking and daily shift cash reconciliation.
- **Gate Dispatcher Tool:** Web-based ticket QR code scanning and boarding manifest management.
- **Target Output:** Fully functional backend REST API and operational Admin Panel ready for agency deployment.

Phase 2: Mobile Application (React Native for Android & iOS) — *Secondary Priority*

- **Cross-Platform Mobile App:** Built with React Native, supporting both **Android** and **iOS** devices.
- **Passenger Features:** Real-time trip search, interactive bus seat selection, booking engine, and digital ticket wallet with QR codes.

- **Mobile Money Integration:** Direct integration with **CamPay** and **Fapshi** payment APIs for instant mobile money payment processing.
- **Push Notifications:** Instant trip departure alerts, ticket confirmations, and schedule updates.

Phase 3: Landing Page & App Redirect Website — *Tertiary Priority*

- **Public Web Presence:** Modern marketing website presenting TravelCam features, partner travel agencies, and route networks.
- **App Store Redirects:** Smart download badges and QR links directing visitors to download the official mobile apps on the **Google Play Store** and **Apple App Store**.

Phase 4: Full Passenger Web Application (React JS) — *Final Phase*

- **Web App SPA:** Built with React JS, bringing complete feature parity with the React Native mobile app to desktop and mobile web browsers.
- **Web Passenger Portal:** Allows desktop users to search trips, select seats, complete CamPay/Fapshi mobile money payments, and view digital tickets directly on the web.

---

8. Master 27-Entity Domain Model Summary

1. `SystemAdmin` (Governance)
2. `Agency` (Tenant Root)
3. `Region` (Organizational Hierarchy)
4. `Branch` (Station Terminal Base)
5. `Staff` (Unified Employee Model)
6. `StaffScopeAssignment` (Dynamic Scope Scoping)
7. `Role` (Tenant Custom Role)
8. `Permission` (Global Master Feature Capability)
9. `RolePermission` (Junction Table)
10. `StaffRole` (Multi-Role Junction)
11. `Bus` (Fleet Vehicle)
12. `BusMaintenance` (Work Orders & Servicing)
13. `Seat` (Bus Layout Matrix)
14. `Route` (Branch-to-Branch Route)
15. `RouteBranchStop` (Intermediate Branch Order)
16. `Trip` (Scheduled Departure)
17. `TripStop` (Timetable Pickup/Dropoff)
18. `User` (Registered Passenger)
19. `Booking` (Guest/Auth Reservation)

20. `Ticket` (Passenger QR Code E-Ticket)
21. `Luggage` (Baggage Registration & Sequential Tag ID)
22. `BranchCashAccount` (Daily Counter Reconciliation)
23. `Notification` (Polymorphic Alert Engine)
24. `ActivityLog` (Polymorphic Audit Trail)
25. `Attachment` (Polymorphic Document Engine)
26. `LedgerTransaction` (Polymorphic Double-Entry Accounting)
27. `Review` (Polymorphic Passenger Rating & Feedback)
