RideCompare – Technical PRD Companion

1. Purpose

This document is the technical companion to the RideCompare Product Requirements Document. It translates product intent into concrete system architecture, service boundaries, data handling policies, and evaluation criteria for engineering execution.

2. System Architecture Overview

RideCompare is designed as a stateless, request-driven system. All provider estimates are fetched in parallel, normalized, ranked, and returned to the client within a single request lifecycle. No booking or transaction logic exists within the platform.

Core Components

- Client Application (Web / Mobile)
- API Gateway
- Estimate Aggregation Service
- Provider Adapter Layer
- Comparison & Ranking Engine
- Redirect & Deep-Link Generator

3. Service Boundaries

Provider integrations are fully isolated behind adapters. This ensures that API or policy changes from any provider do not affect core business logic or other integrations.

4. API Contracts

POST /compare

{ pickup_lat, pickup_lng, drop_lat, drop_lng, category }
Response Payload

{ provider, normalized_category, estimated_price, eta_minutes, redirect_url }

5. Data Handling & Storage

RideCompare persists minimal data. All price and ETA estimates exist only in memory during request execution and are discarded immediately after response.

- Persistent: optional user preferences
- Ephemeral: in-memory aggregation results
- Explicitly forbidden: historical pricing storage

6. Provider Integration Strategy

Each provider adapter manages authentication, retries, timeouts, schema translation, and rate limiting. Adapters must strictly adhere to provider Terms of Service.

7. Non-Functional Requirements

- p95 latency < 3 seconds
- Graceful degradation if provider is unavailable
- Stateless services enabling horizontal scaling
- Full observability via metrics, logs, and traces

8. Security & Compliance

The system does not store provider credentials, does not initiate bookings, and does not handle payments. Redirect-based handoff is enforced as a hard boundary both at the API and UI layers.

9. Evaluation & Monitoring

- Price estimate accuracy monitoring
- Latency SLO tracking
- Redirect success rate measurement
- Provider API error rate alerts

10. MVP Technical Scope

The MVP consists of a single comparison endpoint, two provider adapters, price-based ranking, and redirect-only booking. The architecture is intentionally minimal while remaining extensible.