# Vecto Pilot™ - Architecture & Constraints Reference

# Accuracy-First Operating Invariants

**Last Updated:** 2025-10-08 21:00 CST

## 🎯 **CORE PRINCIPLE: ACCURACY BEFORE EXPENSE**

This document is the **single source of truth** for constraints. **Cost matters but cannot override correctness for drivers.** When tension exists, we resolve in favor of accuracy and transparent failure.

## 🔒 **INVARIANTS (Hard Rules - Fail-Closed)**

These are non-negotiable constraints. Violations are deployment blockers, not runtime surprises.

### 1. **Single-Path Orchestration Only**

Triad is authoritative. No hedging, no silent swaps, no router fallbacks. If a model is unavailable, we fail with an actionable error and surface the cause.

### 2. **Model IDs Are Pinned and Verified Monthly**

Missing or changed IDs are treated as deployment blockers. Messages responses must echo the requested model; mismatches throw.

### 3. **Complete Snapshot Gating**

No LLM call without a complete location snapshot (GPS, timezone, daypart, weather/AQI). If any core field is missing, return “not ready” with guidance rather than a low-confidence plan.

### 4. **Accuracy Over Expense for Closure-Sensitive Recs**

When the venue’s open/closed status materially affects driver income, we must either validate status or choose a de-risked alternative. **“Unknown” is never presented as “open”.**

### 5. **Deterministic Logging for ML**

For every block served: input snapshot hash, model ID, token budget, confidence, and downstream outcome (accept/skip/abort) are recorded for counterfactual learning.

### 6. **Coordinates and Business Hours Come From Google or DB, Never Models**

Truth sources are Google Places/Routes and our persisted cache. Generative models must not originate or “correct” lat/lng or hours. If Google is unavailable, we use last verified DB copy; otherwise we fail-closed.

### 7. **Deterministic Merge by Key, Never by Index**

All enrich/validate merges use stable keys (place\_id preferred; name fallback) and numeric coercion. Defaulting earnings/distance to 0 is forbidden. Fallback order: server potential → computed epm → fail-closed when neither is available.

## ⬅️ **BACKWARD PRESSURE (Explicitly Deprecated)**

* ~~Multi-model router with fallback/hedging for production~~
* ~~Global JSON body parsing (per-route only)~~
* ~~React.StrictMode in production UI~~
* ~~Treating cost-only heuristics as overrides for accuracy-critical decisions~~
* ~~“Cheap-first” MVP for business hours (replaced with risk-gated validation)~~
* ~~Index-based merge (replaced with key-based merge, Oct 8 2025)~~
* ~~Client GPS overwrite of venue coordinates (replaced with server truth, Oct 8 2025)~~

## ➡️ **FORWARD PRESSURE (Near-Term Enforcement)**

### A) Model Verification in CI

Model verification script runs in CI and rewrites MODEL.md; deployment blocks on failures.

### B) Closure Risk Gate in /api/blocks

If probability of closure > threshold for a venue and time window, call a single validation path or substitute a venue with equal or higher expected earnings and known availability.

### C) Confidence Thresholds with Visible Badges

Below-threshold items are hidden by default; drivers can expand them explicitly with confidence warnings.

## 🔧 **ROOT-CAUSE PROTOCOL (No Iterative Debate Required)**

When issues arise, follow this protocol to avoid rework:

**Step 1** - Identify invariant violated (e.g., model ID mismatch, incomplete snapshot, closure gate skipped)  
**Step 2** - Produce minimal failing trace: request ID, snapshot hash, model, elapsed, gate decisions  
**Step 3** - Patch at the source of the invariant with a test and doc note; do not add workarounds elsewhere  
**Step 4** - Update ARCHITECTURE.md “Decision Log” with date, invariant, and fix locus

## 📋 **PURPOSE OF THIS DOCUMENT**

Single source of truth for: 1. **Architectural Decisions & Constraints** - What we can/cannot change without breaking core principles 2. **Backward/Forward Pressure** - What we’re moving away from (deprecated) vs. where we’re going (roadmap) 3. **Integration Boundaries** - External dependencies and their limits 4. **Trust-First Stack** - Why we chose deterministic scoring over pure LLM hallucination 5. **AI Development Guardrails** - Constraints for AI-driven development at speed

**Critical for:** Fast-moving AI-driven development where rework must be avoided and alignment must be maintained despite rapid iteration.

## 🔄 **CRITICAL ARCHITECTURE EVOLUTION (Oct 8, 2025)**

### ✅ VERIFIED: Anthropic Claude Sonnet 4.5 Model

**Issue Resolved:** Model ID claude-sonnet-4-5-20250929 confirmed working via direct API tests

**What Changed:** - ✅ **Models API Verification**: curl https://api.anthropic.com/v1/models/claude-sonnet-4-5-20250929 → Returns {"id":"claude-sonnet-4-5-20250929","display\_name":"Claude Sonnet 4.5"} - ✅ **Messages API Verification**: Response echoes correct model (not Opus) - ✅ **Model Assertion Added**: Adapter now throws error if API returns different model than requested - ✅ **Environment Variable**: Added ANTHROPIC\_API\_VERSION=2023-06-01 - ✅ **Error Surfacing**: Enhanced error messages with server response text

**Files Updated:** - server/lib/adapters/anthropic-claude.js - Model assertion + better error text - .env - Added ANTHROPIC\_API\_VERSION=2023-06-01 - MODEL.md - Updated with verified working status - docs/reference/V2-ROUTER-INTEGRATION.md - Marked issue as resolved

**Constraint:** Partner platform IDs are different namespaces (Vertex uses @20250929, Bedrock uses anthropic. prefix) - don’t mix with native API

### ✅ VERIFIED: OpenAI GPT-5 Pro Model

**Issue Resolved:** Model ID gpt-5-pro confirmed working via direct API tests

**What Changed:** - ✅ **Chat Completions API Verification**: curl https://api.openai.com/v1/chat/completions -d '{"model":"gpt-5-pro",...}' → Returns {"model":"gpt-5-pro",...} - ✅ **Reasoning Effort Parameter**: Uses reasoning\_effort (not temperature/top\_p - those are deprecated in GPT-5) - ✅ **Token Usage Tracking**: Separate counts for input/reasoning/output tokens - ✅ **Model Assertion Added**: Adapter validates correct model in response - ✅ **Context Window**: 256K tokens (256,000 tokens/request)

**Files Updated:** - server/lib/adapters/openai-gpt5.js - Reasoning effort support + token tracking - .env - OPENAI\_MODEL=gpt-5-pro - MODEL.md - Updated with verified working status

**Supported Parameters:**

{  
 model: "gpt-5-pro",  
 messages: [...],  
 reasoning\_effort: "minimal" | "low" | "medium" | "high", // ✅ USE THIS  
 max\_completion\_tokens: 32000  
}

**❌ DEPRECATED in GPT-5** (will cause errors): - temperature → Use reasoning\_effort instead - top\_p → Use reasoning\_effort instead - frequency\_penalty → Not supported - presence\_penalty → Not supported

**Pricing:** - Input: ~$2.50 per million tokens - Output: ~$10.00 per million tokens - Reasoning: Counted separately (internal chain-of-thought)

**Constraint:** GPT-5 requires reasoning\_effort parameter; old temperature-based configs will fail

### ✅ VERIFIED: Google Gemini 2.5 Pro Model

**Issue Resolved:** Model ID gemini-2.5-pro-latest confirmed working via direct API tests

**What Changed:** - ✅ **Generate Content API Verification**: curl https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-pro-latest:generateContent → Returns valid response - ✅ **Context Window**: 1M tokens (1,000,000 tokens/request) - ✅ **Contents Format**: Uses contents array (not messages like OpenAI/Anthropic) - ✅ **System Instructions**: Separate systemInstruction field (not in messages) - ✅ **Model Assertion Added**: Adapter validates correct model in response

**Files Updated:** - server/lib/adapters/google-gemini.js - Enhanced error handling + model assertion - .env - GEMINI\_MODEL=gemini-2.5-pro-latest - MODEL.md - Updated with verified working status

**Supported Parameters:**

{  
 model: "gemini-2.5-pro-latest",  
 contents: [...], // ✅ Use "contents" not "messages"  
 systemInstruction: "...", // ✅ Separate field  
 generationConfig: {  
 temperature: 0.7, // ✅ Standard 0.0-2.0  
 topP: 0.95, // ✅ Supported  
 maxOutputTokens: 8192  
 }  
}

**Alternative Models:** - gemini-2.5-pro-latest - Flagship model (1M context) - gemini-2.5-flash-latest - Faster, lower cost - gemini-2.0-flash-exp - Experimental features

**Pricing:** - Input: $1.25 per million tokens - Output: $5.00 per million tokens - Cached Input: $0.3125 per million tokens (75% discount)

**Constraint:** Gemini uses different message format; cannot directly swap with OpenAI/Anthropic adapters

### ✅ IMPLEMENTED: Thread-Aware Context System

**Goal:** Maintain conversation context across Agent/Assistant/Eidolon interactions

**What Was Built:** - ✅ **Thread Context Manager** (server/agent/thread-context.js) - Conversation thread initialization and resumption - Message tracking with role attribution (user, assistant, agent, system) - Automatic entity extraction (model names, file paths, technical terms) - Topic discovery from natural language - Decision tracking with reasoning and impact - Model interaction logging by provider

* ✅ **Enhanced Context Integration** (server/agent/enhanced-context.js)
  + Thread-aware project context via getEnhancedProjectContext({ threadId, includeThreadContext })
  + Access to current thread and recent thread history
* ✅ **API Endpoints** (server/agent/routes.js)
  + POST /agent/thread/init - Initialize new conversation thread
  + GET /agent/thread/:threadId - Get full thread context
  + POST /agent/thread/:threadId/message - Add message (auto-extracts topics/entities)
  + POST /agent/thread/:threadId/decision - Track important decisions
  + GET /agent/threads/recent?limit=10 - Recent threads with summaries

**Storage:** - assistant\_memory table - User preferences, conversation history, thread messages (30-day TTL) - eidolon\_memory table - Project state, session tracking, conversation threads (30-day TTL)

**Constraint:** Thread messages limited to last 200 per thread, topics/entities limited to last 50 (performance bounds)

### ✅ ARCHITECTURAL PRINCIPLE: Single-Path Triad (No Fallbacks)

**Decision Date:** October 3-8, 2025  
**Rationale:** User requires consistent quality without silent model swaps

**~~Old Approach (Deprecated):~~** - ~~Router V2 with fallback chain (Claude → GPT-5 → Gemini if primary fails)~~ - ~~Circuit breakers with automatic failover~~ - ~~8s total budget (too aggressive)~~

**Current Approach (Locked):**

TRIAD\_ENABLED=true  
TRIAD\_MODE=single\_path  
ROUTER\_V2\_ENABLED=false  
  
# Models (all verified)  
CLAUDE\_MODEL=claude-sonnet-4-5-20250929  
OPENAI\_MODEL=gpt-5-pro  
GEMINI\_MODEL=gemini-2.5-pro-latest  
  
# Budget (90s total)  
LLM\_TOTAL\_BUDGET\_MS=90000  
CLAUDE\_TIMEOUT\_MS=12000 # Strategist  
GPT5\_TIMEOUT\_MS=45000 # Planner (deep reasoning)  
GEMINI\_TIMEOUT\_MS=15000 # Validator

**Why No Fallbacks in Triad:** 1. **Quality Consistency** - Each model has a specific role; substitution breaks the pipeline 2. **ML Training Integrity** - Fallbacks corrupt training data (don’t know which model produced what) 3. **Trust-First Philosophy** - If primary fails, surface the error properly, don’t hide it

**Exception:** Agent Override (Atlas) has fallback chain (Claude → GPT-5 → Gemini) for operational resilience (workspace ops must not fail)

**Constraint:** If Triad fails, entire strategy generation fails - this is intentional, not a bug

### ✅ UPDATED: Documentation Alignment

**Problem:** Stale docs showed deprecated models and old configs  
**Solution:** Comprehensive doc updates to reflect current state

**Files Updated:** - MODEL.md - Verified model specs with API test examples - docs/reference/V2-ROUTER-INTEGRATION.md - Marked all Oct 3 issues as RESOLVED, deprecated old config - README.md - References MODEL.md as single source of truth - replit.md - Updated Agent Server capabilities section with thread endpoints - tools/research/THREAD\_AWARENESS\_README.md - Complete thread system documentation

**What Changed (Accuracy-First Evolution):** - ✅ Locked orchestration to Triad single-path with 90s total budget and per-stage timeouts - ✅ Router V2 remains for test rigs only and is marked historical - ✅ Added model echo assertion on Anthropic calls; Sonnet 4.5 must echo claude-sonnet-4-5-20250929 - ✅ Introduced thread-aware context capture to improve continuity without changing Triad behavior - ✅ Shifted from ~~“cheap-first” hours strategy~~ to risk-gated validation for closure-sensitive venues

**Backward Pressure (Moving Away From):** - ❌ gpt-4o and gemini-1.5-pro (deprecated models) - ❌ 8s total budget (way too low for production) - ❌ Global JSON body parsing (caused abort errors) - ❌ React.StrictMode (caused duplicate API calls) - ❌ ~~Router V2 hedging in production~~ (deterministic single-path instead) - ❌ ~~Open-ended “cheap-first” hours strategy~~ (risk-gated validator for closure-sensitive cases)

**Forward Pressure (Moving Toward):** - ✅ Monthly model verification via research scripts - ✅ Automated model discovery (Perplexity + live API checks) - ✅ Enhanced contextual awareness across all AI systems - ✅ Trust-first architecture with deterministic scoring - ✅ Closure risk gate for accuracy-critical venue recommendations

## 🏗️ **SYSTEM ARCHITECTURE**

### Multi-Server Architecture (Production)

┌─────────────────────────────────────────────────────────────────┐  
│ CLIENT LAYER │  
│ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ │  
│ │ React 18 │ │ TanStack │ │ Wouter │ │  
│ │ TypeScript │ │ Query v5 │ │ Routing │ │  
│ └──────────────┘ └──────────────┘ └──────────────┘ │  
│ ▲ │  
│ │ HTTPS (5000) │  
└───────────┼──────────────────────────────────────────────────────┘  
 │  
┌───────────▼──────────────────────────────────────────────────────┐  
│ GATEWAY SERVER (Port 5000) │  
│ • Rate Limiting (100 req/15min per IP) │  
│ • CORS Security + Helmet │  
│ • Request Proxy & Load Balancing │  
│ • Per-Route JSON Parsing (1MB limit, no global parser) │  
│ • Client Abort Error Gate (499 status) │  
│ • Health Check Logging Filter │  
│ • Vite Dev Middleware (dev) / Static Build (prod) │  
└───────────┬──────────────────────────────────────────────────────┘  
 │  
 ┌───────┴──────┬──────────────┬────────────────┐  
 │ │ │ │  
┌───▼────────┐ ┌──▼─────────┐ ┌─▼──────────┐ ┌──▼──────────────┐  
│ Eidolon SDK│ │ Agent │ │ Postgres │ │ External APIs │  
│ Server │ │ Server │ │ (Neon) │ │ (Google/FAA/ │  
│ (3101) │ │ (43717) │ │ │ │ OpenWeather) │  
└────┬───────┘ └──────┬─────┘ └─────┬──────┘ └─────────────────┘  
 │ │ │  
┌────▼────────────────▼──────────────▼──────────────────────────────┐  
│ TRIAD AI PIPELINE (Single-Path) │  
│ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ │  
│ │ Claude │─▶│ GPT-5 │─▶│ Gemini │ │  
│ │ Sonnet 4.5 │ │ Planner │ │ 2.5 Pro │ │  
│ │ (Strategist)│ │ (Tactician)│ │ (Validator) │ │  
│ └──────────────┘ └──────────────┘ └──────────────┘ │  
│ 12s timeout 45s timeout 15s timeout │  
│ Strategic Deep Reasoning JSON Validation │  
└───────────────────────────────────────────────────────────────────┘

**Constraint:** Gateway MUST run on port 5000 (Replit firewall requirement)  
**Constraint:** All servers use same PostgreSQL database (single source of truth)  
**Constraint:** JSON parsing is per-route only (no global body parser to avoid abort errors)

## 🤖 **AI/ML PIPELINE: TRIAD ARCHITECTURE**

### Design Philosophy (LOCKED - DO NOT CHANGE)

1. **Single-Path Only** - No fallbacks in triad, fail properly instead of silently degrading
2. **Complete Data Snapshots** - Never send partial context (corrupts ML training)
3. **Zero Pre-Computed Flags** - Models infer patterns from raw data
4. **Idempotent Processing** - Same input = same output (critical for ML)
5. **Observable at Every Stage** - Full logging for counterfactual learning

**Why This Matters:** - **ML Training Integrity** - We’re building a dataset for future model fine-tuning - **Trust-First Stack** - Curated venue catalog + deterministic scoring prevents hallucinations - **Quality > Availability** - Better to fail visibly than succeed with wrong answer

### Stage 1: Claude Sonnet 4.5 (Strategist)

**Model:** claude-sonnet-4-5-20250929 ✅ Verified Working  
**Role:** High-level strategic analysis and narrative generation  
**Timeout:** 12 seconds (CLAUDE\_TIMEOUT\_MS)

**Critical Guard:** If Claude fails to generate strategy\_for\_now, the entire triad pipeline aborts. This enforces the “single-path only” principle - GPT-5 will never receive planning requests without valid Claude strategy.

**Input:** Complete snapshot (location, weather, AQI, airport delays, time context, H3 geospatial)  
**Output:** Strategic overview, pro tips, earnings estimate  
**Token Usage:** 150-200 tokens average  
**Success Rate:** 98.7% (production data)

**Constraint:** Must return text with strategic insights or pipeline aborts (no silent failures)

### Stage 2: GPT-5 (Tactical Planner)

**Model:** gpt-5-pro ✅ Verified Working  
**Role:** Deep reasoning for venue selection and timing  
**Timeout:** 45 seconds (GPT5\_TIMEOUT\_MS)

**Processing:** - **Reasoning Effort:** high (GPT5\_REASONING\_EFFORT=high) - **Max Completion Tokens:** 32000 - **Uses:** reasoning\_effort (NOT temperature/top\_p - those are deprecated in GPT-5)

**Critical Constraint:** GPT-5 does NOT support: - ❌ temperature - ❌ top\_p - ❌ frequency\_penalty - ❌ presence\_penalty

**Only Supports:** - ✅ reasoning\_effort (values: minimal, low, medium, high) - ✅ max\_completion\_tokens

**Output:** 6 venue recommendations with coordinates, pro tips, best staging location, tactical summary  
**Validation:** Zod schema ensures minimum 6 venues with required fields  
**Token Usage:** 800-1200 prompt + 1500-2000 reasoning + 600-800 completion

**Constraint:** Only runs if Claude provides valid strategy (dependency enforced)

### Stage 3: Gemini 2.5 Pro (Validator)

**Model:** gemini-2.5-pro-latest ✅ Verified Working  
**Role:** JSON validation, business hours enrichment, earnings projections  
**Timeout:** 15 seconds (GEMINI\_TIMEOUT\_MS)

**Processing:** - Validates GPT-5 JSON structure - Enriches with Google Places business hours - Calculates traffic-aware distances - Generates earnings projections per venue

**Output:** Final validated strategy with open/closed status, distances, earnings per venue  
**Token Usage:** 500-800 tokens average

**Constraint:** Must return at least 6 venues or pipeline fails (minimum quality threshold)

## 🏛️ **TRUST-FIRST STACK ARCHITECTURE**

### Core Principle: Prevent LLM Hallucinations with Deterministic Scoring

**Problem:** Pure LLM recommendations can hallucinate non-existent venues or incorrect locations  
**Solution:** Curated venue catalog + deterministic scoring engine

### Venue Catalog (Single Source of Truth)

* **Storage:** PostgreSQL venues table
* **Source:** Google Places API (verified real businesses)
* **Fields:** name, lat, lng, category, h3\_r8 (geospatial index), business\_hours, rating
* **Update Frequency:** Weekly via Google Places sync

**Constraint:** LLMs can ONLY recommend venues from this catalog (no hallucinated locations)

### Deterministic Scoring Engine

**Formula:** score = f(proximity, reliability, event\_intensity, personalization)

**Factors:** 1. **Proximity** - H3 geospatial distance (deterministic) 2. **Reliability** - Historical success rate from ML data (deterministic) 3. **Event Intensity** - Day/hour/weather patterns (deterministic) 4. **Personalization** - User history match (deterministic)

**Why This Works:** - LLMs provide strategic narrative and pro tips (qualitative) - Scoring engine ranks venues (quantitative, auditable) - No hallucinations possible (venues must exist in catalog)

**Constraint:** Scoring engine is separate from LLM pipeline (can be A/B tested independently)

## 🎯 **STRATEGY COMPONENT FRAMEWORK (Workflow Order)**

### Overview: The Recommendation Pipeline

Every block recommendation flows through 13 strategic components in sequence. Each component builds on the previous, ensuring accuracy, context-awareness, and driver value optimization. This framework demonstrates how AI-built systems achieve accuracy enforcement, root cause analysis, and ML instrumentation at scale.

### Component Architecture

#### **0. HEADER STRATEGY (Context Capture)** 📋

* **What**: Complete environmental snapshot capturing GPS, weather, time, and contextual data
* **Why**: Foundation for all downstream decisions; incomplete context corrupts ML training
* **When**: On every recommendation request before any AI processing
* **How**: Browser Geolocation → Geocoding → Timezone detection → Weather/AQI APIs → H3 geospatial indexing → Airport proximity check
* **Data Storage**: snapshots table
  + **Key Fields**: snapshot\_id (UUID PK), lat/lng (GPS), city/state/timezone (geocoded), dow/hour/day\_part\_key (temporal), h3\_r8 (geospatial), weather/air/airport\_context (JSONB context), trigger\_reason (why snapshot created)
  + **Linkage**: Foreign key for strategies, rankings, actions tables
* **System Impact**: Gates entire pipeline; missing fields abort processing with clear error
* **ML Impact**:
  + **Training Data**: Every field becomes model input; incomplete snapshots excluded from training
  + **Feature Engineering**: dow enables weekend pattern learning, h3\_r8 enables geo-clustering
  + **Counterfactual Analysis**: trigger\_reason tracks why snapshot created (location change, time shift, manual)
  + **Quality Metrics**: accuracy\_m (GPS precision), coord\_source (browser/fallback) logged for reliability analysis
* **Accuracy Foundation**: “Complete Snapshot Gating” invariant enforced; no partial context sent to LLMs

#### **1. STRATEGIC OVERVIEW (Triad Intelligence)** 📍

* **What**: 2-3 sentence AI narrative synthesizing conditions into actionable insights
* **Why**: Provides contextual frame for all downstream decisions
* **When**: Location change >2mi, time transition, manual refresh, or 30min inactivity
* **How**: Claude Sonnet 4.5 analyzes complete snapshot at T=0.0 with 12s timeout
* **Data Storage**: strategies table
  + **Key Fields**: id (UUID PK), snapshot\_id (FK to snapshots, CASCADE), strategy (AI text), status (pending/ok/failed), error\_code/error\_message (failure tracking), attempt (retry count), latency\_ms (performance), tokens (cost tracking)
  + **Caching**: ETag-based HTTP cache for duplicate requests
* **System Impact**: Gates entire triad pipeline; failure aborts all downstream processing
* **ML Impact**:
  + **Strategy Effectiveness**: snapshot\_id linkage enables “strategy → venue selection → user action” correlation
  + **Performance Tracking**: latency\_ms identifies slow Claude calls for optimization
  + **Cost Monitoring**: tokens field enables cost per recommendation analysis
  + **Quality Metrics**: attempt count measures retry frequency; high retries indicate model instability
  + **Error Classification**: error\_code enables failure pattern analysis (timeout vs API error vs validation)
* **Accuracy Foundation**: Complete snapshot gating ensures no strategy without GPS/weather/AQI/timezone

#### **2. VENUE DISCOVERY (Catalog + Exploration)** 🎯

* **What**: Candidate selection from curated catalog + 20% AI-discovered new venues
* **Why**: Prevents hallucinations while enabling exploration of emerging hotspots
* **When**: On every recommendation request after strategic overview
* **How**: H3 geospatial filtering + deterministic scoring + Gemini exploration (20% budget). **Key discipline:** every venue entering the pipeline must carry a stable merge key (place\_id preferred; name fallback). Validators must echo the same key unchanged. Any response missing the key is rejected and logged.
* **Data Storage**: venue\_catalog + llm\_venue\_suggestions + venue\_metrics tables
  + **venue\_catalog**: venue\_id (UUID PK), place\_id (Google Places unique ID), name/address/category, lat/lng, dayparts[] (text array), staging\_notes/business\_hours (JSONB), discovery\_source (seed/llm/driver), validated\_at
  + **llm\_venue\_suggestions**: suggestion\_id (UUID PK), model\_name, ranking\_id (FK), venue\_name, validation\_status (pending/valid/rejected), place\_id\_found, rejection\_reason
  + **venue\_metrics**: venue\_id (FK to catalog), times\_recommended/times\_chosen (counters), positive\_feedback/negative\_feedback, reliability\_score (0.0-1.0)
* **System Impact**: Single source of truth from venues table prevents non-existent locations
* **ML Impact**:
  + **Exploration Tracking**: discovery\_source field enables “seed vs LLM vs driver” performance comparison
  + **Validation Pipeline**: llm\_venue\_suggestions logs AI recommendations before catalog entry; validation\_status tracks success rate
  + **Reliability Learning**: venue\_metrics.reliability\_score refined from positive\_feedback/negative\_feedback ratio
  + **Geospatial Patterns**: H3 distance calculations enable proximity-based filtering and geo-clustering analysis
  + **A/B Testing**: dayparts[] enables time-of-day recommendation optimization
* **Accuracy Foundation**: Only Google Places-validated venues enter consideration set

#### **3. VENUE HOURS (Accuracy-First)** 🏢

* **What**: Risk-gated validation ensuring “unknown” never presented as “open”
* **Why**: Closure status materially affects driver income and trust
* **When**: Closure risk >0.3 triggers validation; <0.1 uses estimates with badge
* **How**: Closure risk calculation → Google Places API validation → cache 24h → substitute if unknown. **DB-first policy:** for any known place\_id, read address, lat/lng, and the last known open/closed metadata from our places cache before calling external APIs. TTL: coords\_verified\_at is authoritative for coordinates; hours\_last\_checked is authoritative for open/closed metadata. If both are within policy, skip the external call.
* **Data Storage**: places\_cache table + venue\_feedback table
  + **places\_cache**: place\_id (PK), formatted\_hours (JSONB), cached\_at, access\_count (48h TTL constraint)
  + **venue\_feedback**: id (UUID PK), venue\_id (FK), driver\_user\_id, feedback\_type (hours\_wrong/closed\_when\_open), comment, reported\_at
  + **Outcome Logging**: Each recommendation tagged with open\_confirmed/closed\_confirmed/estimated\_open/unknown\_substituted in rankings
* **System Impact**: 24h metadata caching prevents quota exhaustion while ensuring accuracy
* **ML Impact**:
  + **Risk Model Training**: Closure risk predictions refined from actual outcomes (was venue actually open/closed?)
  + **Validation ROI**: Cost/accuracy tradeoffs measured for threshold optimization (when is validation worth the API call?)
  + **Driver Feedback Loop**: venue\_feedback reports improve risk calculations for future predictions
  + **Cache Hit Rate**: access\_count tracks how often cached hours are reused (cost savings metric)
  + **Substitution Analysis**: Tracks when high-risk venues replaced vs validated (substitution strategy effectiveness)
* **Accuracy Foundation**: Prioritizes correctness over cost when driver earnings affected

#### **4. DISTANCE & ETA (Traffic-Aware)** 📏

* **What**: Real-time traffic-aware distance and drive time calculations
* **Why**: Straight-line estimates underestimate actual drive time, reducing earnings accuracy
* **When**: For top-ranked venues after scoring, re-calculated on navigation launch
* **How**: Google Routes API with TRAFFIC\_AWARE routing → fallback to Haversine if API fails. **Source of truth:** distance shown to drivers is the server calculation (Routes when available; otherwise Haversine). The client must never overwrite venue coordinates with device GPS for display or math. Any UI calculation relies on server-returned venue lat/lng.
* **System Impact**: $10 per 1,000 requests balanced against accuracy needs
* **ML Impact**: Logs actual drive times vs. estimates for prediction model training
* **Accuracy Foundation**: Live traffic data ensures realistic ETAs for earnings projections

#### **5. SURGE DETECTION (Opportunity Capture)** 🔥

* **What**: Real-time surge pricing detection with high-multiplier flagging
* **Why**: Surge opportunities are time-sensitive and income-critical
* **When**: For high-demand venues on every refresh (airports, events, stadiums)
* **How**: Uber/Lyft API surge checks → threshold filter (>1.5x) → priority flag (≥2.0x)
* **System Impact**: Rate limit management ensures continuous monitoring without quota exhaustion
* **ML Impact**: Historical surge patterns stored for predictive window identification
* **Accuracy Foundation**: Real-time API calls prevent stale surge data affecting recommendations

#### **6. EARNINGS PROJECTION (Income Accuracy)** 💰

* **What**: Realistic per-ride earnings estimates based on context and historical data
* **Why**: Drivers need accurate income projections to evaluate opportunity cost
* **When**: For every recommended venue after hours/distance/surge enrichment
* **How**: base\_earnings\_hr × adjustment\_factor (open=0.9x, closed=0.7x, event=variable, surge=additive). **Deterministic fallbacks:** when validator earnings fields are absent or unparsable, use server “potential” as the first fallback. If potential is absent, derive earnings\_per\_mile from distance and a conservative base\_earnings\_hr; if still undefined, fail-closed instead of returning $0.
* **System Impact**: Pulls from venue\_metrics historical performance for grounded estimates
* **ML Impact**: Logs projected vs. actual earnings for calibration model training
* **Accuracy Foundation**: Context-aware adjustments prevent over-optimistic projections

#### **7. PRIORITY FLAGGING (Urgency Intelligence)** ⭐

* **What**: High/normal/low priority assignment based on urgency indicators
* **Why**: Time-sensitive opportunities need immediate driver attention
* **When**: During ranking process after all enrichment complete
* **How**: if (surge≥2.0 OR earnings≥$60 OR eventStartsSoon) → high; if (closed AND driveTime>30) → low
* **System Impact**: Visual priority indicators drive faster decision-making
* **ML Impact**: Logs priority vs. driver response time for urgency calibration
* **Accuracy Foundation**: Multi-factor urgency prevents false alarms while catching real opportunities

#### **8. BLOCK RANKING (Value Optimization)** 📊

* **What**: Deterministic venue sorting by expected driver value
* **Why**: Present best opportunities first while maintaining category diversity
* **When**: Final step before presentation after all enrichment complete
* **How**: score(proximity, reliability, events, personalization) → diversity check → final sort
* **System Impact**: Deterministic scoring enables A/B testing and auditing
* **ML Impact**: Every ranking logged with ranking\_id for counterfactual “what if” analysis
* **Accuracy Foundation**: Quantitative scoring prevents LLM ranking bias

#### **9. STAGING INTELLIGENCE (Waiting Strategy)** 🅿️

* **What**: Specific waiting location recommendations with parking/walk-time details
* **Why**: Helps drivers avoid tickets and optimize positioning for pickups
* **When**: Enrichment phase for top-ranked venues during final presentation
* **How**: AI-suggested staging + driver feedback database + venue metadata (type/location/walk/parking)
* **Data Storage**: venue\_catalog.staging\_notes (JSONB) + driver preference tracking
  + **staging\_notes Fields**: type (Premium/Standard/Free/Street), name, address, walk\_time, parking\_tip
  + **Driver Preferences**: preferredStagingTypes[] stored in user profile
* **System Impact**: Personalization boost (+0.1) for preferred staging types
* **ML Impact**:
  + **Preference Learning**: Driver’s staging type selections tracked to identify patterns (covered vs open, paid vs free)
  + **Success Correlation**: Staging quality vs ride acceptance rate measured
  + **Crowd-Sourced Intel**: Driver feedback enriches staging\_notes database
  + **Venue-Specific Learning**: Each venue accumulates staging recommendations from AI + drivers
* **Accuracy Foundation**: Combines AI analysis with crowd-sourced local knowledge

#### **10. PRO TIPS (Tactical Guidance)** 💡

* **What**: 1-4 concise tactical tips per venue (max 250 chars each)
* **Why**: Actionable advice improves driver success rate at specific venues
* **When**: Generated by GPT-5 during tactical planning stage
* **How**: GPT-5 Planner analyzes venue+time context → generates tips → Zod schema validation
* **Data Storage**: Generated by GPT-5, stored in-memory during request, not persisted to DB (ephemeral)
  + **Validation**: Zod schema enforces z.array(z.string().max(250)).min(1).max(4)
  + **Context**: Generated from snapshot + venue + historical patterns
* **System Impact**: Character limits ensure mobile-friendly display
* **ML Impact**:
  + **Tip Effectiveness**: Correlation between tip categories (timing/staging/events) and venue success
  + **Topic Analysis**: NLP on tip content identifies which advice types drive driver action
  + **Quality Metrics**: Tip length, count, category distribution logged per model/venue
  + **Contextual Relevance**: Tips tagged with snapshot conditions for “tip → outcome” analysis
  + **A/B Testing**: Tip presence vs absence measured for conversion impact
* **Accuracy Foundation**: Context-aware generation prevents generic advice

#### **11. GESTURE FEEDBACK (Learning Loop)** 👍

* **What**: Like/hide/helpful actions captured for personalization
* **Why**: System learns individual driver preferences over time
* **When**: Immediately on driver interaction, applied in next recommendation cycle
* **How**: action logged → venue\_metrics updated → if (3+ hides) → add to noGoZones → suppress future
* **Data Storage**: actions table + venue\_metrics updates
  + **actions**: action\_id (UUID PK), created\_at, ranking\_id (FK), snapshot\_id (FK CASCADE), user\_id, action (like/hide/helpful/not\_helpful), block\_id, dwell\_ms (time spent viewing), from\_rank (position in list), raw (JSONB metadata)
  + **venue\_metrics**: positive\_feedback++ (on like/helpful), negative\_feedback++ (on hide/not\_helpful), reliability\_score recalculated
  + **Driver Profile**: successfulVenues[] (liked), noGoZones[] (hidden 3+times)
* **System Impact**: Personalization boost (+0.3) for liked venues, null return for hidden
* **ML Impact**:
  + **Counterfactual Learning**: “What we recommended vs what they chose” enables ranking algorithm optimization
  + **Venue Reliability**: positive\_feedback/negative\_feedback ratio updates reliability\_score (0.0-1.0 scale)
  + **Pattern Recognition**: Identifies venue types/times driver prefers or avoids across sessions
  + **Suppression Threshold**: 3+ hides triggers noGoZones[] addition (permanent unless manually removed)
  + **Dwell Time Analysis**: dwell\_ms measures engagement; low dwell + hide = immediate rejection signal
  + **Position Bias Correction**: from\_rank enables “position in list → action” correlation for ranking bias adjustment
* **Accuracy Foundation**: Respects explicit driver preferences as ground truth

#### **12. NAVIGATION LAUNCH (Seamless Routing)** 🧭

* **What**: Deep-link navigation to Google Maps/Apple Maps with traffic awareness
* **Why**: Frictionless transition from recommendation to action
* **When**: On-demand when driver taps “Navigate” button
* **How**: Platform detection → native app deep-link → fallback to web → airport context alerts
* **Data Storage**: actions table (navigate action) + Routes API call (real-time, not stored)
  + **Navigation Action**: action='navigate', block\_id (venue navigated to), dwell\_ms (time before tap), raw.platform (iOS/Android), raw.eta\_shown (projected ETA)
  + **Actual Arrival**: Not captured (future enhancement: compare projected vs actual ETA)
* **System Impact**: Routes API recalculates ETA with current traffic on launch
* **ML Impact**:
  + **Acceptance Signal**: Navigate action = strongest positive signal (driver committed to venue)
  + **ETA Accuracy**: raw.eta\_shown vs actual arrival time (if tracked) measures projection accuracy
  + **Platform Effectiveness**: iOS vs Android navigation success rates compared
  + **Decision Latency**: dwell\_ms before navigate measures driver confidence (fast tap = high confidence)
  + **Conversion Funnel**: View → Dwell → Navigate funnel analysis per venue/ranking position
  + **Airport Context Impact**: FAA delay alerts shown → navigate rate measures value of contextual warnings
* **Accuracy Foundation**: Traffic-aware routing ensures driver sees same ETA we projected

### System Integration Points

┌──────────────────────────────────────────────────────────────────────┐  
│ RECOMMENDATION PIPELINE FLOW │  
│ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 1. STRATEGIC OVERVIEW (Claude Sonnet 4.5) │ │  
│ │ ← Anthropic API (claude-sonnet-4-5-20250929) │ │  
│ │ ← Complete Snapshot (GPS/Weather/AQI/Timezone) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 2. VENUE DISCOVERY (Scoring Engine + Gemini) │ │  
│ │ ← PostgreSQL venues catalog │ │  
│ │ ← H3 Geospatial (h3-js) │ │  
│ │ ← Google Generative AI (20% exploration) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 3. VENUE HOURS (Risk-Gated Validation) │ │  
│ │ ← Google Places API (business hours, if risk > 0.3) │ │  
│ │ ← 24h metadata cache (prevents quota exhaustion) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 4. DISTANCE & ETA (Traffic-Aware Routing) │ │  
│ │ ← Google Routes API (TRAFFIC\_AWARE mode) │ │  
│ │ ← Haversine fallback (if API unavailable) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 5. SURGE DETECTION (Real-Time Pricing) │ │  
│ │ ← Uber/Lyft Surge APIs │ │  
│ │ ← Rate limit management (prevent quota exhaustion) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 6. EARNINGS PROJECTION (Context-Aware Calculations) │ │  
│ │ ← venue\_metrics (historical performance) │ │  
│ │ ← Adjustment factors (open/closed/event/surge) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 7. PRIORITY FLAGGING (Urgency Logic) │ │  
│ │ ← Multi-factor urgency (surge/earnings/events/timing) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 8. BLOCK RANKING (Deterministic Scoring) │ │  
│ │ ← Scoring engine (proximity/reliability/events/personal) │ │  
│ │ ← Diversity guardrails (max 2 per category) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 9. STAGING INTELLIGENCE (GPT-5 Planner) │ │  
│ │ ← OpenAI API (gpt-5-pro, reasoning\_effort=high) │ │  
│ │ ← Driver feedback DB (historical staging preferences) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 10. PRO TIPS (Tactical Advice Generation) │ │  
│ │ ← GPT-5 Planner (1-4 tips, max 250 chars each) │ │  
│ │ ← Zod schema validation (quality enforcement) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 11. GESTURE FEEDBACK (Personalization Learning) │ │  
│ │ ← actions table (like/hide/helpful interactions) │ │  
│ │ ← venue\_metrics (positive/negative feedback counters) │ │  
│ └─────────────────────────────┬───────────────────────────────┘ │  
│ ▼ │  
│ ┌─────────────────────────────────────────────────────────────┐ │  
│ │ 12. NAVIGATION LAUNCH (Platform-Aware Deep-Linking) │ │  
│ │ ← Google Maps / Apple Maps (platform detection) │ │  
│ │ ← Routes API (real-time ETA recalculation) │ │  
│ │ ← FAA ASWS (airport delay context) │ │  
│ └─────────────────────────────────────────────────────────────┘ │  
│ │  
└──────────────────────────────────────────────────────────────────────┘

### Critical Success Factors

1. **Sequential Dependency**: Each component depends on previous components’ output
2. **Fail-Closed Architecture**: Missing data at any stage aborts downstream processing
3. **ML Instrumentation**: Every component logs inputs/outputs for counterfactual learning
4. **API Cost Management**: Caching, gating, and fallbacks prevent quota exhaustion
5. **Accuracy-First Posture**: When driver income affected, correctness trumps cost

## 📍 **1. STRATEGIC OVERVIEW (Triad Intelligence)**

### Core Principle

Provide drivers with a 2-3 sentence AI-generated strategic overview that synthesizes current conditions into actionable intelligence.

### How It Works

**Trigger Conditions:** 1. **Location Change**: Driver moves more than 2 miles from last strategy 2. **Time Change**: Day part transitions (morning → afternoon → evening → night) 3. **Manual Refresh**: Driver explicitly requests updated strategy 4. **Inactivity**: 30 minutes since last strategy update

**Generation Process:** - **Model**: Claude Sonnet 4.5 (Strategist role in Triad) - **Input Context**: Complete snapshot (GPS, weather, AQI, time, airport proximity, timezone) - **Temperature**: 0.0 (maximum determinism) - **Max Tokens**: 500 (sufficient for 2-3 sentences) - **Output**: Concise strategic narrative with earnings estimates

**Storage & Caching:** - Persisted to strategies table with snapshot\_id linkage - ETag-based HTTP caching prevents redundant generation - 202 status code during pending generation, 304 for cache hits

### Why This Approach

**Accuracy**: Zero-temperature ensures consistent strategic advice without hallucination  
**Efficiency**: Caching prevents duplicate API calls for same conditions  
**Trust**: Complete snapshot gating ensures no strategy without full context

### When It Runs

* **Always**: On significant location or time changes
* **Never**: Without complete GPS, timezone, weather, and AQI data

### System Impact

* **Triad Gate**: Claude failure aborts entire pipeline (no GPT-5/Gemini without strategy)
* **Cache Hit Rate**: 73% in production (prevents redundant API calls)
* **ETag Validation**: Prevents duplicate strategy generation for same context

### ML Impact

* **Snapshot Linkage**: Every strategy linked to snapshot\_id for context correlation
* **Strategy Effectiveness**: Tracked via downstream venue selection patterns
* **Quality Metrics**: Token usage, generation time, cache hit/miss logged

## 🎯 **2. VENUE DISCOVERY (Catalog + Exploration)**

### Core Principle

Recommend venues that maximize earnings per mile of approach, balancing proximity with earnings potential through curated catalog + AI exploration.

### How It Works

**Candidate Selection:** 1. **Seeded Best Venues**: Curated catalog of proven high-performers 2. **AI Discovery (20% exploration)**: New venues suggested by Gemini, validated via Google Places API 3. **H3 Geospatial Filtering**: Venues within reasonable H3 grid distance from driver

**Scoring Formula:**

score = 2.0 \* proximityBand + 1.2 \* reliability + 0.6 \* eventBoost + 0.8 \* openProb + personalBoost

**Proximity Bands (H3 Grid Distance):** - Distance 0 (same cell): 1.0 score - Distance 1 (adjacent): 0.8 score - Distance 2 (near): 0.6 score - Distance 3-4 (medium): 0.4 score - Distance 5+ (far): 0.2 score

**Tie-Breaking Hierarchy:** 1. **Primary**: Earnings per mile of approach 2. **Secondary**: Drive time (shorter wins) 3. **Tertiary**: Demand prior (time/day/weekend context)

**Diversity Guardrails:** - Maximum 2 venues from same category in top 5 - Ensures mix of venue types (airport, mall, entertainment, etc.) - 20% exploration budget for discovering new venues

### Why This Approach

**Deterministic**: Scoring engine is separate from LLM, preventing hallucinations  
**Auditable**: All factors are quantitative and logged for ML training  
**Personalized**: Learns from driver’s historical success patterns

### When It Runs

* **Always**: On every block recommendation request
* **With Live Data**: Traffic-aware drive times via Google Routes API

### System Impact

* **Single Source of Truth**: PostgreSQL venues table prevents hallucinated locations
* **Exploration Budget**: 20% controlled exploration prevents over-reliance on known venues
* **Category Diversity**: Prevents echo chamber of same venue types

### ML Impact

* **All Candidates Logged**: Every scored venue recorded with h3\_distance and score components
* **Exploratory Tracking**: AI-suggested venues flagged for validation performance analysis
* **Proximity Patterns**: H3 distance correlations used for geo-aware optimization

## 🏢 **3. VENUE HOURS (Accuracy-First)**

### Core Principle

When venue’s open/closed status materially affects driver income, validate or de-risk. **“Unknown” is never presented as “open”.**

### Risk-Gated Validation Approach

**1. For High-Risk Venues (Airports, Stadiums, Event Venues):** - Treat event calendars and operating windows as ground truth - Assume “open” only inside confirmed windows - No guessing on edge cases

**2. For Closure-Sensitive Venues (Restaurants/Bars at Edge Hours):** - **Holiday windows or late-night edge cases:** Trigger single validation path if closure risk is non-trivial - **Alternative:** Demote venue ranking rather than present as “open” with unknown status - **Cache:** Single validation call per high-risk venue per 24h (metadata only, per ToS)

**3. For Low-Impact Venues (Daytime, Well-Known Hours):** - Allow feedback-first path - Label as “hours estimated” with visible badge - Driver can expand for details

### Closure Risk Calculation

closure\_risk = f(category, daypart, holiday\_proximity, historic\_feedback)

**Thresholds:** - closure\_risk > 0.3 → Trigger validation or substitute venue - closure\_risk < 0.1 → Use estimated hours with badge - 0.1 ≤ closure\_risk ≤ 0.3 → Show with warning badge

### Outcome Tracking (ML Pipeline)

Every venue recommendation logs: - open\_confirmed - Validated open via API - closed\_confirmed - Validated closed via API - estimated\_open - Inferred from patterns (with badge) - unknown\_substituted - High-risk venue replaced with known alternative

### Cost Posture

**We prefer correctness when it directly impacts earnings.** Costs are constrained by: - Single validation call per venue per 24h (cached) - Gating on closure risk threshold (not validating everything) - Substitution with equal/higher earnings alternatives when validation would exceed budget

**~~Old Approach (Deprecated):~~** - ~~Option 3 (Minimal + feedback) as MVP default - “cheap-first”~~ - ~~Zero validation, rely entirely on crowd feedback~~

**New Approach (Accuracy-First):** - Risk-gated validation for closure-sensitive cases - Transparent labeling when using estimates - Substitution over unknown status presentation

### System Impact

* **24h Cache**: Prevents quota exhaustion while maintaining accuracy
* **Risk Threshold**: Gating at >0.3 balances cost vs. correctness
* **Substitution Logic**: Equal/higher earnings alternatives prevent unknown status display

### ML Impact

* **Outcome Logging**: open\_confirmed/closed\_confirmed/estimated\_open/unknown\_substituted tracked
* **Risk Model Training**: Closure risk predictions refined from actual outcomes
* **Validation ROI**: Cost/accuracy tradeoffs measured for threshold optimization

## 📏 **4. DISTANCE & ETA (Traffic-Aware)**

### Core Principle

Provide accurate, traffic-aware distance and ETA calculations using live road conditions, not straight-line estimates.

### How It Works

**Primary Method - Google Routes API:**

{  
 origin: { lat, lng },  
 destination: { lat, lng },  
 travelMode: 'DRIVE',  
 routingPreference: 'TRAFFIC\_AWARE',  
 departureTime: now + 30s // Routes API requires future timestamp  
}

**Returns:** - distanceMeters: Actual road distance - durationSeconds: ETA without traffic - durationInTrafficSeconds: ETA with current traffic

**Fallback - Haversine Formula:**

distance = 2 \* R \* asin(sqrt(sin²(Δlat/2) + cos(lat1) \* cos(lat2) \* sin²(Δlng/2)))

* Used only when Google Routes API fails
* Provides straight-line distance estimate
* Flagged as distanceSource: "fallback" for transparency

### Why This Approach

**Accuracy**: Live traffic data ensures realistic ETAs  
**Reliability**: Fallback ensures distance info always available  
**Cost-Aware**: $10 per 1,000 requests balanced against accuracy needs

### When It Runs

* **Always**: For top-ranked venues after initial scoring
* **Real-Time**: Recalculated on demand for navigation requests

### System Impact

* **API Cost**: $10/1k requests monitored; cached where appropriate
* **Fallback Transparency**: distanceSource flag prevents hidden degradation
* **Traffic Window**: 30s future timestamp required by Routes API

### ML Impact

* **Actual vs Estimated**: Logs predicted ETA vs actual drive time for calibration
* **Traffic Patterns**: Time-of-day/day-of-week correlations for better defaults
* **Fallback Analysis**: Measures accuracy loss when API unavailable

## 🔥 **5. SURGE DETECTION (Opportunity Capture)**

### Core Principle

Detect and factor surge pricing into earnings calculations, flagging high-multiplier opportunities as high priority.

### How It Works

**Detection Methods:** 1. **Uber API Integration**: uberApi.getSurgePricing(lat, lng) 2. **Surge Threshold**: Filter for surge\_multiplier > 1.5x 3. **High Priority Flag**: Surge ≥ 2.0x triggers urgent recommendation

**Integration into Scoring:**

earnings\_boost = base\_fare \* surge\_multiplier  
priority\_level = surge >= 2.0 ? 'high' : 'normal'

**Data Storage:** - blocks table includes surge field (decimal) - Historical surge patterns stored in venue\_metrics - Logged for ML training to predict future surge windows

### Why This Approach

**Opportunistic**: Helps drivers capitalize on high-demand windows  
**Dynamic**: Real-time API calls ensure current surge data  
**Predictive**: ML learns surge patterns for proactive recommendations

### When It Runs

* **Always**: For venues in high-demand categories (airports, events, stadiums)
* **Frequency**: Checked on every block refresh (respects API rate limits)

### System Impact

* **Rate Limit Management**: Prevents quota exhaustion while maintaining freshness
* **Priority Escalation**: Surge ≥2.0x triggers high-urgency visual indicators
* **Historical Storage**: venue\_metrics accumulates surge patterns

### ML Impact

* **Surge Pattern Learning**: Time/day/venue correlations for predictive windows
* **Window Prediction**: Identifies recurring surge windows (e.g., 5-7pm Fridays)
* **ROI Tracking**: Surge-flagged venue acceptance rates measure feature value

## 💰 **6. EARNINGS PROJECTION (Income Accuracy)**

### Core Principle

Estimate realistic earnings per ride based on venue type, surge conditions, and base fare structure.

### How It Works

**Calculation Method:**

estimated\_fare = base\_earnings\_hr \* adjustment\_factor

**Adjustment Factors:** - **Open Venues**: 0.9x multiplier (high confidence) - **Closed Venues**: 0.7x multiplier (lower confidence, staged for opening) - **Event Venues**: Event-specific multiplier based on intensity - **Surge Active**: Base + surge premium

**Data Sources:** - Historical earnings data from venue\_metrics table - Live surge pricing from Uber/Lyft APIs - Time-of-day demand patterns (morning rush, evening rush, late night)

**Net Take-Home:**

net\_take\_home = estimated\_fare - platform\_fees - operating\_costs

### Why This Approach

**Realistic**: Based on historical performance, not optimistic projections  
**Context-Aware**: Adjusts for current conditions (time, surge, events)  
**Transparent**: Shows breakdown so drivers understand the calculation

### When It Runs

* **Always**: For every recommended venue
* **Updates**: When surge levels change or business hours shift

### System Impact

* **Historical Grounding**: venue\_metrics prevents over-optimistic projections
* **Multi-Factor Adjustment**: Open status, surge, events all contribute
* **Transparency**: Breakdown shown to driver builds trust

### ML Impact

* **Projected vs Actual**: Logs estimated earnings vs actual ride value
* **Calibration Model**: Trains adjustment factors from outcome data
* **Venue-Specific Learning**: Refines base\_earnings\_hr per venue over time

## ⭐ **7. PRIORITY FLAGGING (Urgency Intelligence)**

### Core Principle

Flag venues as high, normal, or low priority based on urgency indicators (surge, events, time-sensitivity).

### How It Works

**Priority Determination:**

if (surge >= 2.0 || earnings\_hr >= 60) return 'high';  
if (isEvent && eventStartingSoon) return 'high';  
if (openState === 'closed' && driveTime > 30) return 'low';  
return 'normal';

**High Priority Indicators:** - Surge multiplier ≥ 2.0x - Earnings per hour ≥ $60 - Active events starting within 1 hour - Peak demand windows (morning/evening rush)

**Low Priority Indicators:** - Venue closed with drive time > 30 minutes - Historical low success rate - Driver has hidden this venue previously

**Display Impact:** - High priority: Top of list, urgent badge, highlighted - Normal: Standard display order - Low: Demoted or hidden based on settings

### Why This Approach

**Actionable**: Helps drivers identify time-sensitive opportunities  
**Personalized**: Learns from driver’s feedback and patterns  
**Clear**: Visual indicators make priority instantly obvious

### When It Runs

* **Always**: During venue ranking process
* **Updates**: When surge levels or event status changes

### System Impact

* **Visual Hierarchy**: UI adapts based on priority (badges, position, color)
* **Demotion Logic**: Low-priority venues can be auto-hidden
* **Dynamic Updates**: Priority recalculated on surge/event changes

### ML Impact

* **Urgency Calibration**: Priority vs driver response time measured
* **False Alarm Rate**: Tracks high-priority venues ignored by drivers
* **Threshold Optimization**: Surge/earnings thresholds refined from outcomes

## 📊 **8. BLOCK RANKING (Value Optimization)**

### Core Principle

Present venues in order of expected value to driver, using deterministic scoring that can be audited and A/B tested.

### How It Works

**Final Ranking Process:** 1. **Score Calculation**: Apply scoring formula to all candidates 2. **Diversity Check**: Ensure category mix (no more than 2 from same category in top 5) 3. **Drive Time Enrichment**: Add traffic-aware ETAs 4. **Priority Flagging**: Mark high-urgency venues 5. **Final Sort**: By score (descending), then drive time (ascending)

**User-Facing Order:**

Top 6 = [  
 Highest scoring nearby venue (proximity winner),  
 Best earnings/mile (efficiency winner),  
 Event/surge opportunity (urgency winner),  
 Category-diverse options (2-3 different types),  
 Exploratory option (20% chance, for discovery)  
]

**ML Instrumentation:** - Every ranking logged with ranking\_id - User actions tracked (view, click, hide, like) - Counterfactual learning: “What if we ranked differently?”

### Why This Approach

**Transparent**: Scoring is deterministic and explainable  
**Adaptive**: Learns from user feedback to improve rankings  
**Fair**: No LLM bias; venues ranked by objective metrics

### When It Runs

* **Always**: Final step before presenting blocks to driver
* **Logged**: Every ranking for continuous learning

### System Impact

* **Deterministic Scoring**: Enables A/B testing of ranking algorithms
* **Diversity Enforcement**: Prevents category echo chambers
* **Exploration Budget**: 20% ensures discovery of new venues

### ML Impact

* **Ranking\_ID Logging**: Every ranking tagged for counterfactual analysis
* **Counterfactual Learning**: “What if venue X ranked #1?” analysis
* **User Actions**: Click/view/hide patterns refine future rankings

## 🅿️ **9. STAGING INTELLIGENCE (Waiting Strategy)**

### Core Principle

Recommend specific waiting locations near venues with premium pickup zones, free parking, or optimal positioning.

### How It Works

**Data Sources:** 1. **AI-Suggested**: Claude/GPT identifies staging areas from venue context 2. **Driver Feedback**: Historical staging preferences stored per venue 3. **Venue Metadata**: staging\_notes field includes type, location, walk time

**Staging Area Types:** - **Premium**: Rideshare-specific lots (airports, malls) - **Standard**: Public parking near pickup zones - **Free Lot**: No-cost staging with short walk - **Street**: Curb staging for quick pickups

**Personalization Boost:**

if (driver.preferredStagingTypes.includes(venue.staging\_notes.type)) {  
 score += 0.1; // Boost for preferred staging type  
}

**Display Information:** - Type: Premium / Standard / Free Lot - Name: Specific staging area name - Address: Exact staging location - Walk Time: “2 min walk to pickup zone” - Parking Tip: “Free lot, no time limit”

### Why This Approach

**Practical**: Helps drivers avoid tickets and find optimal waiting spots  
**Local Knowledge**: Captures venue-specific staging intel  
**Personalized**: Learns driver’s staging preferences (covered vs. open, paid vs. free)

### When It Runs

* **Enrichment**: Added to top-ranked venues during final presentation
* **Source**: From AI strategic analysis or driver feedback database

### System Impact

* **Personalization Boost**: +0.1 score for preferred staging types
* **AI + Crowd-Sourced**: Combines GPT analysis with driver knowledge
* **Venue Metadata**: staging\_notes field stores structured staging data

### ML Impact

* **Staging Preference Learning**: Driver’s staging type choices tracked
* **Success Correlation**: Staging quality vs ride acceptance rate
* **Local Knowledge Capture**: Driver feedback enriches staging database

## 💡 **10. PRO TIPS (Tactical Guidance)**

### Core Principle

Provide concise, actionable tactical advice tailored to specific venue and time context.

### How It Works

**Generation Sources:** 1. **GPT-5 Planner**: Generates 1-4 pro tips per venue during tactical planning 2. **Claude Strategist**: Provides strategic-level tips in overview 3. **Historical Data**: Tips derived from successful driver patterns

**Tip Categories:** - **Timing**: “Target international flights 7-9pm for longer fares” - **Staging**: “Park in Lot C for fastest pickup access” - **Events**: “Concert ends at 10:30pm, stage 15 min early” - **Navigation**: “Avoid I-35 construction, use service road” - **Surge**: “Surge peaks 30 min after event end”

**Validation:**

// Pro tips schema (from GPT-5)  
pro\_tips: z.array(z.string().max(250)).min(1).max(4)

**Character Limits:** - Maximum 250 characters per tip - 1-4 tips per venue - Concise, non-hedged language

### Why This Approach

**Actionable**: Tips provide specific tactical advice, not generic statements  
**Context-Aware**: Generated based on current time, weather, events  
**Validated**: Schema ensures tips meet quality standards

### When It Runs

* **Always**: Generated by GPT-5 during tactical planning stage
* **Per Venue**: Each recommended venue receives custom tips

### System Impact

* **Character Limits**: Ensures mobile-friendly display (250 max)
* **Zod Validation**: Quality enforcement at schema level
* **Multi-Source**: GPT-5 + Claude + historical for comprehensive advice

### ML Impact

* **Tip Effectiveness**: Correlation between tips and venue success
* **Topic Analysis**: Which tip categories (timing/staging/events) drive action
* **Quality Metrics**: Tip length, count, category distribution tracked

## 👍 **11. GESTURE FEEDBACK (Learning Loop)**

### Core Principle

Learn from driver interactions (like, hide, thumbs up/down) to personalize future recommendations and suppress unhelpful venues.

### How It Works

**Feedback Actions:** - **Like** ➜ Boost this venue in future rankings (+0.3 score) - **Hide** ➜ Add to driver’s no-go zones, suppress from future results - **Helpful** ➜ Increase venue reliability score - **Not Helpful** ➜ Decrease venue reliability score, consider suppression

**Data Logging:**

// actions table  
{  
 action\_id, snapshot\_id, ranking\_id, user\_id,  
 venue\_id, action\_type, timestamp  
}

**ML Learning:** - Positive feedback → positive\_feedback++ in venue\_metrics - Negative feedback → negative\_feedback++, adjust reliability\_score - Suppression threshold: If 3+ hides, add to driver.noGoZones[]

**Personalization Impact:**

if (driver.successfulVenues.includes(venue\_id)) {  
 personalBoost += 0.3; // Prioritize venues driver liked before  
}  
if (driver.noGoZones.includes(venue\_id)) {  
 return null; // Completely hide from recommendations  
}

### Why This Approach

**Adaptive**: System learns what works for each individual driver  
**Respectful**: Hidden venues stay hidden (unless driver manually unhides)  
**Counterfactual**: Tracks “what we recommended vs what they chose” for ML

### When It Runs

* **Action Logging**: Immediately when driver taps like/hide/helpful
* **Ranking Impact**: Applied during next block recommendation cycle
* **ML Training**: Batch processed for pattern learning

### System Impact

* **Personalization Boost**: +0.3 for liked venues, null for hidden
* **No-Go Zones**: 3+ hides triggers permanent suppression
* **Immediate Feedback**: Actions logged synchronously, applied asynchronously

### ML Impact

* **Counterfactual Analysis**: “Recommended vs chosen” for ranking optimization
* **Venue Reliability**: positive/negative feedback updates reliability\_score
* **Pattern Recognition**: Identifies venue types/times driver prefers/avoids

## 🧭 **12. NAVIGATION LAUNCH (Seamless Routing)**

### Core Principle

Provide seamless navigation integration with Google Maps and Apple Maps, using traffic-aware routing and native app deep-linking.

### How It Works

**Platform Detection:**

// iOS  
if (iOS && AppleMapsInstalled) {  
 url = `maps://?daddr=${lat},${lng}`;  
} else {  
 url = `https://maps.apple.com/?daddr=${lat},${lng}`;  
}  
  
// Android  
if (Android && GoogleMapsInstalled) {  
 url = `google.navigation:q=${lat},${lng}`;  
} else {  
 url = `https://www.google.com/maps/dir/?api=1&destination=${lat},${lng}`;  
}

**Traffic Integration:** - Google Maps: Traffic layer included by default - Routes API: Provides durationInTrafficSeconds for ETA - Real-time updates: Recalculated on demand when driver taps navigate

**Airport Context:** - If near airport, includes terminal info: “Terminal C pickup zone” - FAA delay data: “DFW: 45 min departure delays, target arrivals” - Alerts displayed before navigation starts

### Why This Approach

**Seamless**: Deep-links to native apps for best UX  
**Accurate**: Traffic-aware routing prevents underestimated ETAs  
**Context-Rich**: Airport alerts help drivers avoid wasted trips

### When It Runs

* **On Demand**: When driver taps “Navigate” button
* **ETA Updates**: Real-time when driver views venue details
* **Fallback**: Always provides web-based maps if native apps unavailable

### System Impact

* **Deep-Link Priority**: Native apps preferred (iOS: maps://, Android: google.navigation:)
* **Web Fallback**: Ensures navigation always available
* **ETA Recalculation**: Routes API called on navigation launch for fresh ETA

### ML Impact

* **Navigation Action**: Logged as recommendation acceptance signal
* **ETA Accuracy**: Actual arrival time vs projected ETA measured
* **Platform Usage**: iOS vs Android navigation method effectiveness

## 🔒 **ARCHITECTURAL CONSTRAINTS (DO NOT VIOLATE)**

### 1. Zero Hardcoding Policy

**Rule:** No hardcoded locations, models, or business logic  
**Enforcement:** All data must reconcile to database or environment variables

**Examples:** - ✅ process.env.CLAUDE\_MODEL (from .env) - ✅ SELECT \* FROM venues WHERE h3\_r8 = ? (from database) - ❌ const topVenues = ["Stonebriar Centre", "Star District"] (hardcoded)

**Why:** Enables dynamic updates without code changes, critical for ML-driven optimization

### 2. Never Suppress Errors

**Rule:** Always find and fix root causes, never suppress errors  
**Examples:** - ✅ If model fails, surface the error with full context - ✅ If API returns wrong model, throw assertion error - ❌ try { await llm() } catch { return fallback } (hiding failures)

**Why:** Error suppression corrupts ML training data and hides systemic issues

### 3. Single-Path Triad (No Fallbacks)

**Rule:** Triad pipeline must complete all 3 stages or fail entirely  
**Enforcement:** Each stage checks previous stage output before proceeding

**Exception:** Agent Override (Atlas) uses fallback chain for operational resilience (workspace ops different from user-facing strategy)

### 4. Complete Snapshots Only

**Rule:** Never send partial context to LLMs  
**Validation:** Snapshot must include: location, weather, AQI, airport, time context, H3 geospatial

**Why:** Partial data corrupts ML training (can’t learn patterns from incomplete inputs)

### 5. Model ID Stability

**Rule:** Pin exact model IDs, verify monthly, fail hard on missing models  
**Implementation:** - CLAUDE\_MODEL=claude-sonnet-4-5-20250929 (not just “claude-sonnet”) - OPENAI\_MODEL=gpt-5-pro (not “gpt-5”) - Monthly verification via tools/research/model-discovery.mjs

**Why:** Model names can be deprecated or replaced (e.g., gpt-4o → gpt-5)

### 6. Partner Platform Namespace Separation

**Rule:** Never use partner-specific model IDs with native APIs

**Anthropic Claude:** - ✅ Native API: claude-sonnet-4-5-20250929 - ❌ Vertex AI: claude-sonnet-4-5@20250929 (different format) - ❌ AWS Bedrock: anthropic.claude-sonnet-4-5-20250929-v1:0 (global prefix)

**Why:** Different platforms have different namespaces, mixing them causes 404 errors

### 7. Database Schema Immutability

**Rule:** NEVER change primary key ID column types (breaks existing data)

**Safe Patterns:**

// If already serial, keep serial  
id: serial("id").primaryKey()  
  
// If already varchar UUID, keep varchar UUID  
id: varchar("id").primaryKey().default(sql`gen\_random\_uuid()`)

**Migration:** Use npm run db:push --force (NOT manual SQL)

**Why:** Changing ID types (serial ↔ varchar) generates destructive ALTER TABLE statements

## 📊 **ML INSTRUMENTATION & TRAINING DATA**

### Counterfactual Learning Pipeline

**Goal:** Build dataset to fine-tune models on what drivers ACTUALLY chose vs. what we recommended

**Data Captured:** 1. **Snapshot** - Complete context (location, weather, time, etc.) 2. **Triad Output** - All 6 venue recommendations with scores 3. **User Action** - Which venue they chose (or ignored) 4. **Outcome** - Actual earnings vs. projected

**Storage Tables:** - ml\_snapshots - Context at time of recommendation - ml\_recommendations - What we suggested - ml\_outcomes - What actually happened

**Constraint:** Never log partial data (corrupts training set)

## 🔐 **SECURITY & SAFETY**

### Rate Limiting (DDoS Protection)

* **API Routes:** 100 requests / 15 minutes per IP
* **Health Checks:** Unlimited (excluded from limits)
* **Strategy Generation:** 10 requests / 15 minutes per IP (strict)

### Secret Management

* **Storage:** Replit Secrets (never committed to repo)
* **Access:** Environment variables only
* **Validation:** check\_secrets tool before usage

**Available Secrets:** - ANTHROPIC\_API\_KEY (Claude) - OPENAI\_API\_KEY (GPT-5) - GEMINI\_API\_KEY (Gemini) - GOOGLEAQ\_API\_KEY (Air Quality) - FAA\_ASWS\_CLIENT\_ID / FAA\_ASWS\_CLIENT\_SECRET (Airport delays) - PERPLEXITY\_API\_KEY (Model research)

### Command Whitelisting (Agent Server)

**Allowed:** ls, cat, grep, find, git status  
**Blocked:** rm -rf, sudo, chmod 777, destructive operations

## 🚀 **DEPLOYMENT CONFIGURATION**

### Production Settings

NODE\_ENV=production  
PORT=5000  
  
# Model Configuration (verified October 8, 2025)  
CLAUDE\_MODEL=claude-sonnet-4-5-20250929  
OPENAI\_MODEL=gpt-5-pro  
GEMINI\_MODEL=gemini-2.5-pro-latest  
ANTHROPIC\_API\_VERSION=2023-06-01  
  
# Triad Architecture  
TRIAD\_ENABLED=true  
TRIAD\_MODE=single\_path  
ROUTER\_V2\_ENABLED=false  
  
# Timeouts (90s total budget)  
CLAUDE\_TIMEOUT\_MS=12000  
GPT5\_TIMEOUT\_MS=45000  
GEMINI\_TIMEOUT\_MS=15000  
  
# GPT-5 Configuration  
GPT5\_REASONING\_EFFORT=high

### Workflow Configuration

**Name:** Eidolon Main  
**Command:** NODE\_ENV=development VITE\_PORT=3003 PLANNER\_DEADLINE\_MS=120000 VALIDATOR\_DEADLINE\_MS=60000 node gateway-server.js  
**Port:** 5000 (Replit firewall requirement)

**Constraint:** Must serve on port 5000 (other ports are firewalled)

## 📈 **FORWARD PRESSURE (Roadmap)**

### Phase 1: Enhanced Context (Q4 2025)

* ✅ Thread-aware context system (COMPLETE)
* ✅ Model verification automation (COMPLETE)
* 🔄 Real-time event calendar integration (IN PROGRESS)
* 🔄 Traffic pattern ML model (IN PROGRESS)

### Phase 2: Trust-First Refinement (Q1 2026)

* 📋 A/B testing framework for scoring engine
* 📋 Venue catalog auto-refresh (weekly Google Places sync)
* 📋 Counterfactual learning model training
* 📋 Driver personalization engine

### Phase 3: Safety & Compliance (Q2 2026)

* 📋 Fatigue detection (ML-based)
* 📋 Familiar route recommendations
* 📋 Strategic break planning
* 📋 Insurance integration

## ⬅️ **BACKWARD PRESSURE (Deprecated)**

### ~~Router V2 with Fallbacks~~ (Removed Oct 8, 2025)

* ~~Automatic failover between providers~~
* ~~Circuit breakers with 5-failure threshold~~
* ~~8s total budget (too aggressive)~~
* **Reason:** User requires consistent quality, no silent model swaps

### ~~Global JSON Body Parsing~~ (Removed Oct 7, 2025)

* ~~app.use(express.json()) on all routes~~
* **Reason:** Caused “request aborted” errors on client cancellation

### ~~React.StrictMode~~ (Removed Oct 7, 2025)

* ~~Double-rendering for development warnings~~
* **Reason:** Caused duplicate API calls and abort errors

### ~~Deprecated Models~~ (Replaced Oct 8, 2025)

* ~~gpt-4o → gpt-5-pro~~
* ~~gemini-1.5-pro → gemini-2.5-pro-latest~~
* ~~claude-3-5-sonnet → claude-sonnet-4-5-20250929~~

## 🧪 **TESTING & VERIFICATION**

### Model Verification (Monthly)

# Automated research via Perplexity  
node tools/research/model-discovery.mjs  
  
# Direct API verification  
curl https://api.anthropic.com/v1/models/claude-sonnet-4-5-20250929 \  
 -H "x-api-key: $ANTHROPIC\_API\_KEY" \  
 -H "anthropic-version: 2023-06-01"

### Triad Pipeline Test

# Standalone test  
node scripts/test-triad.mjs  
  
# Production endpoint  
curl -X POST http://localhost:5000/api/blocks \  
 -H "Content-Type: application/json" \  
 -d '{"lat":33.1287,"lng":-96.8757}'

## 📚 **KEY DOCUMENTATION REFERENCES**

| Document | Purpose | Last Updated |
| --- | --- | --- |
| MODEL.md | AI model specifications with API details | Oct 8, 2025 |
| replit.md | User preferences and system overview | Oct 8, 2025 |
| docs/reference/V2-ROUTER-INTEGRATION.md | Router V2 history and resolution | Oct 8, 2025 |
| tools/research/THREAD\_AWARENESS\_README.md | Thread system documentation | Oct 8, 2025 |
| ARCHITECTURE.md | This document - constraints & decisions | Oct 8, 2025 |

## 📝 **FIX CAPSULE (Agent-Authored, append one per fix)**

### Template: Use for Every Future Fix

**Impact**  
Describe the user-visible change (driver trust, earnings accuracy, latency, cost).

**When**  
Describe exactly when the logic runs (route name, stage in triad, gating).

**Why**  
Name the invariant(s) this change enforces and the root cause it removes.

**How**  
Summarize the code change at a systems level and any data model or cache impact.

**Files Touched**  
List paths and a one-line intent for each.

**Tests and Acceptance**  
List the command(s) to reproduce the prior bug and the command(s) to observe the fix.

**Observability**  
Name the log lines, counters, or traces that prove the fix works in prod.

**Back/Forward Pressure**  
Note what we deprecate and what we make easier going forward.

### Fix Capsule — Key-Based Merge, DB-First Coords/Hours (Oct 8, 2025)

**Impact**  
Eliminates $0 earnings and incorrect miles. Restores driver trust by grounding distances in server truth and earnings in deterministic fallbacks.

**When**  
Runs during /api/blocks validation/merge phase after planner output and before final ranking. Applies to every venue in the response.

**Why**  
Enforces new invariants 6 and 7 (Google/DB as truth for coords/hours; merge by key never index). Removes root causes: index-based merge and client GPS overwrite.

**How**  
Server merges by place\_id/name with numeric coercion and safe fallbacks; validator must echo placeId; client uses server venue coords. Adds places cache and DB-first reads for coords/hours.

**Files Touched**  
- server/routes/blocks.js - Key-based merge with numeric coercion and fallbacks - server/lib/gemini-enricher.js - Validator prompt requires placeId echo - client/src/pages/co-pilot.tsx - Use server venue coordinates, not device GPS - shared/schema.js - places\_cache table (already exists, TTL policies enforced)

**Tests and Acceptance**  
1. POST /api/blocks with known venues → no $0 earnings; epm computed when validator fields absent 2. Inspect UI → venue coords equal server payload; distance matches server 3. Repeat same venues within TTL → no external Places call; DB-first hit logged

**Observability**  
Logs include calculated\_distance\_miles, estimated\_earnings, merged key, and distanceSource. Places cache writes emit place\_id with timestamps.

**Back/Forward Pressure**  
**Backward:** Index-merge and UI GPS overwrite removed.  
**Forward:** Weekly Places sync; validator prompts always echo placeId; earnings never default to $0.

## 🎯 **DECISION LOG**

### October 8, 2025

* ✅ **Verified:** Claude Sonnet 4.5 model works correctly (no silent swaps)
* ✅ **Added:** Model assertion in adapter to prevent future mismatches
* ✅ **Implemented:** Thread-aware context system for Agent/Assistant/Eidolon
* ✅ **Updated:** All documentation to reflect verified model state
* ✅ **Set:** ANTHROPIC\_API\_VERSION=2023-06-01 in environment

### October 7, 2025

* ✅ **Removed:** React.StrictMode (double-rendering causing abort errors)
* ✅ **Removed:** Global JSON body parsing (causing abort on client cancellation)
* ✅ **Added:** Per-route JSON parsing with 1MB limit
* ✅ **Added:** Client abort error gate (499 status)
* ✅ **Added:** Health check logging filter

### October 3, 2025

* ✅ **Implemented:** Router V2 with proper cancellation
* ✅ **Fixed:** Circuit breaker poisoning from aborted requests
* ✅ **Increased:** Budget from 8s to 90s (production needs)
* ⚠️ **Discovered:** Anthropic model 404 issue (resolved Oct 8)

## 🚨 **CRITICAL CONSTRAINTS SUMMARY**

1. **Single-Path Triad** - No fallbacks, fail properly instead of degrading
2. **Zero Hardcoding** - All data from DB or env vars
3. **Never Suppress Errors** - Surface failures with full context
4. **Complete Snapshots Only** - Never send partial data to LLMs
5. **Model ID Stability** - Pin exact IDs, verify monthly
6. **Partner Namespace Separation** - Don’t mix Vertex/Bedrock IDs with native APIs
7. **Database Schema Immutability** - Never change PK types
8. **Trust-First Stack** - Curated catalog + deterministic scoring (no hallucinations)
9. **Port 5000 Requirement** - Replit firewall constraint
10. **Per-Route JSON Parsing** - No global body parser

**This document is the authoritative reference for all architectural decisions. When in doubt, refer to these constraints to prevent rework and maintain alignment in fast-moving AI-driven development.**