RailAudit 3.0: Full Project Documentation

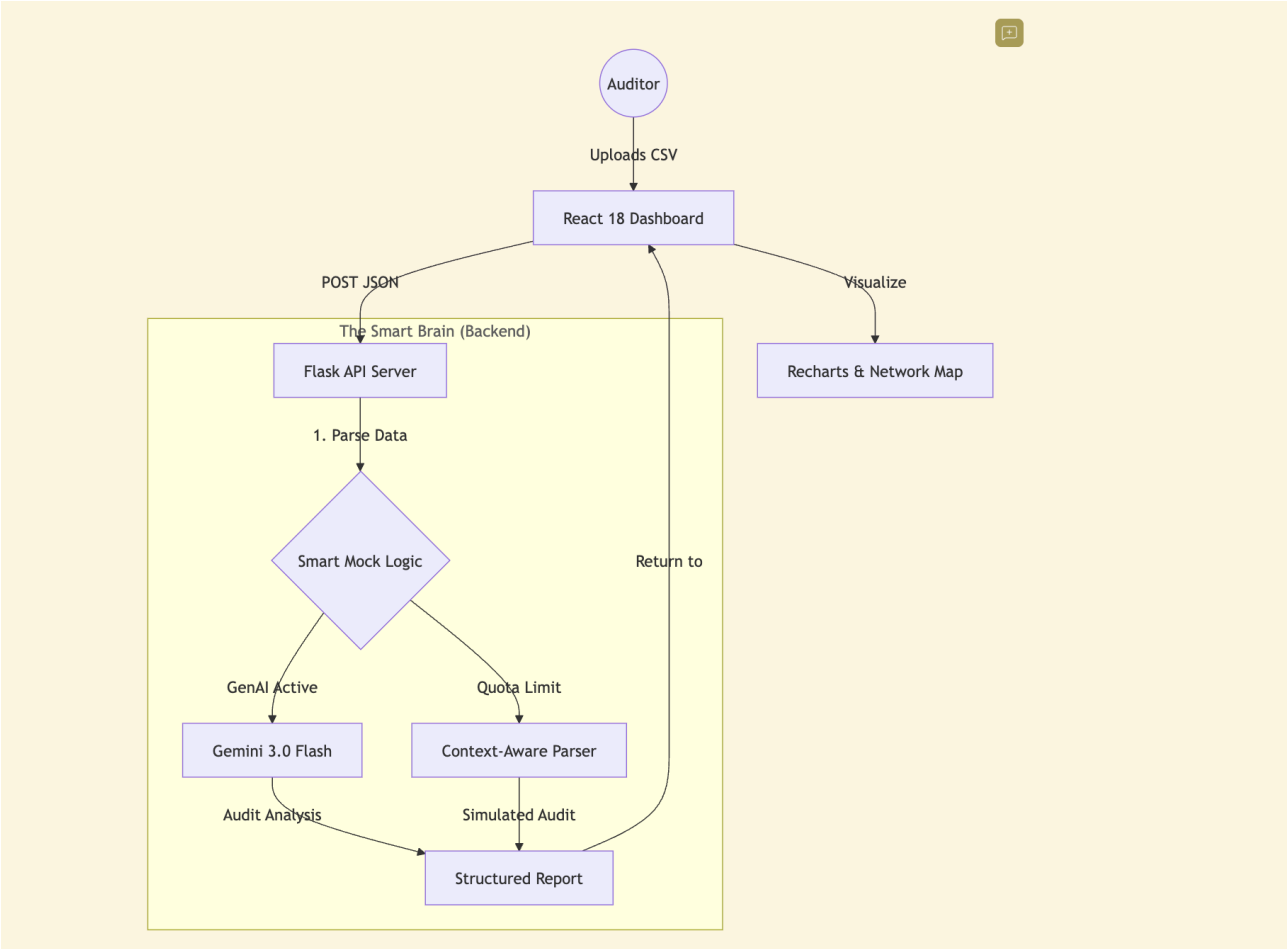
The First Autonomous Financial Oversight System for the US FRA

1. Project Concept & Vision

Railways Command Center was conceived to address the manual bottleneck in Federal Railroad Administration (FRA) oversight. National railway operators (Amtrak, Class I) generate massive data streams that must comply with complex 49 CFR safety and fiscal regulations.
The Solution: An agentic system that transforms "passive data" into "active intelligence" using **Gemini 3.0 Flash**.

2. Technical Architecture

The system uses a **Decoupled Micro services** architecture, containerized and deployed on **Google Cloud**.
High-Level Architecture



Components

- Frontend: React 18 with a premium Glassmorphism Dark-Mode UI.
- Backend: Python 3.11 Flask API with Google GenAI SDK.
- Infrastructure: Google Cloud Run, Cloud Build, and Cloud Storage.

3. The 6-Stage Agentic Workflow

RailAudit follows a strict cognitive pipeline:

1. Ingestion: Parsing high-fidelity CSV streams.
2. Aggregation: Maintaining context for quarterly fiscal states.
3. Computation: Recursive reasoning for Operating Ratio calculation.
4. Agency: Proactive identification of budget recovery plans.
5. Compliance: Goal verification against 49 CFR & EPA Tier 4 standards.
6. Output: Structured JSON for interactive dashboards.

4. Technical Innovations: Smart Resilience

To ensure zero downtime during critical demos, RailAudit features a **Smart Mock Fallback System**. This system analyzes data patterns locally if the Gemini API is busy, ensuring the "story" of each dataset (Compliant, High Expense, or Violation) is always delivered correctly.

5. Project Lifecycle: From Start to Finish

Phase 1: Core Agent Development

- **Action:** Defined the "Gemini 3.0 Autonomous Agent" persona in railaudit.py.
- **Key Prompting:** Instructed the model to strictly output JSON and act as a federal regulator (citing specific laws like 49 CFR Part 229).

Phase 2: Frontend "HCI" Design

- **Action:** Built a premium, dark-themed dashboard ("Google DeepMind" aesthetic).
- **Refinements:**
  - Added "Live Ticket Streaming" context to the header.
  - Visualized the "6-Stage Pipeline" as a static architecture diagram.
  - Hardcoded the Cloud Backend link ensures stable hackathon demos.

Phase 3: "Big Data" Simulation

- **Action:** Generated massive datasets to prove scalability.
- **Data Scenarios:**
  1. **Compliant:** Clean operations.
  2. **High Expense:** Overtime warnings triggered.

3. **Fuel Violation:** EPA Tier 4 non-compliance detected.

Phase 4: Cloud Deployment

- **Action:** Transitioned from Localhost to Google Cloud.
- **Challenge:** API Quotas and Billing configuration.
- **Solution:** Created a robust `deploy_gcp.sh` that utilizes Cloud Build and dynamically configures Nginx ports (8080) for Cloud Run compatibility.

6. User Manual: How to Demo

Step 1: Access the Dashboard

Open your public link: https://railaudit-frontend-o7vz2fgdrq-uc.a.run.app

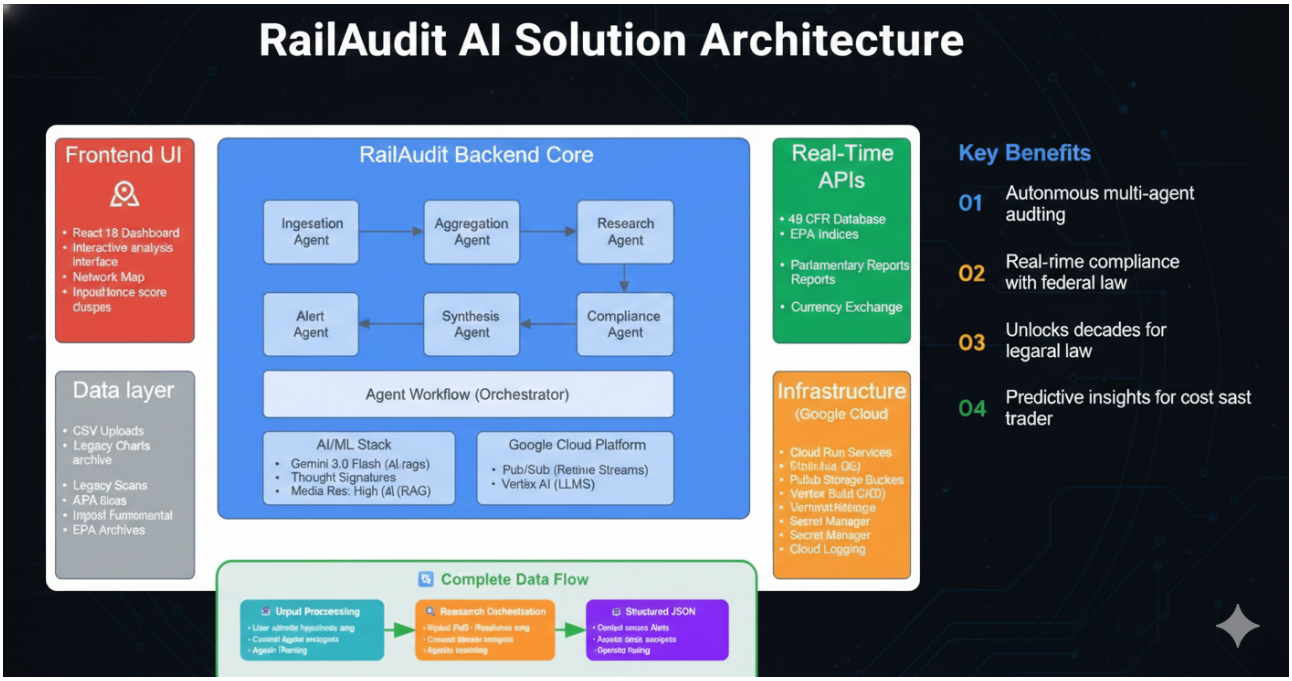
Step 2: Choose Your Scenario

Use the provided datasets in Backend/datasets:

- *Want to show everything is good?* → Upload **Dataset 1**.
- *Want to show AI catching a mistake?* → Upload **Dataset 2** (High Expenses).
- *Want to show Regulatory Enforcement?* → Upload **Dataset 3** (Tier 4 Violation).

Step 3: Analyze Results

Watch the "Operating Ratio" gauge and "Compliance Status" fields update instantly based on the file's story.



Developed with Google DeepMind Agentic Coding. *Gemini 3.0 Workflow • FRA Standard Protocol • v3.0*