# Engineering Assessment — Travel Signals Aggregator

Context
This assessment is designed to evaluate real-world engineering capability beyond framework usage or AI-assisted scaffolding. The goal is to understand how you think about system design, third-party integrations, resilience, and extensibility.

## Problem Statement

Users want a quick situational understanding of a potential trip before committing time or money. Build a system that aggregates multiple travel-related signals and converts them into a structured, explainable summary.

## Core Objective

Given a destination and date range, the system should fetch data from multiple third-party APIs, normalize it, and generate a consolidated travel signal indicating feasibility and risk.

## Required Third-Party Integrations (Minimum 3)

- Weather (OpenWeatherMap, WeatherAPI, etc.)
- Public Holidays (Nager.Date API)
- Currency / Cost Signals (Frankfurter, ExchangeRate API)
- Safety / Travel Advisory (RapidAPI or equivalent)
- Flight Cost Signals (real or mocked)

Mocked APIs are acceptable if they realistically simulate latency, rate limits, and failure scenarios.

## Functional Requirements

- Accept destination, date range, and optional budget as input
- Fetch data asynchronously from multiple APIs
- Normalize responses into a unified internal schema
- Generate a scored and explainable output summary

## Architecture Expectations

- Clear separation of API adapters, aggregation logic, and scoring
- Document trade-offs and future extensibility

## Error Handling & Resilience

- Graceful degradation when APIs fail
- Timeouts, retries, or fallback strategies
- Transparent communication of missing signals

## Performance & Scalability (Conceptual)

- Caching strategies

- Rate-limit handling
- Reducing redundant API calls

## Feature Evolution

Explain how your system could evolve to support preferences, saved queries, historical comparison, or notifications.

## Deliverables

- Working backend code (frontend optional)
- README with setup instructions and API details
- Design notes covering architecture and decisions

## Evaluation Criteria

- Depth of integration and data modeling
- Error handling and resilience
- Architectural clarity and extensibility
- Reasoning and documentation quality

This assessment intentionally avoids product-specific context. The focus is on engineering judgment, not feature completeness.