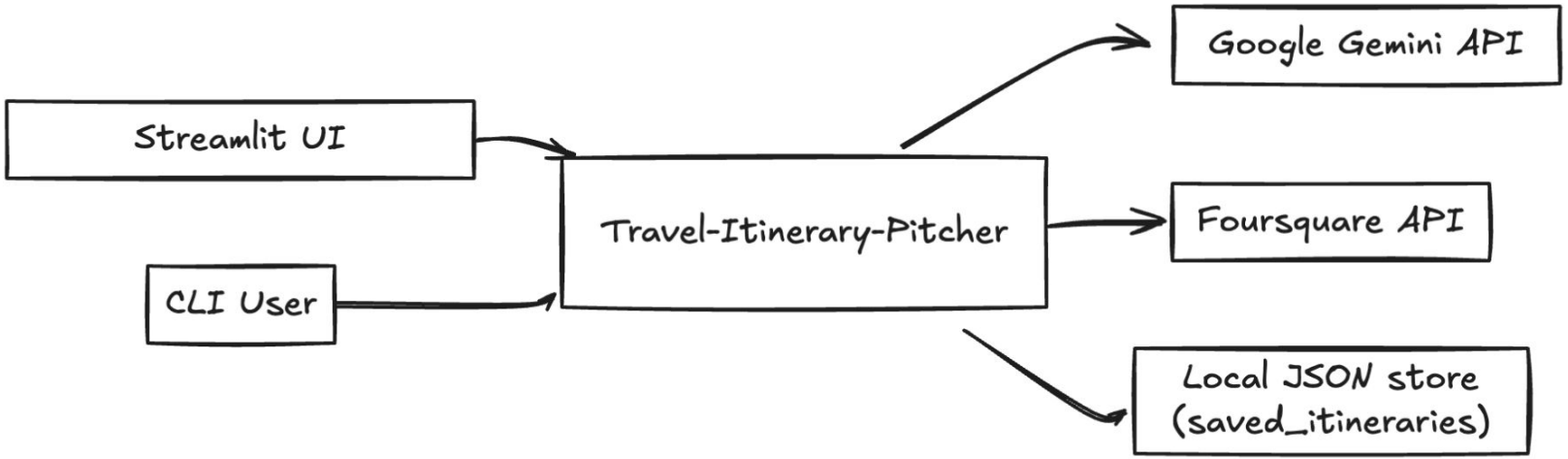
Travel Itinerary Pitcher

Prajna Penmetsa
Aaditya Bhatia

Architecture Overview



Architecture Overview

01

Presentation

Streamlit
Frontend & CLI

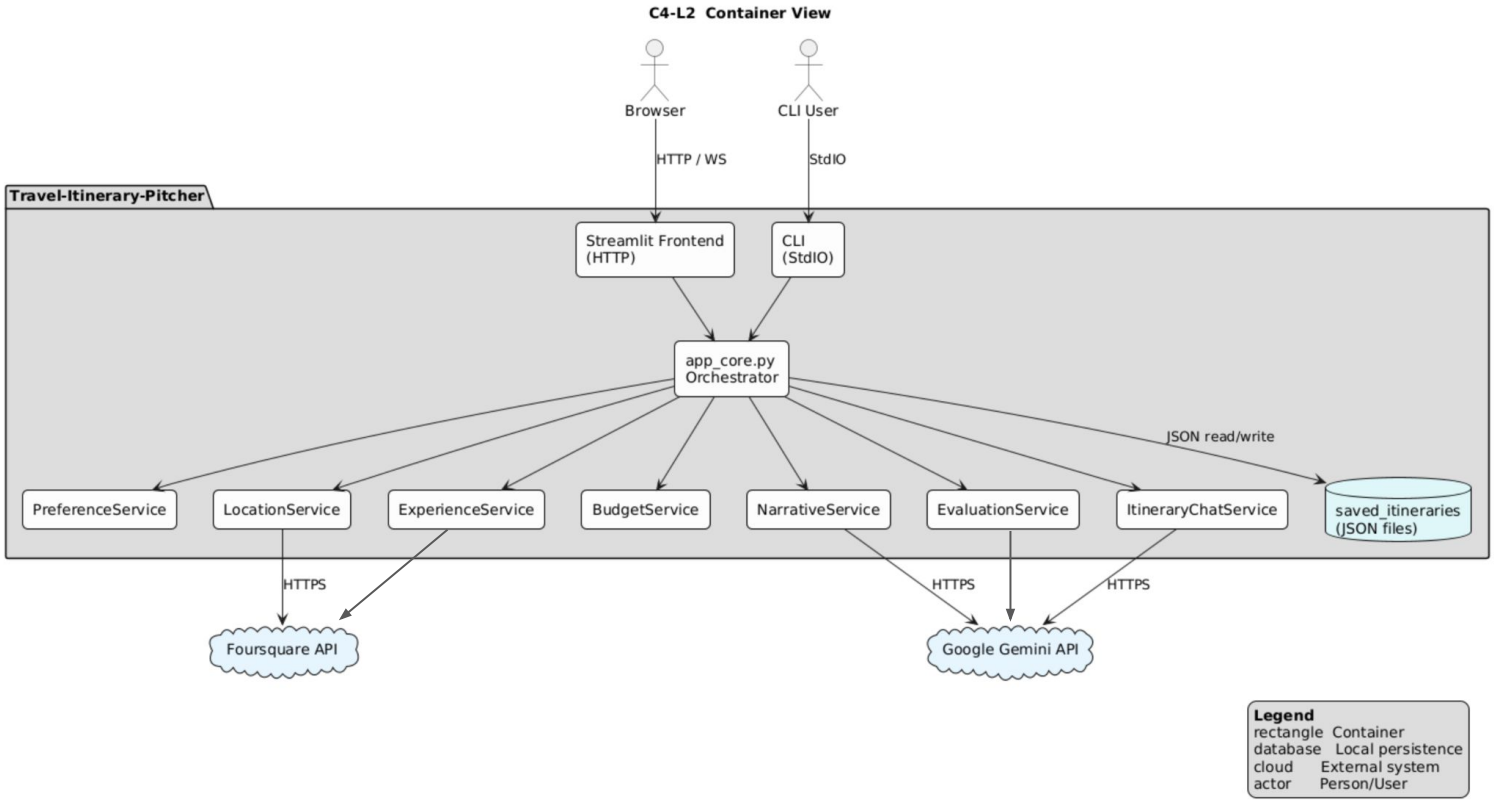
02

Core Orchestration

Coordinates the end-to-end pipeline, wiring each Python microservice together

app_core.generate_itinerary()

Architecture Overview



FourSquare API

- Foursquare Places API is a REST API, which we are using for POI
  - Returns rich metadata about 120 M+ “places” (restaurants, museums, landmarks, parks, etc.) worldwide
  - Enables the Location and Experience services to offer location-specific recommendations instead of AI guesswork

Usage - FourSquare API

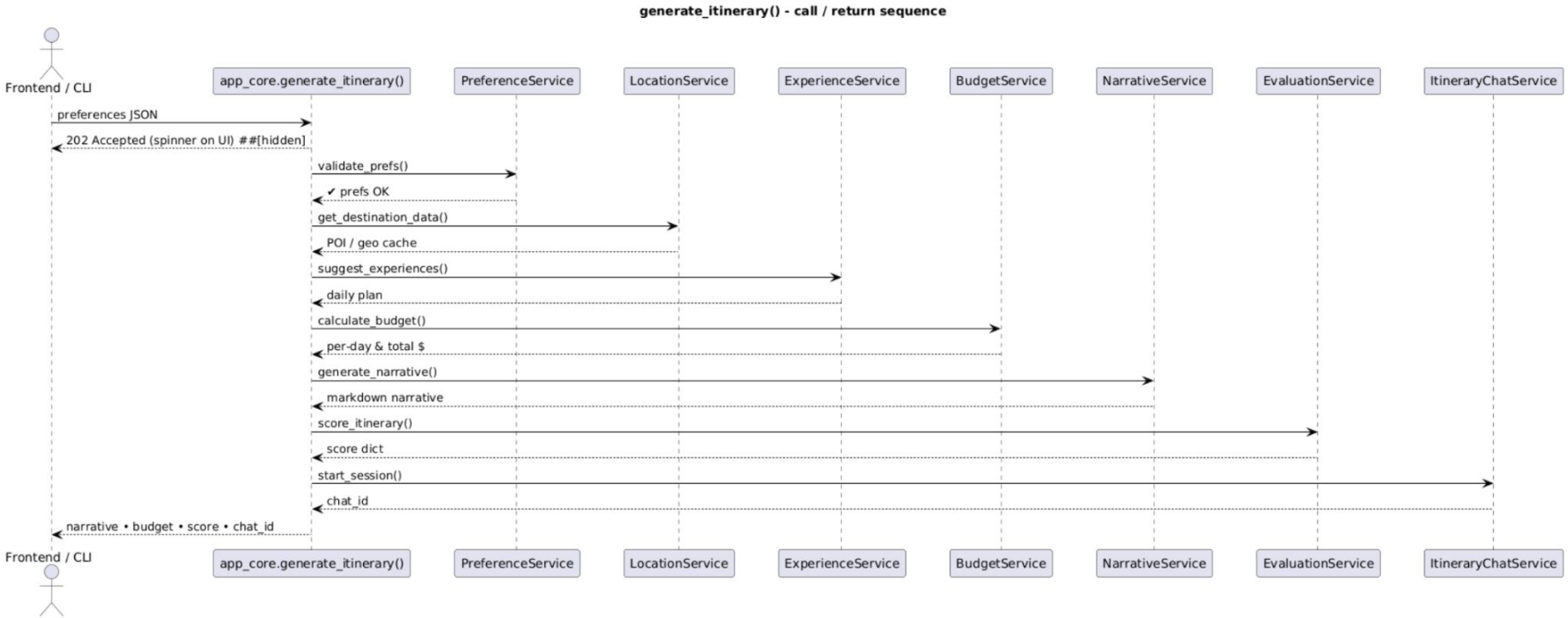
LocationService calls the Search / Details endpoints to:

1. Retrieve a list of candidate venues for the chosen destination
2. Cache the JSON so repeat runs are instant

Curated POIs are passed to ExperienceService

- maps them to user interests and builds a balanced day-by-day plan

Sequence Diagram - Microservices



Google Gemini API

- Cloud-hosted large-language-model (LLM) service from Google
- Accessed google-generativeai Python SDK.
    - You send it a prompt, it sends back intelligently generated text
    - Using for storytelling and conversational assistance

Usage - FourSquare API

NarrativeService turns the raw daily plan + budget into a compelling, human-readable travel story:

1. intro paragraph
2. vivid day descriptions
3. budget paragraph

Through ItineraryChatService, powers an chatbot after the plan is generated

1. Answers follow-up questions (“Can I swap Day 2 evening for a food tour?” “What’s the local tipping culture?”)

Usage - FourSquare API

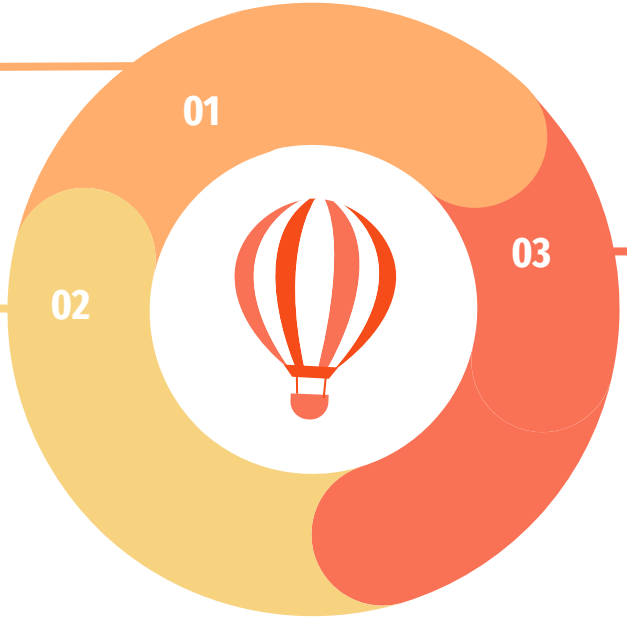
Used in a small part of EvaluationService:

1. Gemini rates the narrative quality sub-score (clarity & engagement, 1-5)

Gen AI Usage

Gemini
Ideation
Prompt: “unique, fun”

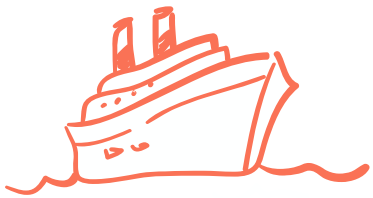
Claude
Initial code generation
Prompt:
1. implement a free micro-services architecture for this idea
2. generate project overview, requirements specification
3. We want to use these APIs



ChatGPT
Code tweaks
Streamlit front end creation, debugging, adding various features, visualizing diagrams

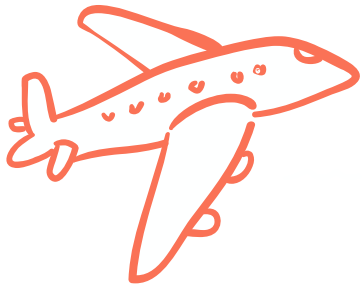
GitHub co-pilot
Code tweaks

what worked



Claude	Very useful for getting all the initial code – gave us a great starting point
APIs	Difficult to configure and find free plans, but they ended up adding to our project in satisfactory way
Streamlit	Seamless, minimal, beautiful frontend creation - Streamlit cloud provided easy deployment

what didn’t



Creating README	LLMs find it difficult to handle markdown so they weren’t able to auto-generate readme
Diagram visualization	either lots of information loss or lots of compiler errors or diagrams are too abstracted