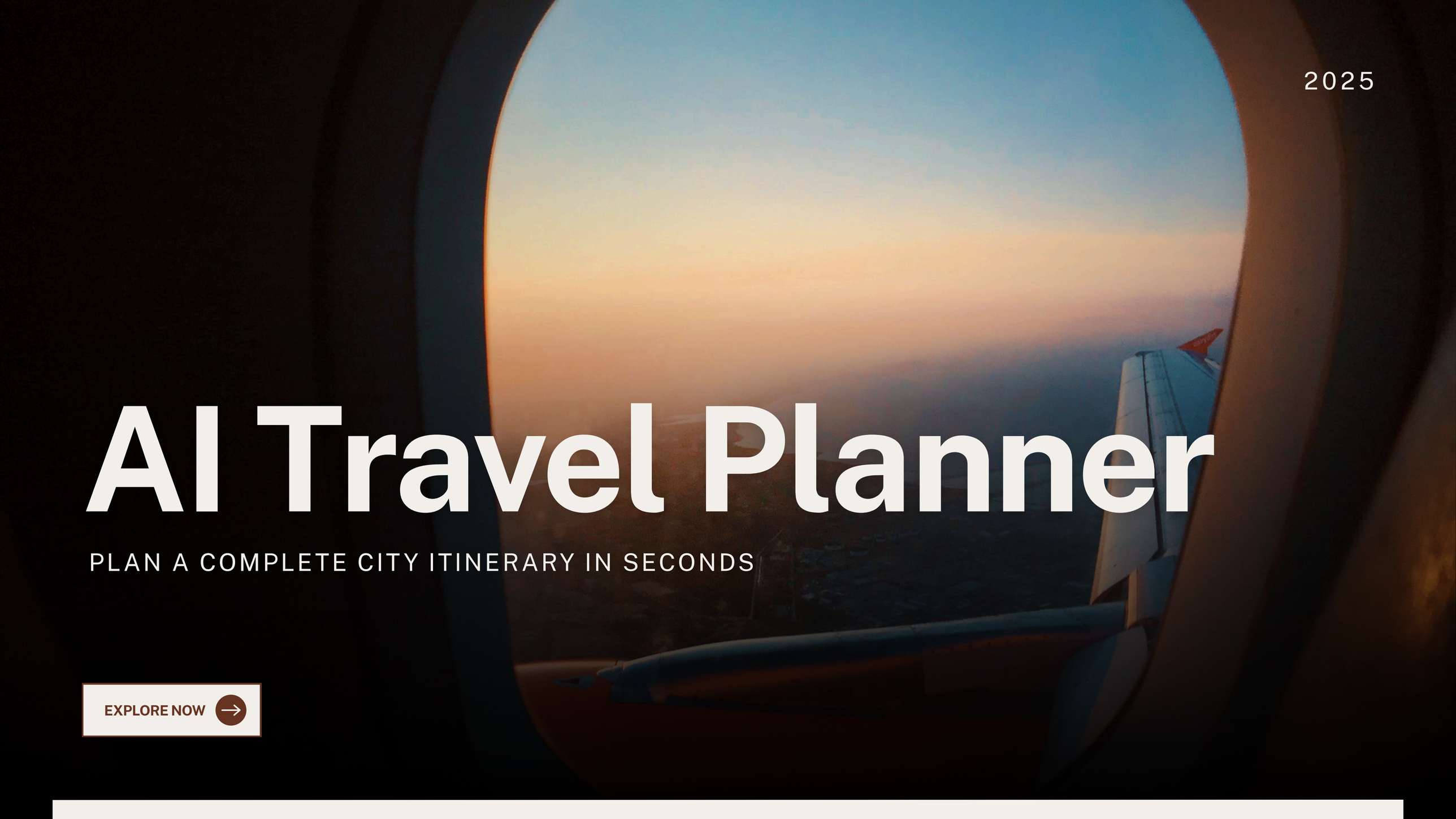

2025

AI Travel Planner

PLAN A COMPLETE CITY ITINERARY IN SECONDS

EXPLORE NOW

PROBLEM

- Planning a short trip is slow: too many blogs, duplicate ideas, inconsistent costs.
- Hard to keep days balanced (cost/time) and family/food constraints.
- Sharing the plan with friends is clunky.

Goal: Get a sensible, structured plan in seconds that’s easy to tweak and export.

SOLUTION

- Input: city, days, budget, preferences & simple filters.
- Output: per-day plan (morning/afternoon/evening) with reasons and estimated costs.
- One click exports: JSON / CSV / PDF.
- Lightweight flags: vegetarian, family friendly, avoid long walks.

DEMO FLOW (UX)

1

Enter destination, days, budget, preferences.

2

Optional filters (vegetarian, family, avoid long walks).

3

Generate itinerary → view table (day/slots/cost).

4

Export: save JSON, download CSV, or export PDF.

ARCHITECTURE

CLIENT (STREAMLIT)

- Collect inputs, call /plan, render DataFrame.
- Recomputes daily totals and handles exports.

API (FASTAPI VIA NGROK)

- Auth header (Bearer <API_KEY>).
- Builds plan with retrieval-augmented generation.
- Applies optional lightweight filters.
- Returns validated JSON.

PLANNER (RAG)

- FAISS index of city attractions (JSON data).
- Retriever (MMR) → diverse, relevant places.
- Mistral-7B Instruct, 4-bit quantized (bitsandbytes) → fits on modest GPU/CPU RAM, ~2-3× smaller with minimal quality loss.
- LLM (Mistral-7B Instruct, 4-bit) → JSON itinerary.

RAG narrows the LLM's choices to known attractions, reducing hallucinations and repetition.

BACKEND: RETRIEVAL → PROMPTING → VALIDATION

RETRIEVAL

- FAISS vector store built from city datasets.
- MMR retriever (k top items) for diversity.

PROMPTING

- System prompt: “Return only one JSON object. Use only retrieved places.”
- User template injects destination, days, budget, preferences + retrieved facts.

VALIDATION

- Pydantic models: Slot, DayPlan, Itinerary.
- If structure is off, the request fails early — UI never sees malformed data.

This loop — retrieve, prompt with constraints, then validate — keeps output grounded and structured.

API /PLAN DETAILS

REQUEST

- city, days, budget, preferences, filters?, constrain_to_budget?

RESPONSE

- {"days":[{day_number, city, morning{place,reason,cost}, afternoon{...}, evening{...}, daily_total_usd}, ...]}

INSIDE

- Calls planner, then post-processes with simple filters:
- add “vegetarian option” tag to restaurant evenings
- mark “family-friendly” for walk-like evenings
- replace “walk” wording with “short visit” if avoiding long walks

FRONTEND (STREAMLIT)

UI

- Inputs at top; generated itinerary as a table.
- Totals: recompute day totals from the three slot costs (robust to minor LLM drift).
- Buttons: Save JSON, Download CSV, Export PDF.

PDF EXPORT

- FPDF (landscape), header row, wrapped multi_cells.
- Manual cursor control to keep columns aligned — no text overlap.

DESIGN CHOICES

- RAG first: keep the LLM honest; reduce repetition.
- Validate with Pydantic: stricter than regex — prevents schema drift.
- Recompute totals: fixes tiny cost inconsistencies.
- PDF ergonomics: multi_cell wrapping + dynamic column widths.

LIMITATIONS

- Datasets are local; not live reviews or hours.
- Costs are estimates, not dynamic prices.
- No route optimization/transit time yet.
- Single-city per plan in v1.

ROADMAP (WHAT I'D ADD NEXT)

- Bring back Regenerate Day with a ban-list of used places and a proper seed.
- Multi-city trips; travel time & distance balancing.
- Live data: Google Places / Yelp / OpenTripMap APIs.
- Maps view + drag-to-swap activities.
- User accounts + shareable links.

2025

The End

THANK YOU FOR LISTENING