1. Project brief

Build a **“Smart Trip Planner”** app in Flutter 3.x (stable) that:

- Lets travellers describe a trip in natural language (e.g., “5 days in Kyoto next April, solo, mid-range budget”)
- Uses an **AI agent** (OpenAI’s models, Gemini, or open-source via Ollama) to generate a day-by-day itinerary with activities, map links, and dining suggestions. Using internet search you can show real time info about them.
- Supports real-time follow-up questions (“Could you swap day 3’s hike with a tea ceremony?”)
- Persists itineraries locally for offline access

2. Core User Stories (MVP)

ID	Story	Acceptance Criteria
S-1	**Create trip via chat**	From a chat screen, the user sends a prompt. The app streams a structured JSON itinerary (see Spec A) from the LLM and renders it as rich cards.
S-2	**Refine itinerary**	Follow-up chat messages modify the plan (agent reasons over previous JSON + user hint). The UI updates diff-style, highlighting changes.
S-3	**Save & revisit**	“Save” button writes the itinerary to a local DB (Hive/Isar). Trips show on a home list; tapping opens a read-only view.
S-4	**Offline view**	If the device is offline, cached trips open; new chat attempts show a graceful error.

S-5	**Basic metrics**	The app logs the number of tokens per request/response and shows it in a debug overlay (helps cost awareness).
S-6	**Web-search**	When generating the itinerary, the AI agent will request real-time information by performing searches using relevant keywords from the itinerary, like “restaurants in Kyoto,” or “best hotels near Fushimi Inari Shrine.**”**

Spec A – Itinerary JSON (minimum fields)

```
{
  "title": "Kyoto 5-Day Solo Trip",
  "startDate": "2025-04-10",
  "endDate": "2025-04-15",
  "days": [
    {
      "date": "2025-04-10",
      "summary": "Fushimi Inari & Gion",
      "items": [
        { "time": "09:00", "activity": "Climb Fushimi Inari Shrine", "location": "34.9671,135.7727" },
        { "time": "14:00", "activity": "Lunch at Nishiki Market", "location": "35.0047,135.7630" },
        { "time": "18:30", "activity": "Evening walk in Gion", "location": "35.0037,135.7788" }
      ]
    }
  ]
```

}

Your agent **must** output this schema.

3. Technical Requirements

Area	Mandatory
Agent logic	Write a *thin* serverless Cloud Function (Dart or TypeScript) or a local isolate that: 1) receives user prompt, previous itinerary JSON, plus chat history; 2) calls the LLM with **function-calling / tool-calling** enabled; 3) validates and returns updated JSON.
Flutter client	Clean architecture (data → domain → presentation). Use **Riverpod 3** *or* **Flutter Bloc**.
Streaming UI	Show token-by-token or chunked streaming in the chat bubble (think ChatGPT typing).
Persistence	Use **Isar** (preferred) or **Hive** behind a repository interface.
Maps integration	Tapping a location opens a Google Maps / Apple Maps intent with coordinates.
Error handling	Handle 401, 429, network loss, JSON-schema errors.

Testing (≥ 60% coverage)	Unit tests for repositories & agent wrapper. Widget tests for chat + itinerary view. Mock HTTP with `http_mock_adapter` or similar.

4. Design

We've created the design to give you an idea regarding how the application should look which you can find here: Figma

5. Nice-to-Have Extras (Optional ⚡)

Idea	Demonstrates
Voice input (speech-to-text) + TTS playback of plan	Multimodal UX
OpenWeather + Currency APIs agent-called for local temps & exchange tips	Tool-calling skills
Local vector store (pinecone-dart / qdrant) to cache site descriptions for faster follow-ups	RAG patterns
Re-rank POIs by walking distance using *A pathfinding**	Algorithmic thinking

---

6. Deliverables

1. **GitHub repo** `smart_trip_planner_flutter`
2. **README.md** with
    - setup (`brew install …`, `flutterfire configure`, etc.)
    - architecture diagram
    - how the agent chain works (prompt, tools, validation)
    - token cost table from your testing (see metrics overlay)
    - Demo video link
3. At least **a few meaningful commits** (no single “final” commit).