Project Blueprint: Gemini-Sponsor Engine

This is a professional-grade stack. Using **Flutter** for the frontend and **Python (FastAPI)** for the backend is the industry standard for high-performance AI applications in 2026. This setup allows you to build a cross-platform app (Web, Android, iOS) while leveraging Python's superior AI libraries.

🏗️ The System Architecture

To make this "online," your Flutter app will communicate with your Python server, which acts as the **Orchestrator**. It talks to Gemini and your Ad Database simultaneously.

🗓️ 1-Month Detailed Plan of Action

Week 1: The "Brains" (Backend & AI)

- **Tech:** Python, FastAPI, *google-generativeai* SDK.
- **Goal:** Create an API that takes a prompt and returns a Gemini response.
- **Tasks:**
    1. Set up **FastAPI** with an */ask* endpoint.
    2. Integrate **Gemini 1.5 Flash** (use the free tier from Google AI Studio).
    3. Build the **Intent Classifier**: A function that asks Gemini, *"Is this user asking about a skill/product? If yes, return the category (e.g., 'C++', 'Law')."*

Week 2: The "Vault" (Ad Retrieval & RAG)

- **Tech:** ChromaDB (Vector DB), Sentence-Transformers.
- **Goal:** Build a system that finds the right ad for the right prompt.
- **Tasks:**
    1. Create a *sponsors.json* with 20 mock ads.
    2. **Vectorize them:** Use Python to turn ad descriptions into math vectors and store them in **ChromaDB**.
    3. **Semantic Search:** When a user asks about "Coding in C++," your backend should "search" the database for the most mathematically similar ad.

Week 3: The "Face" (Flutter Frontend)

- **Tech:** Flutter, *http* package, *flutter_markdown*.
- **Goal:** Build a beautiful chat interface that can render both AI text and Ad cards.

- **Tasks:**
    1. Design a **Chat UI** with a scrollable list.
    2. Create a **Custom Widget** for ads (e.g., a card with a "Sponsored" badge, an image, and a button).
    3. Connect Flutter to your Python API using the http package.

Week 4: The "Launch" (Cloud & Hosting)

- **Tech:** Firebase, Render/Railway, GitHub Actions.
- **Goal:** Move from *localhost* to a live URL.
- **Tasks:**
    1. **Backend Hosting:** Deploy your Python code to **Render** or **Railway**. (They have easy GitHub integration).
    2. **Frontend Hosting:** Deploy the Flutter Web build to **Firebase Hosting** (it's free and extremely fast).
    3. **Environment Variables:** Secure your Gemini API keys using .env files (never upload keys to GitHub!).

💰 Cost Analysis (How to keep it $0)

As a student or developer, you can run this entire project for **free** by staying within these limits:

Service	Provider	Cost in 2026	Free Tier Limits
LLM (AI)	Google AI Studio	**$0**	1,500 requests/day (Gemini 1.5 Flash)
Backend	Render / Railway	**$0**	Free "Spin-down" instances (Active when used)
Frontend	Firebase Hosting	**$0**	10GB storage / 360MB transfer per day
Vector DB	ChromaDB (Local)	**$0**	Running inside your Python backend
Database	MongoDB Atlas	**$0**	512MB to 5GB (Standard for users/logs)

🚀 How to Make It "Online" (Deployment Steps)

1. **Frontend:** * Run *flutter build web --release*.
    - Install Firebase CLI: *npm install -g firebase-tools*.
    - Run *firebase deploy* to get a link like *your-project.web.app*.
2. **Backend:**
    - Push your Python code to a private GitHub repo.
    - Connect the repo to **Render.com**.
    - It will give you a public URL (e.g., *https://api-service.onrender.com*).
3. **The Link:**
    - In your Flutter code, change the API URL from *http://localhost:8000* to your Render URL.

💎 Why This Wins Interviews

When a recruiter asks about this project, don't just say "I used AI." Say this:

> *"I built a **Mediator-Pattern** system where a Python backend orchestrates a RAG-based ad-insertion engine. It uses **Semantic Similarity** to match ads to user intent, ensuring the monetization doesn't degrade the user experience."*

Would you like me to provide the Flutter code for the "Ad Card Widget" so you can see how to display the sponsor inside the chat?