PROJECT PROMPT - DAILY BRIEFING (MVP)

Role:
You are an expert Python backend engineer.
Build a SaaS MVP called Daily Briefing, which automatically generates personalized daily news briefings using ChatGPT-4o-mini via the OpenAI API.

Project Overview:
The app provides daily news summaries for user-selected categories using two AI processes:
1. Selection - picks the most important and early news per category from web search results.
2. Resume - summarizes each selected article into a short briefing.
No user login/signup is required in the MVP.

Workflow:
1. User Input
- User selects up to n = 10 categories (e.g., technology, finance, politics, sports...).

2. Selection Process
- Fetch ~10 latest headlines per category via web search.
- Use ChatGPT-4o-mini to select the top 3 headlines per category.
- Return JSON with title, URL, and short description.

3. Resume Process
- Summarize each selected article via ChatGPT-4o-mini into 2-3 concise sentences.
- Include title, description, URL in summary.

4. Storage
- Store results in SQLite database: category, title, URL, summary, date/time.

5. Output
- Summaries returned via /briefing endpoint or displayed in Streamlit/Dash dashboard grouped by category.

6. Automation
- Daily scheduler (APScheduler) refreshes the news summaries once every 24 hours.

Technical Requirements:
- Language: Python
- Backend: FastAPI
- Frontend: Streamlit or Dash
- Database: SQLite (MVP)
- Scheduler: APScheduler

- LLM: ChatGPT-4o-mini via OpenAI API
- Web Search: News API or equivalent

Project Structure:

```
daily_briefing/
├── app/
│   ├── main.py
│   ├── database.py
│   ├── models.py
│   ├── routes/briefing.py
│   ├── services/selection.py
│   ├── services/summarizer.py
│   ├── services/scheduler.py
│   └── utils/openai_client.py
├── frontend/app.py
├── requirements.txt
├── .env
└── README.md
```

API Endpoints:
- GET /categories -> returns available categories
- POST /briefing -> input: list of categories, output: summarized news JSON

Implementation Notes:
- Use openai.ChatCompletion.create() for Selection and Resume processes.
- Keep code modular: separate Selection and Resume logic.
- Store timestamps in UTC.
- Use .env for secret keys.
- README should include setup and run instructions.

Deliverables:
1. Working FastAPI backend with /briefing endpoint.
2. ChatGPT-4o-mini integration for Selection + Resume.
3. SQLite database with stored summaries.
4. Simple dashboard displaying results by category.
5. Daily refresh job.
6. README with run instructions.