FlowGenX AI — Full Stack Engineer Assignment

Background & Purpose

At FlowGenX AI, we're building an intelligent orchestration platform that blends **data workflows**, **AI agents**, and **modern UI experiences**. As a Full Stack Engineer, you'll work across both frontend and backend technologies — from creating intuitive, visual workflows in **ReactFlow** to integrating with **FastAPI** services powered by LLM, LangGraph, Relational DB, Redis and many other technologies.

This assignment is designed to give you a small taste of the kind of work we do every day — and for us to understand how you approach new challenges, learn new frameworks quickly, and design clean, working solutions.

You're not expected to build a production-grade app; we're more interested in how you structure your code, think through the problem, and connect the dots between frontend and backend.

Project Goal

Build a simple **3-node workflow** using **ReactFlow** (frontend) and **FastAPI** (backend):

`ChatInput → LLM_Call → Update_DB`

The idea is straightforward:

1. A user types a message.
2. The message is sent to an LLM (e.g., OpenAI or Gemini) for a response.
3. The full conversation is saved into a PostgreSQL database.

Core Requirements

1. Backend

Requirements

- `POST /api/chat/message` → Accept a user message and return a response.
- `POST /api/llm/process` → Send input to OpenAI (or Gemini), get AI response.
- `POST /api/db/save` → Save conversation (message + response) to PostgreSQL.
- `POST /api/save-or-update/workflow` → Persist workflow definition.
- Built with **FastAPI** and include basic error handling.
- Database connection (psycopg2) and environment-based configuration.
- Functional OpenAI or Gemini API integration.

Success Criteria

- All endpoints return **HTTP 200**.
- OpenAI/Gemini responses appear correctly.
- Conversation records persist in the database.

2. Frontend

Requirements

- Implement 3 ReactFlow nodes:
    - **ChatInput Node** — Text input for user messages.
    - **LLM_Call Node** — Shows “Processing…” during API call.
    - **Update_DB Node** — Displays “Saved” when data persists.
- Connect the nodes visually with ReactFlow edges.
- Simple chat interface: input + message display.
- Make API calls to backend endpoints.

Success Criteria

- User types a message → Message appears in chat.
- AI response displayed.
- No console errors.
- Data flows logically through the 3 connected nodes.

3. Integration

Requirements

- Frontend communicates with backend APIs successfully.
- Environment variables correctly configured (API key, DB URL).
- Complete flow works end-to-end: Input → LLM → Database.
- Errors surfaced gracefully on the frontend.

Success Criteria

- Sending “Hello” → AI responds → Data appears in DB.
- Works 3 consecutive times without breaking.

You Can Skip

Please **don’t** spend time on:

- Authentication, WebSockets, Tests, Docker, CI/CD, Docs, Monitoring, Styling, Mobile responsiveness, Retry logic, Animations, or Deployment.
- Just focus on the functionality — **make it work**, not perfect.

Definition of Done

- User can type a message in the chat

- Backend receives message and calls OpenAI
- AI response returns and appears in UI
- Conversation saved to database
- Workflow runs successfully 3 times without errors
- Data visible in the database table

If these work — you're done!

Tech Stack

Backend:

- FastAPI
- OpenAI (or Gemini) SDK
- psycopg2
- python-dotenv

Frontend:

- Next.js
- ReactFlow
- fetch / axios

Database:

- PostgreSQL (local)

Environment Variables

```
OPENAI_API_KEY=sk-...
DATABASE_URL=postgresql://user:pass@localhost:5432/convoflow
```

(You may use Gemini from Google AI Studio if you prefer.)

Evaluation Criteria

We'll be looking for:

- Clean, modular, and readable code.
- Working end-to-end functionality.
- Sensible architecture and data flow.
- Ability to pick up ReactFlow quickly.
- Minimal bugs or crashes during demo.

Bonus: Clear commits or comments explaining thought process.

Submission

When complete, please share you gitHub repository. We will schedule a few minutes with you when you can demo it working.

Tip from Us

This exercise isn't about perfection or fancy UI — it's about your ability to **think, learn, and build a working prototype** that connects ideas across the stack.

We're looking for people who can learn fast, reason clearly, and enjoy bringing ideas to life.