**🔹 How the Flow Would Work in FastAPI**

Here’s how your AI Resume Builder would function using FastAPI:

**📌 Step 1: User Inputs Resume Details**

* The frontend (React/Vue/Next.js) sends resume data (name, skills, experience) to FastAPI.
* FastAPI receives the request, processes it, and stores it in a database (PostgreSQL, MongoDB).

**📌 Step 2: Generate Resume (PDF/HTML)**

* FastAPI uses **Jinja2 templates** or **WeasyPrint** to generate resumes dynamically.
* The user can **download the resume as a PDF** or share it via email.

**📌 Step 3: ATS Score Calculation**

* The resume data is passed to an **ML model (Pickle, Scikit-learn, or OpenAI API)** via FastAPI.
* The ATS model **analyzes the resume**, extracts keywords, and assigns an ATS score.

**📌 Step 4: Job Search Using APIs**

* FastAPI fetches **relevant job listings** from LinkedIn, Indeed, or Google Jobs API.
* Jobs are **recommended based on the resume skills** and ATS analysis.
* The user gets a **personalized job feed** directly inside the app.

**🔹 Tech Stack for FastAPI-Based Resume Builder**

| **Component** | **Technology** |
| --- | --- |
| **Backend Framework** | FastAPI 🚀 |
| **Frontend** | React / Next.js (optional) |
| **Database** | PostgreSQL / MongoDB |
| **AI Model (ATS Scoring)** | Pickle, Scikit-learn, TensorFlow |
| **Resume Generation** | Jinja2 (HTML) or WeasyPrint (PDF) |
| **Authentication** | OAuth2 (Google Login, GitHub Login) |
| **Job Search API** | LinkedIn API, Indeed API |

**🔹 Benefits of Using FastAPI**

✔️ **High Performance** – Handles AI processing and API calls efficiently.  
✔️ **Auto-Documentation** – Comes with built-in **Swagger UI** for API testing.  
✔️ **Modern & Asynchronous** – Runs tasks faster than Flask/Django.  
✔️ **Easy Deployment** – Works seamlessly with **Docker, Kubernetes, and Cloud (Azure, AWS, GCP)**.