Great—let's actually *build your first AI app together*, step by step.

I'll keep it beginner-friendly and practical so you can finish something real.

Project: AI Resume Analyzer (Simple & Useful)

This app will:

- Take a resume (text)
- Analyze it using AI
- Give feedback (skills, improvements, score)

Step 1: Setup Your Environment

Install:

- Python (3.9+)
- Code editor (VS Code recommended)

Create project folder:

```
ai-resume-analyzer
cd ai-resume-analyzer
```

Step 2: Install Dependencies

```
pip install openai flask python-dotenv
```

We're using:

- OpenAI API for AI
- Flask for backend

□ Step 3: Get API Key

1. Go to OpenAI website
2. Create account
3. Generate API key

Create a `.env` file:

```
OPENAI_API_KEY=your_key_here
```

□ Step 4: Create Backend (Flask App)

Create `app.py`:

```
from flask import Flask, request, jsonify
import os
from dotenv import load_dotenv
from openai import OpenAI

load_dotenv()

app = Flask(__name__)
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

@app.route("/analyze", methods=["POST"])
def analyze_resume():
    data = request.json
    resume_text = data.get("resume")

    prompt = f"""
    Analyze this resume and give:
    1. Strengths
    2. Weaknesses
    3. Suggestions
    4. Score out of 10

    Resume:
    {resume_text}
    """

    response = client.chat.completions.create(
        model="gpt-4.1-mini",
        messages=[{"role": "user", "content": prompt}]
    )

    return jsonify({
```

```
        "analysis": response.choices[0].message.content
    })

if __name__ == "__main__":
    app.run(debug=True)
```

□ Step 5: Create Simple Frontend

Create `index.html`:

```
<!DOCTYPE html>
<html>
<head>
    <title>AI Resume Analyzer</title>
</head>
<body>
    <h1>AI Resume Analyzer</h1>
    <textarea id="resume" rows="10" cols="50"></textarea><br><br>
    <button onclick="analyze()">Analyze</button>

    <pre id="result"></pre>

    <script>
        async function analyze() {
            const resume = document.getElementById("resume").value;

            const res = await fetch("http://127.0.0.1:5000/analyze", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify({ resume })
            });

            const data = await res.json();
            document.getElementById("result").innerText = data.analysis;
        }
    </script>
</body>
</html>
```

▶□ Step 6: Run Your App

Start backend:

```
python app.py
```

Open `index.html` in browser.

 Paste a resume → click Analyze → AI feedback appears!

Step 7: Improve It (Important!)

Now upgrade from “basic project” → “real product”:

Add:

- File upload (PDF)
- Better UI (use React)
- Save results (database)
- Login system

Step 8: Turn It Into a Product

Ideas:

- Free: 3 resume analyses
- Paid: Unlimited + advanced feedback

Step 9: Deploy It

Use:

- Backend → Railway / Render
- Frontend → Vercel

What You Just Learned

- How to call AI APIs
- How to build backend + frontend

- How to turn AI into a real product

□ Next Step

Tell me ONE thing:

1. “I’m beginner” → I’ll simplify this further
2. “I want advanced” → I’ll help you add:
    - AI agents
    - memory
    - multi-feature SaaS

Or I can help you build a **different AI app** (like chatbot, Instagram tool, etc.)

File upload (PDF)

Perfect—that’s the next big upgrade. Let’s add **PDF upload + text extraction** to your AI app so users can upload resumes instead of pasting text.

□ Step-by-Step: Add PDF Upload

□ Step 1: Install Required Libraries

Run:

```
pip install PyPDF2
```

We’ll use:

- PyPDF2 to read PDFs

□ Step 2: Update Backend (Flask)

Modify your `app.py`:

□ Add import

```
from PyPDF2 import PdfReader
```

□ Replace your `/analyze` route with this:

```
@app.route("/analyze", methods=["POST"])
def analyze_resume():
    file = request.files.get("file")

    if not file:
        return jsonify({"error": "No file uploaded"}), 400

    # Extract text from PDF
    reader = PdfReader(file)
    resume_text = ""

    for page in reader.pages:
        resume_text += page.extract_text() or ""

    if not resume_text.strip():
        return jsonify({"error": "Could not read PDF"}), 400

    prompt = f"""
    Analyze this resume and give:
    1. Strengths
    2. Weaknesses
    3. Suggestions
    4. Score out of 10

    Resume:
    {resume_text}
    """

    response = client.chat.completions.create(
        model="gpt-4.1-mini",
        messages=[{"role": "user", "content": prompt}]
    )

    return jsonify({
        "analysis": response.choices[0].message.content
    })
```

□ Step 3: Update Frontend (HTML)

Replace your `index.html` with this:

```
<!DOCTYPE html>
<html>
<head>
    <title>AI Resume Analyzer</title>
</head>
<body>
    <h1>Upload Resume (PDF)</h1>

    <input type="file" id="fileInput" accept="application/pdf"><br><br>
    <button onclick="analyze()">Analyze</button>

    <pre id="result"></pre>

    <script>
        async function analyze() {
            const fileInput = document.getElementById("fileInput");
            const file = fileInput.files[0];

            if (!file) {
                alert("Please upload a PDF file");
                return;
            }

            const formData = new FormData();
            formData.append("file", file);

            const res = await fetch("http://127.0.0.1:5000/analyze", {
                method: "POST",
                body: formData
            });

            const data = await res.json();
            document.getElementById("result").innerText = data.analysis ||
data.error;
        }
    </script>
</body>
</html>
```

▶ Step 4: Run It

1. Start backend:

```
python app.py
```

2. Open HTML file
3. Upload a PDF resume
4. Click Analyze → Done!

let’s go step-by-step and make sure you can **install Flask and run your app properly** without confusion.

□ 1. Install Python (if not already)

Check if Python is installed:

```
python --version
```

If not, download from:

- Python official site

□ 2. Create Your Project Folder

```
mkdir ai-resume-analyzer
cd ai-resume-analyzer
```

□ 3. (Recommended) Create Virtual Environment

This keeps your project clean.

```
python -m venv venv
```

Activate it:

Windows:

```
venv\Scripts\activate
```

Mac/Linux:

```
source venv/bin/activate
```

□ 4. Install Flask + Other Packages

Now install everything:

```
pip install flask openai python-dotenv PyPDF2
```

You are installing:

- Flask → backend server
- PyPDF2 → read PDFs

□ 5. Create Your App File

Create a file called:

```
app.py
```

Paste your Flask code there.

▶□ 6. Run Flask App

Option 1 (simplest):

```
python app.py
```