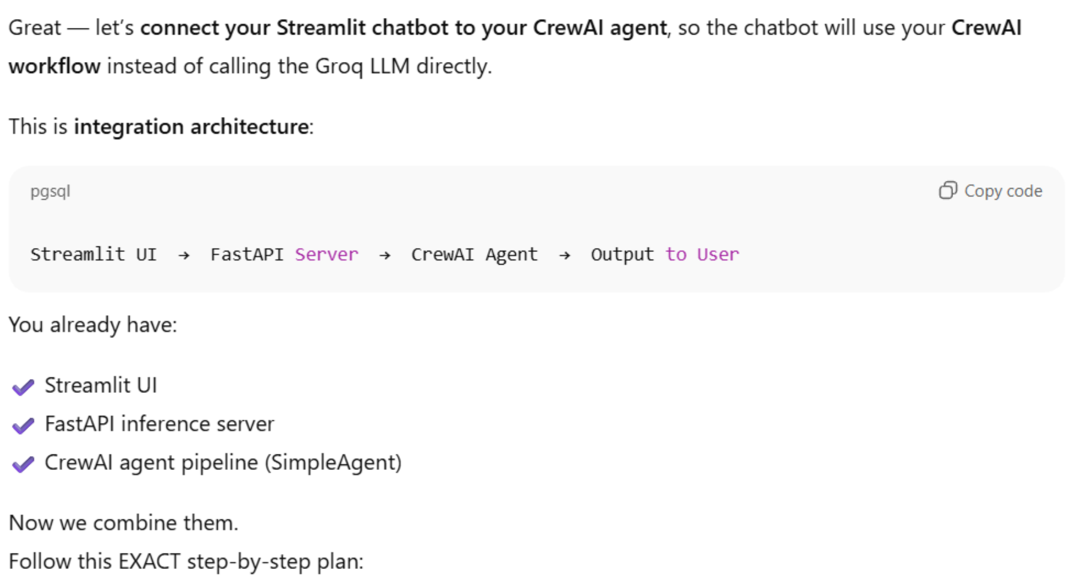
Chatbot and crewai integration.



Great — let's **connect your Streamlit chatbot to your CrewAI agent**, so the chatbot will use your **CrewAI workflow** instead of calling the Groq LLM directly.

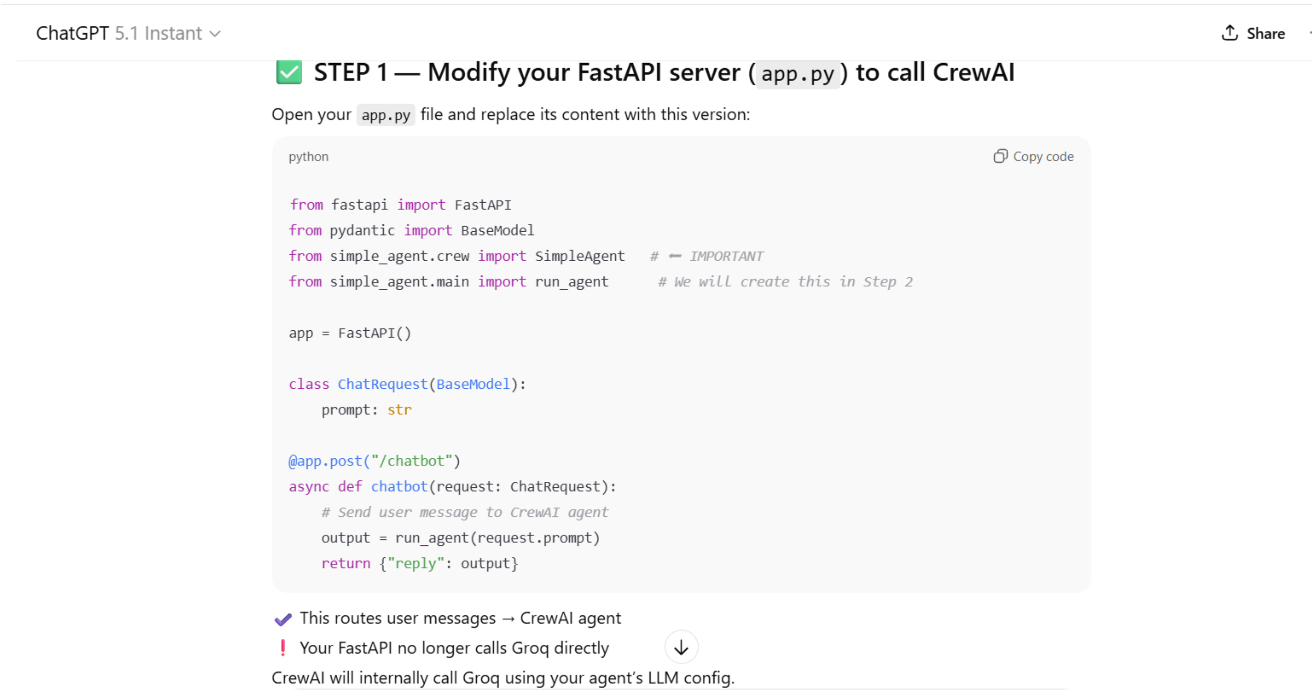
This is **integration architecture**:

```
Streamlit UI → FastAPI Server → CrewAI Agent → Output to User
```

You already have:

✔ Streamlit UI
✔ FastAPI inference server
✔ CrewAI agent pipeline (SimpleAgent)

Now we combine them.
Follow this EXACT step-by-step plan:



✅ STEP 1 — Modify your FastAPI server (`app.py`) to call CrewAI

Open your `app.py` file and replace its content with this version:

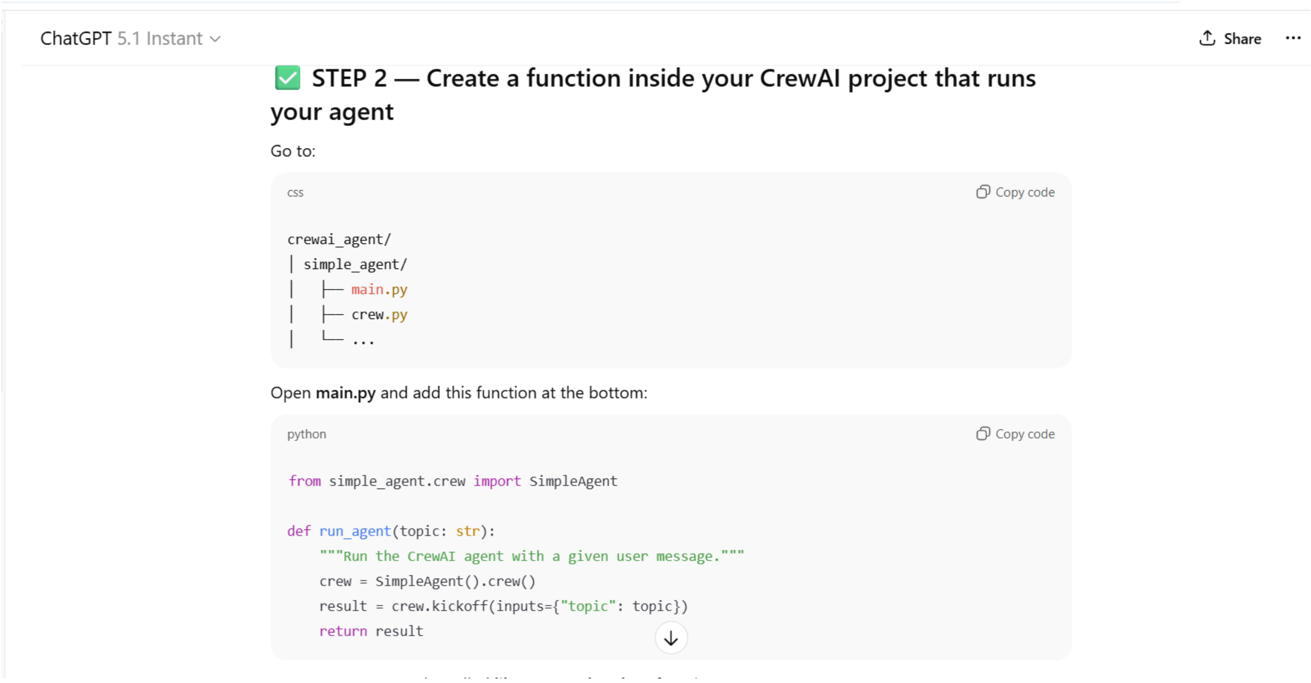
```python
from fastapi import FastAPI
from pydantic import BaseModel
from simple_agent.crew import SimpleAgent    # ← IMPORTANT
from simple_agent.main import run_agent      # We will create this in Step 2

app = FastAPI()

class ChatRequest(BaseModel):
    prompt: str

@app.post("/chatbot")
async def chatbot(request: ChatRequest):
    # Send user message to CrewAI agent
    output = run_agent(request.prompt)
    return {"reply": output}
```

✔ This routes user messages → CrewAI agent
❗ Your FastAPI no longer calls Groq directly
CrewAI will internally call Groq using your agent's LLM config.



✅ STEP 2 — Create a function inside your CrewAI project that runs your agent

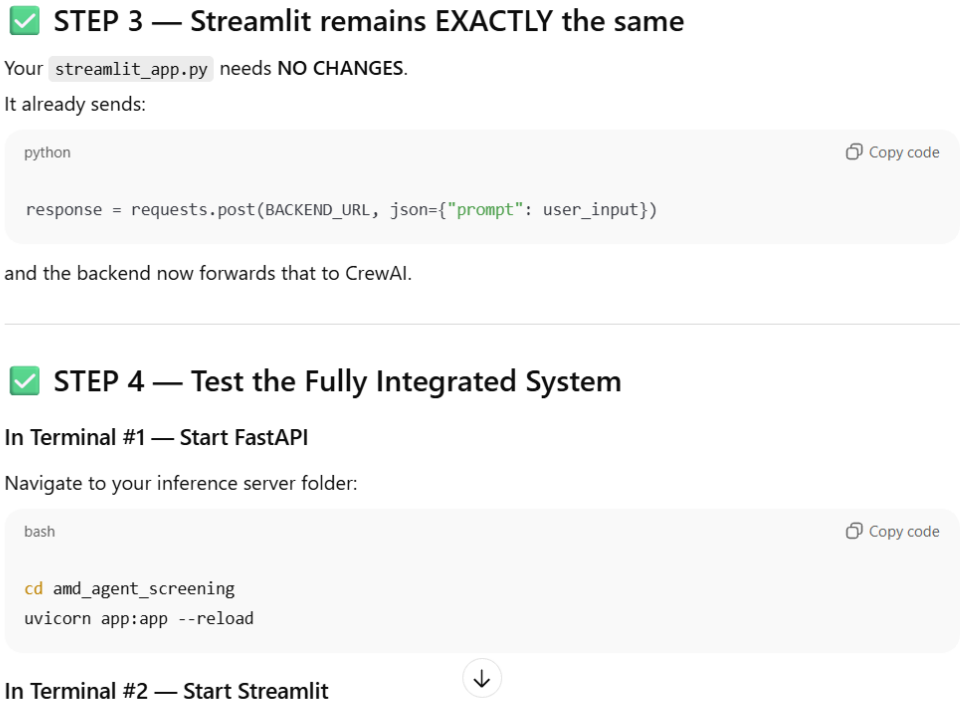
Go to:

```css
crewai_agent/
│ simple_agent/
│   ├── main.py
│   ├── crew.py
│   └── ...
```

Open **main.py** and add this function at the bottom:

```python
from simple_agent.crew import SimpleAgent

def run_agent(topic: str):
    """Run the CrewAI agent with a given user message."""
    crew = SimpleAgent().crew()
    result = crew.kickoff(inputs={"topic": topic})
    return result
```



✅ STEP 3 — Streamlit remains EXACTLY the same

Your `streamlit_app.py` needs **NO CHANGES**.

It already sends:

```python
response = requests.post(BACKEND_URL, json={"prompt": user_input})
```

and the backend now forwards that to CrewAI.

✅ STEP 4 — Test the Fully Integrated System

In Terminal #1 — Start FastAPI

Navigate to your inference server folder:

```bash
cd amd_agent_screening
uvicorn app:app --reload
```

In Terminal #2 — Start Streamlit

In Terminal #2 — Start Streamlit

Navigate to your UI folder:

```arduino
cd streamlit-ui-project
streamlit run streamlit_app.py
```

Now open your browser:
👉 http://localhost:8501 ↗

🎉 Your chatbot now uses CrewAI to generate answers.

📌 What Happens Now?

1️⃣ User types a message in Streamlit
2️⃣ Streamlit sends it to FastAPI as JSON
3️⃣ FastAPI runs the CrewAI agent pipeline
4️⃣ CrewAI uses your Explainer Agent + LLM
5️⃣ Output goes back to Streamlit
6️⃣ User sees the CrewAI-generated answer