□ ProfileCopilot – Approach, Challenges & Solutions

✅ PROJECT GOAL

Build an AI-powered app that analyzes LinkedIn profiles, suggests suitable job roles, checks job fit, rewrites profile sections, and maintains a memory-aware conversation system—all in one smooth user experience.

🔧 DEVELOPMENT APPROACH

1. LinkedIn Scraping

- **Tool Used:** Apify's LinkedIn Profile Scraper Actor
- **Why?** Reliable way to get public profile info like headline, about, skills, and experience.
- **How?** Called using apify-client Python SDK with profileUrls.

---

2. Data Cleaning & Structuring

- **Input:** Raw scraped JSON
- **Output:** Structured profile dictionary
- **Steps:**
    - Extract useful fields (e.g., firstName, headline, skills)
    - Format experiences and skills into clean text

---

3. Job Role Prediction (if empty)

- **Agent:** role_predictor.py
- **Logic:** If user skips job title, AI suggests one using OpenRouter LLM.
- **Prompt:** Profile data sent with instruction: "Suggest ideal job role for this person."

---

4. LangGraph Flow Setup

- **Modules Built:**
    - profile_analyzer.py – Strengths/weaknesses analysis

    - job_matcher.py – Fit score vs target job
    - content_rewriter.py – Suggests improvements + learning path
- **Why LangGraph?**
    - Easy to create modular, state-aware agent pipelines
    - State automatically passes between steps

---

5. Memory Store

- **Tool:** Custom JSON file-based memory per user (via memory_store.py)
- **What It Stores:**
    - Profile info
    - Suggested job role
    - Chat history
    - Past analyses and feedback
- **Goal:** Let user continue talking to bot anytime without losing context

---

6. Chat Assistant Integration

- **Prompt Logic:**
    - Combined LinkedIn summary + job + past conversation
    - Sends everything to OpenRouter LLM
- **Output:** A friendly and memory-aware career mentor chat

🚧 CHALLENGES FACED & SOLUTIONS

⚠ Challenge 1: LinkedIn Scraping Reliability

Problem: Apify scraper sometimes fails or rate-limits on large LinkedIn pages.
Fix:

- Added checks for missing fields
- Used publicIdentifier as fallback for memory key

- Limited request frequency to avoid bans

□ Challenge 2: Broken Profile Data Format

Problem: Some skills and experience fields returned as nested dicts instead of plain text
Fix:

- Used list comprehensions to flatten and extract .get('name') from each item
- Wrapped all joins in safe fallback (if isinstance checks)

□ Challenge 3: Session Persistence in Streamlit

Problem: After a chat response, memory wasn't retaining session correctly
Fix:

- Used st.session_state to store user_id and analyzed flags
- Made chat history update only after analysis was completed

□ Challenge 4: Preventing Double Submit / Button Lag

Problem: On some systems, "Send" button needed to be pressed twice
Fix:

- Refactored input + submit into separate keys
- Ensured input was flushed (st.session_state["chat_input"] = "" workaround)

□ Challenge 5: Conversation Context Prompt Overflow

Problem: Long histories caused token overflow in OpenRouter
Fix:

- Limited chat history to last few turns
- Summarized profile into short version before sending to LLM

□ Challenge 6: Making the Bot Call Users by Name

Problem: LLM response lacked personalization
Fix:

- Extracted firstName from scraped data
- Included it in the system prompt directly

□ Challenge 7: Testing Memory System

Problem: Hard to verify memory retention visually
Fix:

- Designed tests like:
    - "What job role am I aiming for?"
    - "Can you summarize my skills?"
    - "What did I ask earlier?"

✨ **FUTURE IMPROVEMENTS**

- Add PDF resume upload and compare vs LinkedIn
- Integrate multiple job descriptions instead of 1
- Add visual dashboard: skill gap heatmap, roadmap timeline
- Connect learning path to actual course platforms (Coursera, Udemy)
- Deploy with authentication (to protect memory data)

□ CONCLUSION

ProfileCopilot combines scraping, LLMs, LangGraph, and memory to offer an advanced and personal career coaching experience—all in one simple Streamlit app. The modular codebase, strong memory support, and chat integration make it powerful yet easy to extend.