Comparative Reflection: CrewAI vs. ADK

Name: Radhika Bhati
Student ID: 220650
Course: Generative AI Agents – Task Automation with LLM Reasoning

Introduction

Implementing the same six-agent financial analysis workflow in both CrewAI and Google's Agent Development Kit (ADK) was a practical lesson in the fundamental trade-offs in AI agent development. Both of them got me to the same end result, a working investment report, but the way I had to build each one felt completely different. If I had to summarize my experience in one line, I'd say CrewAI makes things faster and smoother for prototyping, while ADK gives you the kind of detailed control you'd need for a production-level system. Below I've compared them across three main aspects: workflow design, coding effort, and system robustness.

1. Workflow Designing

Designing the workflow in CrewAI felt almost effortless. All I really had to do was define the agents, give them clear goals, and then tell the Crew object how they depend on each other. Even running agents in parallel was just a single line of code (async_execution = True). The analogy I would give for it is, It felt like driving a car with automatic transmission, I could focus on the big picture and let CrewAI handle the coordination behind the scenes.

With ADK, it was a completely different vibe. I had to code the entire flow in the main.py file, manually manage the shared state, and even implement parallel execution myself using python's ThreadPoolExecutor. Nothing happened automatically, which meant I had to think through every step. It was a bit overwhelming at first since even in Lab assignments we have did more of them for CrewAI so I was already comfortable with it, but once I got used to it, I liked how much visibility and control I had. More like driving a manual car, you do more work, but you also really understand what's going on.

2. Coding Effort

Here's where the difference really stood out.

- With CrewAI, the coding part was pretty light. Most of my time went into prompt engineering- writing clear roles, goals, and backstories for the agents, and being precise with task descriptions. I could get something working quickly without writing hundreds of lines of code.
- With ADK, the balance shifted. Prompting was still important, but I had to spend a lot more time on traditional software engineering tasks: orchestrating the workflow, managing analysis_state, handling concurrency, and setting up structured logging. At times it felt tedious, but I also felt like I was building a "real" system.

3. System Robustness

Both frameworks produced a final report, but the quality and reliability felt different.

- CrewAI's output was polished and detailed, partly because of how the manager agent refines things internally. Debugging was possible with verbose mode, but I sometimes felt like I was looking into a black box. I couldn't fully control what the memory contained or how it worked.
- ADK, on the other hand, felt much more solid. The explicit analysis_state dictionary gave me a "glass box" memory that I could trust. The logging system I built in workflow.log gave me a clear trace of everything happening step by step. And the compliance check was more reliable because the validator agent could directly cross-reference raw financial data with the draft report. The downside was that the report looked plainer compared to CrewAI's, but it was definitely more predictable and explainable.

Conclusion

Looking back, I don't think it's about which tool is "better," but more about when to use each. CrewAI is perfect for quickly sketching and testing out a workflow idea. It saves a lot of time and lets you focus on the AI side. ADK, while harder and more time-consuming, gave me a system I could actually imagine running in production. If I were building a serious agentic system, I'd probably start with CrewAI for the design phase, and then shift to an ADK-style implementation for deployment.