Digital Twin Assistant (CrewAI)

Author: Ioana Cristescu
Track: Tech Track

Overview
The Digital Twin Assistant is a CrewAI project representing a second-year Master's student in Data Science at Harvard University, originally from Romania, with a B.S. in Computer Science and Mathematics from the University of Richmond. The agent serves as a "digital twin lite" that can introduce me, explain my background, summarize meetings, and analyze code. It processes text, URLs, and PDFs, and for meetings it applies a consistent four-section structure: TL;DR, Decisions, Risks/Blockers, and Next Steps.

Test Results
I tested the agent with multiple prompts:

- **Introduction tasks** ("Introduce yourself to the class" and "Explain my background in 3 sentences") produced accurate and well-structured outputs that reflected my academic path, Romanian background, and personal hobbies.
- **Meeting note summarization** worked reliably with text, PDFs, and URLs. The agent returned structured outputs with the four required sections and used [not found] when information was missing, which improved clarity. Weekly mode was partially successful but did not combine multiple --text inputs correctly.
- **Code analysis** tasks ran successfully and provided detailed explanations with a focus on clarity and security. However, the outputs were often too verbose and could be more concise for practical use.

What Worked Well
- Single-purpose design made the agent reliable across both meeting and code analysis tasks.
- Tools for text, URL, and PDF inputs integrated smoothly and included error handling.
- Persona remained consistent, reflecting both my professional background and personal interests.
- Structured meeting outputs were clear, with graceful handling of missing information.
- Code analysis demonstrated strong explanatory depth and useful security considerations.

Areas for Improvement
- Weekly mode needs to better handle multiple inputs.
- Error messages should be more specific for invalid arguments.
- Code analysis outputs should be shorter and easier to scan.
- Meeting note formatting should be standardized further.
- Input validation should prevent empty or unsupported inputs.

Key Learnings
Through these tests I learned that specialized agents with clear roles perform better than broad, general-purpose ones. Tool selection is critical to ensure reliability across different input types.

Robust error handling improves usability and prevents silent failures. For code analysis, balancing depth with conciseness is important to keep outputs actionable. Finally, it is better to have fewer, well-polished modes than to spread effort across many partially working ones.