Reflection Essay

94-815 Agent Based Modelling and Agentic Technology

Team 8 - Fanxing Bu, Ivan Wiryadi

Tool Selection Trade-offs

To prototype our agent-based financial portal effectively within the constraints of time, budget, and scope, we carefully selected tools across several technical pillars: OCR, language reasoning, agent orchestration, memory/persistence, financial data access, and frontend interface. Our guiding principles were: **maximize learning**, **enable modular iteration**, and **stay within a manageable stack**.

For instance, choosing Mistral VLM over traditional OCR pipelines or handcrafting parsing methods sacrificed fine-grained control, and using SQLite having less features compared to other more robust relational databases. But these dramatically simplified our setup and allow us to focus on learning agents with tools.

Tool selection then should be made based on the context, the developer / user needs, and select the right tools to achieve those said needs.

For more detailed trade-off, please refer to our planning report document.

Ethical Considerations

There are several ethical considerations that immediately come to mind (although there could be more):

Automation Bias

Users may develop excessive trust in AI-generated financial insights the agent. Therefore, the system should implement contextual disclaimers and encourage verification with qualified professionals.

Data Privacy

Financial data reveals sensitive personal information about spending habits, income, and merchant relationships. Sticklet introduces multiple exposure points through:

- Receipt processing via Mistral's OCR
- Queries handled by OpenAI models
- Transaction storage in unsecured SQLite database

Each data processing should be properly vetted and provide sufficient controls.

Security Vulnerabilities

The RAG pattern implemented in Sticklet creates potential attack vectors, particularly through the `SQLQueryTool` which could be vulnerable to SQL injection without proper input sanitization. Validation must be performed.

Moreover, there are inherent risks with Language Models based input and output, such as recent cases in Jailbreaking, harmful outputs, etc. Sufficient guardrails and controls should be put in place.

Accuracy and Responsibility

While the Agent aims to improve data quality through self-reflection and human-reflection, there still can be mistakes. Moreover, the nature of Language Models output may not be necessarily true (hallucinating). Care must be taken into account to make sure the user is aware of such risks.

Lesson Learnt

Our experience building Sticklet—a multi-agent, AI-powered financial portal—revealed several important lessons about aligning product thinking with system architecture and implementation.

1. Product clarity is the foundation of good architecture

We began by clearly defining our business scenario: intelligent receipt management and financial insight generation. This helped us articulate concrete user needs—structured data extraction, monthly summaries, and Q&A interfaces—which in turn guided our technical direction.

2. Design technical architecture based on product scope

Once our product goals were established, we identified the minimum technical requirements to support them—OCR, reasoning, memory, and user interaction. This helped us to quickly narrow down on tools that we need, e.g. using Mistral for OCR, GPT-4 and Mistral for language reasoning, LangChain for agent orchestration, SQLite for persistence, and Streamlit for frontend development.

3. Build the minimum viable system before scaling

We implemented a minimal functional pipeline—OCR to Coordinator Agent to SQL and LLM to UI—as early as possible. The Coordinator Agent, implemented using LangChain, handled task orchestration, memory lookup, and tool invocation. This allowed us to validate the system workflow quickly and manage complexity before adding other features.

4. Scale through modularity and gradual extension

Once the core system was validated, we incrementally introduced new components to extend functionality. These included multi-model dispatching logic (e.g., routing complex reasoning to GPT-4), self-reflection patterns, and specialized agents. The modular design enabled us to add features without destabilizing the overall system.

5. Tool choices are strategic

Tool selection had a significant impact on development speed, as we discussed in the tool trade off reflection.

Conclusion

In summary, the development of Sticklet demonstrated that building effective multi-agent systems requires more than assembling tools and models. A successful architecture must be grounded in product clarity, designed for modularity, and implemented with strategic consideration of trade-offs. Scalability, extensibility, and iterative validation were essential principles that enabled us to deliver a system that was both functional and adaptable to future needs.