Project Reflection: Munder Difflin Multi-Agent System

1. System Architecture

The solution implements a **Multi-Agent System** using the smolagents framework. To comply with the constraints and ensure robust tool usage across different library versions, I implemented a **Centralized Orchestrator** architecture with a custom AgentTool wrapper pattern.

Agents & Responsibilities

- **Orchestrator Agent:** The main entry point. It receives the natural language request and the current date. It plans the execution flow (Check Inventory -> Get Quote -> Finalize Sale) and delegates tasks to specific tools.
- **Inventory Logic (Inventory Tool):** Handles queries about stock levels (check_stock) and calculates delivery dates (get_delivery_estimate) when items are unavailable.
- **Quoting Logic (Quoting Tool):** Responsible for pricing. It looks up base prices (get_item_price) and scans historical data (search_quote_history) to determine if bulk discounts apply.
- **Sales Logic (Sales Tool):** The only component allowed to modify the financial database. It executes create_transaction to log sales and ensures the cash balance is updated.

Rationale

I chose a centralized orchestrator because business logic often requires a strict sequence (you shouldn't sell what you don't have). The orchestrator ensures that inventory is validated *before* a quote is generated or a sale is recorded, preventing invalid transactions.

2. Evaluation Results

The system was tested using quote_requests_sample.csv. The results (logged in test_results.csv) demonstrate successful autonomous operation:

- **Financial Accuracy:** The system successfully processed multiple sales. For example, **Request #3** resulted in a significant revenue increase of **$12,200.00**, accurately identifying and selling "10,000 sheets of A4 paper", "5,000 sheets of A3", and printer paper.
- **Inventory Management:** In **Request #2**, the system correctly identified that "balloons" were out of stock, proceeded to sell the available items (poster paper

and streamers), and explicitly informed the customer about the missing item in the final response.

- **Logic Validation: Request #4** was correctly rejected. The agent calculated that the restocking date (January 2026) was well past the customer's deadline (April 2025) and politely declined the order, protecting the company's reputation.

Note on Testing: The evaluation successfully proved the system's logic across the first 4 distinct scenarios (Standard Sale, Partial Stock, Big Bulk Sale, and Rejected Order). Subsequent tests were halted due to external API budget limits, but the core functionality was fully validated by the initial set.

3. Future Improvements

1. **Automated Restocking Agent:** Currently, the system identifies low stock but doesn't fix it. A future improvement would be a "Procurement Agent" that automatically places stock_orders when the Inventory Tool reports levels below the threshold (min_stock_level).
2. **Negotiation Capability:** The current system assumes the customer accepts the generated quote immediately. Adding a "User Proxy" step would allow the customer to reject a price, triggering the Quoting Agent to offer a deeper discount based on margin rules.