Munder Difflin Multi-Agent System Report

Agent Workflow Diagram

Roles of the Agents

The system is divided into three distinct agents to handle specific aspects of the business logic:

OrchestratorAgent

Serves as the central controller. Its primary role is to manage the lifecycle of the user request. It acts as the bridge between the user and the specialized agents, ensuring that stock is checked before quotes are generated and that transactions are only recorded after a quote is finalized. It does not access the database directly.

QuoteAgent

Handles all Natural Language Processing (NLP) tasks. Its responsibilities include parsing unstructured text into structured JSON (`structure_request`), retrieving relevant historical data (`get_quote_history`), and generating the final pricing and explanation (`calculate_quote`). It focuses on the "creative" and "analytical" side of the process.

InventoryAgent

Manages the "backend" operations. Its responsibilities are strictly defined as checking stock levels (`check_item_stock`) and recording financial transactions (`order_item_stock`). It is the only agent authorized to write to the **transactions** table, ensuring data integrity.

Workflow Description

The workflow follows a linear sequence starting with the Orchestrator sending raw text to the QuoteAgent, which parses it into a structured format containing item names and quantities. The Orchestrator then takes these items and requests the InventoryAgent to check stock levels. Following the stock check, the Orchestrator asks the QuoteAgent to retrieve historical data. The QuoteAgent then uses the structured data and history to infer a final price and explanation, returning this to the Orchestrator. Once the quote is established, the Orchestrator commands the InventoryAgent to record the sales transaction. If the InventoryAgent reports that stock has fallen below the minimum threshold, the Orchestrator triggers a stock order to replenish the inventory.

Evaluation Results

The system processed **20 unique quote requests**, resulting in a net increase in cash reserves. The inventory management logic maintained stock levels, automatically reordering items when they fell below the minimum threshold.

Metric	Start Value	End Value	Net Change
Cash Balance	$42,389.22	**$43,208.33**	+$819.11
Inventory Value	$7,610.78	**$6,791.67**	-$819.11
Total Assets	$50,000.00	**$50,000.00**	$0.00

Analysis

- **Revenue Generation:** The agents generated **$819.11** in net cash flow.
- **Inventory Efficiency:** Inventory value decreased by an equal amount, indicating that sales were made from existing stock. The system balanced stock depletion with revenue generation.
- **Asset Stability:** The total assets (Cash + Inventory) remained perfectly stable at **$50,000.00**, confirming that the accounting logic for sales (converting inventory asset to cash asset) is functioning correctly without data loss or phantom costs.

Specific Strengths:

The QuoteAgent successfully interpreted the requirement for "friendly" pricing, applying discounts for bulk orders and rounding final totals down to integers or multiples of 5 (e.g., rounding $93.50 to $90.00) in strict adherence to the prompt's tone guidelines.

The system correctly identified scenarios where demand exceeded supply. For requests where the stock was insufficient to fulfill the order, the Orchestrator correctly halted the process and returned a message indicating the quote could not be processed, thereby preventing the company from selling phantom inventory.

Suggestions for Further Improvements

Implement backorder capability

Currently, if an item is out of stock, the Orchestrator rejects the entire order. A better approach would be to have the QuoteAgent check the supplier delivery date if stock is insufficient. If the delivery arrives before the client's deadline, the system should allow the sale with a specific warning in the final quote explanation, allowing the company to capture revenue that is currently being lost.

Parallel execution for multi-item orders.

The current Orchestrator iterates through items in a request one by one to check stock, which causes latency for large orders. Modifying the InventoryAgent to accept a list of items for batch processing would allow the Orchestrator to validate the entire order's inventory in a single tool call, significantly reducing execution time and token usage.