Home Assignment

AI Agent Home Assignment: Financial Research Assistant

Overview

This assignment tests your ability to build an intelligent agent that analyzes complex financial documents and supplements that analysis with real-time web search. You'll work with Apple's 10-K filing (their annual financial report) and build an agent that can answer questions by using multiple tools together.

Time Estimate: 3-4 hours

Difficulty: Intermediate to Advanced

What we're testing: Agent development, document processing, tool orchestration, information synthesis

Whatis a 10-K Filing?

A **10-K** is an annual report required by the U.S. Securities and Exchange Commission (SEC) that provides a comprehensive summary of a company's financial performance. It includes: • **Business Overview**: Description of operations, products, and strategy

- **Risk Factors**: Detailed risks that could impact the business (typically 20-40 pages)
    - **Management Discussion & Analysis (MD&A)**: Management's perspective on financial results • **Financial Statements**: Balance sheets, income statements, cash flow statements • **Notes to Financial Statements**: Detailed breakdowns and accounting policies • **Geographic/Segment Data**: Revenue and performance by region and product line

Assignment Objective

Build an agent that can answer complex questions about Apple Inc. by:

1. Parsing and querying Apple's 10-K filing

2. Using web search for real-time data when needed

3. Doing calculations on financial data

4. Combining information from multiple sources

5. Giving accurate answers with proper citations

Requirements

1. Tools You Must Implement

Your agent must be able to use these two tools:

Tool 1: Document QA System

- Parse Apple's 10-K filing (PDF or HTML format)
- Implement a chunking strategy that preserves document structure
- Build a retrieval system (vector database, embeddings, etc.)
- Return relevant sections with citations

Key Considerations:

- How will you handle tables vs. text?
- How will you preserve section context (which part of the 10-K)?
- What chunking size/strategy will you use?
- How will you maintain relationships between chunks that reference each other?
- How will you handle cross-references (e.g., "See Note 5", "As discussed in Risk Factors")? • How will you preserve entity relationships (e.g., revenue by region, products by segment)? • How will you connect narrative discussion to corresponding table data? • What metadata will you attach to chunks to enable multi-hop reasoning?

Tool 2: Tavily Search API

- Integrate Tavily for real-time web search
- Use for current market data, stock prices, competitor info, news
- Free tier provides 1,000 requests/month
- Sign up at: https://tavily.com

When to use Tavily:

- Current stock prices or market caps
- Competitor comparisons
- Recent news or events
- Any data not in the 10-K or that has changed since filing

2. Document Setup

Step 1: Navigate to Apple's SEC EDGAR page:

```
1 https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&CIK=0000320193
```

Step 2: Find the most recent 10-K filing:

- Look for document type "10-K" (not 10-K/A which is an amendment) • Click on the "Documents" button
- Download either:
  - The HTML version (aapl-20230930.htm or similar)
  - Or the PDF version if available

Step 3: Place the file in your project directory

- We recommend creating a `data/` folder
- Name it clearly (e.g., `apple_10k_2023.htm`)

3. Agent Architecture

Your agent should use one of these patterns:

- **ReAct** (Reasoning + Acting)
- **Function Calling** (if using OpenAI/Anthropic APIs)
- **Tool Use** pattern (or build your own)

What your agent needs to do:

- Figure out which tools are needed for a query
- Run tools in the right order
- Handle failures gracefully
- Combine results from multiple tools

- Cite sources for all claims

Some examples of decision logic:

- "What are Apple's risk factors?" → Just use Document QA
- "What's Apple's current stock price?" → Just use Tavily
- "How does Apple's P/E ratio compare to Microsoft?" → Use Document QA + Tavily

Test Queries

Your agent must successfully handle these two queries:

Query 1: Document-Only Analysis

Question:

"What are Apple's top 3 risk factors mentioned in their latest 10-K, and what percentage of total revenue did they spend on R&D?"

What This Tests:

- Multi-section document retrieval (Risk Factors + Financial Statements)
- Extracting specific information (top 3 items)
- Finding numerical data in tables
- Performing simple calculations
- Combining information from different document sections

Expected Tools:

1. Document QA (Risk Factors section)

2. Document QA (R&D expenses and total revenue from financial statements)

Query 2: Hybrid Analysis (Document + Web)

Question:

"How does Apple's gross margin compare to Microsoft's current gross margin, and what reasons does Apple cite in their 10-K for any margin pressure?"

What This Tests:

- Extracting financial metrics from 10-K
- Using web search for competitor data
- Understanding when document data vs. current data is needed
- Combining internal (10-K) and external (web) sources
- Comparative analysis

Expected Tools:

1. Document QA (Apple's gross margin from financial statements)

2. Document QA (MD&A section for margin discussion)

3. Tavily (Microsoft's current gross margin)

Technical Specifications

Language & Frameworks

- **Language:** Python only
- **Frameworks:** Use any framework you prefer (LangChain, LlamaIndex, custom implementation, etc.)

LLM Provider

Use any LLM API or model you prefer.

If you don't have access to paid APIs (OpenAI, Anthropic, etc.), you can use **Google's Gemini API for free**:

1. Visit **Google AI Studio**: https://aistudio.google.com

2. Sign in with your Google account

3. Click "Get API key" in the left sidebar

4. Generate your API key (no credit card required)

Free tier includes:

- 15 requests per minute for Gemini 2.5 Flash
- 1,500 requests per day
- 1 million token context window
- Completely free - no billing setup needed

Documentation: https://ai.google.dev/gemini-api/docs

User Interface

Your agent should run via command line interface (CLI). No need to build a web UI or chat application. Users should be able to:

- Run your agent from the terminal
- Type queries and get responses
- See which tools the agent is using (optional but helpful for debugging)

Example interaction:

```
$ python src/main.py
> What are Apple's top 3 risk factors?
[Agent processing...]
[Tool: Document QA - Searching Risk Factors section] Here are Apple's top 3 risk factors:
1. ...
```

Document Processing

Use any libraries you wish for parsing PDFs or HTML.

Vector Database (for RAG)

Use any vector database you prefer.

Key Constraints

- Must handle cases where information is not available • Must provide citations (section name or page number from 10-K) • Error handling for API failures, missing data, etc.

Bonus Features (Optional)

These aren't required, but they'd be nice to see if you have time:

Bonus 1: Structured Table Extraction

- Keep table structure intact from financial statements
- Parse multi-column tables accurately
- Maintain table headers and row/column relationships

Bonus 2: Page-Level Citations

- Reference specific page numbers
    - Link findings to exact locations in the 10-K

Bonus 3: Planning Step

- Add a "plan" phase where the agent explains its approach before executing • Show reasoning: "To answer this, I will: 1) ... 2) ... 3) ..."
- Makes debugging easier and behavior more transparent

Bonus 4: Cross-Reference Handling

- Handle references like "See Note 5 to Consolidated Financial Statements" • Follow internal document links automatically

Deliverables

1. Working Code

Submit a GitHub repository or ZIP file containing:

```
project/
├── README.md # Setup and run instructions ├── requirements.txt # Python dependencies
├── .env.example # Example environment variables ├── src/
│ ├── agent.py # Main agent logic
│ ├── tools/ # Tool implementations
│ │ ├── document_qa.py
│ │ └── tavily_search.py
│ ├── utils/ # Helper functions
│ └── main.py # Entry point
├── prompts/ # Agent prompts in separate files │ ├── system_prompt.txt # Main system prompt for the agent │ ├── tool_descriptions.txt # How to describe tools to the LLM │ └── examples.txt # Few-shot examples (optional) ├── data/
│ └── apple_10k_2023.htm # The 10-K file
└── tests/
└── test_queries.py # Test the 2 required queries
```

Code Quality Expectations:

- Clean, readable code with appropriate comments

- Proper error handling and logging
- Separation of concerns (agent logic, tools, utilities)
- Type hints (if using Python 3.7+)
- No hardcoded API keys (use environment variables)
- **Store prompts in separate text files** (in prompts/ directory) rather than hardcoding them in your code

2. README.md

Your README should include:

Setup Instructions

```
## Setup
1. Clone the repository
2. Install dependencies: `pip install -r requirements.txt`
3. Download Apple 10-K from [link] and place in data/
4. Copy .env.example to .env and add your API keys:
- OPENAI_API_KEY (or ANTHROPIC_API_KEY)
- TAVILY_API_KEY
5. Run: `python src/main.py`
```

How to Run Test Queries

Clear instructions on how to execute the two test queries

Architecture Overview

A simple diagram or description of your agent's architecture:

- How tools are registered
- How the agent decides which tool to use

- How information flows through the system
    - Where prompts are stored (prompts/ directory)

Dependencies

List all major dependencies and why you chose them

Submission

What to Submit:

1. **Code Repository** (GitHub link or ZIP file)
2. **README.md** with setup instructions
3. **Example outputs** for the 2 test queries (optional but helpful) Good luck!