# Solution 1: Foundational Investment Memo Fact-Finding (RAG)

## Project Design & Scaffolding Report

## Section 1: Business Problem Breakdown & Analysis

1. **Problem Statement:** The business requires an automated system to assist analysts in generating investment memos. The core need is to process dense financial documents like SEC filings (10-Ks, 10-Qs) to extract key facts, summarize critical sections, and assemble this information into a structured draft report, reducing the manual effort of data collection.
2. **LLM Suitability Analysis:** This problem is a strong candidate for an LLM-based solution due to its reliance on processing large volumes of unstructured and semi-structured text from financial documents. An LLM's ability to understand natural language queries, identify relevant information within lengthy reports, and generate coherent summaries is a perfect fit for the initial, fact-gathering stages of credit analysis. A standard RAG approach provides a robust mechanism for grounding the LLM's output in the specific source documents, which is critical for maintaining factual accuracy.
3. **Core Tasks:**
    i. **Ingest Source Documents:** Systematically ingest primary financial documents (e.g., 10-K, 10-Q) for a target company.
    ii. **Process and Chunk:** Extract text and tables, clean the data, and divide it into contextually relevant chunks for embedding.
    iii. **Query for Section Content:** For each section of the memo (e.g., "Business Analysis," "Risk Factors"), formulate a query to retrieve relevant chunks from the knowledge base.
    iv. **Generate Section Draft:** Use an LLM to synthesize the retrieved information into a written narrative for that specific memo section.
    v. **Compile Draft Memo:** Assemble the generated sections into a single,

cohesive document for an analyst to review and refine.

## Section 2: Recommended Solution Architecture

1. **Recommended Tool: RAG (Retrieval-Augmented Generation)**
2. **Justification:** A standard RAG architecture is the recommended starting point for this task. It directly addresses the core challenge of extracting and summarizing information from a specific knowledge base (the company's financial filings).
    - **vs. Naive LLM Call:** A naive call to an LLM without RAG would be dangerously unreliable. The model would lack access to the specific, up-to-date financial data in the 10-K and would be forced to rely on its general training data, leading to outdated information and hallucinations. RAG ensures the output is grounded in the provided source documents.
    - **vs. Agentic Workflow:** While an Agentic Workflow is ultimately the superior solution for full memo generation (as detailed in the second report), a standard RAG system is a less complex, foundational first step. It excels at the "retrieve-and-summarize" tasks that form the building blocks of the memo. This approach allows the organization to build the critical data ingestion pipeline and knowledge corpus, delivering immediate value in automating factual extraction before tackling the more complex reasoning and calculation tasks.
3. **Detailed Workflow:**
    i. **Ingestion:** An analyst specifies a target company. The system ingests the latest 10-K and 10-Q filings for that company from a source like the SEC EDGAR database.
    ii. **Indexing:** The documents are parsed, cleaned, and chunked using a semantic chunking strategy (e.g., splitting by sections like "Risk Factors"). The chunks are converted to vector embeddings and stored in a vector database with rich metadata (document name, fiscal period, etc.).
    iii. **Querying:** For a desired memo section, like "Risk Factors," a predefined query is run against the vector database. For example: "What are the primary risks to the business as described in the 'Risk Factors' section of the latest 10-K?"

iv. **Retrieval:** The vector database performs a similarity search and retrieves the most relevant text chunks. Metadata filters are used to ensure the chunks come from the correct document and section.
v. **Generation:** The retrieved chunks are compiled into a context block and passed to an LLM along with a prompt, instructing it to synthesize the information into a coherent narrative for the "Risks & Mitigants" section of the memo.
vi. **Iteration:** This process is repeated for each section of the memo, and the outputs are compiled into a final draft.

# Section 3: Input and Output Specification

1. **Input Documents:**
    - **Types:** PDF files from the SEC EDGAR database, specifically Form 10-K and Form 10-Q.
    - **Sample Input:** A small excerpt from a fictional 10-K's "Risk Factors" section.

```
**Risk Factors**

**Risks Related to Our Business and Industry**

Our operating results may be adversely affected by evolving and uncer

Furthermore, our business is subject to complex and evolving U.S. and
```

2. **Output Deliverable:**
    - **Format:** A single Markdown (`.md`) file.
    - **Structure & Content:** The file will be structured with standard headers corresponding to the memo sections. The content will be the LLM-generated narrative based on the source documents.

```
# Investment Memo Draft: [Company Name]

## 1. Business & Market Analysis
(LLM-generated text synthesizing the 'Business' description from the

## 2. Financial Analysis Summary
(LLM-generated text summarizing key trends from the MD&A section)

## 3. Risks & Mitigants
(LLM-generated text summarizing the 'Risk Factors' section)

---
**Source Documents:**
- [e.g., AAPL_10K_2023.pdf]
```

## Section 4: Python Scaffolding (Tenant & ElementTemplates)

```python
# Assume necessary imports and Beanie initialization are done.
# from models import TenantConfiguration, ElementTemplate, TenantType, TaskType, ElementTy

# --- Define Enums for this use case ---
class TenantType(str, Enum):
    INVESTMENT_MEMO_GENERATION_RAG = "INVESTMENT_MEMO_GENERATION_RAG"
    # Other tenant types would be added here

# 1. Tenant Configuration
# This configures the general category for this task.
investment_memo_rag_tenant = TenantConfiguration(
    tenant_type=TenantType.INVESTMENT_MEMO_GENERATION_RAG,
    display_name="Investment Memo Generation (Standard RAG)",
    description="Generates sections of an investment memo by retrieving and summarizing fa
    default_task_type=TaskType.RAG,
    auto_provision_templates=True,
    is_active=True,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# In a real application, you would save this to the database:
# await investment_memo_rag_tenant.insert()


# 2. Element Template
# This single, powerful prompt template drives the generation for any given memo section.
section_generator_template = ElementTemplate(
    name="RAG_Memo_Section_Generator",
    description="A RAG prompt that takes a memo section title and source documents, and ge
    tenant_type=TenantType.INVESTMENT_MEMO_GENERATION_RAG,
    task_type=TaskType.RAG,
    element_type=ElementType.PROMPT_TEMPLATE,
    generation_prompt="""
You are an expert financial analyst assistant. Your task is to write a clear, concise, and

**Instructions:**
1.  You will be given a `section_title` and a set of `source_documents` retrieved from the
2.  Your entire response MUST be based ONLY on the information provided in the `source_doc
3.  Synthesize the information from the provided context to write a professional-grade sum
4.  Structure your output using paragraphs and bullet points for clarity.
```

```
5.  If the provided context is insufficient to write the section, you must state: "Insuffi

**Company:** {company_name}
**Memo Section to Generate:** {section_title}

**Source Documents (Context):**
---
{source_documents}
---

**Generated Memo Section:**
""",
    retrieval_prompt="Information required to write the '{section_title}' section of an in
    variables=["company_name", "section_title", "source_documents"],
    execution_config={
        "model": "gpt-4-turbo",
        "temperature": 0.2,
        "max_tokens": 1500,
    },
    is_system_default=True,
    version="1.0.0",
    tags=["finance", "investment memo", "rag", "summarization", "10-k"],
    status=ElementStatus.ACTIVE,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# In a real application, you would save this to the database:
# await section_generator_template.insert()
```

## Section 5: Evaluation Strategy

1. **Human-in-the-Loop Evaluation:**
    - **Methodology:** A senior financial analyst will be presented with a generated memo section side-by-side with the source document text chunks that were used as context. The analyst will score the output using the rubric below and provide qualitative feedback on errors or areas for improvement.
    - **Evaluation Rubric:**
      | Criterion | 1 (Poor) | 3 (Acceptable) | 5 (Excellent) |

| :--- | :--- | :--- | :--- |
| **Faithfulness** | Output contains significant hallucinations or facts not supported by the source text. | Output is mostly faithful, with minor, non-critical deviations. | All statements in the output are verifiably grounded in the provided source text. |
| **Completeness** | Output misses critical information present in the source text for the requested section. | Output captures the main points but may miss some nuances from the source. | Output comprehensively covers all relevant information from the source text. |
| **Coherence** | The generated text is disjointed, poorly structured, or difficult to read. | The text is understandable and logically structured. | The text is clear, well-written, and flows logically like a professional report. |
| **Relevance** | The output includes irrelevant information not pertaining to the requested section title. | The output is focused on the topic but may include minor tangential points. | The entire output is highly relevant and directly addresses the section title. |

2. **LLM-as-a-Judge Evaluation:**
    - **Methodology:** A powerful model like GPT-4o or Claude 3 Opus will be used as a "Judge." It will receive the source context, the original query (i.e., the section to be generated), and the system's generated output. It will be prompted to assess the faithfulness of the generation and output a structured JSON object with its verdict and rationale.
    - **Judge LLM Prompt:**

```
You are an impartial evaluator. Your task is to determine if a genera

You will be provided with a "Context" from a financial document and a

Analyze the "Generated Answer" sentence by sentence. For each sentenc

Respond in the following JSON format:
{
  "is_faithful": boolean, // true if the entire answer is supported b
  "reasoning": "Provide a step-by-step analysis here. If not faithful
  "unsupported_sentences": ["List the exact sentences that are not su
}

**Context:**
{source_context}

**Generated Answer:**
{generated_answer}

**Evaluation (JSON Output Only):**
```

- **Sample Code Snippet:**

```python
import openai

client = openai.OpenAI(api_key="YOUR_API_KEY")

def evaluate_faithfulness(source_context, generated_answer):
    judge_prompt = f"""
    You are an impartial evaluator... [rest of the prompt from above]

    **Context:**
    {source_context}

    **Generated Answer:**
    {generated_answer}

    **Evaluation (JSON Output Only):**
    """

    response = client.chat.completions.create(
        model="gpt-4o",
        messages=[{"role": "user", "content": judge_prompt}],
        temperature=0.0,
        response_format={"type": "json_object"}
    )
    return response.choices[0].message.content
```

## Section 6: User Input

This report addresses the initial request to design a system for AI-powered investment memo generation. It focuses on establishing a foundational RAG-based solution as a first step, capable of handling the core tasks of document ingestion, fact retrieval, and summarization, as outlined in the user's comprehensive strategic blueprint.

---

---

Here is the second, more advanced report.

---

# Solution 2: Advanced Investment Memo Generation (Agentic Workflow)

## Project Design & Scaffolding Report

---

## Section 1: Business Problem Breakdown & Analysis

1. **Problem Statement:** The business requires a sophisticated AI system to generate a comprehensive, investment-grade credit memo. This task transcends simple fact retrieval and demands multi-step reasoning, quantitative analysis (e.g., ratio calculation), qualitative judgment, risk identification, and the synthesis of information from diverse sources into a coherent, defensible investment opinion.
2. **LLM Suitability Analysis:** This complex problem is an excellent fit for an advanced LLM-based solution, specifically an agentic workflow. While basic LLMs can summarize text, the core of this task involves *process execution* that mimics a human analyst's workflow. An agentic framework allows the LLM to move beyond a simple "generator" role to become an "orchestrator" and a team of "specialist workers." It can decompose the complex goal, use external tools (like a calculator), perform multi-hop reasoning across documents, and iteratively build and refine a complex analytical product, which is precisely what is required.
3. **Core Tasks:**
    i. **Decompose Memo Task:** Break down the high-level request ("Generate memo for Company X") into a structured plan of sequential and parallel sub-tasks.
    ii. **Specialized Information Retrieval:** Execute targeted queries to find specific data types (e.g., financial figures, risk statements, management bios) from various documents.
    iii. **Perform Quantitative Calculations:** Extract numerical data and use a calculation tool to compute key financial ratios (e.g., Debt/EBITDA, ICR).
    iv. **Synthesize Narrative Sections:** Assign specialist agents to draft individual memo sections (e.g., Financial Analysis, Market Overview) based on the retrieved and calculated data.

v. **Critique and Refine:** Review the generated sections and the overall draft for logical consistency, accuracy, and narrative flow.
vi. **Assemble Final Report:** Compile all finalized sections into a single, professional-grade document with verifiable citations.

# Section 2: Recommended Solution Architecture

1. **Recommended Tool: Agentic Workflow**
2. **Justification:** An Agentic Workflow is not just recommended; it is required to meet the problem's complexity.
    - **vs. Standard RAG:** A standard RAG system is fundamentally inadequate. It operates on a linear "retrieve-then-generate" path and cannot perform the necessary intermediate steps. It cannot natively use a calculator to compute financial ratios, dynamically plan a multi-step analysis, or synthesize findings from a 10-K, an earnings call transcript, and a news article to form a forward-looking judgment. The task requires a dynamic "reason-act-retrieve" loop, which is the hallmark of an agentic system.
    - **The Orchestrator-Worker Pattern:** This problem maps perfectly to the Orchestrator-Worker agentic pattern described in the user's blueprint. A central **Orchestrator** can create the memo plan, and specialized **Worker Agents** (Financial Extractor, Ratio Calculator, Risk Identifier, Section Writer) can execute their expert functions in parallel. This modularity makes the system more powerful, robust, and easier to maintain and debug than a single monolithic agent.
3. **Detailed Workflow:**
    i. **Initiation:** An analyst provides a high-level goal: "Generate an investment memo for Company XYZ's $500M note offering, focusing on debt service capacity."
    ii. **Planning (Orchestrator Agent):** The Orchestrator Agent receives the goal. It uses its "Memo_Orchestrator_Planner" template to generate a structured plan, outlining the required sections and the agents needed to create them. This plan might be a JSON object or a DAG.
    iii. **HITL: Plan Approval (Optional but Recommended):** The plan is presented to the human analyst for review and approval before

execution proceeds.

iv. **Delegation & Execution (Worker Agents):** The Orchestrator dispatches tasks to the worker agents:
    - **Financial Data Extraction Agent:** Scans the 10-K/10-Q to extract key figures like Revenue, EBITDA, and Total Debt into a structured format.
    - **Ratio Calculation Agent (Tool-Using):** Receives the structured data, invokes a calculator tool, and computes Debt-to-EBITDA, ICR, etc.
    - **Risk Identification Agent:** Scans all documents for keywords like "litigation," "supply chain disruption," etc., and creates a list of potential risks.

v. **Synthesis (Writing Agents):** The Orchestrator passes the outputs from the previous agents to specialized writing agents. For example, the **Financial Analysis Writer** receives the historical figures and calculated ratios to draft its section.

vi. **Assembly & Editing:** The Orchestrator compiles the drafted sections. A dedicated **Editor Agent** reviews the full draft for consistency, tone, and flow.

vii. **Final Review (HITL):** The complete draft memo, with every key claim linked to its source citation, is presented to the senior analyst for final review, editing, and sign-off.

# Section 3: Input and Output Specification

1. **Input Documents:**
    - **Types:** A diverse set of documents including PDFs (10-K, 10-Q, S-1), text from APIs (News Feeds), and potentially internal documents (previous memos).
    - **Sample Input:** A sample from a fictional earnings call transcript.

```
**Analyst Question:** "Can you provide some color on the margin compr

**CFO Response:** "Certainly. The margin pressure was primarily drive
```

2. **Output Deliverable:**

- **Format:** A professional, multi-page PDF report or an interactive web interface.
- **Structure & Content:** A comprehensive, well-structured document with verifiable citations for every claim. The structure follows the detailed breakdown in the user's blueprint (Executive Summary, Transaction Overview, etc.).

  **Example Section: 7. Risks & Mitigants**
    - **Market Risk: Margin Compression:** The company experienced margin compression in the most recent quarter, driven by rising raw material costs (Source: Q3 2023 Earnings Call Transcript, p. 5). Management expects this pressure to be temporary and normalize within two quarters.
    - **Operational Risk: Customer Concentration:** A significant portion of revenue is derived from the company's top three customers (Source: 2023 10-K, p. 12).
    - **Mitigant:** The proposed loan includes covenants that monitor liquidity and profitability on a quarterly basis.

## Section 4: Python Scaffolding (Tenant & ElementTemplates)

```python
# Assume necessary imports and Beanie initialization are done.
# from models import TenantConfiguration, ElementTemplate, TenantType, TaskType, ElementTy

# --- Define Enums for this use case ---
class TenantType(str, Enum):
    INVESTMENT_MEMO_AGENTIC_WORKFLOW = "INVESTMENT_MEMO_AGENTIC_WORKFLOW"
    # Other tenant types would be added here

# 1. Tenant Configuration
investment_memo_agentic_tenant = TenantConfiguration(
    tenant_type=TenantType.INVESTMENT_MEMO_AGENTIC_WORKFLOW,
    display_name="Investment Memo Generation (Agentic Workflow)",
    description="A multi-agent system that plans, analyzes, calculates, and writes a compl
    default_task_type=TaskType.AGENTIC_WORKFLOW,
    auto_provision_templates=True,
    is_active=True,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# await investment_memo_agentic_tenant.insert()


# 2. Element Templates (One for each specialized agent task)

# Template for the Orchestrator Agent to create the plan
orchestrator_planner_template = ElementTemplate(
    name="Agentic_Memo_Orchestrator_Planner",
    description="Generates a structured, step-by-step plan for the agentic workflow to cre
    tenant_type=TenantType.INVESTMENT_MEMO_AGENTIC_WORKFLOW,
    task_type=TaskType.AGENTIC_WORKFLOW,
    generation_prompt="""
You are a master project manager for an AI-powered financial analysis team.
Your goal is to create a detailed, step-by-step plan to generate a comprehensive investmen

**Company:** {company_name}
**Transaction Details:** {transaction_details}
**Source Documents Available:** {document_list}

Based on the above, generate a plan as a JSON object. The plan should be a list of tasks,
```

```
Available agent_types:
- 'FinancialDataExtractor'
- 'RatioCalculator'
- 'RiskIdentifier'
- 'MarketAnalysisWriter'
- 'FinancialAnalysisWriter'
- 'SWOTWriter'
- 'Editor'

Example Task:
{{
  "name": "Extract Q3 2023 Financials",
  "agent_type": "FinancialDataExtractor",
  "dependencies": []
}}

**JSON Plan Output:**
""",
    variables=["company_name", "transaction_details", "document_list"],
    execution_config={"model": "gpt-4-turbo", "response_format": {"type": "json_object"}},
    tags=["agent", "orchestrator", "planner", "finance"],
    status=ElementStatus.ACTIVE,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# await orchestrator_planner_template.insert()

# Template for the Financial Data Extraction Agent
financial_extractor_template = ElementTemplate(
    name="Agentic_Financial_Data_Extractor",
    description="Extracts specific financial figures from text and returns them as a struc
    tenant_type=TenantType.INVESTMENT_MEMO_AGENTIC_WORKFLOW,
    task_type=TaskType.AGENTIC_WORKFLOW,
    generation_prompt="""
You are a highly accurate data extraction bot. Your task is to read the provided financia
If a figure is not present, use null.

**Financial Period:** {fiscal_period}
**Figures to Extract:** {figures_list}
**Context (Source Text):**
```

```
---
{source_documents}
---

**Output (JSON object with extracted figures):**
""",
    variables=["fiscal_period", "figures_list", "source_documents"],
    execution_config={"model": "gpt-4-turbo", "response_format": {"type": "json_object"}},
    tags=["agent", "extractor", "finance", "kpi"],
    status=ElementStatus.ACTIVE,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# await financial_extractor_template.insert()

# Template for a Synthesis/Writing Agent
financial_writer_template = ElementTemplate(
    name="Agentic_Financial_Analysis_Writer",
    description="Synthesizes structured data (historical figures, calculated ratios) into
    tenant_type=TenantType.INVESTMENT_MEMO_AGENTIC_WORKFLOW,
    task_type=TaskType.AGENTIC_WORKFLOW,
    generation_prompt="""
You are an expert credit analyst. You will be given historical financial data and key calc
Your task is to write the 'Financial Analysis' section of an investment memo.

**Instructions:**
1.  Begin with a high-level summary of the company's financial performance.
2.  Discuss the trends in revenue, profitability, and cash flow, citing the specific data
3.  Analyze the provided financial ratios, explaining what they indicate about the company
4.  Maintain a professional, objective, and analytical tone.
5.  All claims must be supported by the data provided below.

**Company:** {company_name}

**Input Data:**
---
**Historical Figures (JSON):**
{historical_data_json}

**Calculated Ratios (JSON):**
```

```
{calculated_ratios_json}

**Management Commentary (Text):**
{management_commentary_context}
---

**Generated 'Financial Analysis' Section:**
""",
    variables=["company_name", "historical_data_json", "calculated_ratios_json", "managem
    execution_config={"model": "gpt-4-turbo", "temperature": 0.4},
    tags=["agent", "writer", "synthesis", "finance"],
    status=ElementStatus.ACTIVE,
    created_by="AI_SOLUTIONS_ARCHITECT_LLM",
)
# await financial_writer_template.insert()
```

## Section 5: Evaluation Strategy

1. **Human-in-the-Loop Evaluation:**
    - **Methodology:** A senior credit analyst will review the final generated memo using the detailed rubric below. Crucially, the review interface will allow the analyst to inspect the *reasoning chain* for any given claim—seeing which agent generated it, based on what specific retrieved evidence and intermediate calculations. This evaluates both the output and the process.
    - **Evaluation Rubric:** (Adapted from the user's detailed blueprint)
      | Criterion | 1 (Unacceptable) | 3 (Acceptable) | 5 (Exceptional) |
      | :--- | :--- | :--- | :--- |
      | **Factual Accuracy** | Contains significant factual errors or hallucinations. | Mostly accurate, with minor, non-material inaccuracies. | All claims are 100% accurate and verifiably grounded in sources. |
      | **Quantitative Analysis** | Contains material calculation errors or uses inappropriate metrics. | Calculations are correct and metrics are adequate. | Uses the most appropriate ratios and calculations are flawless. |

| **Soundness of Conclusions** | Recommendations are implausible or unsupported by the analysis. | Recommendations are reasonable and flow from the analysis. | Recommendations are highly convincing and directly supported by the evidence. |
| **Traceability & Grounding** | Claims lack citations; the agent's reasoning chain is opaque or illogical. | Most claims are cited; reasoning is generally understandable. | Every key assertion is linked to a verifiable citation; the reasoning chain is transparent and logical. |

2. **LLM-as-a-Judge Evaluation:**
    - **Methodology:** Use multiple, specialized Judge LLMs to evaluate different facets of the agentic process. One judge can evaluate the initial plan's quality, while another can verify the accuracy of the final synthesized text against a collection of all source evidence used throughout the process.
    - **Judge LLM Prompt (for evaluating the Orchestrator's Plan):**

```
You are an expert in financial analysis workflows. Your task is to ev

A high-quality plan is:
- **Comprehensive:** It includes all necessary steps, from data extra
- **Logical:** The dependencies between tasks make sense (e.g., you m
- **Efficient:** It correctly identifies tasks that can be run in par

You will be provided with the initial request and the generated plan
Please evaluate the plan and provide your feedback in the following J

{
  "plan_score": "integer score from 1 (poor) to 5 (excellent)",
  "reasoning": "Explain your score, highlighting strengths and weakne
  "suggested_improvements": ["List any tasks that are missing or any
}

**Initial Request:**
{initial_request}

**Generated Plan (JSON):**
{generated_plan_json}

**Evaluation (JSON Output Only):**
```

- **Sample Code Snippet:**

```python
import openai
import json

client = openai.OpenAI(api_key="YOUR_API_KEY")

def evaluate_agentic_plan(initial_request, generated_plan_json):
    judge_prompt = f"""
    You are an expert in financial analysis workflows... [rest of pro

    **Initial Request:**
    {initial_request}

    **Generated Plan (JSON):**
    {json.dumps(generated_plan_json, indent=2)}

    **Evaluation (JSON Output Only):**
    """

    response = client.chat.completions.create(
        model="gpt-4o",
        messages=[{"role": "user", "content": judge_prompt}],
        temperature=0.0,
        response_format={"type": "json_object"}
    )
    return response.choices[0].message.content
```

## Section 6: User Input

This report provides a technical design and scaffolding for the advanced, agentic AI-powered investment memo generation system described in the user's strategic blueprint. It adopts the recommended Orchestrator-Worker pattern, decomposes the problem into specialized agent tasks, and proposes a sophisticated evaluation framework that assesses not just the final output but the logical soundness of the AI's reasoning process.