Technical Assessment for Gen-AI Intern

Background

Welcome to the technical assessment for the Gen-AI Internship at Traderware. In this assessment, you will demonstrate your ability to create and analyze generative AI solutions using modern frameworks such as LangChain or LangGraph.

Assessment Guidelines

- The deadline for submission is **Friday, May 16th**. If you foresee difficulties meeting this deadline, please inform us promptly.
- Usage of LangChain or LangGraph is mandatory to ensure consistency and reproducibility.
- The evaluation will focus on creativity, analytical thinking, robustness, and practicality of your AI implementations.
- Early submission is recommended, as successful candidates will receive an invitation by **May 19th** for a final one-on-one 15-minute interview.
- The interview period for successful candidates is scheduled from **May 19th to May 30th**.
- The internship commencement date is set for **June 9th**.

Deliverable Format

- Submit one Jupyter notebook (.ipynb) or a `.py` script with an accompanying markdown README.
- Each task must conclude with a brief reflection paragraph addressing:
    - Why I chose this method or approach.
    - What surprised me during development.
    - What my next steps would be given additional time.

- All random seeds must be fixed, ensuring reproducibility upon re-execution.
- Use only public packages including `langchain`, `langgraph`, `sec-parser`, `sec-downloader`, `pandas`, `numpy`, `scipy`, `scikit-learn`, `matplotlib`, or `plotly`.

Task 1: 10-K Retrieval QA

Goal: Build a Retrieval-Augmented Generation (RAG) pipeline to answer questions about the latest 10-K filings for ten companies of your choice.

Steps (required)

Data Ingestion:

- Download the latest 10-K filings for 10 companies using the SEC EDGAR RSS link or `sec-parser` and `sec-downloader` libraries.
- Choose and implement a chunking method suitable for your pipeline.

Vector Store:

- Embed document chunks using an embedding method of your choice.
- Store embeddings in a vector store suitable for your chosen approach.

Chain Creation:

- Develop a RetrievalQA chain using an LLM of your choice.
- Limit total retrieved information to ensure concise responses.

Demo Questions:

- For each company, answer:
    1. "What does [Company] list as its three primary sources of revenue?"

2. "Summarize the biggest risk [Company] cites about supply chain concentration."

Evaluation Criteria:

- Quality and clarity of chunking strategy.
- Cite-aware answers with precise source referencing.
- Effective management of retrieved content ensuring concise responses.
- Reflection on recall vs. precision trade-offs and potential improvements.

Task 2: LangGraph Financial Tool Router

Goal: Create a LangGraph pipeline where an LLM dynamically routes finance-related user queries to appropriate tools.

Steps (required)

Tools to Expose:

- **price_lookup** `(ticker: str) -> str`: Returns the latest stock price using public sources.
- **news_headlines** `(ticker: str, n: int) -> str`: Returns recent news headlines.
- **stat_ratios** `(ticker: str) -> str`: Returns P/E, P/S, ROE ratios.

Graph Topology:

```
Unset
User → LLM-Router → (Tool A | Tool B | Tool C) → Answer-Composer → End
```

- **Router Node:** An LLM call informed by tool descriptions, outputs a dict specifying the

chosen tool and arguments.

- **Tool Node(s):** Executes synchronously, returning string results.
- **Composer Node:** Another LLM call formats tool responses into user-friendly output.
- **Termination:** Single-shot.

Demo Interaction (Sample):

- User: "Give me the P/E ratio for NVDA."
- Assistant: Routes request to `stat_ratios`.

(Interaction can vary based on your chosen data.)

Constraints:

- Proper LangGraph implementation with clear data-flow and robust type hints.
- Graceful error handling for undefined tools.
- Provide downloaded data if sourced externally.

Evaluation Criteria:

- Correct LangGraph implementation.
- Clear, robust routing logic.
- Effective error management.
- Reflection on choices, surprises, and improvements.

Task 3: Automatic Chain Evaluator & Cost Ledger

Goal: Create an evaluation framework that unit-tests a LangChain Chain against predefined golden Q-A pairs, tracking total cost in tokens and USD.

Steps (required)

- Load at least five predefined Q-A test cases (CSV or list of dicts) based on Apple's 10-K from Task 1.
- Run each test through your RetrievalQA chain, computing exact-match F1 scores against provided answers (ignore case/punctuation).
- Track total prompt and completion tokens; calculate USD cost using response metadata (`response.usage`).
- Present a summary table: Question | F1 Score | Cost (cents).
- Raise an `AssertionError` if mean F1 score < 0.6 or total cost exceeds $0.10.

Evaluation Criteria:

- Accurate evaluation methodology.
- Effective token budgeting.
- Robust error handling.
- Reflection on evaluation insights and improvements.
