Appendix

The initial draft of this report was created using (Claude.ai, Sonnet 3.7) with the following prompt:

```
See the following assignment prompt as well as several of these code and
results.



I think the code is mostly done, but feel free to review for any mistakes.

The runs (run1-run4) are results from running the python code and ">>"'ed
onto txt files. Run 1 was failed due to an error, also attached.



create me a readme.md that fulfills the assignment requirements and also a
draft report and analysis. Create the outline based on the assignment,
ensure that it answers everything. Cater it well.



For challenges, some of the stuff I found that is not attached here:

1. Sometimes the LLM would call tools incorrectly, and this is also seen
on the third tool having dummty input parameter to make sure the tool
work. Other times, it may input "$AAPL" instead of just "AAPL"

2. It seems Llama 70b rarely do that, but openai mini and llama 8b is no
stranger

3. Therefore, either we are using suboptimal langchain library (there amy
be better otions to use, e.g. not zero shot) but also its best practice to
create more robust tools.



Remember: the north star is the assignment prompt and rubric.
```

```
Use web search as needed
```

I also attached the files in `financial_agent.py` and `runs/`

To which it produces the following output:

```
# Technical Report: AI-Powered Financial Portfolio Rebalancer



## 1. Introduction



This report provides a comprehensive analysis of the AI-Powered Financial
Portfolio Rebalancer, a system designed to assist investors in maintaining
balanced portfolios. The implementation leverages LangChain's agent
framework to integrate multiple Large Language Models (LLMs) and compare
their effectiveness for financial analysis tasks.



The system was built to perform three primary functions:

1. Retrieve real-time stock prices

2. Analyze portfolio balance based on an equal-weight strategy

3. Provide market trend insights



## 2. Implementation Details



### 2.1 Core Components



The implementation consists of three main tools that work together to
```

~~~
provide financial analysis capabilities:



#### Stock Price Lookup Tool

This tool fetches the latest stock price data using the Yahoo Finance API.
It returns comprehensive information including:

- Current price

- Previous close price

- Daily change (absolute and percentage)

- Timestamp of the data



The implementation includes data cleaning to handle potential formatting
issues with ticker symbols (removing '$' symbols or quotes). The tool
returns structured data in a dictionary format for easy consumption by the
agent.



```python

def get_stock_price(symbol):

try:

# data cleansing just in case.

symbol = symbol.replace("$", "").replace("'", "")

# Create a Ticker object

ticker = yf.Ticker(symbol)



# Get the latest stock data
~~~

```
hist = ticker.history(period="1d")



if hist.empty:

return f"No data found for symbol {symbol}"



# Get info for additional data

info = ticker.fast_info



# Extract relevant information

current_price = info.last_price if hasattr(

info, 'last_price') else hist['Close'].iloc[-1]

previous_close = info.previous_close if hasattr(

info, 'previous_close') else None



# Calculate daily change if possible

if previous_close:

daily_change = current_price - previous_close

daily_change_percent = (daily_change / previous_close) * 100

else:

daily_change = None

daily_change_percent = None
```

```
# Return the data in a structured format

result = {

"symbol": symbol,

"current_price": current_price,

"previous_close": previous_close,

"daily_change": daily_change,

"daily_change_percent": daily_change_percent,

"timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")

}



return result



except Exception as e:

return f"Error retrieving stock price for {symbol}: {str(e)}"



#### Portfolio Rebalancing Tool

This tool analyzes a given portfolio against an equal-weight strategy and
suggests necessary adjustments. It:

- Parses the portfolio input (handling both string and dictionary formats)

- Calculates target weights based on the number of assets

- Determines necessary buy/sell actions for each asset
- Returns detailed recommendations in a structured format



The tool implements robust error handling and formatting to ensure
consistent results regardless of input format variations.



#### Market Trend Analysis Tool

This tool analyzes the S&P 500 (using SPY as a proxy) to provide insights
on market trends over the past week. It calculates:

- 5-day returns

- Volatility (standard deviation of daily returns)

- Market trend characterization (strongly bullish, mildly bullish, mildly
bearish, strongly bearish)

- Start and current prices



A notable implementation detail is the use of a dummy input parameter to
ensure compatibility with the LangChain agent framework, which requires
all tools to accept at least one parameter.



### 2.2 LLM Integration



Three different LLM configurations were implemented:



1. **OpenAI GPT-4o-mini**

- Used as the default option
```

- Implemented via `langchain_openai.ChatOpenAI`

2. **Groq LLaMA3-8B**

- Implemented via `langchain_groq.ChatGroq`

- Using model "llama3-8b-8192"

3. **Groq LLaMA3-70B**

- Implemented via `langchain_groq.ChatGroq`

- Using model "llama3-70b-8192"

All models were configured with a temperature of 0 to maximize deterministic, factual responses.

The LangChain agent was configured using the `AgentType.ZERO_SHOT_REACT_DESCRIPTION` framework, which allows the LLM to determine which tool to use based on the query without requiring examples.

## 3. Challenges Encountered and Solutions

Several significant challenges were encountered during development:

### 3.1 LLM Output Parsing and Tool Selection

```
One of the most persistent challenges was ensuring that LLMs correctly
called tools with valid parameters. This manifested in several ways:



1. **Inconsistent Input Formatting**: The LLMs, particularly LLaMA3-8B and
GPT-4o-mini, sometimes included extraneous characters in stock symbols
(e.g., '$AAPL' instead of 'AAPL').

- **Solution**: Implemented robust input cleaning in the
`get_stock_price()` function to remove common extraneous characters.



2. **Invalid Tool Calls**: The LLMs occasionally attempted invalid actions
or provided incorrectly formatted parameters.

- **Solution**: Added a dummy parameter to the Market Trend Analysis tool
to ensure compatibility with the agent framework's requirements.



3. **Output Parsing Errors**: As seen in the error logs, LLaMA models
sometimes generated output formats that LangChain's parser couldn't
interpret correctly:
```

langchain_core.exceptions.OutputParserException: Could not parse LLM output: `Thought: I have all the current prices of the stocks in the portfolio. Now, I need to recalculate the weights based on the current prices.

Action: (No action needed, just calculation)

```
- **Solution**: A more robust approach would be to use
`handle_parsing_errors=True` in the AgentExecutor configuration, though
this wasn't implemented in the current version.



### 3.2 API Rate Limiting and Reliability
```

Working with multiple external APIs introduced several stability challenges:

1. **API Rate Limiting**: The Groq API occasionally returned 429 "Too Many Requests" errors during testing:

httpx.HTTPStatusError: Client error '429 Too Many Requests' for url 'https://api.groq.com/openai/v1/chat/completions'

- **Solution**: Implementing exponential backoff and retry logic would improve stability, though this wasn't included in the current implementation.

2. **Yahoo Finance API Reliability**: The Yahoo Finance API occasionally returned 404 errors when ticker symbols were incorrectly formatted:

404 Client Error: Not Found for url: https://query2.finance.yahoo.com/v10/finance/quoteSummary/'MSFT'?...

- **Solution**: The input cleaning implementation helped address this issue, but further robust error handling could improve reliability.

### 3.3 LangChain Deprecation Warnings

The implementation triggered several deprecation warnings from LangChain:

```
LangChainDeprecationWarning: LangChain agents will continue to be
supported, but it is recommended for new use cases to be built with
LangGraph.



LangChainDeprecationWarning: The method `Chain.run` was deprecated in
langchain 0.1.0 and will be removed in 1.0.



A future iteration could address these by transitioning to LangGraph for
agent implementation and using `.invoke()` instead of `.run()`.


## 4. Analysis of LLM Performance


### 4.1 Comparative Metrics


Based on the test runs, the following performance metrics were observed:


| Metric | OpenAI GPT-4o-mini | Groq LLaMA3-8B | Groq LLaMA3-70B |

|--------|-------------------|----------------|-----------------|

| Response Time (avg) | 15.98s | 4.01s | 71.49s |

| Response Accuracy | High | Medium | High |

| Tool Selection Efficiency | High | Medium | High |
| Market Analysis Depth | Medium | Low | High |

| Reasoning Capability | Medium | Low | High |



### 4.2 Performance Analysis



#### OpenAI GPT-4o-mini

**Strengths:**

- Reliably identified when portfolios were already balanced

- Provided detailed market context for recommendations

- Consistently selected appropriate tools for each query

- Produced concise, actionable recommendations



**Weaknesses:**

- Occasionally failed to retrieve stock data (as seen in run4.txt)

- Sometimes performed redundant analyses even when portfolios were
balanced

- Medium response times (averaging 15.98 seconds)



**Notable Behavior:**

In one instance (run4.txt), GPT-4o-mini was unable to retrieve stock data
but still provided a correct recommendation based on the portfolio
weights:


It seems that I am unable to retrieve stock prices for any of the stocks
```

```
in the portfolio. However, I can still analyze the portfolio based on the
weights provided. The current weights are AAPL (50%), TSLA (30%), and
GOOGL (20%), which are clearly imbalanced compared to an equal-weight
strategy of approximately 33.33% for each stock.



#### Groq LLaMA3-8B

**Strengths:**

- Extremely fast response times (averaging 4.01 seconds)

- Provided simple, clear recommendations

- Minimal computational overhead



**Weaknesses:**

- Less thorough analysis compared to other models

- Occasionally struggled with tool selection

- Sometimes attempted invalid actions (e.g., "Action: None")

- Limited market context in recommendations



**Notable Behavior:**

LLaMA3-8B consistently provided the most concise responses, focusing
exclusively on the rebalancing task without additional market context or
detailed analysis.



#### Groq LLaMA3-70B

**Strengths:**
```

```
- Most comprehensive analysis of all models

- Consistently retrieved current prices for all stocks

- Provided detailed, actionable recommendations with price context

- Included relevant market trend insights in recommendations



**Weaknesses:**

- Extremely slow response times (averaging 71.49 seconds)

- Highest computational cost

- Occasional API rate limit issues



**Notable Behavior:**

LLaMA3-70B consistently performed the most thorough analysis, retrieving
current prices for all stocks even when the portfolio was already
balanced, and providing detailed market context:


Based on the analysis, the portfolio is already balanced according to the
equal-weight strategy, and no rebalancing actions are necessary. However,
considering the current market trends, which are strongly bearish, it may
be a good idea for the user to consider diversifying their portfolio or
adjusting their investment strategy to mitigate potential losses.




## 5. Recommendations for LLM Selection



Based on the comparative analysis, the following recommendations can be
made for different financial analysis scenarios:
```

### 5.1 For Quick Portfolio Checks

Recommended LLM: Groq LLaMA3-8B

The 8B model's extremely fast response time makes it ideal for quick portfolio balance checks where detailed analysis isn't required. It provides adequate results for simple rebalancing tasks with minimal computational overhead, making it cost-effective for high-volume, low-complexity scenarios.

### 5.2 For Comprehensive Financial Analysis

Recommended LLM: Groq LLaMA3-70B

For detailed financial analysis that requires nuanced understanding and comprehensive recommendations, the 70B model provides the most thorough responses. Its ability to analyze market trends in conjunction with portfolio data makes it suitable for more complex financial decision-making, despite its significantly longer processing time.

### 5.3 For Balanced Performance

Recommended LLM: OpenAI GPT-4o-mini

For users needing a balance between speed and quality, the GPT-4o-mini model offers the best compromise. It provides detailed and accurate recommendations with reasonable response times, making it suitable for most everyday portfolio management tasks.

```
## 6. Future Improvements



Several enhancements could improve the system's functionality and
reliability:



1. **Implement LangGraph**: Transition from the deprecated LangChain agent
framework to LangGraph for improved flexibility and error handling.



2. **Add Retry Logic**: Implement exponential backoff and retry mechanisms
for API calls to handle rate limiting and temporary failures.



3. **Expand Rebalancing Strategies**: Currently, the system only supports
equal-weight rebalancing. Adding support for other strategies (e.g.,
market-cap weighted, risk-parity) would enhance flexibility.



4. **Implement Portfolio Visualization**: Adding data visualization
capabilities would improve the interpretability of recommendations.



5. **Enhanced Error Handling**: Implementing more robust error handling,
particularly for LLM output parsing, would improve stability.



6. **Input Validation Layer**: Adding a dedicated input validation layer
before processing would reduce errors related to incorrectly formatted
inputs.



7. **Cost Optimization**: Implementing a tiered approach that uses smaller
```

models for simple tasks and larger models only when needed could optimize computational costs.

## 7. Conclusion

The AI-Powered Financial Portfolio Rebalancer demonstrates the potential of LLM-based financial tools to assist investors with portfolio management. Each LLM configuration offers distinct advantages in terms of speed, accuracy, and depth of analysis, allowing for flexibility in different use cases.

While the GPT-4o-mini model offers the best balance of performance characteristics for general use, the availability of faster (LLaMA3-8B) and more comprehensive (LLaMA3-70B) options provides valuable alternatives for specialized scenarios. The implementation challenges highlight the importance of robust error handling and API management when developing LLM-based financial tools.

Overall, the system successfully demonstrates how LangChain's agent framework can be leveraged to create intelligent financial assistants capable of providing practical portfolio management recommendations.