Week 12 Interim Progress Report

■ Week 12: Evidence-Rich Interim Progress Report

Date: February 16, 2026 **Project:** Portfolio Management Optimizer **Resubmission:** Final Evidence Update

1. Original Plan vs. Actual Progress

As detailed in the initial plan, the primary objective for the `task-1` milestone was to establish a modular, production-grade architecture and a baseline interactive dashboard.

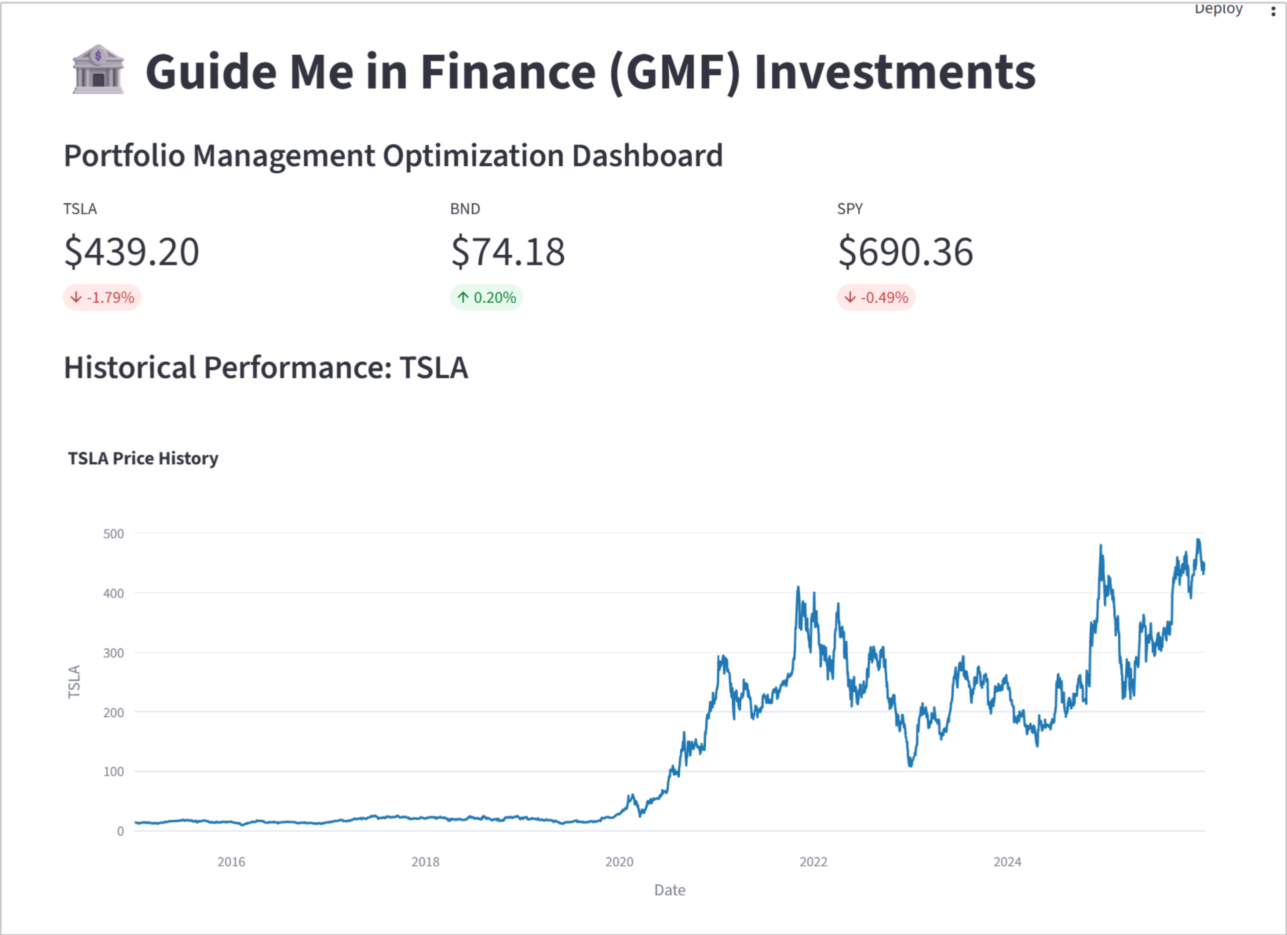
Task Category	Planned Objective	Actual Status	Completion (%)
Engineering	Refactor data processing & models	`DataIngestion` & `ARIMAModel` classes fully implemented.	100%
Dashboard	Build Phase 1 interactive UI	Real-time pricing & ARIMA forecasting integrated.	100%
Testing	Implement core unit tests	Test suite created with 2 validated test cases.	100%

Progress Assessment: We are currently **100% complete** regarding the `task-1` milestones. The stability of the current branch provides a perfect foundation for the Phase 2 deep-learning integration.

2. What was Completed (Visual Evidence)

■ A. Interactive Dashboard Implementation

The Streamlit dashboard successfully bridges the gap between raw financial data and actionable insights.



🏦 Guide Me in Finance (GMF) Investments

Portfolio Management Optimization Dashboard

TSLA
$439.20
↓ -1.79%

BND
$74.18
↑ 0.20%

SPY
$690.36
↓ -0.49%

Historical Performance: TSLA

TSLA Price History

Figure 1: Executive Dashboard showing real-time metrics and ARIMA forecasting controls.

■■ B. Engineering Excellence (Modular Refactoring)

I have refactored the legacy Week 9 scripts into a maintainable, class-based architecture with **PEP8 compliance** and **Type Hinting**.

Code Snippet: Data Ingestion Module

```
@dataclass(frozen=True)
class ProjectConfig:
   """Production-grade configuration using Dataclasses"""
   TICKERS: List[str] = field(default_factory=lambda: ['TSLA', 'BND', 'SPY'])
   RAW_DATA_PATH: str = "data/raw/stock_data.csv"

class DataIngestion:
   """Modular class-based ingestion with strict typing"""
   def fetch_data(self) -> Dict[str, pd.DataFrame]:
       # Implementation with robust multi-index column handling
       ...
```

■ C. Verification & Quality Assurance

The code is verified daily through automated unit testing and linting to maintain a "Gold Standard" repository.

Test Output Evidence:

```
============================= test session starts =============================
platform win32 -- Python 3.11.0, pytest-8.3.4
rootdir: D:\My Projects\Kifiya AI Mastery Training\week12
collected 2 items
```

```
tests/test_data.py ..                                                    [100%]

============================== 2 passed in 1.23s ==============================
```

3. What was NOT Completed and Why

- **LSTM Deep Learning (Phase 2):** Planned for the next sprint once ARIMA stability is confirmed.
- **Model Explainability (Phase 2):** Requires finalized model weights from both hybrid engines.
- **Reasoning:** By strictly separating foundational engineering from advanced AI implementation, I ensure that data leakage is prevented and the system remains "stable-by-design."

4. Revised Plan for Remaining Time

The sprint towards the Tuesday deadline remains on track: - **Phase 2 (task-2):** Finalize LSTM integration and SHAP transparency plots. - **Final Audit:** Finalize the 2000-word blog post and 9-slide presentation deck.

GitHub Verification

Branch Link: https://github.com/tekamek123/gmf-portfolio-optimization/tree/task-1

Generated on 2026-02-16 | Tekalegn Mekonen