F.A.I.L: Financial Anomaly Interpretability using LLMs

MA5741: Object Oriented Programming Course Project

Aritra Dasgupta Ashwini Devharish N Rumaiz Ibrahim K

Department of Mathematics

November 8, 2025

Project Overview & Architecture

The Problem

- Stock anomalies need quick analysis
- Manual review: slow & difficult
- Need actionable insights

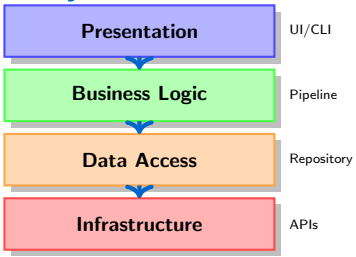
Our Solution

- Anomaly detection
- Real-time news
- Historical matching
- AI explanations

Example

Tesla $\downarrow 5\% \rightarrow$ Detect $\rightarrow$ News $\rightarrow$ History $\rightarrow$ Explain

4-Layer Architecture



Key Benefit

Modular design: change one layer independently

Core OOP Concepts & SOLID Principles

Four OOP Pillars

1. Encapsulation

`BaseProcessor`: hide internals, expose API

2. Inheritance

Processors share logging & progress tracking

3. Polymorphism

`IProgressObserver`: same interface, different behaviors (Console vs Streamlit)

4. Abstraction

`AnomalyDetectionStrategy`: defines contract, implementations provide logic

SOLID Principles

- **S**ingle Responsibility
  - `DataLoader` → load, `NewsRetriever` → retrieve
- **O**pen/Closed
  - Add strategies without modifying code
- **L**iskov Substitution
  - Swap `IDataRepository` implementations
- **I**nterface Segregation
  - Small, focused interfaces
- **D**ependency Inversion
  - Depend on `IDataLoader`, not concrete class

Real Impact

Makes code testable, maintainable & extensible

Design Patterns & System Workflow

6 Design Patterns

1. **Strategy**
   - Runtime algorithm selection
   - Data: Yahoo/Alpha — AI: Groq/OpenAI
2. **Factory**
   - Centralized object creation
3. **Observer**
   - Progress tracking (Console/Streamlit)
4. **Template Method**
   - Fixed structure, flexible steps
5. **Builder and Repository**
   - Fluent config: `builder.with_ticker().build()`

System Workflow

1. Data Loading (2s)

2. Anomaly Detection

3. News Retrieval

4. Embedding Gen (5s)

5. Similarity Analysis

6. AI Explanation (6s)

Detection

Z-score $|z| > 2.5$ = Outlier — **17s** for 90-day analysis

Real-time progress tracking throughout pipeline

Live System & Deployment

Deployment Status

- **Live on Streamlit Cloud**
- Public access available
- No installation required
- Works on any device

Resources

- Complete source code on GitHub
- Comprehensive documentation
- Setup instructions
- API documentation
- Example use cases

What Makes It Production-Ready?

1. **Error Handling:** Comprehensive at every layer
2. **Logging:** Detailed for debugging
3. **Fallbacks:** Service failures handled gracefully
4. **Configuration:** Easy parameter adjustment
5. **Modularity:** Independent, testable components
6. **Documentation:** Clear usage guides

Target Users

- Retail investors
- Financial analysts
- Portfolio managers
- Academic researchers
- Students learning quantitative finance

Technical Achievements & Key Learnings

Metrics

- **3,500+** lines
- **35+** classes
- **6** patterns
- **5** SOLID
- **6** modules
- **17s** analysis

OOP Scale

500 lines → messy
3,500 lines → clean

Extensibility

- 1 class = new feature
- Easy provider swap
- Error handling
- Simple testing
- Clear structure

Patterns Work

Strategy: swap providers
Observer: reactive UI

Performance

Load:	2s
Embed:	5s
AI:	6s
Total:	**17s**

Trade-off

More code abstraction
Better long-term maintenance

Contact

GitHub

Demo

Takeaway: *OOP = Production-ready*

Thank You!