High Frequency Trading Application

Patrick Flanigan
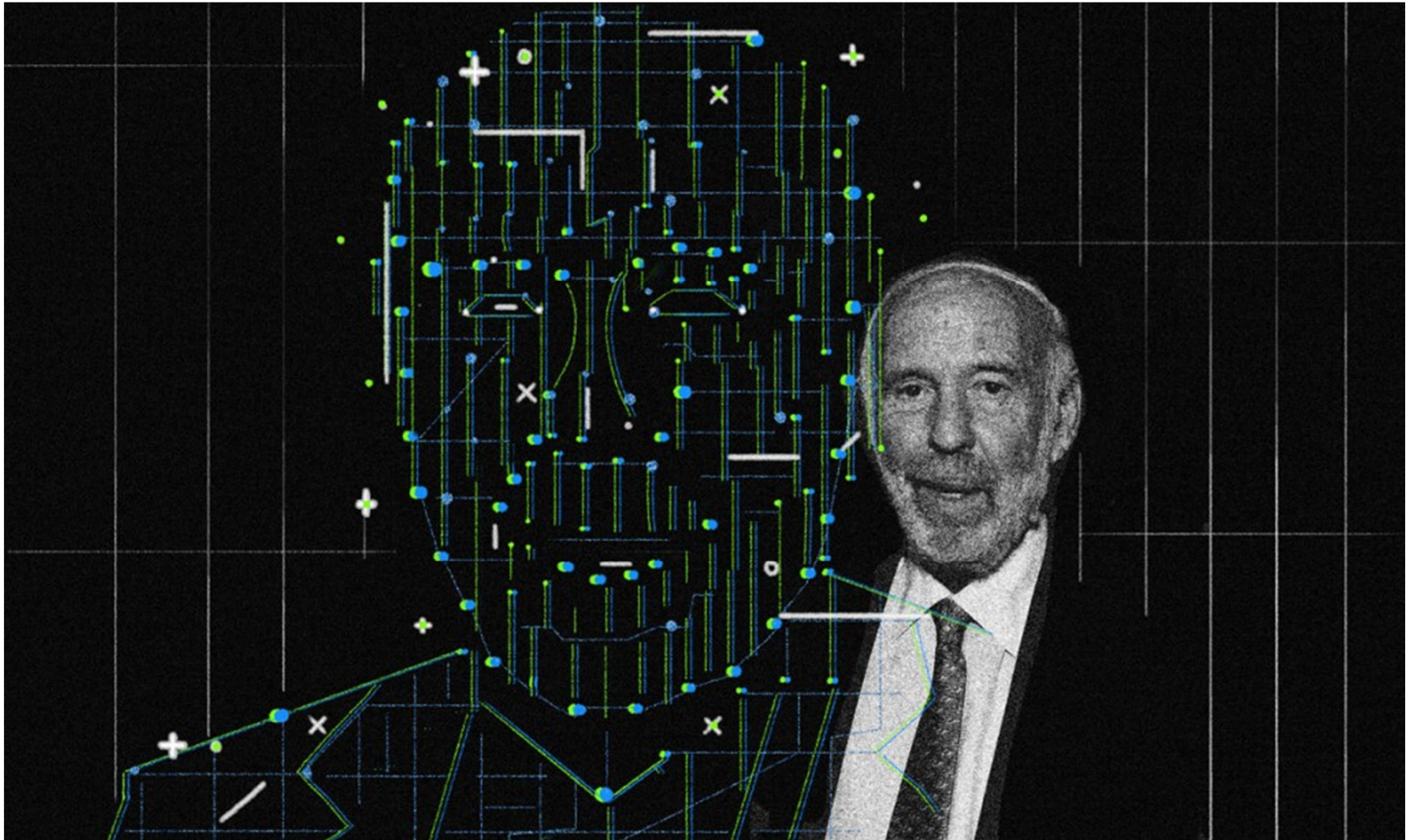
SWENG 837

What is high frequency trading?
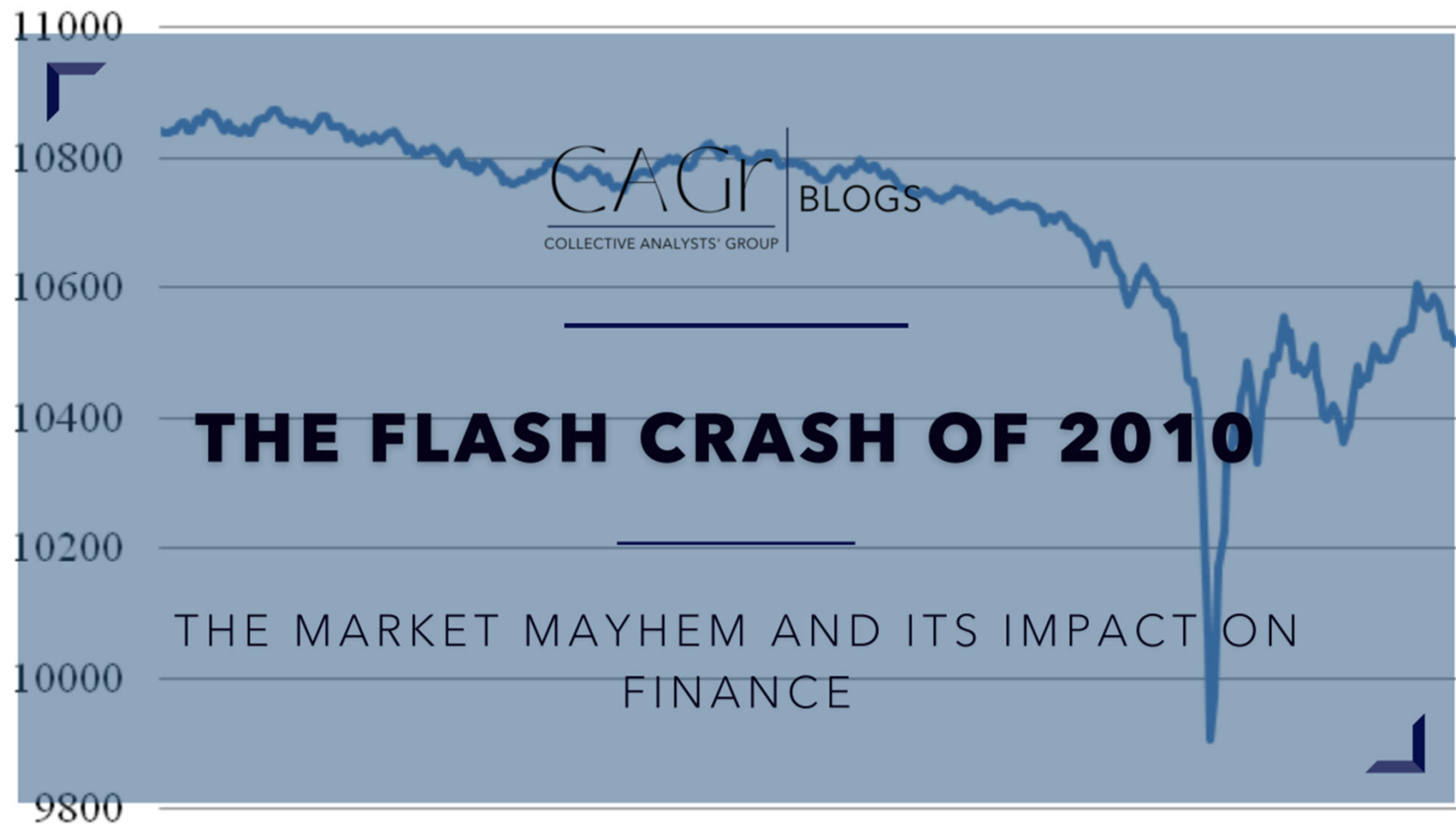

ProfitView 2024

Implemented high frequency trading

A "flash" crash

Problem Statement and Requirements

The primary objective of this system is to address the need for high-frequency trading by providing robust platform to train , host and execute trainable market data algorithms for high frequency trading.

Business Requirements:

- The system must support rapid order management, seamless market data integration, quick trade execution and advanced portfolio management.

- **Target Users:**
  – Individual traders who need quick trade execution with access to real-time data.
  – Institutional traders requiring scalable systems to handle high trade volumes with integrated risk management.
  – Compliance officers who need to monitor and ensure trading activities comply with regulations.

- **Business Goals:**
  – Increase trade volume by reducing latency.
  – Improve decision-making through analytics.
  – Enhance user satisfaction with reliable performance.

Non-functional Requirements

- **Performance:** The system must be scalable, with millisecond-level response times and the ability to handle high volumes of trades.

- **Security:**
  - **Authentication**: Provide for scalable user authentication.
  - **Authorization**: Implement role-based access control.
  - **Data Encryption**: Ensure data protection, encrypting data both at rest and in transit.
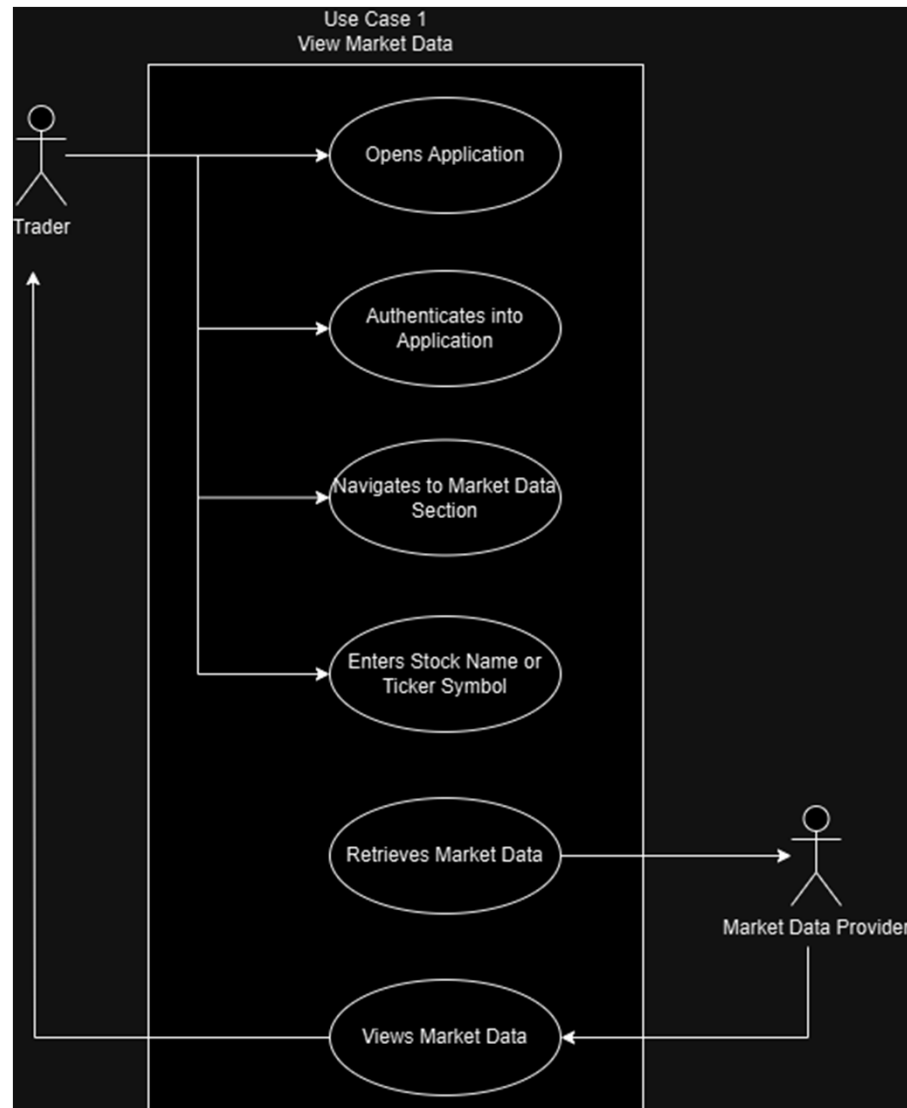
Non-functional Requirements cont'd

- **Maintainability:**
  - **Database Normalization**: Optimize database structure for efficiency and scalability.
  - **Modularization**: Ensure clear separation of concerns within the system.
  - **Non-Overlapping Functions**: Design with distinct, non-redundant components.
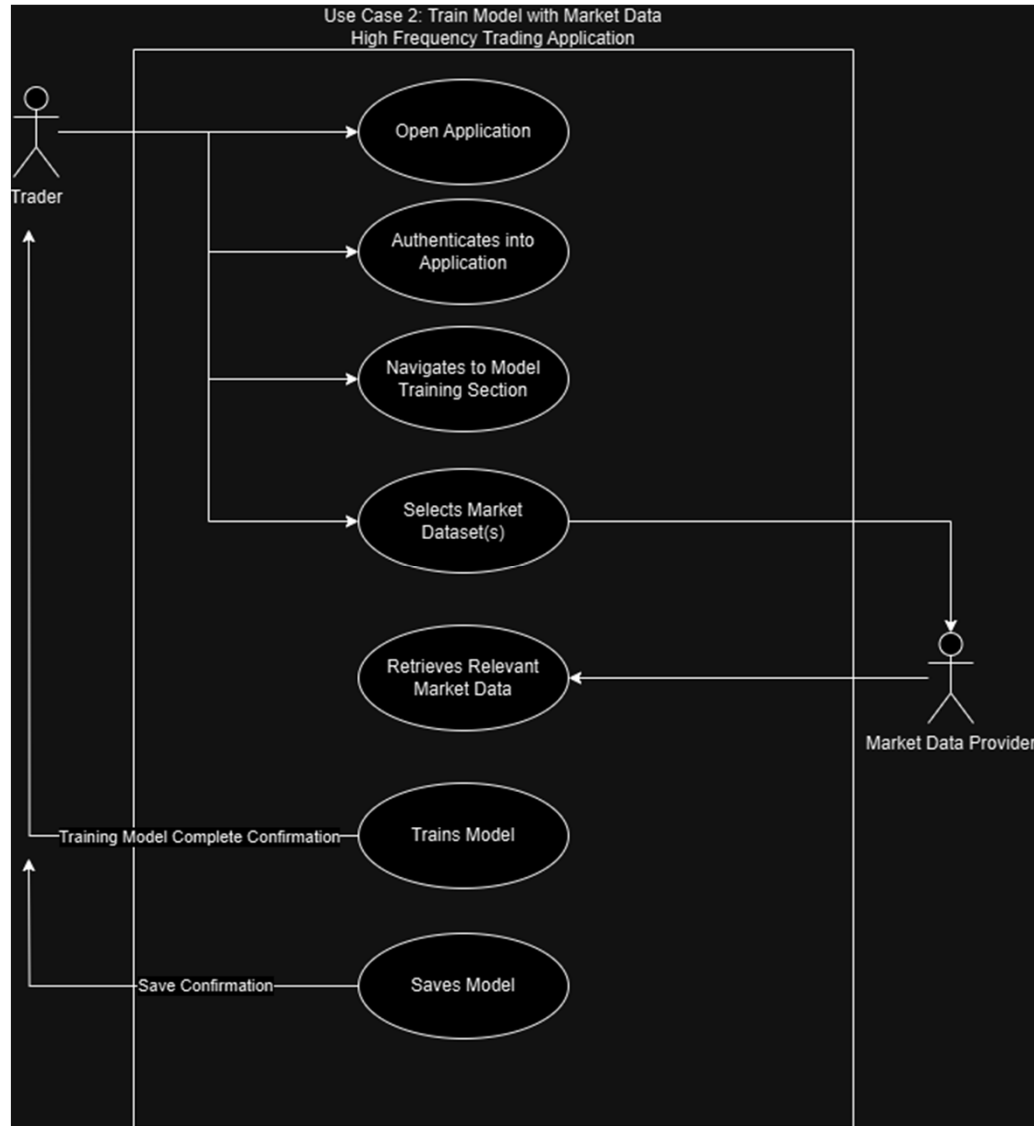
Non-functional Requirements cont'd

- **Other Considerations:**
  - **Compliance**: Leverage technologies for adherence to financial regulations.
  - **High Availability**: Ensure redundancy and failover.
  - **Reliability**: Utilize auto scaling and load balancing for consistent performance.
  - **Usability**: Optimize user experience via services tailored to reduce latency.
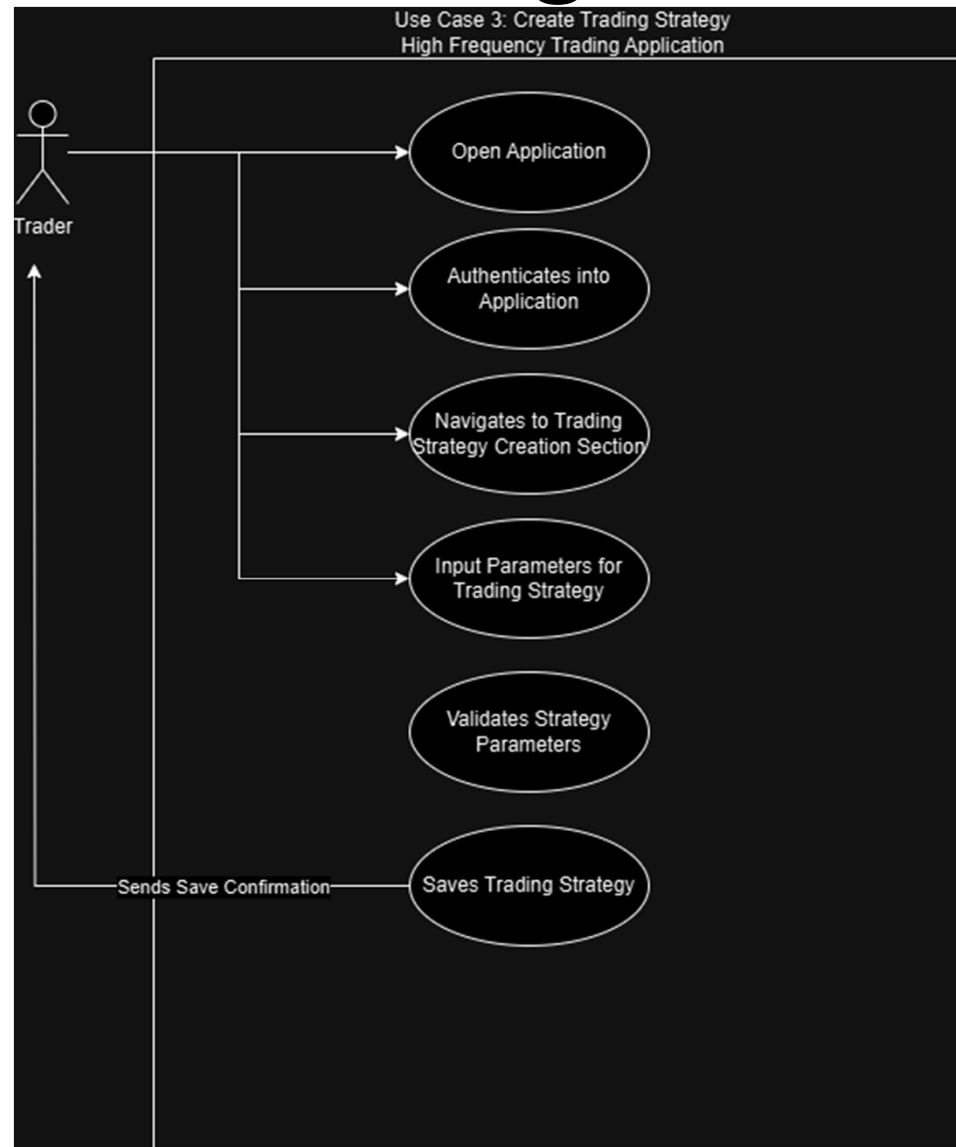
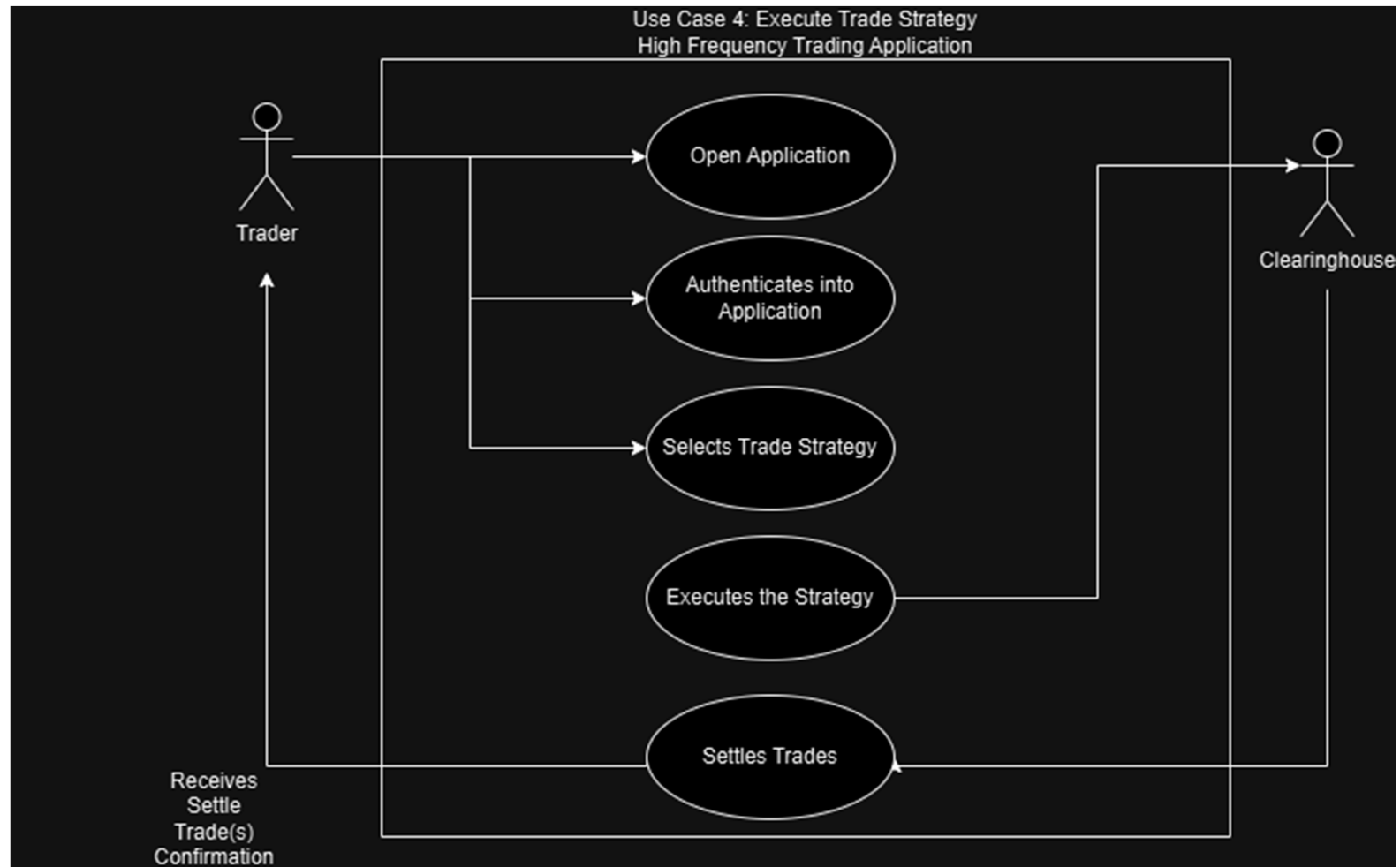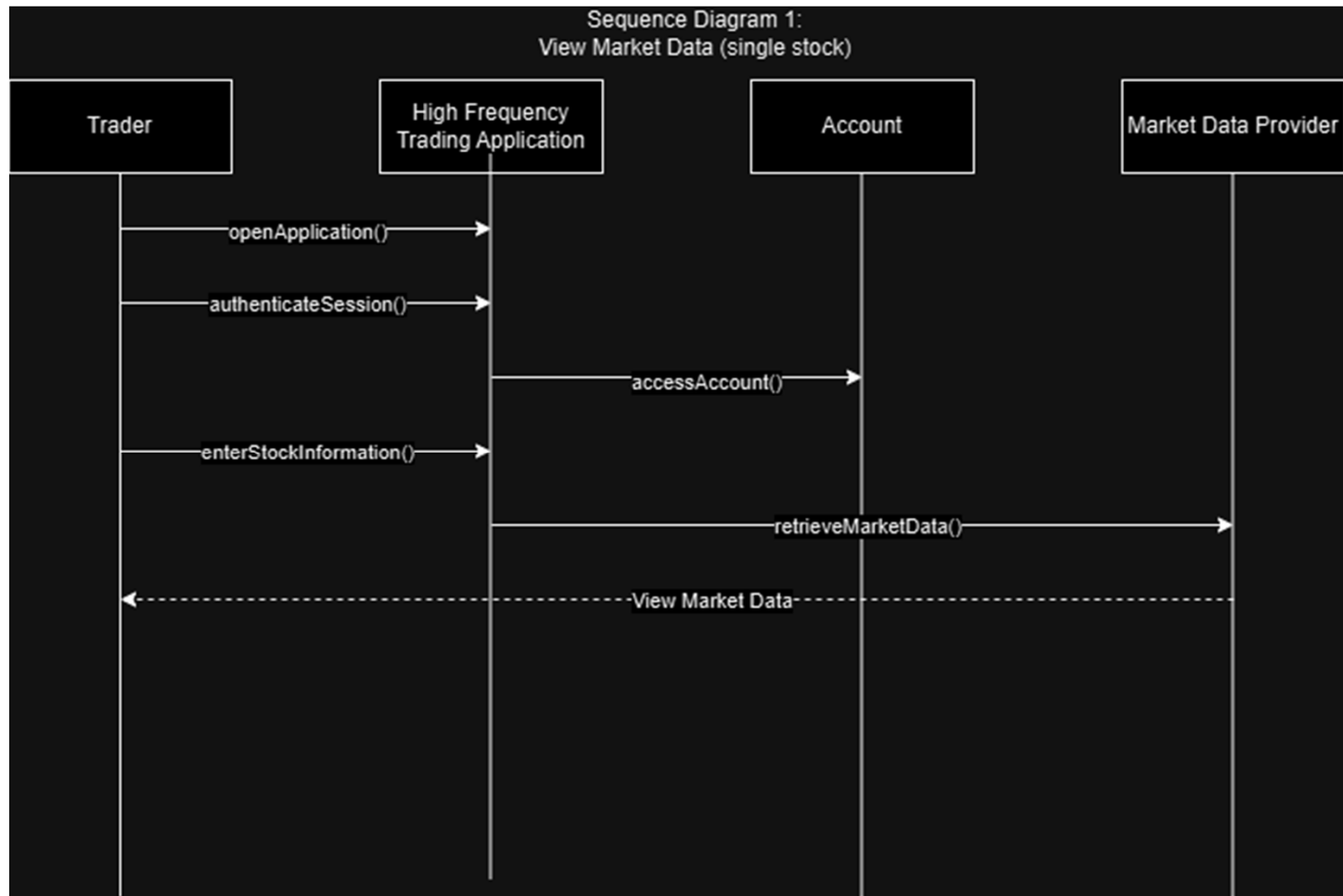# Use Case Diagram

Use Case 1
View Market Data

Trader

- Opens Application
- Authenticates into Application
- Navigates to Market Data Section
- Enters Stock Name or Ticker Symbol
- Retrieves Market Data
- Views Market Data

Market Data Provider

Use Case Diagram cont'd

Use Case 2: Train Model with Market Data
High Frequency Trading Application

Trader

Open Application

Authenticates into Application

Navigates to Model Training Section

Selects Market Dataset(s)

Retrieves Relevant Market Data

Market Data Provider

Training Model Complete Confirmation — Trains Model

Save Confirmation — Saves Model

Use Case Diagram cont'd

Use Case 3: Create Trading Strategy
High Frequency Trading Application

Trader

- Open Application
- Authenticates into Application
- Navigates to Trading Strategy Creation Section
- Input Parameters for Trading Strategy
- Validates Strategy Parameters
- Saves Trading Strategy

Sends Save Confirmation

Use Case Diagram cont'd

Sequence Diagram

Sequence Diagram 1:
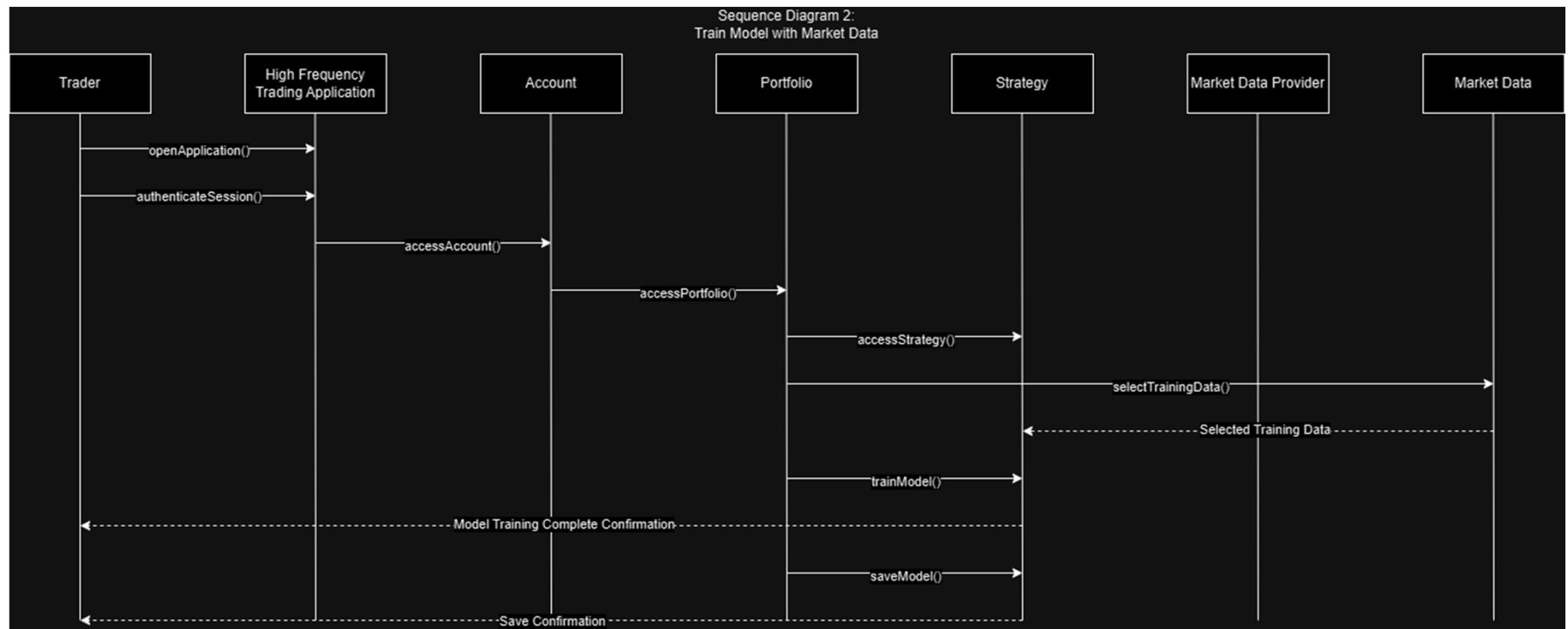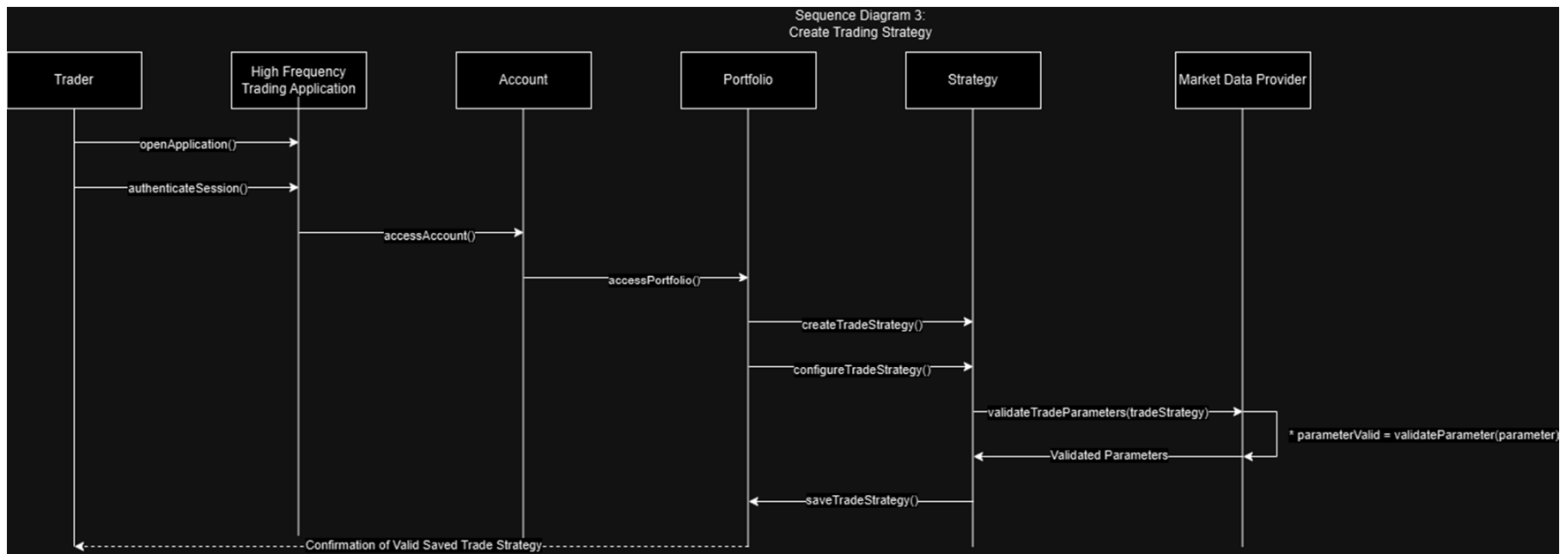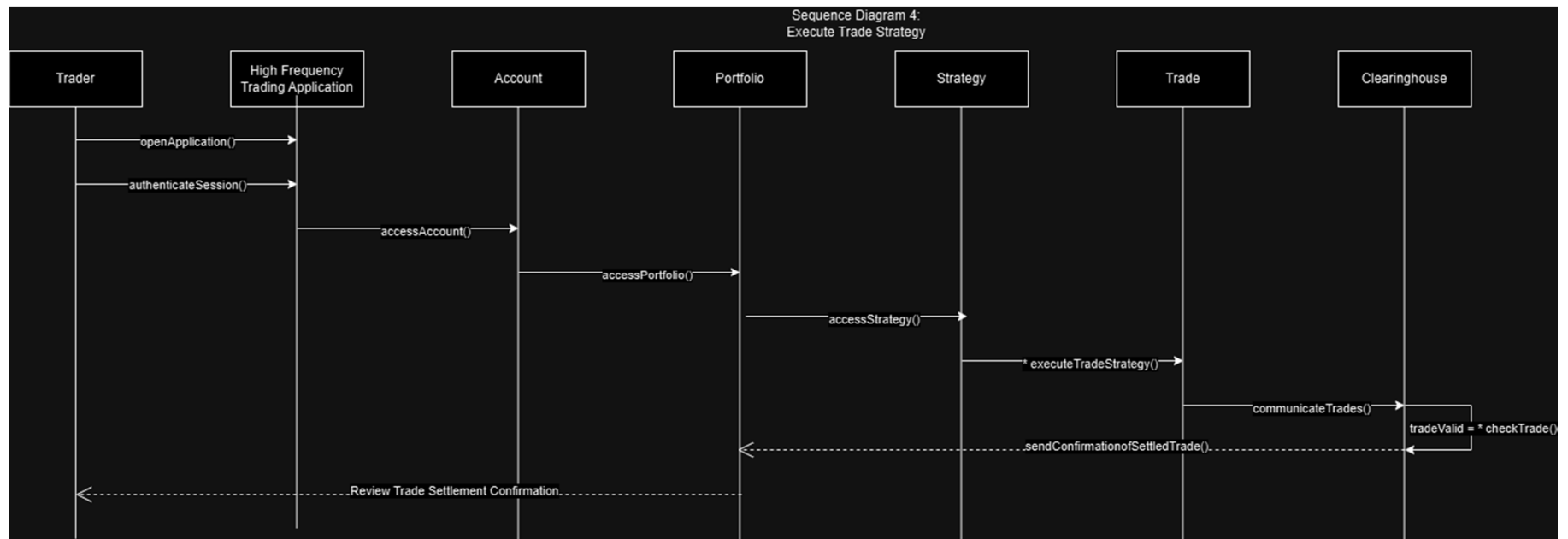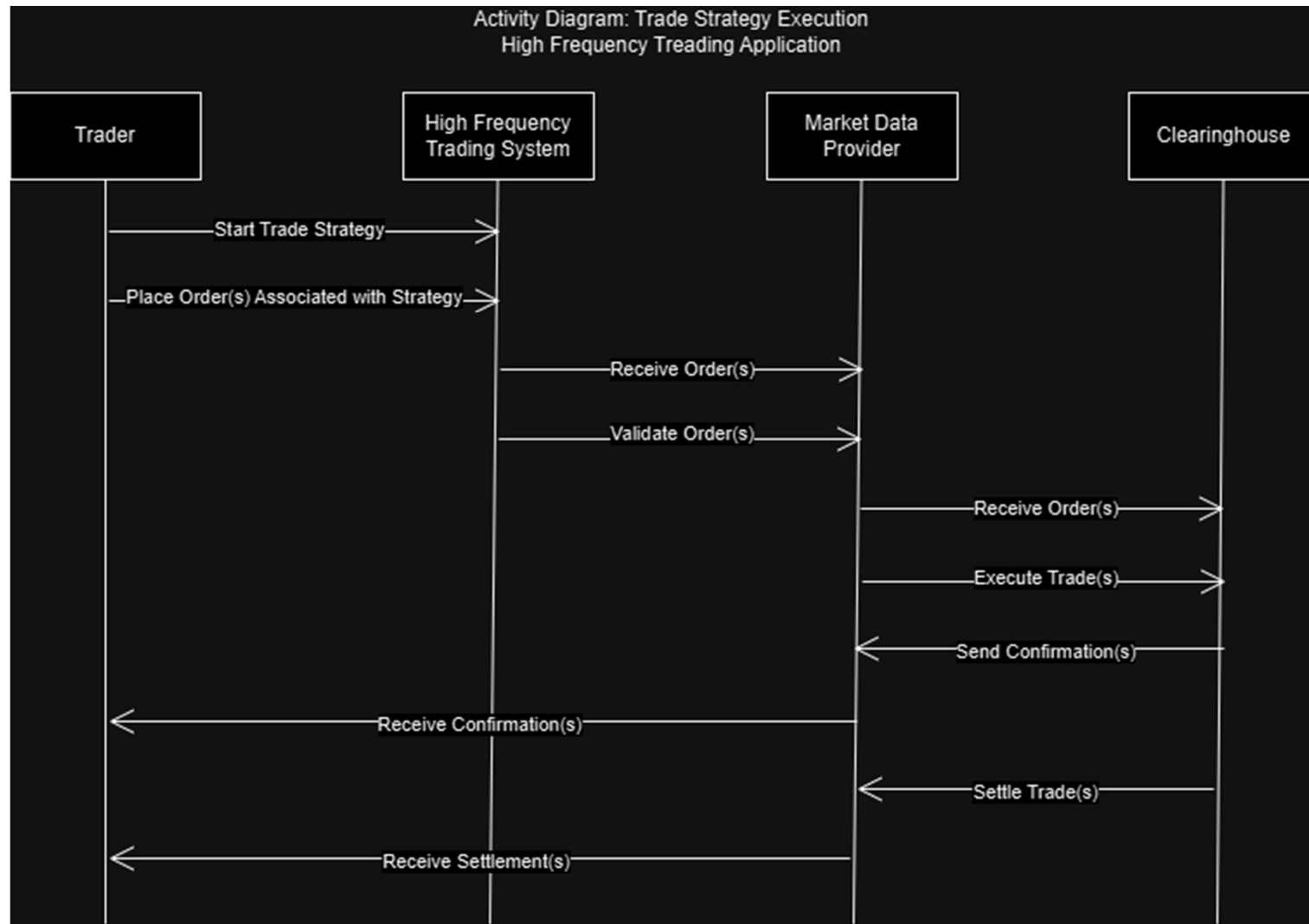View Market Data (single stock)

Sequence Diagram cont'd
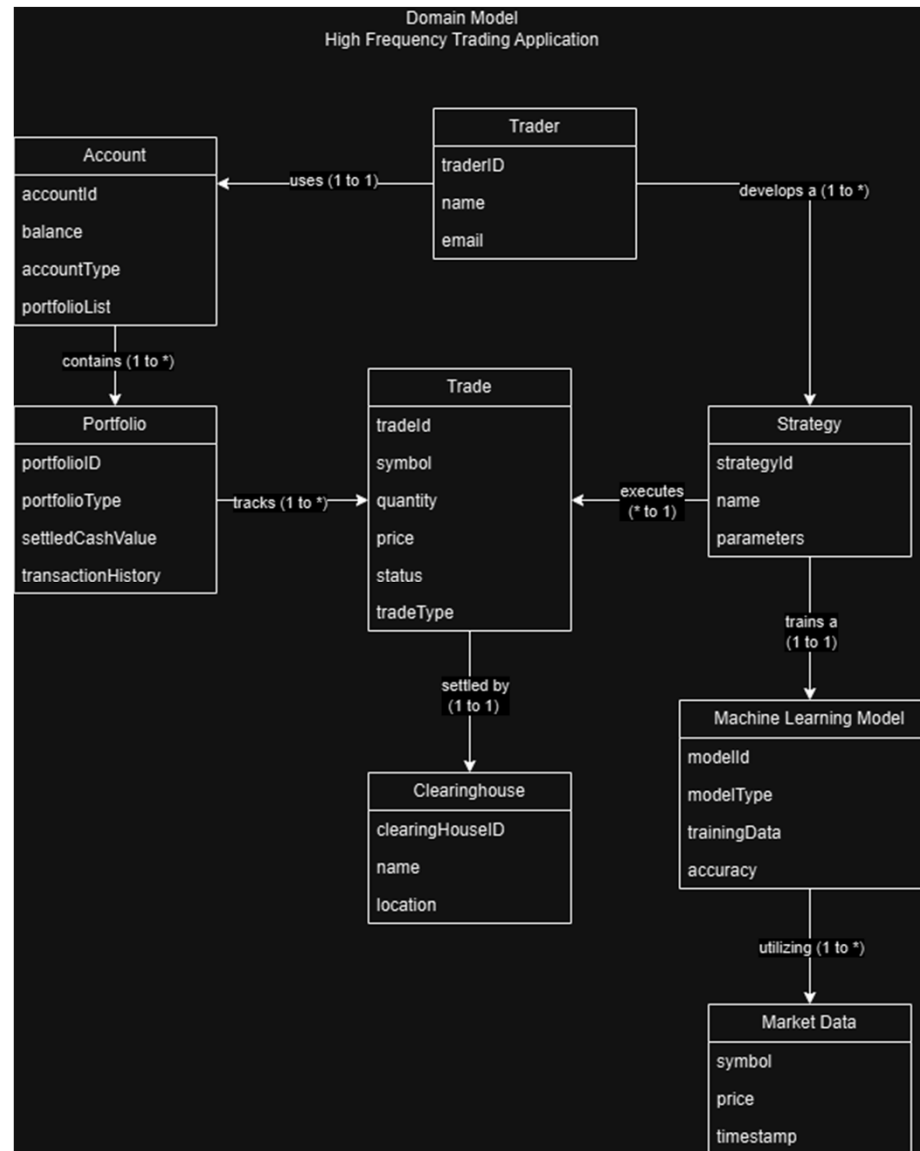
Sequence Diagram cont'd

Sequence Diagram cont'd
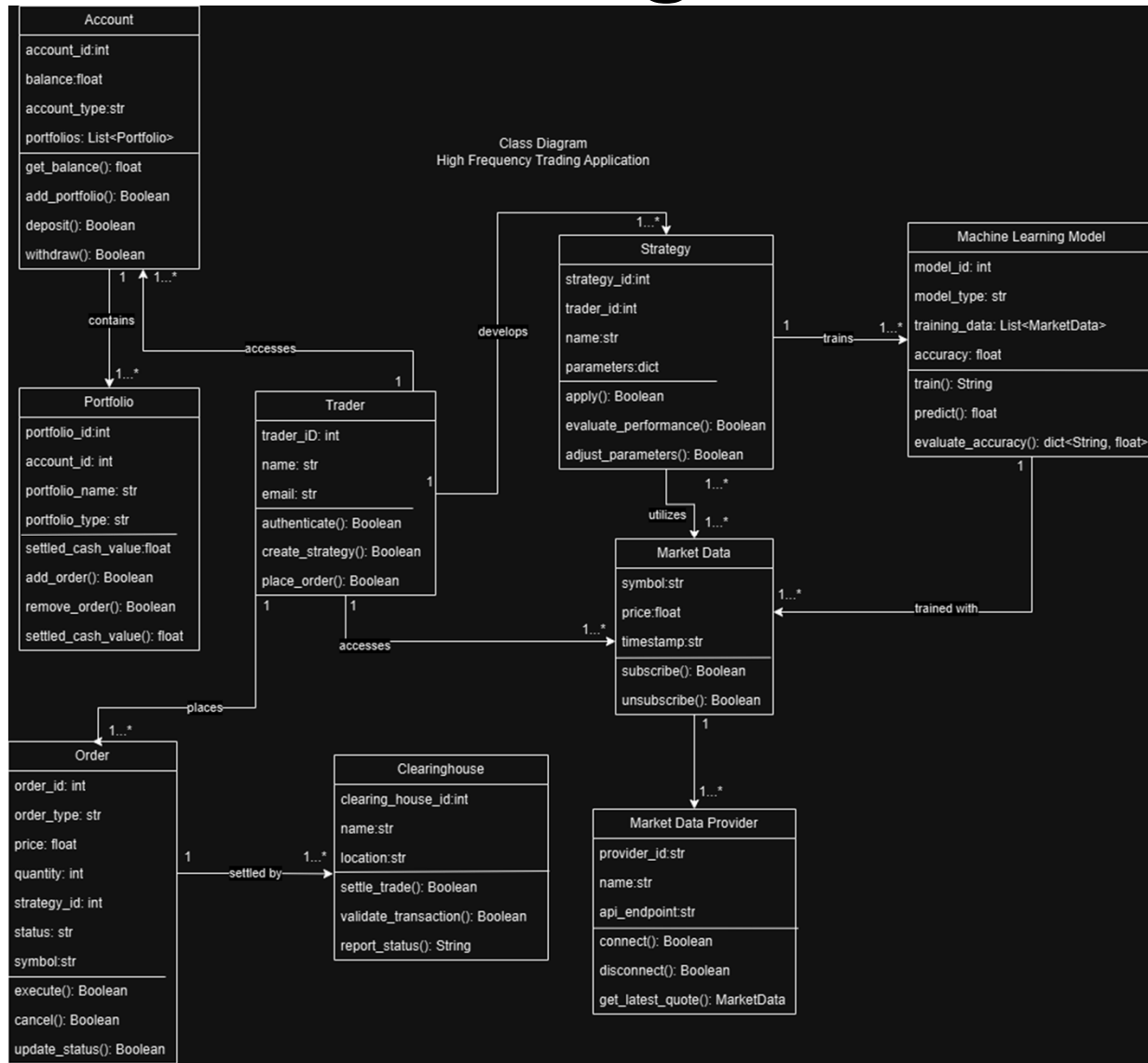

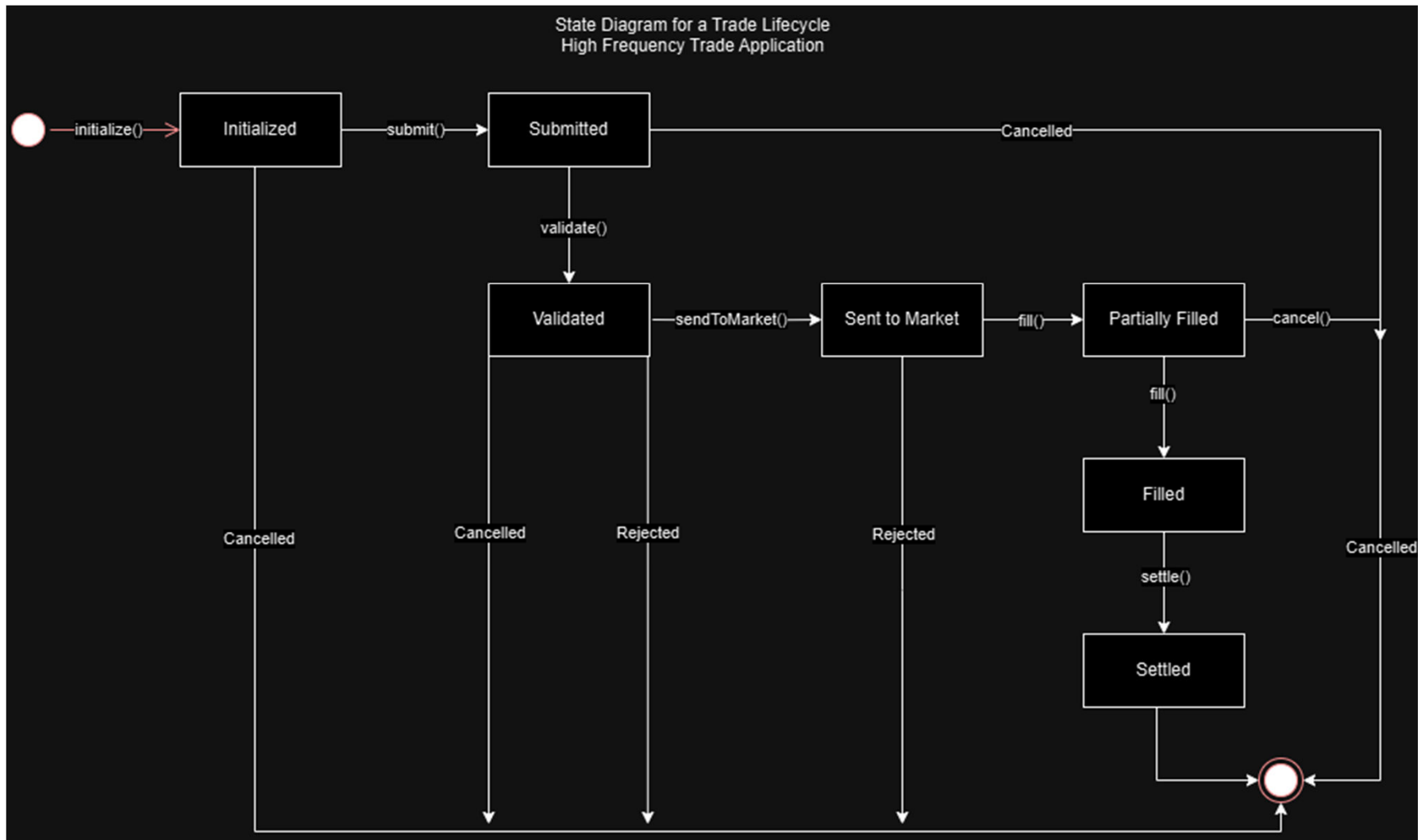Sequence Diagram 4:
Execute Trade Strategy

Activity Diagram

Activity Diagram: Trade Strategy Execution
High Frequency Treading Application

Domain Model

Domain Model
High Frequency Trading Application

Class Diagram

State Chart

State Diagram for a Trade Lifecycle
High Frequency Trade Application

Component Diagram

Component Diagram: High Frequency Trading Platform

- Trader Service
  - is authenticated by → Authentication Service
    - accesses → Account Service
      - uses → Portfolio Service
        - tracks → Trade Service
          - executes → Strategy Service
            - trains → Machine Learning Service
              - utilizes → Market Data Service
          - settled by → Clearinghouse Service

Architecture Pattern

Deployment Diagram
High Frequency Trading Application

Design Principles

- GRASP
- SOLID
- GOF
- Microservices
- Database Normalization
- YAGNI and KISS

Conclusion and Next Steps

- Key Takeaways:
  - More organized "design" approach aside from xtreme programming techniques.
  - Not "building" the ship during the storm.
- Project Benefits:
  - Literature and thorough examples of progressive design just before development and implementation.
- More detailed information available via pdf.

Github Page

- Hosted content page
  - https://github.com/vampireLibrarianMonk/837_hft

Works Cited

Ambler, Scott W. "Data Normalization: Beyond 3rd Normal Form (3NF)." *Agile Data*, www.agiledata.org/essays/dataNormalization.html#1NF. Accessed 10 Aug. 2024.

Arora, Saiyam. "The Flash Crash of 2010: Market Mayhemand Its Impact." *LinkedIn*, 22 Sept. 2023, www.linkedin.com/pulse/flash-crash-2010-market-mayhem-its-impact/. Accessed 10 Aug. 2024.

Calvello, Angelo. "The Medallion Fund, Skepticism and a Failure to Comprehend." *Institutional Investor*, 4 Mar. 2020, www.institutionalinvestor.com/article/2bsx55uaovbhxkya57h8g/opinion/the-medallion-fund-skepticism-and-a-failure-to-comprehend/. Accessed 10 Aug. 2024.

KIRILENKO andREI, et al. "The Flash Crash: High-Frequency Trading in an Electronic Market." *The Journal of Finance*, vol. 72, no. 3, 2017, pp. 967–98. *JSTOR*, http://www.jstor.org/stable/26652722. Accessed 10 Aug. 2024.

"The Concept of Concepts: HFT." *ProfitView*, 12 Sept. 2021, profitview.net/blog/the-concept-of-concepts-hft. Accessed 10 Aug. 2024.