UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE LOS VALLES

SOFTWARE CONFIGURATION MANAGEMENT

Student: Josué Isaías Zepeda Martínez
Code: 209450572

Master: Software Engineering

Professor: Omar Ali Zatarain Duran

3rd Semester 2023-B

Project: Development of an Algorithmic Trading Program

Baseline Date: September 5, 2023

Project Vision:
To develop an algorithmic trading program that allows users to design, test, and execute automated investment strategies in financial markets, with the goal of improving the efficiency and profitability of financial operations.

Project Objectives:

1. Design and develop a functional and scalable algorithmic trading program.
2. Provide users with an intuitive platform for creating and programming custom trading strategies.
3. Integrate an API from at least one trading platform for automated order execution.
4. Ensure data and financial transaction security.
5. Implement risk management and investment protection measures.
6. Provide real-time monitoring and analysis tools.
7. Offer comprehensive documentation and user support.

Project Scope:
The project scope includes the design, development, testing, and deployment of the algorithmic trading program. This encompasses user interfaces, trading algorithm programming, integration of trading platform API, security and risk management measures, and documentation and support provision.

Functional Requirements:

Data Capture and Analysis:
The program must have the ability to get real-time data from at least one financial market, including asset prices, trade volumes, and other relevant indicators. It should analyze this data to identify patterns, trends, and investment opportunities.

Algorithmic Model Development:
Multiple algorithmic models need to be designed and implemented that decide when to buy, sell, or hold financial assets. These models should be adaptable to fit the market better.

Automatic Execution of Trades:
The program should execute buy and sell orders automatically through a trading platform using APIs. It must follow predefined investment strategies.

Intuitive User Interface:
A user interface must be provided, enabling users to monitor the program performance, view executed trades, and access detailed reports.

Non-Functional Requirements:

Low Speed and Latency:
The system must make fast decisions and execute trades within fractions of a second to capitalize on real-time opportunities.

Security and Data Protection:
The program must operate securely, safeguarding financial and personal data involved in trades.

Adaptability to Different Markets:
The program must work effectively in at least one financial asset market, adapting to its specific characteristics.

- Stock markets (Like Apple or Tesla shares for example)
- Forex market (fiat money from different countries)
- Cryptocurrency markets (such as Bitcoin or Ethereum)

With the possibility of expanding the markets and platforms where the program operates in the future.

Reliability and Stability:
The system should be reliable and stable, operating without interruptions and avoiding critical failures.

Understandable Documentation:
Detailed documentation should be provided explaining the program's functionality, implemented models, and how to interpret generated reports.

Key Deliverables:

1. Functional algorithmic trading program.
2. Intuitive user interface.
3. Detailed documentation.
4. Test and analysis reports.
5. Technical user support.

Major Milestones:

1. Completion of program design and specifications.
2. Completion of program development.
3. Completion of unit and integration testing.
4. Security and performance testing conducted and approved.
5. Initial program launch.
6. Ongoing technical support and monitoring.

Risks and Mitigations:

- **Risk:** Volatility in financial markets.
    - **Mitigation:** Implementation of risk management strategies and loss limits.

- **Risk:** Data security breaches.
    - **Mitigation:** Implementation of robust security measures and regular security reviews.

- **Risk:** Changes in financial regulations.
    - **Mitigation:** Staying updated with regulations and adapting the program as needed.

Key Resources:

- Software development team.
- Trading and financial market specialists.
- Access to trading platform APIs.
- Server infrastructure and cloud technology.

Assumptions:

- Necessary permissions and licenses will be obtained for accessing financial markets and executing trades.
- Market data will be available in real-time and reliably through APIs.

Software Design for Algorithmic Trading

Software Architecture:

1. **Microservices Architecture:** The software could adopt a microservices architecture to modularize core functions and facilitate scalability. Each microservice could handle a specific task, such as data capture, analysis, order execution, and monitoring.

2. **Database:** A reliable and scalable database would be used to store market data, transaction histories, user profiles, and strategy configurations.

Key Components:

1. **User Interface (UI):** The software could feature an intuitive user interface that allows users to configure and monitor strategies, view performance reports, and make adjustments as needed.

2. **Data Capture Module:** This module would be responsible for fetching real-time data from multiple market sources, such as stock exchanges and data providers. It would use APIs to access this data.

3. **Analysis Engine:** The analysis engine would process and analyze market data to identify relevant patterns and signals for trading strategies.

4. **Order Execution Module:** This module would connect to trading platform APIs to automatically execute buy and sell orders based on defined strategies.

5. **Risk Management:** A critical component would be the risk management module, which evaluates the risk associated with each trade and applies measures like loss limits and stop-loss to protect the investment.

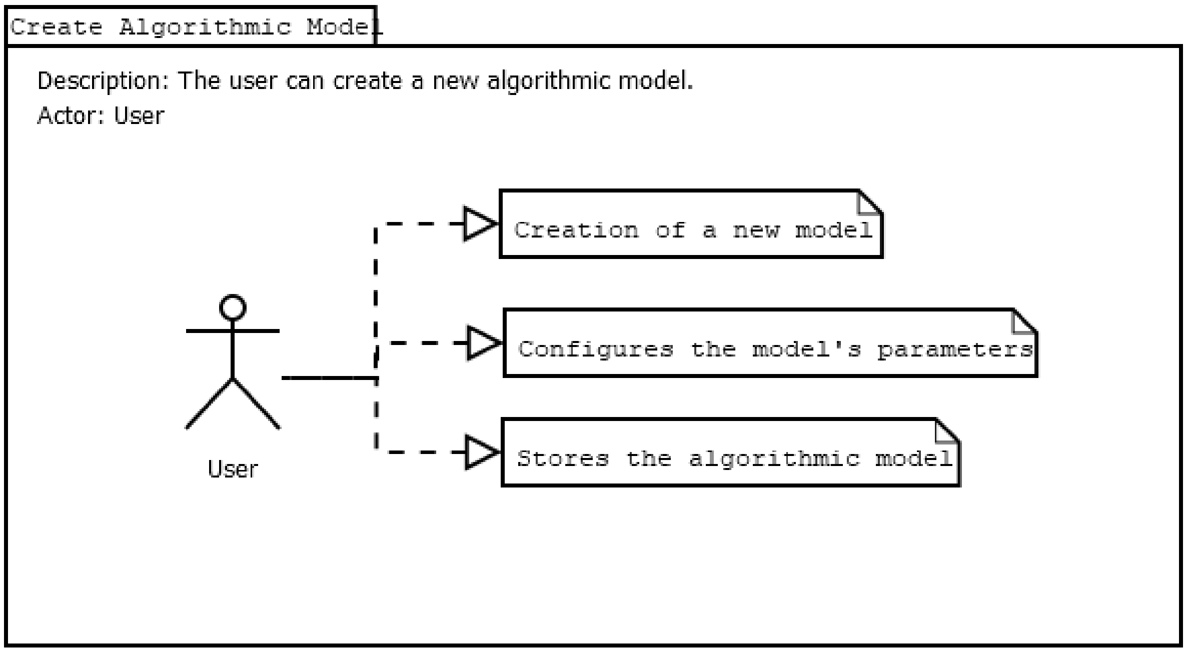
Security and Protection:
Robust security protocols would be implemented to safeguard financial and personal data. This includes user authentication, data encryption, and measures to prevent unauthorized access.

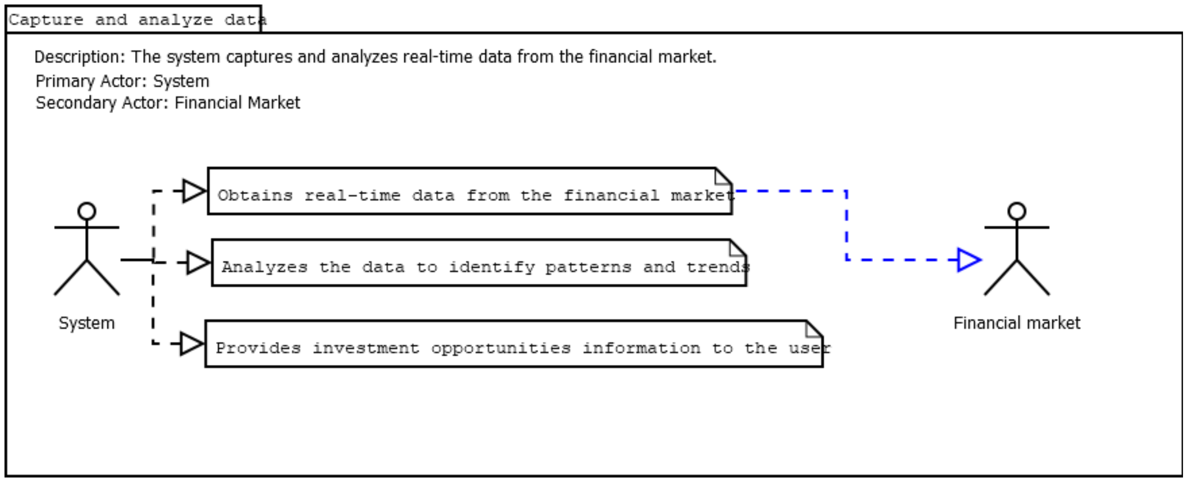
Scalability:
The design would be planned to make the software scalable, capable of handling increased data volume and transactions as it grows.

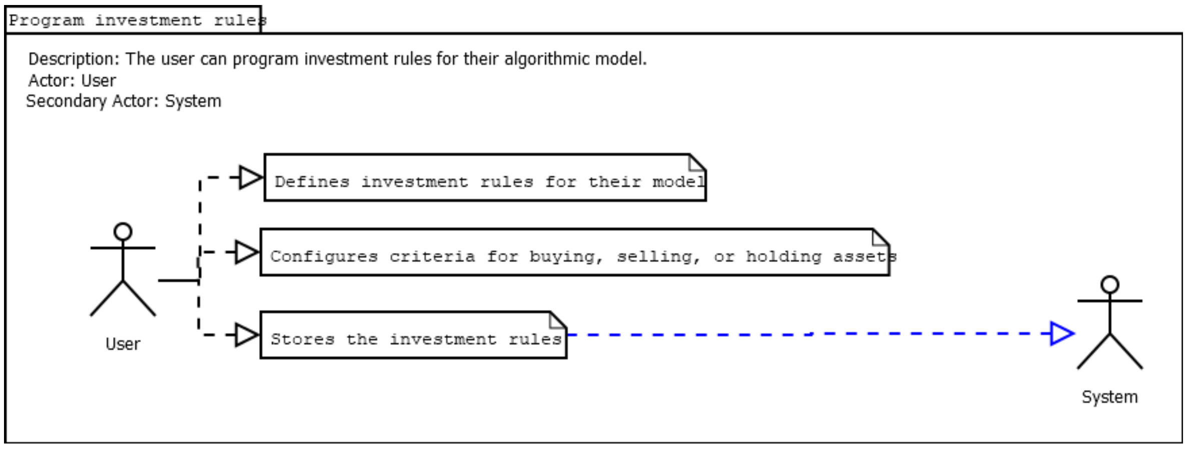
Documentation:
Comprehensive documentation would be provided, describing the architecture, functionality, and guidelines for strategy development.

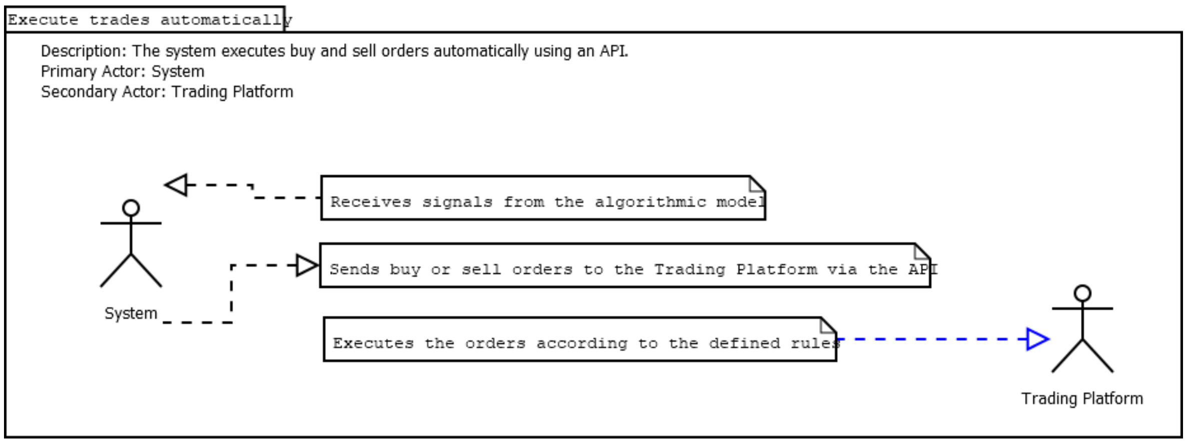
Testing:
Thorough testing, including unit, integration, security, and performance testing, would be conducted to ensure the software operates reliably.

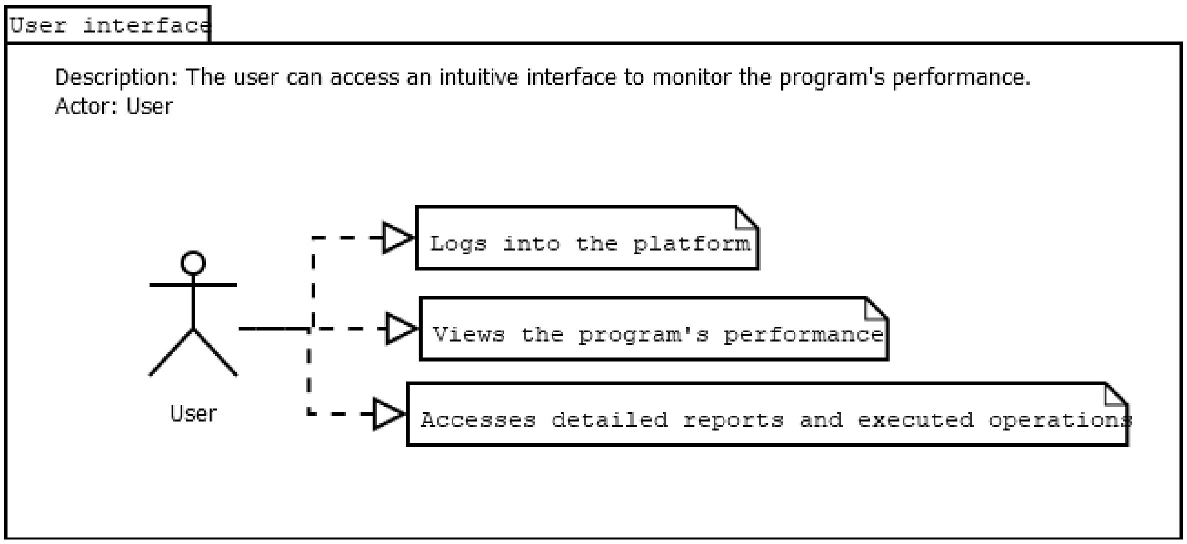
Use case diagrams

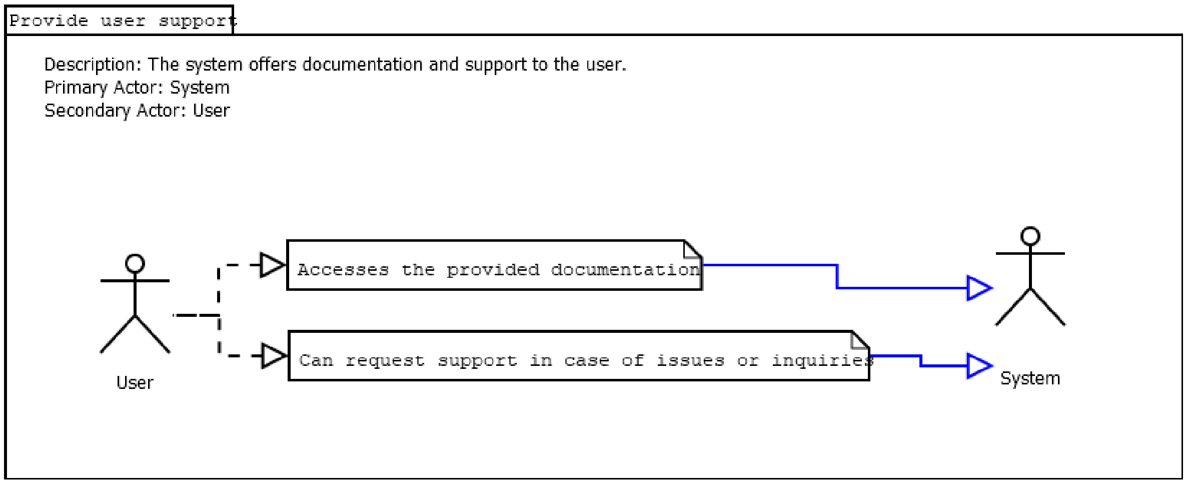

Create Algorithmic Model
Description: The user can create a new algorithmic model.
Actor: User
Creation of a new model
Configures the model's parameters
Stores the algorithmic model
User

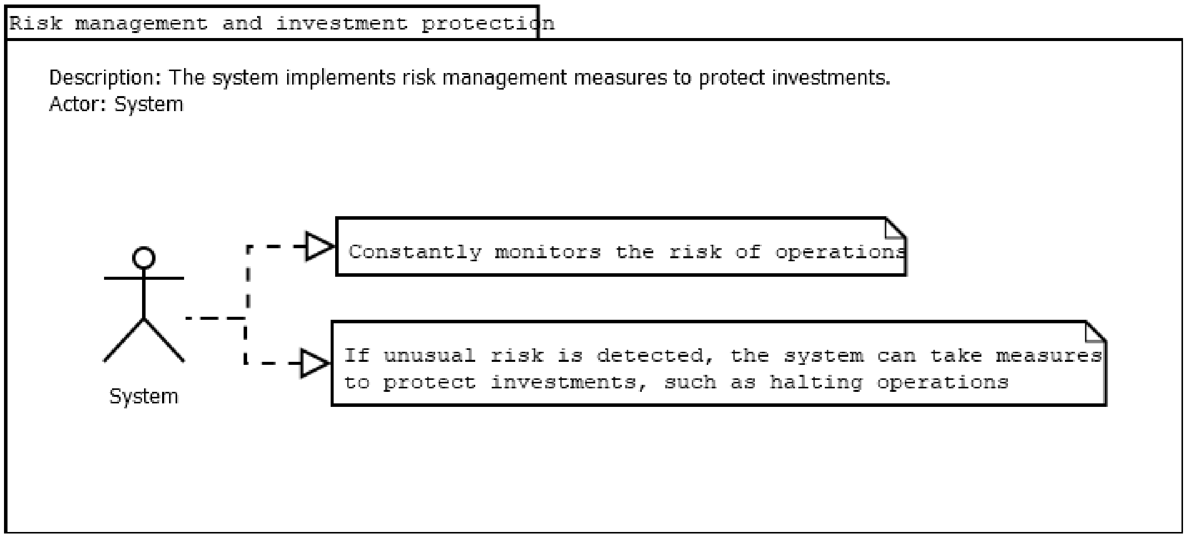

Capture and analyze data
Description: The system captures and analyzes real-time data from the financial market.
Primary Actor: System
Secondary Actor: Financial Market
Obtains real-time data from the financial market
Analyzes the data to identify patterns and trends
Provides investment opportunities information to the user
System
Financial market


Description: The user can program investment rules for their algorithmic model.
Actor: User
Secondary Actor: System
Program investment rules
User
Defines investment rules for their model
Configures criteria for buying, selling, or holding assets
Stores the investment rules
System


Description: The system executes buy and sell orders automatically using an API.
Primary Actor: System
Secondary Actor: Trading Platform
Execute trades automatically
System
Receives signals from the algorithmic model
Sends buy or sell orders to the Trading Platform via the API
Executes the orders according to the defined rules
Trading Platform


Description: The user can access an intuitive interface to monitor the program's performance.
Actor: User
User interface
User
Logs into the platform
Views the program's performance
Accesses detailed reports and executed operations


Provide user support
Description: The system offers documentation and support to the user.
Primary Actor: System
Secondary Actor: User
User
Accesses the provided documentation
Can request support in case of issues or inquiries
System


Risk management and investment protection
Description: The system implements risk management measures to protect investments.
Actor: System
System
Constantly monitors the risk of operations
If unusual risk is detected, the system can take measures to protect investments, such as halting operations