Simulation of Stock Exchange and operations of HFThorizontal line

# Table of Contents:

## horizontal line[Introduction](#_ks82b9yq73a5)

### [Team Details](#_z1y9tifczg7t)

### [Project Overview](#_6k5oo7mopspn)

### [Scope](#_ra54rrgfpnt5)

## [Objectives](#_c361ghh8ac0o)

### [Implement high-performance trading algorithms in C++](#_2bjjslyi1vjr)

### [Develop network monitoring capabilities in C++](#_qkhag4zmpau)

### [Build a backend server and user interface in Java](#_4x8k73k3f2cm)

### [Ensure secure and stable network connections](#_jamcwsuf1ynj)

## [System Overview](#_gvcmxlg5muxo)

### [Technical Specifications](#_cj3akh3g1792)

### [Input/Output Requirements](#_w758u2vqq249)

## [Functional Requirements](#_glezff3gxltv)

### [Detailed Features](#_j59g7uvgaypk)

### [Use Cases](#_v09d7y9l3hxt)

## [Development Setup](#_favtevxsm3fb)

## [Workflow](#_m6cmx820tnhf)

## [Important Files & Folders](#_57t59armeciu)

## [Testing & Logging](#_ba7gpkihk2o5)

## [Conclusion](#_vumri6pyhio1)

## [Non-functional Requirements](#_3efmtsks10c9)

## [Optional Features](#_i9uw6pt9aqph)

# Introduction

## Team Details

### Team name: **DigitalDynamos**

| Name | Email | Roll Number |
| --- | --- | --- |
| Parsania Ramya Parulbhai | ParsaniaRamya.Parulbhai@iiitb.ac.in | IMT2023015 |
| Chaitya Nirmal Shah | chaityanirmal.shah@iiitb.ac.in | IMT2023055 |
| Velidanda Krishna Sai | velidanda.sai@iiitb.ac.in | IMT2023094 |
| Hitanshu Seth | Hitanshu.Seth@iiitb.ac.in | IMT2023100 |
| Aaryan Rajesh Antala | Aaryan.Antala@iiitb.ac.in | IMT2023548 |
| Satyam Ambi | Satyam.Ambi@iiitb.ac.in | IMT2023623 |

## Project Overview:

This project aims to simulate how an HFT (High-Frequency Trading) firm executes its strategies to profit through the stock exchanges. The project also simulates how a stock exchange manages its order book to match and execute orders efficiently using advanced data structures and algorithms to minimize latency.

### ***Purpose***:

* The project offers a realistic simulation of an HFT system and order book management, helping users understand how the entire system functions.
* The project uses strategies such as arbitrage, market making, etc., and advanced algorithms and data structures to reduce the latency of managing the order book.

### ***High-Level Functionality*:**

* Performance analysis: Provides insights into the profits and losses of the firm.
* Simulating strategies of HFT firms.
* Process incoming buy/sell orders and match them based on price and quantity.

### 

### ***Scope*:**

#### *Included*:

* Live management of order books that keep track of buy/sell prices.
* Simulation of basic HFT strategies.
* Implementation of the classes involved in Java, and C++ implementation of the algorithms being used by the HFT firm.

#### Not included:

* Real-time data from actual financial markets.
* Execution of real trades or financial transactions.

# Objectives

*The primary objectives of this project are:*

### Implement high-performance trading algorithms in C++:

* + Maintain the order book containing thousands of orders.
  + Develop and implement efficient data structures to match, add and delete orders.
  + Based on the order, utilise the appropriate data objects and execute algorithms to identify all profitable trading strategies.

### Build a backend and user interface in Java:

* + Users can see the orders placed by the HFT.
  + Users can see the profit/loss of the HFT.
  + Plotting graphs for various data like profit/loss.

1. Use multithreading to let the HFT use multiple strategies when trading to maximise profit
   * Because multiple strategies are implemented for the HFT, it is more optimal for the HFT to apply the appropriate approach to maximise profit.
   * Based on the order book data, the HFT would use the strategy most fit for the scenario or even use multiple strategies.

# 

# System Overview

### ***Technical Specifications*:**

* Frontend: Java (UI)
* Backend: Java (Server-side logic): For implementing HFT strategies and to simulate different cases
* Core Logic: C++ (Trading algorithms and network monitoring): For order matching and efficient data structures

### ***Input/Output Requirements*:**

Input:

* Buy and sell orders placed by the HFT system.
* Order details: Order ID, Order type (buy/sell), price and quantity

Output:

* Data of Matched orders with executed price, quantity, and timestamp.
* Logs showing how the HFT strategies perform under different market conditions.
* Detailed report on the profits and number of matched orders on the performance of the HFT strategies

# Functional Requirements

## Detailed Features:

* **Order Book Management**: The system will maintain an order book in real-time containing current bid and ask prices while managing new orders, cancelling and modifying existing orders, and sorting current orders on priority (the highest bid will match the lowest ask).
* **Order Matching Engine**: Orders are matched according to prices (If a bidding price of an order matches or exceeds the ask price the trade is executed) and quantity and then removes the order from the order book and logs the trade.
* **Arbitrage Strategy**: The arbitrage strategy involves exploiting price differences of the same asset across different markets. HFT systems buy the asset at a lower price in one market and sell it at a higher price in another. Thus profiting from the discrepancy.
* **Market-Making Strategy**: The market-making strategy involves continuously placing buy and sell limit orders on a stock to profit from the bid-ask spread. HFT earns small, consistent profits by capturing the difference between the buying and selling prices.

## Use Cases:

1. **Arbitrage Trading:** Suppose the HFT system detects a price discrepancy between two stock exchanges. For example:

* Stock exchange A lists a stock at ₹10.50, and exchange B lists the stock at ₹10.55. The HFT places a buy order at ₹10.50 in exchange A and a sell order for the same stock in exchange B at ₹10.55. Thus, the order-matching engine executes both trades, profiting the HFT from the arbitrage opportunity.

1. **Market Making:**

* The HFT places a buy order slightly below the current market price and places a sell order slightly above the current market price.
* As the market prices change the system adjusts these orders to stay close to the market prices.

# Development Setup

## Instructions:

## Git Commands:

# Workflow

## Backend:

## Frontend:

## Middleware:

# Important Files & Folders

# Testing & Logging

# Conclusion

# Non-functional Requirements

## Performance Requirements:

## Usability:

## Security Requirements:

# Optional Features

* Network Monitoring: Detects network anomalies and potential security threats.
  + Set up a network monitoring system that monitors the network traffic between the HFT and the Exchange, and analyzes the order execution time, and other performance metrics.
  + Based on a predefined set of threshold values for the metrics, it is determined whether or not the order is feasible.
* Order Entry:
  + The user enters the order details, and following this, the order is added to the order book.
  + If any sell orders match then the system executes the trade updating the order book and logs the trade.

# UML Class Diagram:

