Advanced algorithmic trading system

A PROJECT REPORT

Submitted by

Ayush Raj (20BCS7576)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

MAY 2022

Advanced algorithmic trading system

A PROJECT REPORT

Submitted by

Lekhraj Sharma (20BCS7553)

Ayush Raj (20BCS7576)

Shubham Kumar (20BCS7572)

Shashwat Verma (20BCS7532)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

MAY 2022

BONAFIDE CERTIFICATE

Certified that this project report **“Advanced algorithmic trading system ”** is the bonafide work of “**Lekhraj Sharma, Ayush Raj, Shubham Kumar and Shashwat Verma”** who carried out the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
Dr. Puneet Kumar	Er. Shuvendu Das
Head of The Department	**Supervisor**
Computer Science and Engineering	Computer Science and Engineering

Submitted for the project viva-voce examination held on ____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

On this great occasion of accomplishment of our project on Advance Algorithmic trading system , we would like to sincerely express our gratitude to our Project Supervisor **ER SHUVENDU DAS** and Co-Supervisor **ER SHIKHA ATWAL** who has been supported through the completion of this project.

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced our thinking, behaviour and acts during the course of study.

We express our sincere gratitude for providing us an opportunity to undergo this Project as the part of the curriculum.

We are thankful for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

Finally, as one of the team members, I would like to appreciate all my group members for their support and coordination, I hope we will achieve more in our future endeavours.

Regards!

Ayush raj
Lekhraj Sharma
Shubham Kumar
Shashwat Verma

TABLE OF CONTENT

CHAPTER -3(DESIGN FLOW/PROCESS)

CHAPTER -4(RESULT ANALYSIS AND VALIDATION)

CHAPTER -5(CONCLUSION AND FUTURE SCOPE)

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Algorithm trading is all about the use of computers programs to automate one or more stages of the trading process: pre trade analysis (data analysis), trading signal generation (buy and sell recommendations), and trade execution. Today, Algorithmic trading is amongst the most talked about technologies in the recent years. It has given trading Firms more power in the rapidly evolving markets by eliminating human errors and changing the way financial markets are interlinked today. Its usage is credited to most markets and even to commodity trading. For this project we will use python and machine learning. Machine learning models are becoming increasingly prevalent in algorithmic trading and investment management. The spread of machine learning in finance challenges existing practices of modelling and model use and creates a demand for practical solutions for how to manage the complexity pertaining to these techniques. Python has become a preferred choice for trading recently as Python is open-source and all the packages are free for commercial use. Python has gained traction in the quant finance community. Python makes it easy to build intricate statistical models with ease due to the availability of sufficient scientific libraries. The process to retrieve, format and use data is an essential part of trading using Python, as without data there is nothing we can go ahead with. Financial data is available on various online websites. This data is also called as time-series data as it is indexed by time (the timescale can be monthly, weekly, daily, 5 minutely, minutely, etc.). Apart from that, we can directly upload data from Excel sheets too which are in CSV format, which stores tabular values and can be imported to other files and codes.

CHAPTER 1: INTRODUCTION

1.1) INTRODUCTION

Advances in telecommunications and computer technologies during the past decade have created increasingly global, dynamic, and complex financial markets, which in turn have stimulated trading by computer programs and the rise of systems for algorithmic trading—also known as AT, algo, or black-box- to automate one or more stages of the trading process. These systems seek to capture fleeting anomalies in market prices, profit from statistical patterns within or across financial markets, optimally execute orders, disguise a trader's intentions, or detect and exploit rivals' strategies. Ultimately, profits drive any algorithmic trading system— whether in the form of cost savings, client commissions, or proprietary trading. Algorithmic trading and artificial stock markets have generated huge interest not only among brokers and traders in the financial markets but also across various disciplines in the academia. The emergence of algorithmic trading has created a new environment where the classic way of trading requires new approaches. In order to understand the impact of such a trading process on the functioning of the market, new tools, theories and approaches need to be created. Thus, artificial stock markets have emerged as simulation environments to test, understand and model the impact of algorithmic trading, where humans and software agents may compete on the same market. The purpose of this project is to create a framework to test and analyze various trading strategies in a dedicated artificial environment. Among the hottest programming languages for algo trading, are R and Python, alongside languages such as C++, C#, and Java. In this project, we'll use Python for Algorithmic trading system. The project will cover the following: • The basics that we need to get started: We'll first know more about the stocks and trading strategies, what time series data is and what we need to do to set up our workspace. • An introduction to time series data and some of the most common financial analyses, such as moving windows, volatility calculation, with the Python package Pandas. • The development of a simple momentum strategy: We'll first go through the development process step-by-step and start by formulating and coding up an algorithmic trading strategy. • Next, you'll backtest the formulated trading strategy with Pandas, zipline and Quantopian. • Afterward, we'll see how we can do optimizations to our strategy to make it perform better, and we'll eventually evaluate our strategy's performance and robustness.

1.2) PROBLEM IDENTIFICATION

Algorithmic trading is a method of executing orders using automated pre-programmed trading instructions accounting for variables such as time, price, and volume. This type of trading attempts to leverage the speed and computational resources of computers relative to human traders. In the twenty-first century, algorithmic trading has been gaining traction with both retail and institutional traders. It is widely used by investment banks, pension funds, mutual funds, and hedge funds that may need to spread out the execution of a larger order or perform trades too fast for human traders to react to. A study in 2019 showed that around 92% of trading in the Forex market was performed by trading algorithms rather than humans.

The term algorithmic trading is often used synonymously with automated trading system. These encompass a variety of trading strategies, some of which are based on formulas and results from mathematical finance, and often rely on specialized software.

Examples of strategies used in algorithmic trading include market making, inter-market spreading, arbitrage, or pure speculation such as trend following. Many fall into the category of high-frequency trading (HFT), which is characterized by high turnover and high order-to-trade ratios. HFT strategies utilize computers that make elaborate decisions to initiate orders based on information that is received electronically, before human traders are capable of processing the information they observe. As a result, in February 2012, the Commodity Futures Trading Commission (CFTC) formed a special working group that included academics and industry experts to advise the CFTC on how best to define HFT. Algorithmic trading and HFT have resulted in a dramatic change of the market microstructure and in the complexity and uncertainty of the market macrodynamic, particularly in the way liquidity is provided.

1.3) TASK IDENTIFICATION

Algorithmic trading (also called automated trading, black-box trading, or algo-trading) uses a computer program that follows a defined set of instructions (an algorithm) to place a trade. The trade, in theory, can generate profits at a speed and frequency that is impossible for a human trader.

The defined sets of instructions are based on timing, price, quantity, or any mathematical model. Apart from profit opportunities for the trader, algo-trading renders markets more liquid and trading more systematic by ruling out the impact of human emotions on trading activities.

Using these two simple instructions, a computer program will automatically monitor the stock price (and the moving average indicators) and place the buy and sell orders when the defined conditions are met. The trader no longer needs to monitor live prices and graphs or put in the orders manually. The algorithmic trading system does this automatically by correctly identifying the trading opportunity.

CHAPTER 2: LITERARY SURVEY

2.1) SURVEY

The global algorithmic trading market is expected to grow from 11.1 billion in 2019 to 18.8 billion by 2024. The growth is likely to be driven by rising demand for quick, reliable, and effective order execution. Lowered transactional costs, heightened government regulations, and increased demand for market surveillance are also few of the major catalysts for the growth of the Algo Trading market.

2.2) <u>BIBLIOMETRIC ANALYSIS</u>

(a) We are using Fyers API in our project. At first we have to login to the fyers dashboard with our id and password. If the user is new then he/she have to create an account on fyers trading app. Then the user has to create an app in fyers dashboard through which user can get their client id and secret key which helps them to connect their strategy with the fyers dashboard. Then we have to install fyers api library in our python directory i.e. on Jupyter notebook. After that we have to get our access token and authorization code to connect our system with the app that we have created on fyers dashboard. Now that we have connected our Jupyter notebook with the app that we have created on fyers dashboard, we will start to work with our strategy to trade. We will be creating a stock screener that will screen out the stocks from the NIFTY 500 index based on our strategy.

For screening the stocks, the dashboard will need the market data so in order to have market data we will be using the web socket that is being provided by the fyers team. We will be writing our code so that we can access historical and live market data for different time frame for our trading system. After that we will have to prepare the code for our trading strategy. Now that when we have already got market data and also, we have prepared the code for our trading system, the system will start screening the stocks based on our strategy. After that we will be creating a trading bot in telegram where will be getting our screened stocks and also, we will be getting updates about the buying and selling of our screened stocks. Then we will start preparing our code for buying and selling of the stocks.

After preparing the code for buying/selling we will be feeding the code with the screened-out stocks and also, we will be assigning the volume of stocks that we have to buy or sell. Accordingly, the system will buy or sell the stocks.

We will continue getting our updates through our telegram bots. This process will continue on the daily basis according to the code that we have fed to the system.

2.3) <u>LITERATURE REVIEW</u>

Algorithmic or Quantitative trading can be defined as the process of designing and developing statistical and mathematical trading strategies. It is an extremely sophisticated area of finance.

Following are the steps to make an algorithmic trading system:

1. Learn Python Programming

In order to start developing an algorithmic trading system, we need solid fundamentals. In Python, we have to thoroughly understand certain topics in the language.

Here’s what it takes to master in the Python ecosystem for data science:

• **Environment Setup—**this includes creating a virtual environment, installing required packages, and working with Jupyter notebooks.

• **Data Structures—**some of the most important python data structures are lists, dictionaries, NumPy arrays, tuples, and sets.

• **Object-Oriented Programming—**As a quant analyst, we should make sure we are good at writing well-structured code with proper classes defined. We must learn to use objects and their methods while using external packages like Pandas, NumPy, SciPy, and so on.

2. Learn How to Crunch Financial Data

Data analysis is a crucial part of finance. Besides learning to handle dataframes using Pandas, there are a few specific topics that we have to pay attention to while dealing with trading data.

How to exploring data using Pandas

One of the most important packages in the Python data science stack is undoubtedly Pandas. We can accomplish almost all major tasks using the functions defined in the package. We have to focus on creating dataframes, filtering (loc, iloc, query), descriptive statistics (summary), join/merge, grouping, and subsetting.

How to deal with time-series data

Trading data is all about time-series analysis. We have to learn to resample or reindex the data to change the frequency of the data, from minutes to hours or from the end of day OHLC data to end of week data.

3. Figuring out how to Write Fundamental Trading Algorithms

A project in quantitative finance requires a solid understanding of statistical hypothesis testing and mathematics. A good grip over concepts like multivariate calculus, linear algebra, probability theory will help us lay a good foundation for designing and writing algorithms. We can start by calculating moving averages on stock pricing data, writing simple algorithmic strategies like moving average crossover or mean reversion strategy and learning about relative strength trading.
After taking this small yet significant leap of practicing and understanding how basic statistical algorithms work, we can look into the more sophisticated areas of machine learning techniques. These require a deeper understanding of statistics and mathematics.

4. Learn About Backtesting

Once we are done coding our trading strategy, we can’t simply put it to the test in the live market with actual capital.
The next step is to expose this strategy to a stream of historical trading data, which would generate trading signals. The carried-out trades would then accrue an associated profit or loss (P&L) and the accumulation of all the trades would give us the total P&L. This is called backtesting.

Backtesting requires you to be well-versed in many areas, like mathematics, statistics, software engineering, and market microstructure. Some concepts we will learn to get a decent understanding of backtesting:

• We can start by understanding technical indicators. Explore the Python package called TA_Lib to use these indicators.
• Employ momentum indicators like parabolic SAR, and try to calculate the transaction cost and slippage.
• Learn to plot cumulative strategy returns and study the overall performance of the strategy.
• A very important concept that affects the performance of the backtest is bias. We should learn about optimization bias, look-ahead bias, psychological tolerance, and survivorship bias

5. Performance Metrics—How to Evaluate Trading Strategies

It's important for us to be able to explain our strategy concisely. If we don't understand our strategy, chances are on any external modification of regulation or regime shift, our strategy will start behaving abnormally.
Once we understand the strategy confidently, the following performance metrics can help us learn how good or bad the strategy actually is:

• **Sharpe Ratio—**heuristically characterizes the risk/reward ratio of the strategy. It quantifies the return we can accrue for the level of volatility undergone by the equity curve.
• **Volatility—**quantifies the "risk" related to the strategy. The Sharpe ratio also embodies this characteristic. Higher volatility of an underlying asset often leads to higher risk in the equity curve and that results in smaller Sharpe ratios.
• **Maximum Drawdown—**the largest overall peak-to-trough percentage drop on the equity curve of the strategy. Maximum drawdowns are often studied in conjunction with momentum strategies as they suffer from them. We will learn to calculate it using the NumPy library.
• **Capacity/Liquidity—**determines the scalability of the strategy to further capital. Many funds and investment management firms suffer from these capacity issues when strategies increase in capital allocation.
• **CAGR—**measures the average rate of a strategy's growth over a period of time. It is calculated by the formula: (cumulative strategy returns) ^ (252/number of trading days)

2.4) <u>PROBLEM DEFINATION</u>

Algorithmic trading, also referred to as algo-trading, is a variant of automated trading that involves the usage of computerized platforms and advanced math and computer programming tools to drive trading transactions in the financial markets. The system utilizes a mathematical model or algorithm or standardized instruction set that facilitates buying or sell signals in the financial markets and facilitates trade without humans' involvement.

Components of Algorithmic Trading

1. **An Algorithm**

   An algorithm can be defined as instructions that perform certain repetitive functions. It can also be developed to cater to certain problem-solving situations. For example, it helps in the easy facilitation of data processing and identifying trends.

2. **Computer Program & Automated Trading Platforms**

   An automated trading platform provides a means to execute the algorithm developed by the programmers. It, as a platform, manages the computer programs designed by the programmers and algo-traders, thereby facilitating buy and sell orders in the financial markets. These platforms also help in the backtesting of algorithms developed by the algo-traders or programmers before they can be deployed.

3. **Technical Analysis**

   The technical analysis involves studying and analyzing the price movements of the listed securities in the financial markets. There are several methods, such as 150-day moving average, 200-days moving average, a double exponential moving average, random oscillators, which helps in the identification of price trends for a particular security.

   The methods of technical analysis can be developed as an algorithm. They can, in turn, be transformed into a computer program that one can then deploy into the financial markets to automate the trading function.

4. **Backtesting**

Backtesting is the process of testing the algorithm and verifying whether the strategy would deliver the results anticipated by the trader. It involves testing the programmer’s approach on the historical market data. In addition, the Backtesting lets the trader identify the pitfalls that could have emerged if one used the strategy with the live market trades.

2.5) <u>OBJECTIVES</u>

Algo-trading is the fastest evolving technology with incredible and intelligent functionalities that improve trading performance and speed for market participants. Today, financial institutions such as banks, broking houses and investment funds employ sophisticated algorithms to establish and liquidate positions at breakneck speed. This setup enables such institutions to exploit minor discrepancies in live market conditions, eclipsing what’s possible manually.

Algo-trading constantly monitors markets and places orders when conditions match a set of parameters such as volume, price, resistance, support, or any other factor that the trader or market participant is comfortable with. One of the advantages of algo-trading is that by using available data, it can easily and quickly identify an ongoing trend. It is otherwise a challenge for market participants to act fast by analyzing a large pool of data in quick time. Besides, market participants can use multiple strategies at once and decide the net outcome of the strategy.

The main objective of algo-trading is not just to profit by trading but to save costs, minimize market impact, and the execution risk of a trading order. Traders don’t need to watch stocks or send slices manually.

Algorithmic trading is mainly used by institutional investors and big brokerage houses to cut down on costs associated with trading. According to research, algorithmic trading is especially beneficial for large order sizes that may comprise as much as 10% of overall trading volume. Typically, market makers use algorithmic trades to create liquidity.

Algorithmic trading also allows for faster and easier execution of orders, making it attractive for exchanges. In turn, this means that traders and investors can quickly book profits off small changes in price. The scalping trading strategy commonly employs algorithms because it involves rapid buying and selling of securities at small price increments.

CHAPTER 3: -DESIGN FLOW/PROCESS

3.1) CONCEPT GENERATION

Algorithmic trading means turning a trading idea into an algorithmic trading strategy via an algorithm. The algorithmic trading strategy thus created can be backtested with historical data to check whether it will give good returns in real markets. The algorithmic trading strategy can be executed either manually or in an automated way.

3.2) DESIGN CONSTRAINTS

(a) REGULATION

(i) Hardware Requirements

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and requirements are also important.

Here is the list of hardware's required:

Processor	Intel CORE i5
RAM	8.0 GB
Hard Disk Drive	200 GB
Monitor	Resolution 1024 x 768
Internal free space	10.0 GB

(ii) Software Requirements

The selection of software often helps us to get the things done in smooth and easy way. Proper availability of up-to-date software is often a boon for programmers.

Here is the list of software's required:

No.	Description
1	Windows 7,8,10

a) Python

Python is a high-level programming language that is more deployed in machine learning and for automation of trading systems. Python has got exclusive library functions that facilitate ease of coding the algorithmic trading strategies. Python has become a preferred choice for trading recently as Python is open-source and all the packages are free for commercial use. Python has gained traction in the quant finance community. Python makes it easy to build intricate statistical models with ease due to the availability of sufficient scientific libraries.

Let us list down a few benefits of Python in algorithmic trading: -

- Parallelization and huge computational power of Python give scalability to the trading portfolio.
- Python makes it easier to write and evaluate algo trading structures because of its functional programming approach. Python code can be easily extended to dynamic algorithms for trading.
- Python can be used to develop some great trading platforms whereas using C or C++ is a hassle and time-consuming job.
- Trading using Python is an ideal choice for people who want to become pioneers with dynamic algo trading platforms.
- For individuals new to algorithmic trading, the Python code is easily readable and accessible.
- It is comparatively easier to fix new modules to Python language and make it expansive in trading.

- The existing modules also make it easier for algo traders to share functionality amongst different programs by decomposing them into individual modules which can be applied to various trading architectures.
- When using Python for trading it requires fewer lines of code due to the availability of extensive Python libraries.
- Python makes coding comparatively easier in trading. Quant traders can skip various steps which other languages like C or C++ might require.
- This also brings down the overall cost of maintaining the trading system.
- With a wide range of scientific libraries in Python, algorithmic traders can perform any kind of data analysis at an execution speed that is comparable to compiled languages like C++.

b) FYERS API Bridge

FYERS API Bridge is a lightweight and easy to use portable application. It can be integrated with the FYERS order management engine through our Trading API to place orders or requests from the front-end charting/Algo platform/Python etc.

A great analogy is to think of an API Bridge as an interface. You need to interact with it to feed information (transmitting trade alerts from the platform) and in return place/confirm a specific order or access orderbook and Tradebook etc. Basically, the API bridge acts as a connection between the front-end charting/Algo platform or an alert generating platforms such as Amibroker, TradingView, MT4 and the order matching engine. The interface connection is extended using a trading API.

FYERS API Bridge facilitates the Algo Trading in the most simplistic way as you can send trade alerts to OMS with smooth order execution without any delay. The platform like Amibroker or TradingView unlocks the power of Programmatic or Algo Trading since you can code your algorithm/strategies using AFL (Amibroker Formula Language) in Amibroker or Pine script in TradingView. Also, you can deploy your conditional Algo straightway through any of the platforms for signal generation.

Traders who want to shift their discretionary trade setup, they can deploy their strategies easily in Amibroker or TradingView. Moreover, in TradingView you can generate trade alerts on the basis of the indicators or oscillators values by providing certain conditions. For instance, if you follow a simple RSI indicator on SBIN symbol, then deploy your own strategy like when RSI value is greater than 28 generate a long entry signal, whenever the said condition meets “Buy” alert will pop-up and send to API Bridge to execute. In the bridge, you can define a few parameters under the Symbol settings like order type, product type, target and stop-loss etc. Also, you can use Bridge to manage your risk in terms of max order per minute, max trades per day or daily max loss i.e., it allows you to manage risk both at a strategy level and global (account) level. Another significant advantage of integrating FYERS API is you can trade “ABC” symbol via alerts generated by the “XYZ” symbol, which is very useful when you trade in Index option and you are away or difficult to keep an eye on it.

c) Websocket

Websocket is just a digital version of Ticker Tapes; a WebSocket is basically a connection between us and the broker servers where we request the continuous stock prices information to the server, and it responds with a continuous stream of prices until the broker or client stops the connection.

Most of the Algo-Trading strategies would rely on live stock prices and use WebSocket’s to get those live prices stream and use it to satisfy their strategy conditions and place orders. If a particular broker does not provide a WebSocket feed, you will have to arrange your own feed to run strategies which can be very inconvenient and expensive.

d) Selenium WebDriver

Selenium WebDriver is a collection of open-source APIs which are used to automate the testing of a web application. Selenium WebDriver tool is used to automate web application testing to verify that it works as expected. It supports many browsers such as Firefox, Chrome, IE, and Safari. However, using the Selenium WebDriver, we can automate testing for web applications only. It does not qualify for window-based applications. It also supports different programming languages such as C#, Java, Perl, Python, PHP and Ruby for writing test scripts. Selenium Webdriver is platform-independent since the same code can be used on different Operating Systems like Microsoft Windows, Apple OS and Linux. It is one of the components of the selenium family, which also includes Selenium IDE, Selenium Client API, Selenium Remote Control and Selenium Grid.

Selenium WebDriver does not handle window component, but this limitation can be overcome by using external tools such as AUTO IT tool, Sikuli etc. It has different location strategies as well such as ID, Name, Link text, Partial link text, Class name, CSS selector and Xpath. It also has better support dynamic web pages like Ajax, where elements of the web page may change without the page itself being reloaded. By using different jar files, we can also test API, Database Test etc. using Selenium WebDriver

e) WebDriver Manager

WebDriverManager is an open-source Java library that carries out the management (i.e., download, setup, and maintenance) of the drivers required by Selenium WebDriver (e.g., chromedriver, geckodriver, msedgedriver, etc.) in a fully automated manner. In addition, as of version 5, WebDriverManager provides other relevant features, such as the capability to discover browsers installed in the local system, building WebDriver objects (such as Chromedriver, FirefoxDriver, EdgeDriver, etc.), and running browsers in Docker containers seamlessly.

WebDriverManager provides a fluent API available using the class WebDriverManager (package io.github.bonigarcia.wdm). This class provides a group of static methods to create managers, i.e., objects devoted to providing automated driver management and other features.

The primary use of WebDriverManager is the automation of driver management. For using this feature, you need to select a given manager in the WebDriverManager API (e.g., chromedriver () for Chrome) and invoke the method setup ().

f) Pandas

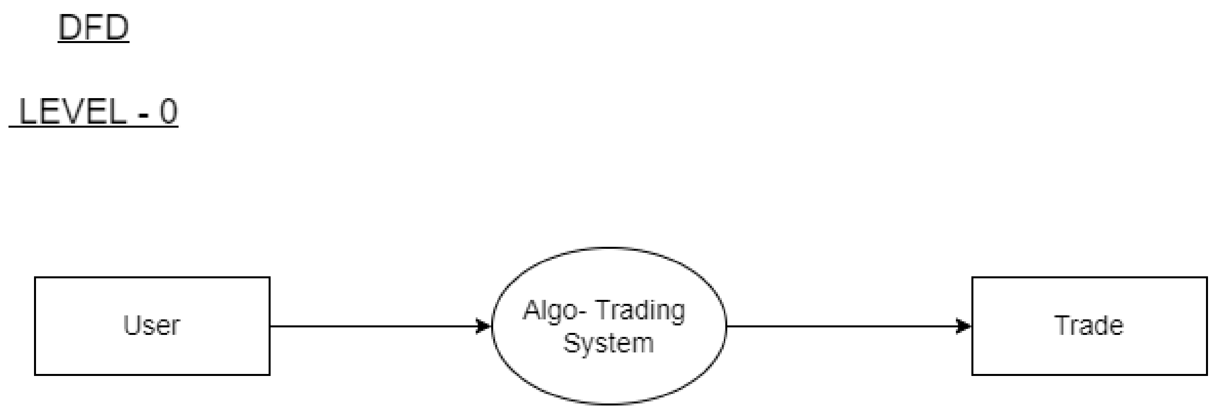
Pandas is a vast Python library used for the purpose of data analysis and manipulation and also for working with numerical tables or data frames and time series, thus, being heavily used in for algorithmic trading using Python. Pandas can be used for various functions including importing .csv files, performing arithmetic operations in series, Boolean indexing, collecting information about a data frame etc.

g) TA-Lib

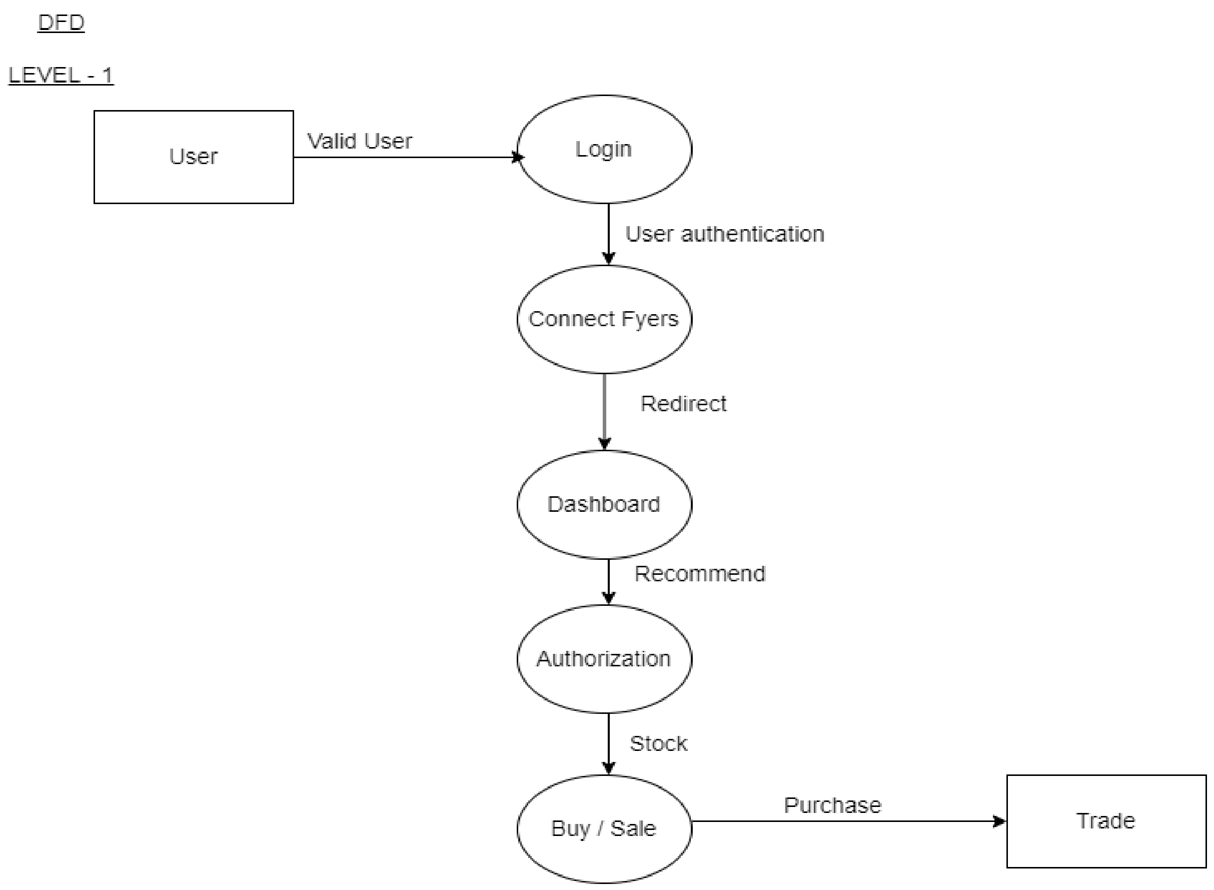
TA-Lib or Technical Analysis library is an open-source library and is extensively used to perform technical analysis on financial data using technical indicators such as RSI (Relative Strength Index), Bollinger bands, MACD etc. It not only works with Python but also with other programming languages such as C/C++, Java, Perl etc. TA-Lib is an open-source python library that is used in analyzing the stock market's historical data like share price, volume, etc. in order to predict the future price or the market direction so that we can make our investments accordingly. Ta-Lib contains a large variety of technical indicators that are used to study.

3.3) DESIGN FLOW

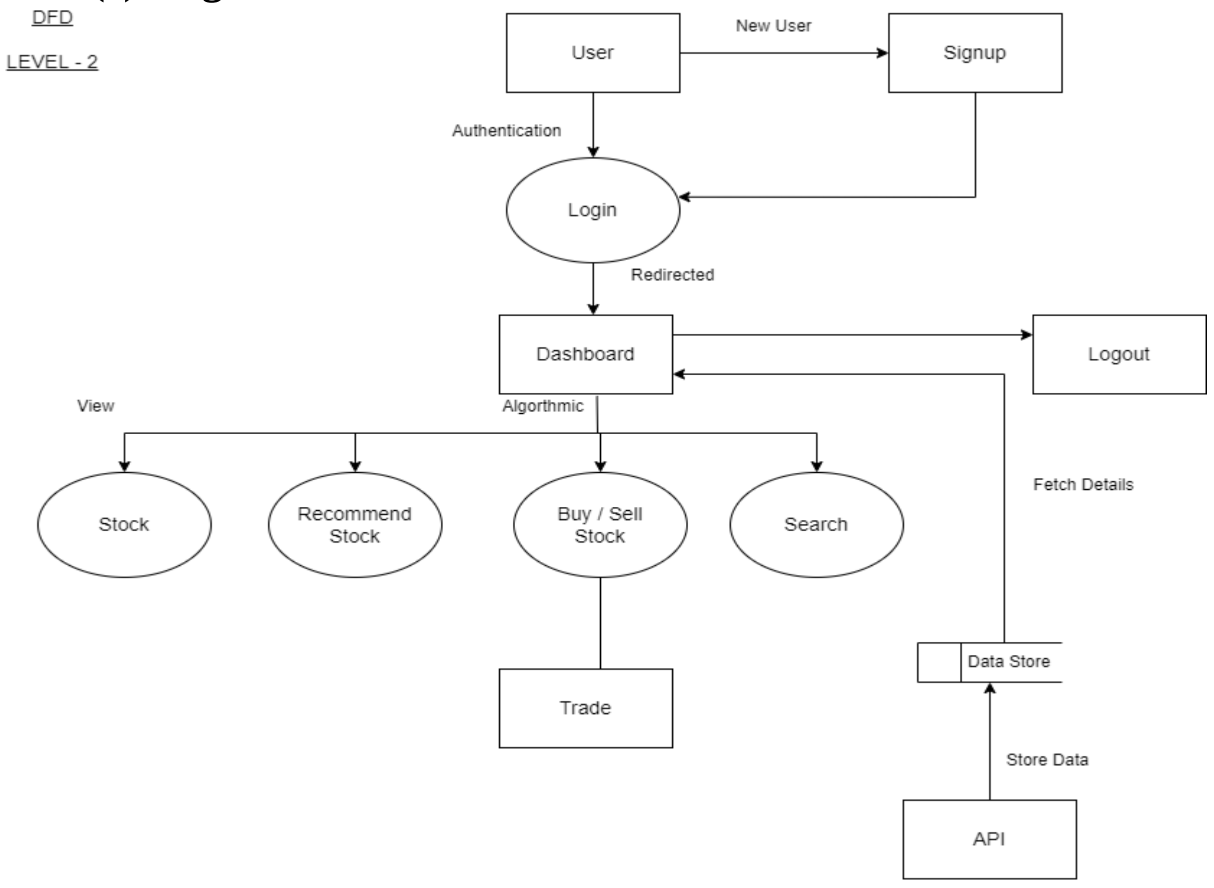
(a) Diagram - 1



(b)Diagram _ 2

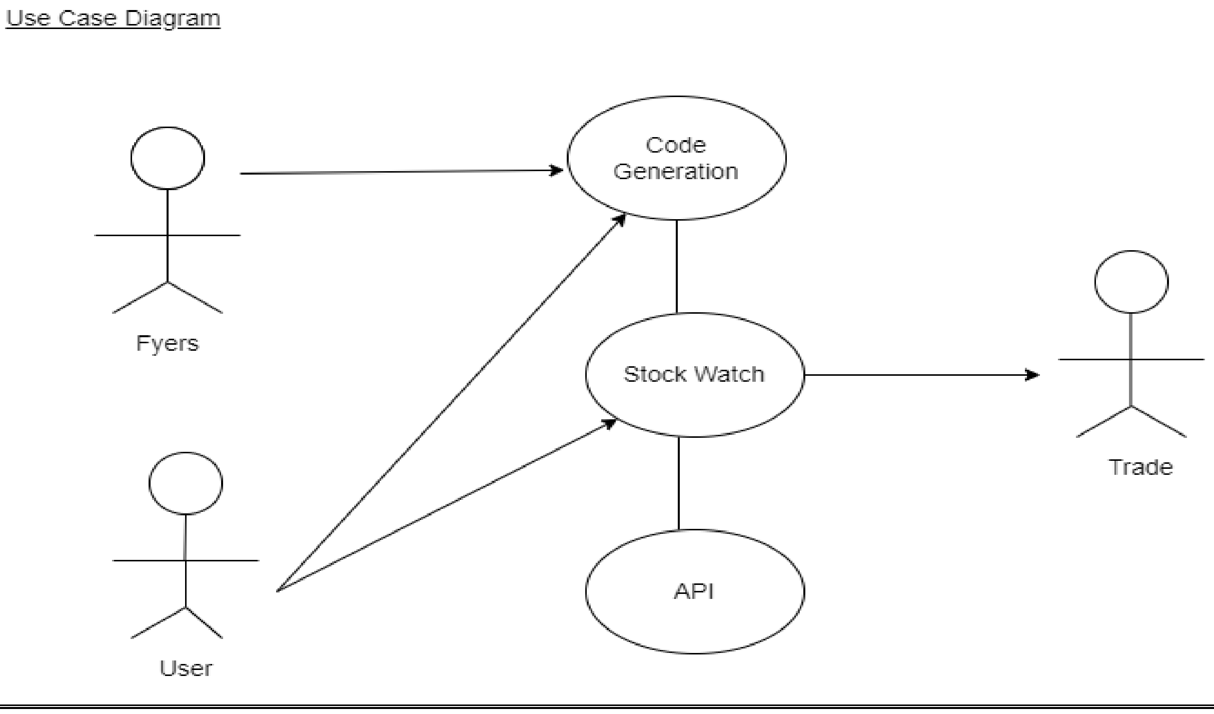


(c)Diagram – 3

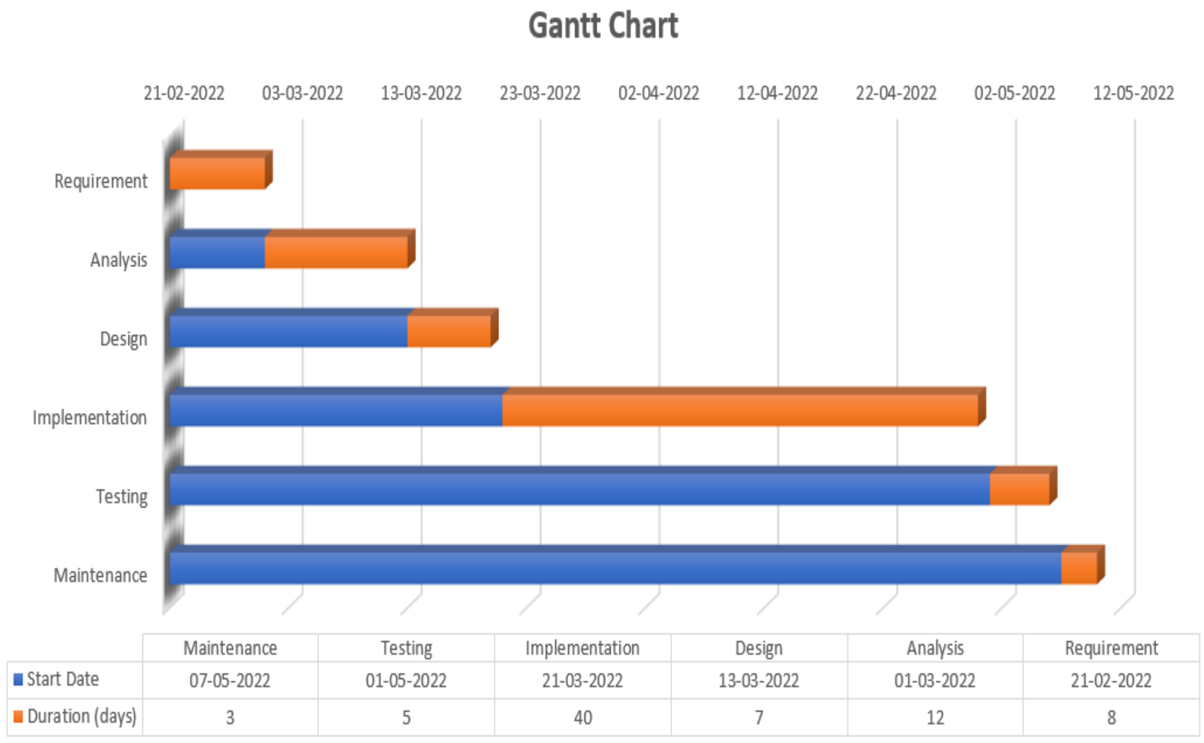


3.4) IMPLEMENTAION PLAN

a) Use Case Diagram



b) Gantt Chart



	Maintenance	Testing	Implementation	Design	Analysis	Requirement
🟦 Start Date	07-05-2022	01-05-2022	21-03-2022	13-03-2022	01-03-2022	21-02-2022
🟧 Duration (days)	3	5	40	7	12	8

CHAPTER-4 (RESULT ANALYSIS AND VALIDATION)

The automated trading system or Algorithmic Trading has been at the center-stage of the trading world for more than a decade now. A “trading system”, more commonly referred as a “trading strategy” is nothing but a set of rules, which is applied to the given input data to generate entry and exit signals (buy/sell).

Algo trading, short for algorithm trading, is a machine-driven trading service that can either suggest trades based on the data that is fed into the system, or automatically execute orders on your behalf. For instance, let’s assume you have come up with a strategy for a trade in a particular stock. Say, you want to sell Reliance Industries’ shares if it falls below 3% and if the sell-side volumes are higher than the 20-day moving average.

You can go to your broker’s website or app and place a ‘Limit’ order with the quantity and price details. However, you won’t be able to specify the volume condition that we mentioned above.

This is where algorithms come in – you can feed both the conditions into an algorithm and tell it what to do when those conditions are met. An order will be placed automatically based on your inputs, with the machine doing your job for you.

Any trading system, conceptually, is nothing more than a computational block that interacts with the exchange on two different streams.

1. Receives market data
2. Sends order requests and receives replies from the exchange.

The market data that is received typically informs the automated trading system of the latest order book. It might contain some additional information like the volume traded so far, the last traded price and quantity for a scrip. However, to make a decision on the data, the trader might need to look at old values or derive certain parameters from history. To cater to that, a conventional system would have a historical database to store the market data and tools to use that database. The analysis would also involve a study of the past trades by the trader. Hence another database for storing the trading decisions as well. Last, but not least, a GUI interface for the trader to view all this information on the screen.

However, it was found that traditional architecture could not scale up to the needs and demands of Automated trading with DMA. The latency between the origin of the event to the order generation went beyond the dimension of human control and entered the realms of milliseconds and microseconds. Order management also needs to be more robust and capable of handling many more orders per second. Since the time frame is minuscule compared to human reaction time, risk management also needs to handle orders in real-time and in a completely automated way.

The New System architecture of the Algorithmic Trading System

To overcome the limitations of the traditional system architecture, the engine which runs the logic of decision making, also known as the ‘Complex Event Processing’ engine, or CEP, moved from within the application to the server. The Application layer is now a little more than a user interface for viewing and providing parameters to the CEP.

The problem of scaling in an automated trading system also leads to an interesting situation. Let us say 100 different logics are being run over a single market data event (as discussed in the earlier example). However, there might be common pieces of complex calculations that need to be run for most of the 100 logic units, let’s say, the calculation of Greeks for options. If each logic were to function independently, each unit would do the same Greek calculation, which would unnecessarily use up processor resources. In order to optimize on the redundancy of calculation, complex redundant calculations are typically hived off into a separate calculation engine which provides the Greeks as an input to the CEP in the automated trading system.

The order is encrypted in the language which the exchange can understand, using the APIs which are provided by the exchange. There are two kinds of APIs provided by the exchange: native API and FIX API. Native APIs are those which are specific to a certain exchange. The FIX (Financial Information Exchange) protocol is a set of rules used across different exchanges to make the data flow in security markets easier and effective. We will talk about FIX further in the next section.

Since automated trading systems work without any human intervention, it becomes pertinent to have thorough risk checks to ensure that the trading systems perform as designed. The absence of risk checks or faulty risk management can lead to enormous irrecoverable losses for a quantitative firm. Thus, a risk management system (RMS) forms a very critical component of any automated trading system.

There are 2 places where Risk Management is handled in automated trading systems:

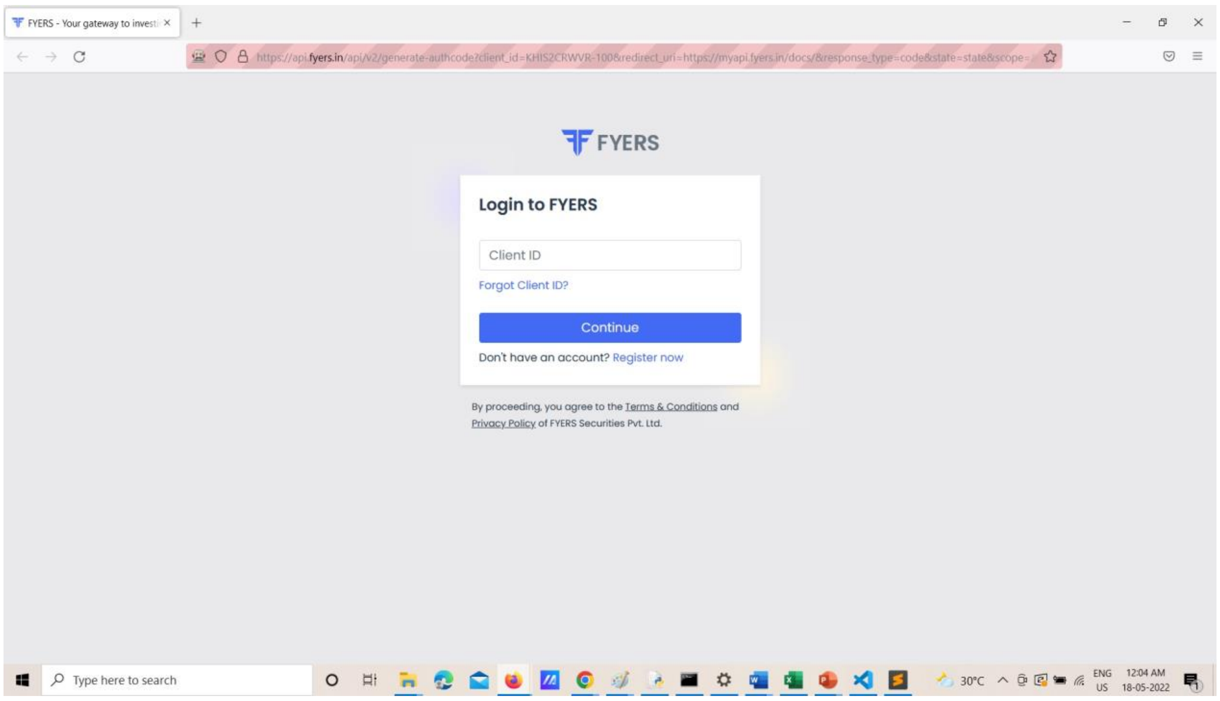
Within the application – We need to ensure those wrong parameters are not set by the trader. It should not allow a trader to set grossly incorrect values nor any fat-finger errors.

Before generating an order in OMS – Before the order flows out of the system we need to make sure it goes through some risk management system. This is where the most critical risk management check happens.

Besides all the internal functions automated login system are also being used so that user did not have to login daily before market opens. Apart from the automated login the user can also schedule their trading system so that it can function automatically on a daily basis.

User can also store historical data in database for future references and they can also receive updates about the buying or selling of the stocks through Telegram, WhatsApp etc.

Screenshots :


FYERS - Your gateway to investi
https://api.fyers.in/api/v2/generate-authcode?client_id=KHIS2CRWVR-100&redirect_uri=https://myapi.fyers.in/docs/&response_type=code&state=state&scope=
FYERS
Type here to search
30°C
ENG US
12:04 AM 18-05-2022

Login to FYERS

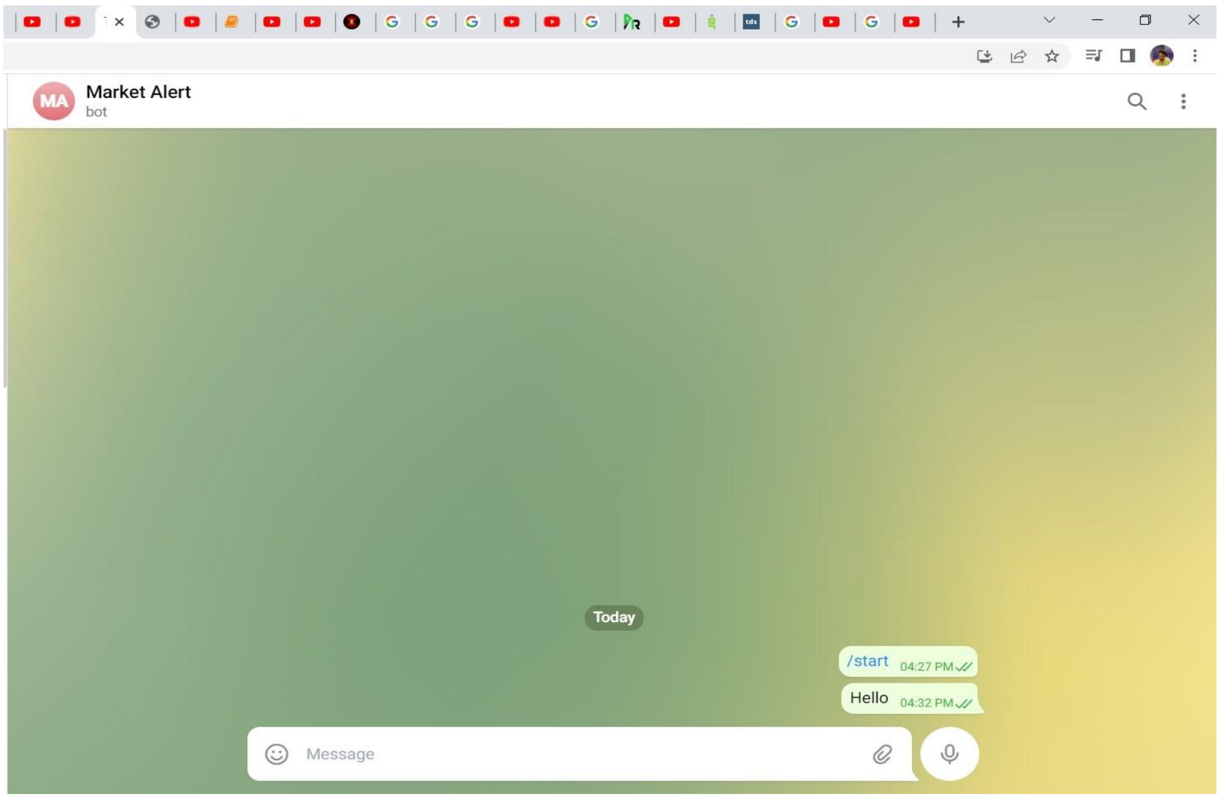
Client ID

Forgot Client ID?

Continue

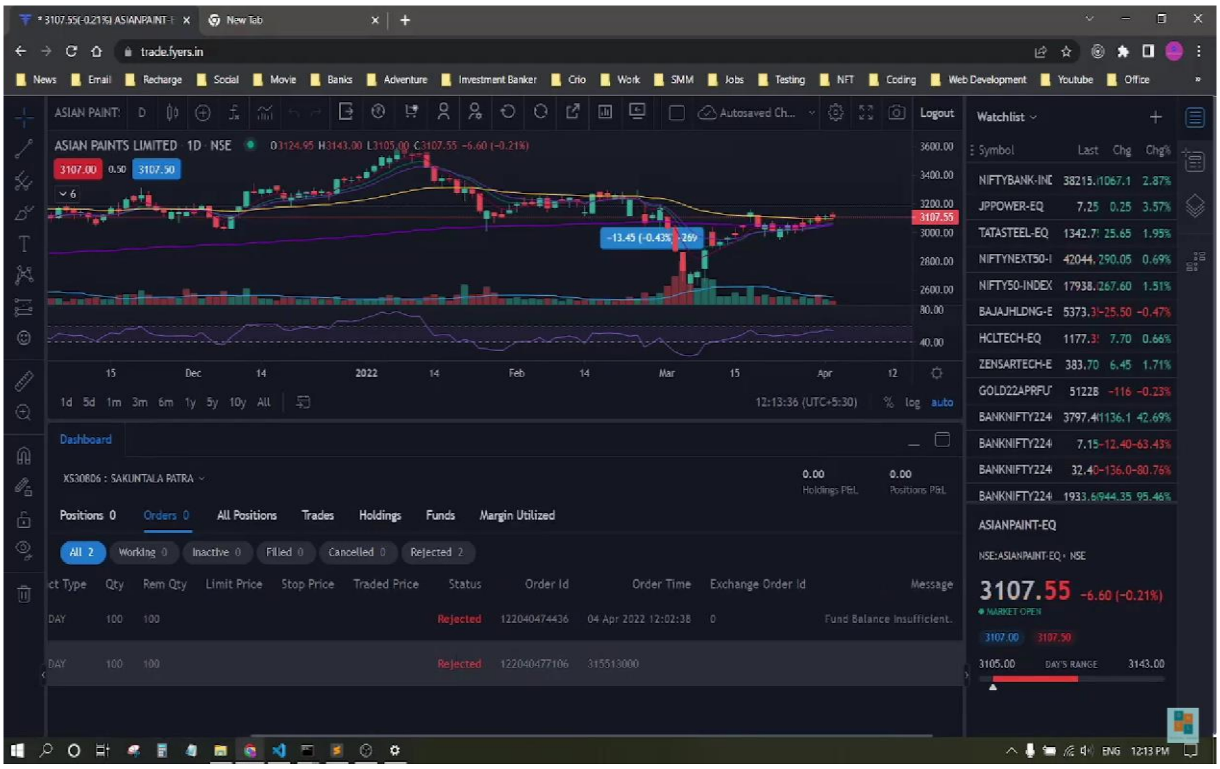
Don't have an account? Register now

By proceeding, you agree to the Terms & Conditions and Privacy Policy of FYERS Securities Pvt. Ltd.



/start 04:27 PM

Hello 04:32 PM



XS30806 : SAKUNTALA PATRA

0.00 Holdings P&L 0.00 Positions P&L

ct Type	Qty	Rem Qty	Limit Price	Stop Price	Traded Price	Status	Order Id	Order Time	Exchange Order Id	Message
DAY	100	100				Rejected	122040474436	04 Apr 2022 12:02:38	0	Fund Balance Insufficient.
DAY	100	100				Rejected	122040477106	315513000		

Symbol	Last	Chg	Chg%
NIFTYBANK-INC	38215.	1067.1	2.87%
JPPOWER-EQ	7.25	0.25	3.57%
TATASTEEL-EQ	1342.7	25.65	1.95%
NIFTYNEXT50-I	42044.	290.05	0.69%
NIFTY50-INDEX	17938.	267.60	1.51%
BAJAJHLDNG-E	5373.3	-25.50	-0.47%
HCLTECH-EQ	1177.3	7.70	0.66%
ZENSARTECH-E	383.70	6.45	1.71%
GOLD22APRFU	51228	-116	-0.23%
BANKNIFTY224	3797.4	1136.1	42.69%
BANKNIFTY224	7.15	-12.40	-63.43%
BANKNIFTY224	32.40	-136.0	-80.76%
BANKNIFTY224	1933.6	944.35	95.46%

ASIANPAINT-EQ
NSE:ASIANPAINT-EQ · NSE
3107.55 -6.60 (-0.21%)
MARKET OPEN
3107.00 3107.50
3105.00 DAY'S RANGE 3143.00

CHAPTER -5: CONCLUSION AND FUTURE WORK

Algorithm trading is all about the use of computers programs to automate one or more stages of the trading process: pre trade analysis (data analysis), trading signal generation (buy and sell recommendations), and trade execution. Today, Algorithmic trading is amongst the most talked about technologies in the recent years. It has given trading Firms more power in the rapidly evolving markets by eliminating human errors and changing the way financial markets are interlinked today. Its usage is credited to most markets and even to commodity trading. For this project we will use python and machine learning. Machine learning models are becoming increasingly prevalent in algorithmic trading and investment management. The spread of machine learning in finance challenges existing practices of modelling and model use and creates a demand for practical solutions for how to manage the complexity pertaining to these techniques. Python has become a preferred choice for trading recently as Python is open-source and all the packages are free for commercial use. Python has gained traction in the quant finance community. Python makes it easy to build intricate statistical models with ease due to the availability of sufficient scientific libraries. The process to retrieve, format and use data is an essential part of trading using Python, as without data there is nothing we can go ahead with. Financial data is available on various online websites. This data is also called as time-series data as it is indexed by time (the timescale can be monthly, weekly, daily, 5 minutely, minutely, etc.). Apart from that, we can directly upload data from Excel sheets too which are in CSV format, which stores tabular values and can be imported to other files and codes.

Future Scope

Algorithmic trading is such a powerful system. As we all know tech has created a huge impact in evolution and development altogether. The future of Algo-trading seems to be really tantalising. All the historical data that we have archived over the course of the entire trading history could be examined by future programs, analysed with ease to figure out the patterns, what would work and what would not. It may also learn to reliably forecast future markets when trading various accounts and strategies to distribute risk and reject or approve real-time bids and offers.

1. Trading systems will use benchmarking to provide intelligence on which algo to use in real-time within the EMS/OMS.

2. Increasing emphasis on real-time tools that will interpret TCA results into practical system configurations.

3. Core algos will become more intelligent and reactive to market conditions.

4. Algos will expand across asset classes - including cross-asset automation.

5. Rules will evolve from simple routing instructions to context-based rulebooks measuring market conditions.

REFERENCES

1. https://ieeexplore.ieee.org/abstract/document/5696713/authors#authors

2. http://faculty.haas.berkeley.edu/hender/ATInformation.pdf

3. https://www.activestate.com/blog/how-to-build-an-algorithmic-trading-bot/

4. https://www.datacamp.com/community/tutorials/finance-python-trading

5. https://blog.quantinsti.com/python-trading/

6.https://www.freecodecamp.org/news/how-to-get-started-with-algorithmic-trading-in-python/

7. https://www.wallstreetmojo.com/algorithmic-trading/

8. https://economictimes.indiatimes.com/markets/stocks/news/how-algo-trading-makes-it-easier- to-grow-protect-wealth-in-stock-market/articleshow/86701379.cms?from=mdr

9. https://journals.sagepub.com/doi/full/10.1177/2053951720926558

10. https://ideas.repec.org/p/nig/wpaper/0139.html

11. https://vixra.org/pdf/1912.0492v1.pdf

12