Project Phase IV Report

On

Advanced algorithmic trading system

Submitted for the requirement of

Project course

BACHELOR OF ENGINEERING

COMPUTER SCIENCE & ENGINEERING

Submitted to:
Project Teacher (Supervisor)
Shuvendu Das (12496)

Shuvendu Das

Submitted By:
Student Group

NAME	**UID**
Ayush Raj	**20BCS7576**
Lekhraj Sharma	**20BCS7553**
Shashwat Verma	**20BCS7532**
Shubham Kumar	**20BCS7572**

Co Supervisor Signature
Shikha Atwal

Shikha

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH UNIVERSITY, GHARUAN
June 2022

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.

After that at the end we will have to do Backtesting. Backtesting a trading strategy is the process of testing a trading hypothesis/strategy on the historical data. Let’s say we formed a hypothesis. This hypothesis states that securities that have positive returns over the past one year are likely to give positive returns over the next one month. By using historical data, we can backtest and see whether our hypothesis is true or not. It helps assess the feasibility of a trading strategy by discovering how it performs on the historical data. If we backtest our strategy on the historical data and it gives good returns, we will be confident to trade using it. If the strategy is performing poorly on the historical data, we will discard or re-evaluate the hypothesis.

TABLE OF CONTENT

1. Use of Modern tools in design and analysis

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.

2. Discussion and report/results analysis

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.

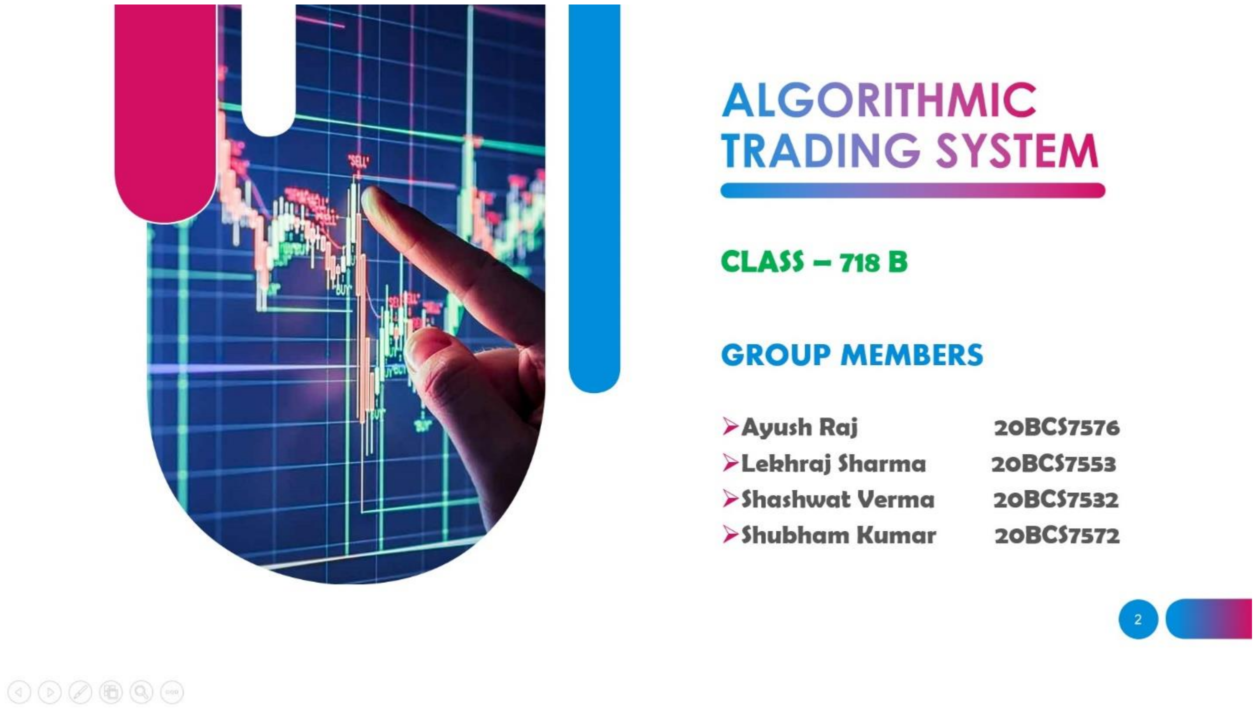
Before generating an order in OMS – Before the order flows out of the system we need to make sure it goes through some risk management system. This is where the most critical risk management check happens.

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.

3. <u>Project management and Professional communication (Presentation)</u>

ALGORITHMIC TRADING SYSTEM

the financial pandora

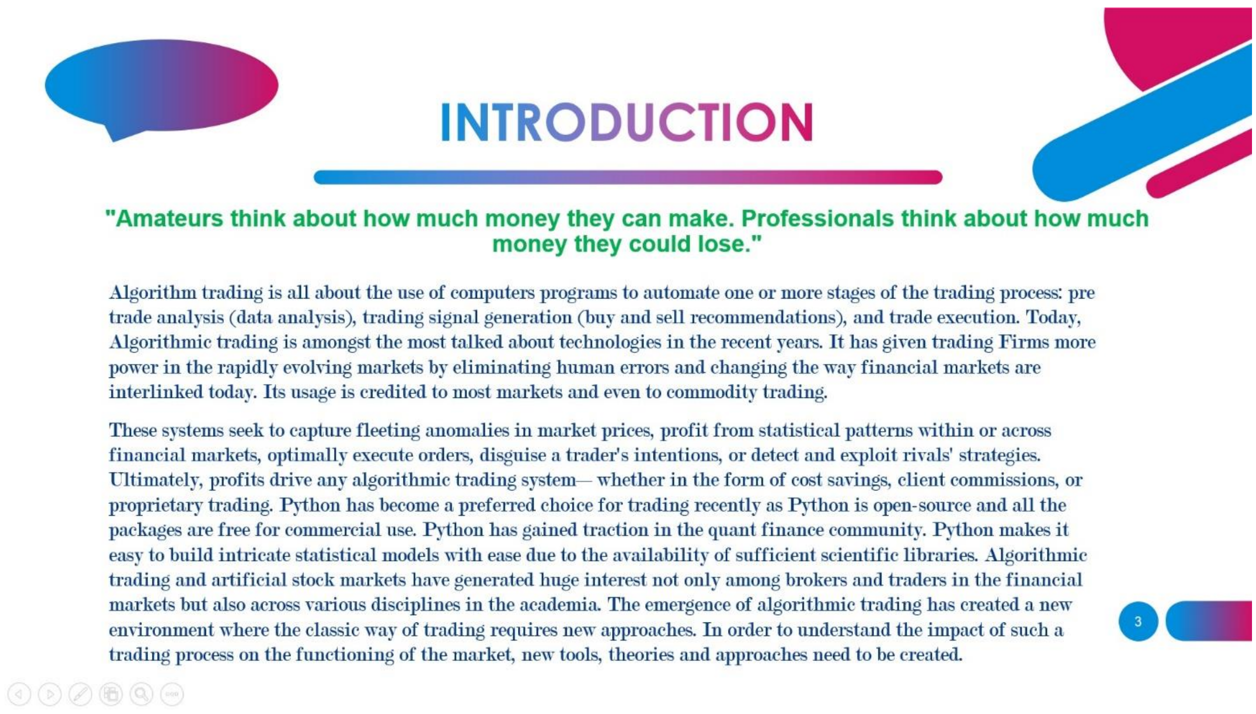

ALGORITHMIC TRADING SYSTEM

CLASS – 718 B

GROUP MEMBERS

➢Ayush Raj	20BCS7576
➢Lekhraj Sharma	20BCS7553
➢Shashwat Verma	20BCS7532
➢Shubham Kumar	20BCS7572

2



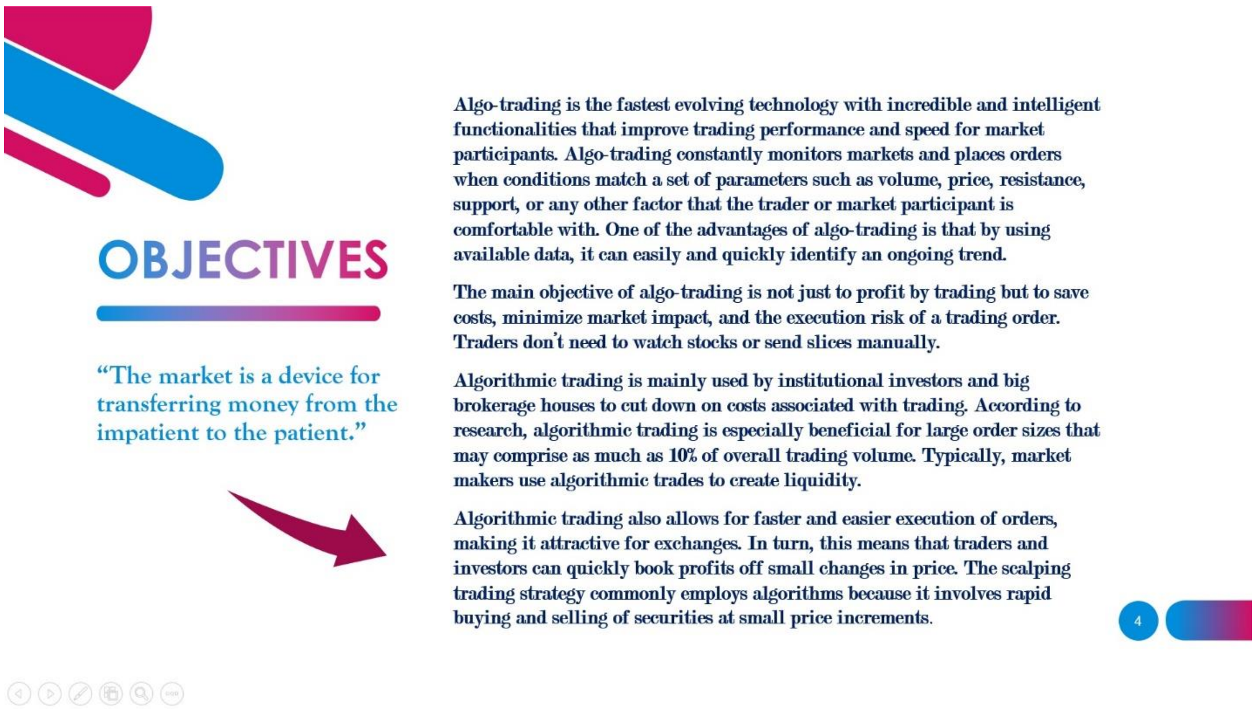
INTRODUCTION

"Amateurs think about how much money they can make. Professionals think about how much money they could lose."

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.

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. 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. 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.

3



OBJECTIVES

“The market is a device for transferring money from the impatient to the patient.”

Algo-trading is the fastest evolving technology with incredible and intelligent functionalities that improve trading performance and speed for market participants. 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.

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.

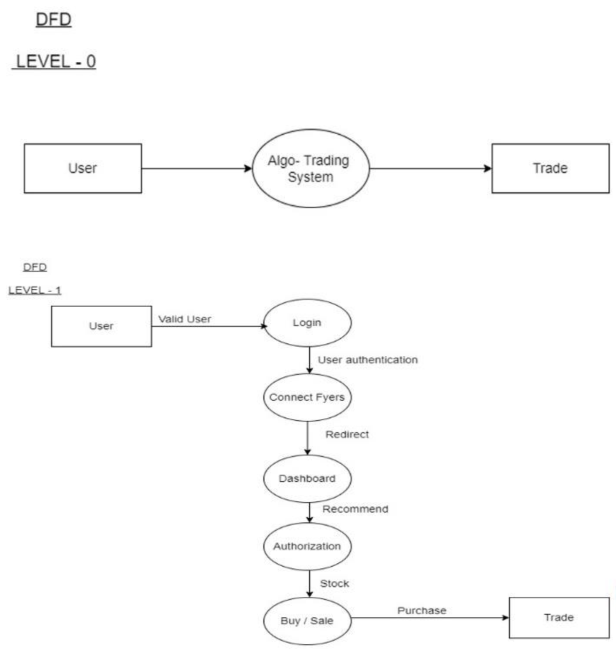
4

TECHNOLOGY USED

- ➢Python
- ➢FYERS API Bridge
- ➢Websocket
- ➢Selenium WebDriver
- ➢WebDriver Manager
- ➢Pandas
- ➢Ta-Lib
- ➢Flask

DESIGNS (Diagrams)

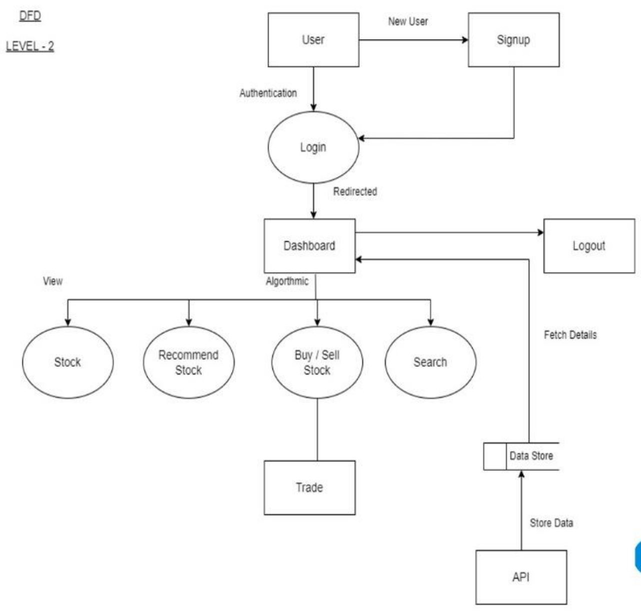
DFD (LEVEL – 0 & LEVEL – 1)



6

DESIGNS (Diagrams)

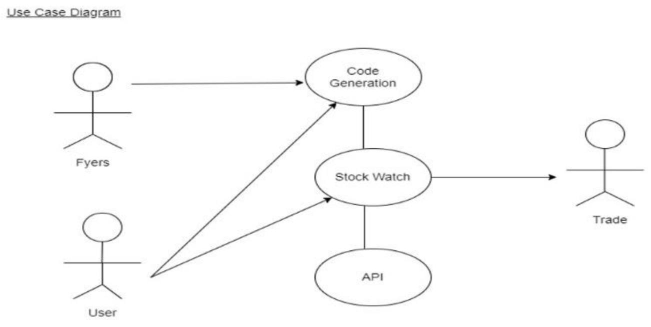
DFD (LEVEL – 2)

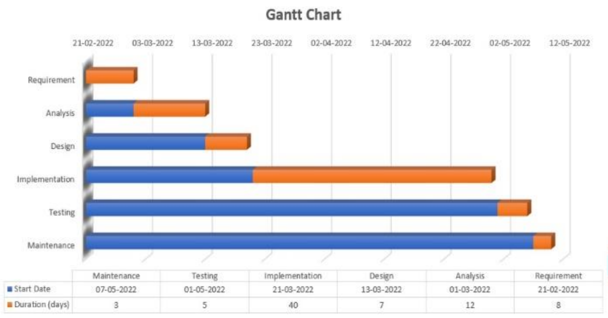


7

DESIGNS (Diagrams)

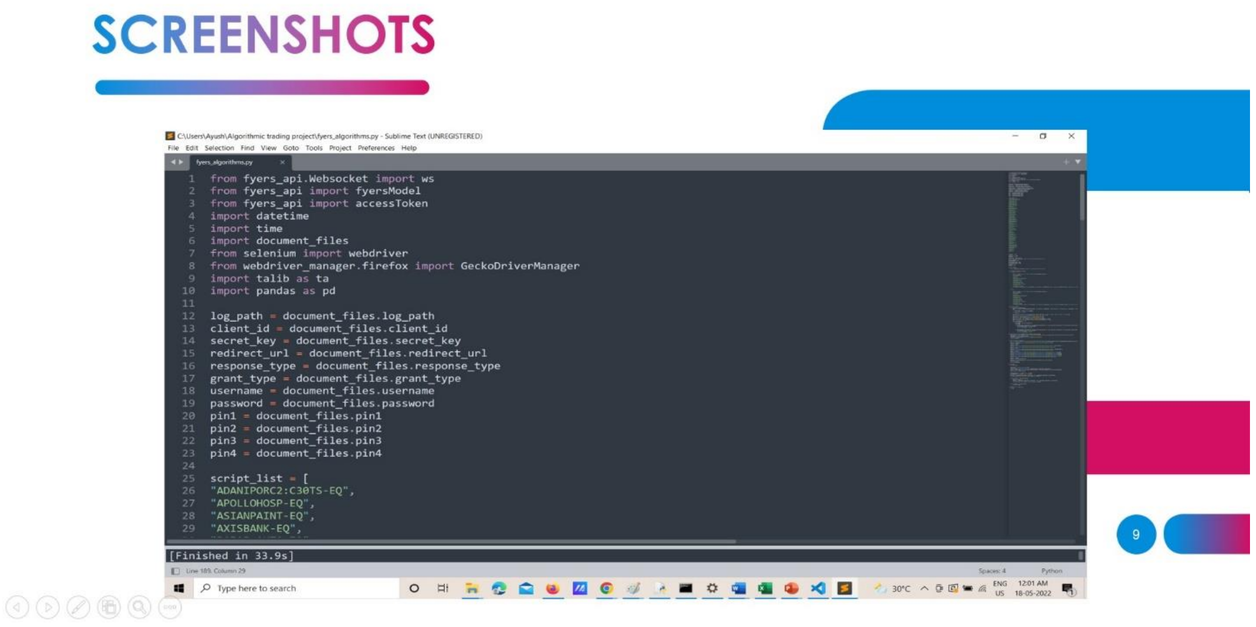
USE CASE DIAGRAM & GANTT CHART





	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

8



SCREENSHOTS

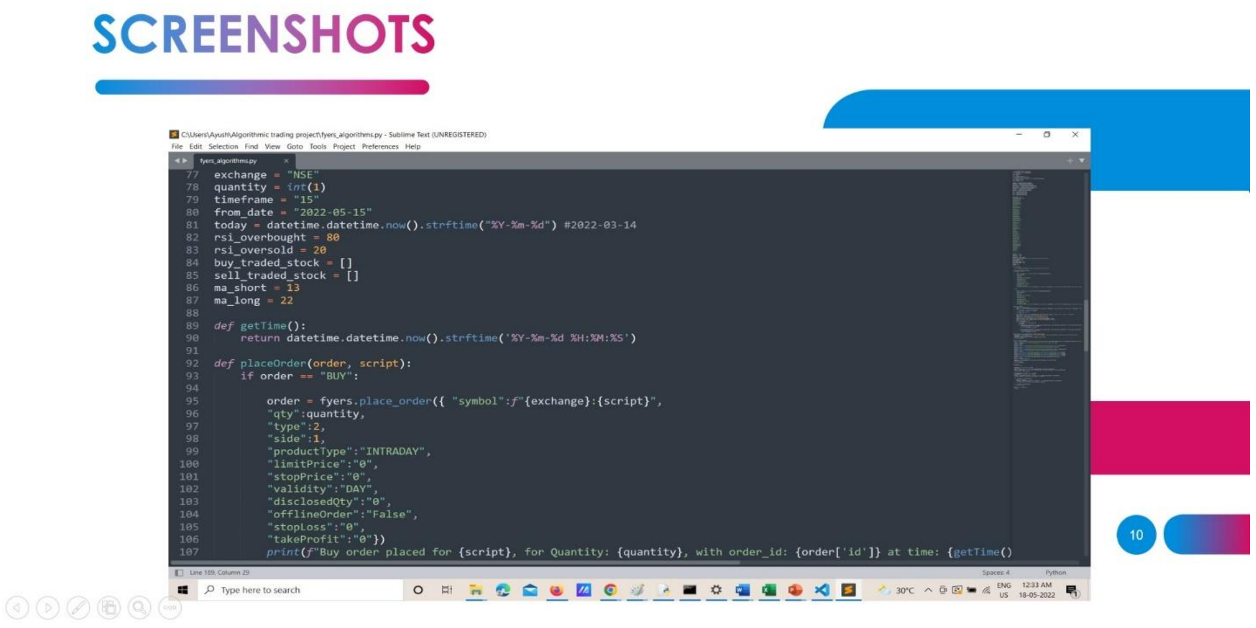
C:\Users\Ayush\Algorithmic trading project\fyers_algorithms.py - Sublime Text (UNREGISTERED)

```python
from fyers_api.Websocket import ws
from fyers_api import fyersModel
from fyers_api import accessToken
import datetime
import time
import document_files
from selenium import webdriver
from webdriver_manager.firefox import GeckoDriverManager
import talib as ta
import pandas as pd

log_path = document_files.log_path
client_id = document_files.client_id
secret_key = document_files.secret_key
redirect_url = document_files.redirect_url
response_type = document_files.response_type
grant_type = document_files.grant_type
username = document_files.username
password = document_files.password
pin1 = document_files.pin1
pin2 = document_files.pin2
pin3 = document_files.pin3
pin4 = document_files.pin4

script_list = [
"ADANIPORC2:C30TS-EQ",
"APOLLOHOSP-EQ",
"ASIANPAINT-EQ",
"AXISBANK-EQ",
```

```
[Finished in 33.9s]
```



SCREENSHOTS

C:\Users\Ayush\Algorithmic trading project\fyers_algorithms.py - Sublime Text (UNREGISTERED)

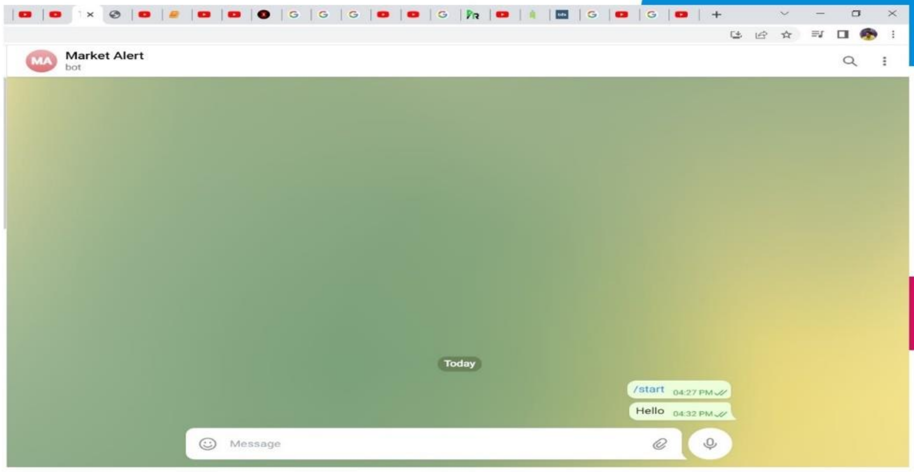
```python
exchange = "NSE"
quantity = int(1)
timeframe = "15"
from_date = "2022-05-15"
today = datetime.datetime.now().strftime("%Y-%m-%d") #2022-03-14
rsi_overbought = 80
rsi_oversold = 20
buy_traded_stock = []
sell_traded_stock = []
ma_short = 13
ma_long = 22

def getTime():
    return datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

def placeOrder(order, script):
    if order == "BUY":

        order = fyers.place_order({ "symbol":f"{exchange}:{script}",
        "qty":quantity,
        "type":2,
        "side":1,
        "productType":"INTRADAY",
        "limitPrice":"0",
        "stopPrice":"0",
        "validity":"DAY",
        "disclosedQty":"0",
        "offlineOrder":"False",
        "stopLoss":"0",
        "takeProfit":"0"})
        print(f"Buy order placed for {script}, for Quantity: {quantity}, with order_id: {order['id']} at time: {getTime()
```

SCREENSHOTS

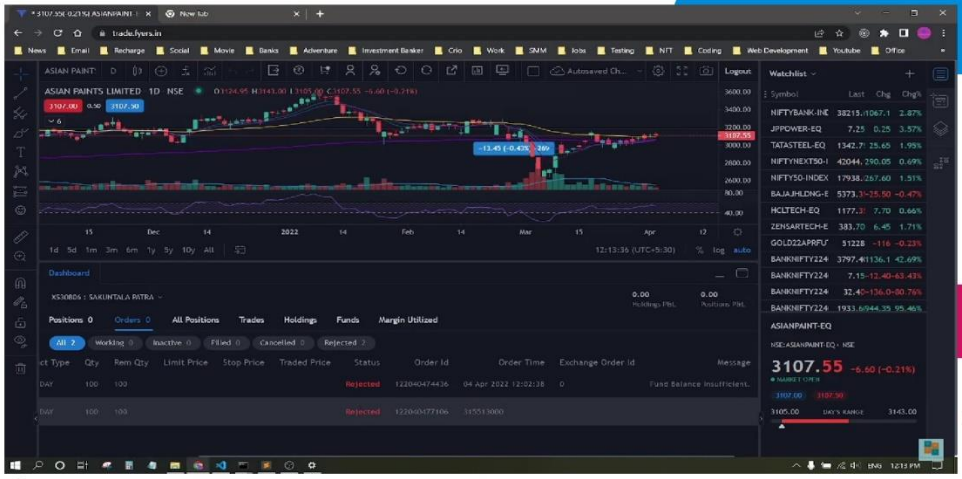


/start 04:27 PM

Hello 04:32 PM

12

SCREENSHOTS



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-INE	38215.1	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

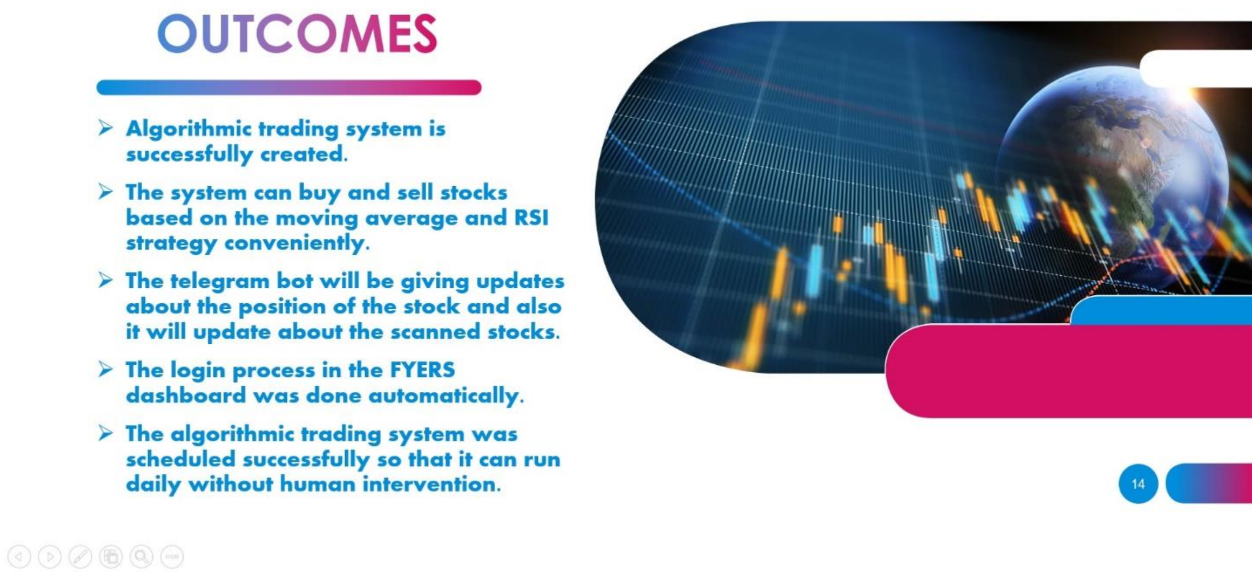
NSE:ASIANPAINT-EQ · NSE

3107.55 -6.60 (-0.21%)

MARKET OPEN

3105.00 DAY'S RANGE 3143.00

13



OUTCOMES

- Algorithmic trading system is successfully created.
- The system can buy and sell stocks based on the moving average and RSI strategy conveniently.
- The telegram bot will be giving updates about the position of the stock and also it will update about the scanned stocks.
- The login process in the FYERS dashboard was done automatically.
- The algorithmic trading system was scheduled successfully so that it can run daily without human intervention.

14

4. Attainment of stated outcomes

- Algorithmic trading system is successfully created.
- The system can buy and sell stocks based on the moving average and RSI strategy conveniently.
- The telegram bot will be giving updates about the position of the stock and also it will update about the scanned stocks.
- The login process in the FYERS dashboard was done automatically.
- The algorithmic trading system was scheduled successfully so that it can run daily without human intervention.

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/