

Industrial Training on Algorithmic Trading on Kite SUBMITTED BY

Jayasuryan Mutyala

210962009

Roll no:5

CSE(AI&ML) - B

jsmu.dev@gmail.com

9902358432

Under the Guidance of:

Mani Katta

CEO

Department

Shivansh Solutions Asia



MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

Department of Computer Science and Engineering

July 2024

Internship Certification



Shivansh Solutions (ASIA) Ltd
Rm 1003 Yu Yuet Lai Building,
43-55 Wyndham Street, Central
Hong Kong
Phone: +852-51182457
www.shivanshk.com

Date: July 15, 2024

Certification Of Completion Of Internship

This is to certify that **Mr. JAYASURYAN MUTYALA (210962009)**, a student of **B.Tech (CSE AI & ML)**, **Manipal Institute of Technology, Manipal** has successfully completed his internship program at **Shivansh Solutions (Asia) Limited**. His internship tenure was from **15th May 2024 to 15th July 2024** and presented his work on **15th July 2024**. He has been rigorously trained in **AlgoTrading using the KiteConnect API** and he executed all the assigned tasks swiftly.

During the span, we found him a punctual and hardworking person. His learning powers are good, and he picks up swiftly. His feedback and evaluation proved that he learned keenly. Moreover, his interpersonal and communication skills are brilliant.

We wish him a bright future.

For Shivansh Solutions (Asia) Limited, Hong Kong



Authorized Signatory

1. Acknowledgements

I am deeply grateful for the transformative experience I had during my two-month summer internship at Shivansh Solutions Asia. This journey was very knowledgeable and transformative, and I owe my appreciation to numerous individuals and the supportive environment at Shivansh Solutions.

First and foremost, I extend my heartfelt thanks to my mentors and supervisors. Their guidance, patience, and willingness to share their expertise have been invaluable. Their mentorship has not only honed my technical skills but also deepened my understanding of the industry. I am very grateful for their constructive feedback, which has significantly contributed to my personal and professional growth.

I also wish to thank my colleagues for warmly welcoming me into the team. Their collaborative spirit, readiness to answer my questions, and the opportunity to engage in meaningful discussions have made the internship truly rewarding. The workforce culture at Shivansh Solutions created a conducive learning environment, and I am grateful for the camaraderie I experienced.

Additionally, I must acknowledge the organizational support that ensured a seamless internship experience. The efforts of the administrative staff, HR team, and everyone involved in coordinating this internship deserve recognition.

Lastly, I want to express my gratitude to my family and friends for their unwavering support throughout this journey. Their encouragement and belief in me motivated me to give my best.

As I conclude my internship and reflect on this experience, I leave with a wealth of knowledge, memorable experiences, and a network of wonderful people. My time at Shivansh Solutions has been instrumental in shaping my aspirations and has further fueled my passion for technology and innovation.

Thank you, Shivansh Solutions, for this incredible opportunity. I look forward to the future with newfound confidence and enthusiasm.

2. Abstract

This Industrial Training involved designing and developing an algorithmic trading strategy using Zerodha's Kite API. Kite is a powerful and user-friendly trading platform developed by Zerodha, India's leading stock brokerage. It offers a seamless trading experience with advanced charting, technical analysis tools, and a robust API for algorithmic trading, catering to both beginner and professional traders.

The project integrates price action concepts, along with the use of support-resistance lines derived from historical stock data to generate buy and sell signals. By automating the trading strategy, it enables traders prevent traders confiding with their emotions while trading, which plays a critical role in stock markets as it is largely driven by human behavior. By placing automated orders for long positions (buy the asset expecting an increase in value), the algorithm aims to maximize profits while managing risk through predefined stop loss and take profit levels.

Table Of Contents

Internship Certification	2
1. Acknowledgements	3
2. Abstract	4
3. Details of the Organization	6
4. Problem Statement	7
4.1 Technologies Used	8
5. Description	9
5.1 Methodology	14
5.1.1 Automating user login and retrieving the access token	15
5.1.2 Setting up the retrieval functions for the historic data	16
5.1.3 Computing the price action patterns	16
5.1.4 Computing the support and resistance levels	20
5.1.5 Simulating orders on historic data	24
5.1.6 Performing the back-test simulation using Backtesting.py	25
5.1.6 Placing orders on live streaming data using web sockets	27
6. Conclusion	28
6.1 Environmental and Societal Impact	29
7. Ethics in Engineering Certification	29
8. NBA/IET Mapping	30
9. References	35

3. Details of the Organization

Shivansh Solutions is an IT service provider based in Hong Kong, established in 2011 by experienced SAP professionals. The company specializes in expert consulting and tailored solutions for the intricate world of SAP implementation, support, and related assignments. Recently, Shivansh has expanded its focus to include Cloud Computing, Data Analytics, and Program Management. The company is dedicated to meeting clients' needs, constraints, and objectives by delivering customized solutions with flexible usability features. They collaborate closely with clients to help them become high-performance businesses, simplifying complexities to provide solutions that facilitate smooth change management.

In previous projects, Shivansh Solutions has delivered a comprehensive range of SAP services, including SAP implementation, offshore and onsite support, business process consultancy, SAP system audit, business intelligence, expert consulting, custom developments, workflow, data migration, and more. They have a robust team of cloud consultants in Hong Kong and India, whose primary goal is to assist clients with their cloud migration strategies. The company is also actively expanding in the fields of data analytics and program management. Shivansh Solutions was formed with the shared vision of ensuring that the newer generation is well-versed in these technologies.

4. Problem Statement

The main problem statement addressed is as follows:

- Design and develop an algorithmic trading strategy using:
 - **Price Action Concepts:** Hammer, Bullish engulfing, Shooting star and Bearish engulfing.
 - **Support and Resistance lines** using fractal candlestick patterns
- The trading algorithm is designed based on the fundamental price action concepts and support, resistance lines. While placing a buy or sell order:
 - **Buy Order:** The algorithm looks for possible entry points to buy when the closing price is near the support line, and it examines if there is a hammer or bullish engulfing candlestick pattern.
 - **Sell Order:** The algorithm looks for possible exit points to buy when the closing price is near the resistance line, and it examines if there is a shooting star or bearish engulfing candlestick pattern. Additionally, there are predefined parameters for take profit and stop-loss so when the closing price meets those target points it will place a sell order.
- Perform backtesting using historical stock market data to evaluate the trading algorithms performance with respect to a few statistics.
- Deploy the trading strategy locally on the system.

4.1 Technologies Used

- **Python:** Python is the core programming language used for this project due to its simplicity, versatility, and extensive libraries for data analysis and automation. It facilitates seamless integration of various tools and frameworks necessary for implementing and testing the trading strategy.
- **Jupyter Notebook:** Jupyter Notebook is an interactive environment that allows for code execution, visualization, and documentation in a single interface. It is used for developing, testing, and demonstrating the trading strategy, enabling real-time feedback and iterative improvement.
- **Pandas:** Pandas is a powerful data manipulation and analysis library in Python. I used it to process stock market data, enabling efficient data analysis, signal generation, and performance tracking within the trading strategy. I primarily used this library for data analysis, computations for candlestick patterns and visualizations.
- **Matplotlib:** Matplotlib is a plotting library for Python, used to visualize data. This was mainly used for data visualization of the stocks closing prices, support-resistance and additionally to visualize the buy sell signals generated over.
- **KiteConnect :** KiteConnect is a trading API provided by Zerodha, enabling automated trading through programmatic access to market data and order management. It is used to fetch real-time market data, execute trades, and manage positions, forming the backbone of the live trading functionality.
- **Selenium:** Selenium is a browser automation tool used for web scraping and testing web applications. In this project, I used to automate the user-login in Zerodha retrieve the access token in order to execute the basic functions such as placing orders and the
- **NumPy:** NumPy is a fundamental package for numerical computation in Python. It supports efficient array operations and mathematical functions, which are crucial for technical analysis, signal processing, and backtesting of the trading strategy, ensuring high performance and accuracy.

- **Backtesting.py:** Backtesting.py is a Python library for simulating and testing trading strategies on historical data. It is used to validate the trading strategy by running it against past market data, allowing for performance evaluation and optimization before deploying it in a live trading environment.

5. Description

Developing successful trading techniques is an ongoing endeavour in the dynamic and frequently volatile realm of financial markets. To enhance technical analysis and maximize trade entries and exits, this project presents a novel trading method that makes use of support and resistance lines in conjunction with candlestick patterns. The approach is intended to optimize returns while skilfully minimizing risks, with a primary focus on long holdings.

Technical analysis in the stock market involves evaluating securities by analyzing statistical trends from trading activity, such as price movements and volume. Unlike fundamental analysis, which assesses a company's financial health and market position, technical analysis focuses on identifying patterns and trends in price charts to predict future market behavior. Key tools and techniques include moving averages, trend lines, and chart patterns like head and shoulders or candlestick formations. Technical analysts believe that historical trading data can offer valuable insights into future price movements, as market psychology and behaviors often repeat over time. This method is widely used by traders to make informed buy and sell decisions, aiming to capitalize on short-term market fluctuations and optimize their trading strategies.

The aim of the project is to utilise the price-action concepts and support-resistance to generate buy and sell signals. In stock markets, price action concepts relate to examining and deciphering price movement on charts without depending on fundamental data or indicators. When making trading decisions, traders pay close attention to patterns, trends, and important levels like support and resistance. Candlestick patterns, trend lines, and chart formations like double tops and bottoms are examples of common price action tools. With this method, traders can forecast future movements based on past price behaviour and market psychology. It is based on the idea that all relevant market information is reflected in the price itself.

The project makes use of various price action concepts:

- **Hammer candles:** The prior candles follow a downward trend; the hammer is a bullish reversal pattern. Its modest body at the upper end of the trading range and long lower wick suggest that although buyers were able to drive prices back up towards the opening level during the session, sellers drove prices lower.



Fig1.1: Hammer candlestick

- **Bullish engulfing candle:** Two candlesticks make up this pattern. A smaller bearish candle appears first, and then a larger bullish candle engulfs the body of the earlier candle. Strong purchasing interest and a possible reversal from a downward to an upward trend are indicated by this pattern.

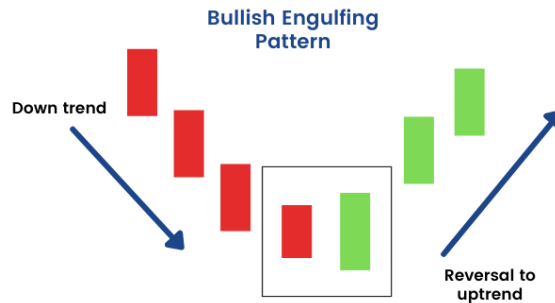


Fig1.2: Bullish Engulfing candlestick

- **Shooting Star Pattern:** The prior candles follow an upward trend, the shooting star pattern is a bearish reversal pattern. Its long upper wick and small body toward the lower end of the trading range indicate that although buyers initially drove prices higher, sellers later drove the back down to the opening level.



Fig1.3:Shooting Star candlestick

- **Bearish Engulfing Pattern:** This pattern is an inversion of the bullish engulfing pattern and is characterized by a small bullish candle that is followed by a larger bearish candle that engulfs the body of the preceding candle. It suggests intense selling pressure and the possibility of an upswing turning into a decline.

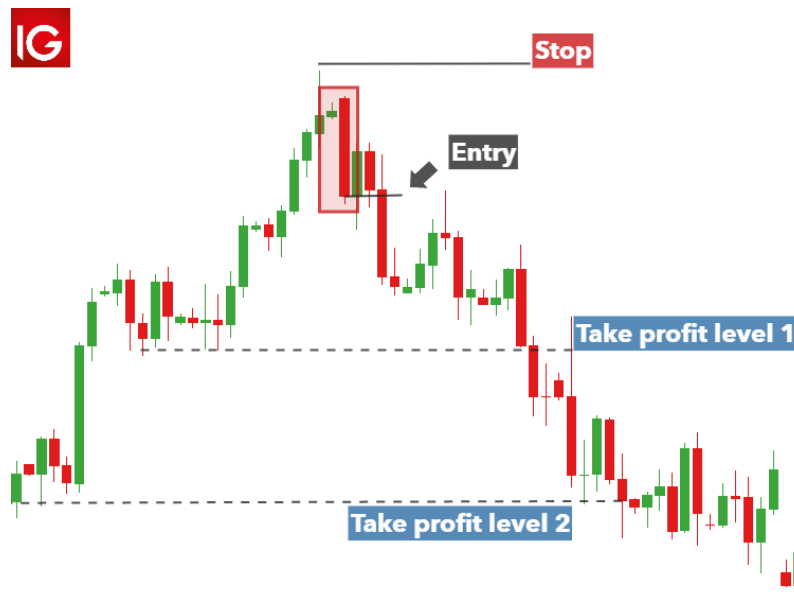


Fig1.4: Bearish Engulfing candlestick

- **Fractal candlesticks:** Fractal candles in trading are a technical analysis tool used to identify potential reversal points in the market. Developed by Bill Williams, fractals consist of a series of at least five consecutive candlesticks where the middle candlestick has the

highest high or lowest low, forming a distinctive pattern. An up fractal indicates a potential sell point, while a down fractal suggests a potential buy point. Fractals help traders spot key turning points and confirm trend reversals, often used in conjunction with other indicators like the Alligator to enhance trading strategies and improve decision-making.



Fig 1.5: Fractal Candlestick pattern

- **Support Line**



Fig1.6: Support Line

Support is a price action zone that stops the price from falling any lower. On the chart, the support level denotes a price where traders expect a substantial increase in demand (buying interest) for the stock or index. It is anticipated that the price would rise after it hits this support line. Interestingly, the support level is consistently located below the going rate in the market. It is highly likely that the price will drop to the support level, level off, take in all the demand, and then start to rise. Support is an important technical level that traders keep a careful eye on in a dropping market. It frequently triggers purchases.

- **Resistance Line**

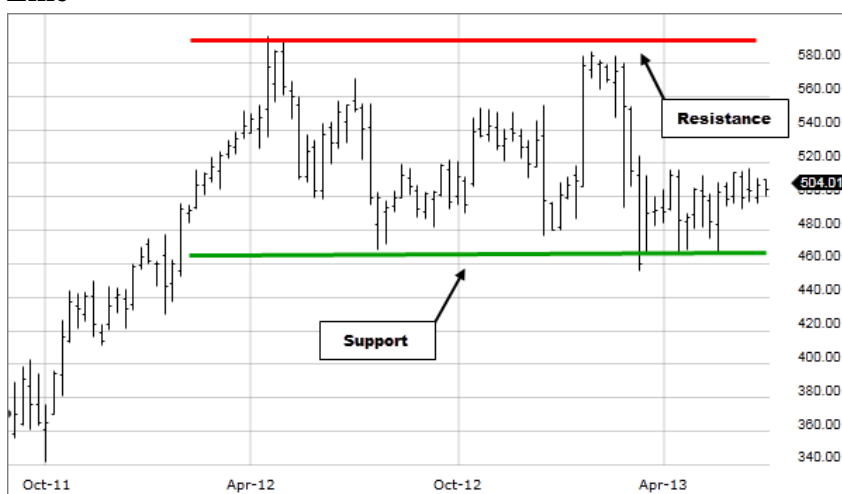


Fig 1.7: Resistance Line

Resistance is a price action zone that prevents the price from rising. The resistance level is a point the chart where the traders expect to see significant amount of supply (high selling power of bears) for the stock. The resistance zone is always above the current market closing price. There is a strong probability that the price will ascend to the resistance level, consolidate, absorb the available supply, and then start to decline. In a rising market, resistance is a key technical level that market participants closely observe. It frequently acts as a signal to sell. Fractal candlestick patterns were used to determine the support and resistance zones and filtering mask was used to filter out excessive lines.


5.1 Methodology

The code is available at GitHub: <https://github.com/Jayasuryan0821/AlgoTrading-on-Kite/tree/main/Final%20Project>

The proposed methodology for the project is as follows:

1. Automation of login in Zerodha to set up the access token which enables traders to place buy and sell orders.
2. Setting up instrument token retrieval functions and retrieving historical data.
3. Computing the price action patterns for hammer, bullish engulfing, shooting star and bearish engulfing.
4. Computing the support and resistance levels using fractal candlestick pattern.
5. Examining the entry points to place buy and sell orders based on the defined conditions simulate this based on historic market data.
6. Perform back-testing on historical data to evaluate the trading algorithm's performance.
7. Execute the trading algorithm on live market data,

5.1.1 Automating user login and retrieving the access token



zerodha_api_test
Kite connect + Publisher (Created on 24 may 2024)

App details

App name:

zerodha_api_test

App icon:

Choose File

No file chosen
64x64 px PNG images only

Redirect URL:

https://127.0.0.1

The url to which a user will be redirected after the login flow for Kite Connect. This has to be an HTTPS url. You can use a http://127.0.0.1 URL for testing. For Publisher, the redirect url does not matter until you want to do advanced offsite basket execution.



Postback URL:

https://

The URL where realtime order completion updates for orders placed by you will be POSTed. This has to be an HTTPS url.


You have 1 app(s).

My apps

	API Key	Expiry
 <div>zerodha_api_test Kite connect + Publisher</div>		2 days 25 Jul 2024


Activate app

API key



Show API secret

Re-generate API secret

 Delete this app

Create new app

Fig 2.0 Configuring the kiteconnect api key

Automating login process for Zerodha and retrieving the access token is done by using Selenium WebDriver and the KiteConnect API. Initially, I set up the necessary credentials in a file, including the API key, API secret, username, password, and TOTP key. The auto_login function then

initiates the KiteConnect API with the API key and configures the Chrome WebDriver using ChromeDriverManager. The WebDriver navigates to the Zerodha login page, inputs the username and password, and waits briefly before entering the TOTP generated from the provided key. Upon successful login, the script extracts the request_token from the URL and saves it to a file, then terminates the WebDriver session.

The next step involves generating the access token. The generate_access_token function reads the saved request token and uses it along with the API key and secret to create a session through the KiteConnect API. This session generation process results in an access token, which is then saved to a file for future API calls. By automating the login process, we can eliminate the need for manual entry, making it efficient and user-friendly for traders who need to authenticate and interact with the Zerodha API regularly.

5.1.2 Setting up the retrieval functions for the historic data

I defined a few functions called tokenLookup, tickerLookup and fetchOHLC.

- The tokenLookup function returns a list of instrument tokens for given trading symbols from a DataFrame, while tickerLookup fetches the trading symbol for a given token.
- The instrumentLookup function retrieves the instrument token for a given symbol, returning -1 if not found.
- The fetchOHLC function extracts historical Open, High, Low, and Close (OHLC) data for a given ticker and duration, outputting it as a DataFrame.
- The fetchOHLCExtended function performs a similar task but handles larger data ranges by iteratively fetching data in 100-day chunks from a specified inception date

5.1.3 Computing the price action patterns

- **Hammer:** I applied several conditions: the range between the high and low prices must be more than three times the difference between the open and close prices, the close and open prices must be closer to the high, and the body length must be significant, at least 10% of the total range. These conditions ensure the identification of a hammer pattern, characterized by a small body and a long lower shadow. If all the conditions are satisfied we return 1 or if it fails to satisfy it returns 0.


```
def hammer(df):
    """returns dataframe with hammer candle column"""
    df = df.copy()
    df["hammer"] = (((df["high"] - df["low"]) > 3 * (df["open"] - df["close"])) & \
                    ((df["close"] - df["low"]) / (.001 + df["high"] - df["low"]) > 0.6) & \
                    ((df["open"] - df["low"]) / (.001 + df["high"] - df["low"]) > 0.6)) & \
                    (abs(df["close"] - df["open"]) > 0.1 * (df["high"] - df["low"]))).astype(int)
    return df
```

Fig 2.1: Computing hammer candlestick pattern

```
1 signals_df[signals_df['hammer'] == 1].head()
```

	open	high	low	close	volume	hammer
date						
2023-02-10 00:00:00+05:30	1612.95	1616.30	1596.45	1608.55	4786762	1
2023-03-08 00:00:00+05:30	1490.55	1496.00	1481.10	1492.70	4137080	1
2023-04-17 00:00:00+05:30	1250.30	1261.15	1185.30	1258.30	53171705	1
2023-04-24 00:00:00+05:30	1228.00	1230.00	1218.00	1226.30	8313774	1
2023-06-27 00:00:00+05:30	1280.00	1282.00	1274.20	1279.15	3240024	1

Fig 2.2: Hammer candlestick signals which indicate potential buy signals

- **Shooting star:** shooting star candlestick patterns is computed based on specific conditions: the total range (high - low) must be more than three times the body length (open - close), the upper shadow (high - close and high - open) must be more than 60% of the total range, and the body length must be significant, at least 10% of the total range. These conditions ensure the identification of a shooting star pattern, characterized by a small body near the low and a long upper shadow. If all the conditions are satisfied, we return 1 or if it fails to satisfy it returns 0.

```
def shooting_star(df):
    """returns dataframe with shooting star candle column"""
    df = df.copy()
    df["shooting_star"] = (((df["high"] - df["low"]) > 3 * (df["open"] - df["close"])) & \
                            ((df["high"] - df["close"]) / (.001 + df["high"] - df["low"]) > 0.6) & \
                            ((df["high"] - df["open"]) / (.001 + df["high"] - df["low"]) > 0.6)) & \
                            (abs(df["close"] - df["open"]) > 0.1 * (df["high"] - df["low"]))).astype(int)
    return df
```

Fig 2.3: Computing the shooting star candlestick signals

```
1 signals_df[signals_df['shooting_star'] == 1].tail()
```

date	open	high	low	close	volume	hammer	shooting_star
2024-05-06 00:00:00+05:30	1420.00	1445.7	1417.10	1425.90	6445422	0	1
2024-05-18 00:00:00+05:30	1445.00	1450.0	1442.00	1443.65	318277	0	1
2024-05-21 00:00:00+05:30	1430.00	1444.7	1428.40	1434.15	6752663	0	1
2024-06-18 00:00:00+05:30	1497.00	1507.0	1495.40	1498.20	5420939	0	1
2024-06-28 00:00:00+05:30	1572.55	1588.5	1564.25	1566.75	8197544	0	1

Fig 2.4: Shooting star signals that indicate potential cell signals

- Bullish and Bearish Engulfing:** I initialized lists to store the high, low, close, and open prices, as well as a signal list indicating the pattern type: 0 for no engulfing, 1 for bearish, and 2 for bullish. We calculate the body length of the candle (absolute difference between open and close prices) and ensure it exceeds a minimum threshold (0.003). For a bearish engulfing pattern, it checks that the previous candle is bullish (open < close) and the current candle is bearish (open > close), with the current open price not just matching but exceeding the previous close price, and the current close price being below the previous open price. For a bullish engulfing pattern, it verifies that the previous candle is bearish (open > close) and the current candle is bullish (open < close), with the current open price not just matching but falling below the previous close price, and the current close price exceeding the previous open price. The detected patterns are then stored in a signal list.

```

def engulfing_signal(df):
    df = df.copy()
    length = len(df)
    high = list(df['high'])
    low = list(df['low'])
    close = list(df['close'])
    open = list(df['open'])
    signal = [0] * length # [0,1,2] 0: no engulfing, 1: bearish, 2: bullish
    bodydiff = [0] * length # abs diff of the close and opening

    for row in range(1, length):
        bodydiff[row] = abs(open[row]-close[row])
        bodydiffmin = 0.003
        if (bodydiff[row]>bodydiffmin and bodydiff[row-1]>bodydiffmin and
            open[row-1]<close[row-1] and
            open[row]>close[row] and
            #open[row]>=close[row-1] and close[row]<open[row-1]):
            (open[row]-close[row-1])>=+0e-5 and close[row]<open[row-1]):
            signal[row] = 1
        elif (bodydiff[row]>bodydiffmin and bodydiff[row-1]>bodydiffmin and
            open[row-1]>close[row-1] and
            open[row]<close[row] and
            #open[row]<=close[row-1] and close[row]>open[row-1]):
            (open[row]-close[row-1])<=-0e-5 and close[row]>open[row-1]):
            signal[row] = 2
        else:
            signal[row] = 0
            #signal[row]=random.choice([0, 1, 2])
            #signal[row]=1
    return signal

# engulf_df['engulf_bullish_bearish'] = signal(engulf_df)

```

Fig 2.4: Computing the Engulfing pattern signals (Bearish and Bullish)

<pre> 3 #bullish engulfing candles 4 signals_df[signals_df['bullish_bearish_engulfing'] == 2].tail() </pre>								
	open	high	low	close	volume	hammer	shooting_star	bullish_bearish_engulfing
date								
2024-02-06 00:00:00+05:30	1686.75	1733.0	1673.95	1729.45	7694265	0	0	2
2024-03-12 00:00:00+05:30	1600.00	1625.0	1597.65	1612.95	4614222	0	0	2
2024-06-27 00:00:00+05:30	1538.45	1578.4	1532.05	1573.35	14757754	0	0	2
2024-07-01 00:00:00+05:30	1559.50	1599.9	1559.50	1590.80	6801771	0	0	2
2024-07-08 00:00:00+05:30	1643.10	1666.0	1640.00	1661.65	5880533	0	0	2
<pre> 1 #bearish engulfing candles 2 signals_df[signals_df['bullish_bearish_engulfing'] == 1].tail() </pre>								
	open	high	low	close	volume	hammer	shooting_star	bullish_bearish_engulfing
date								
2023-11-08 00:00:00+05:30	1405.00	1405.00	1386.45	1391.00	3550635	0	0	1
2023-12-01 00:00:00+05:30	1459.90	1459.90	1446.15	1452.30	3489187	0	0	1
2023-12-05 00:00:00+05:30	1465.80	1466.90	1442.00	1453.95	5123039	0	0	1
2024-01-20 00:00:00+05:30	1665.55	1669.25	1645.00	1648.85	1158144	0	0	1
2024-03-27 00:00:00+05:30	1496.00	1499.00	1481.55	1483.85	6857402	0	0	1

Fig 2.5: Potential buy and sell signals from the engulfing pattern (Bearish and Bullish)

5.1.4 Computing the support and resistance levels

The procedure for determining and filtering support and resistance levels using fractal candlestick patterns involves several steps. First, I computed the support and resistance levels by detecting the lowest lows and highest highs within centered 5-period rolling windows, respectively. These levels are then combined into a single DataFrame and sorted chronologically.

Next, a threshold (s), calculated as the mean range between high and low prices, is used to filter out closely clustered levels. This ensures that only levels sufficiently distant from each other are retained by checking each level against the threshold. Finally, the filtered levels are classified as support or resistance based on their original identification, resulting in a refined set of significant and non-redundant support and resistance levels.

```
def find_support(df):
    df = df.copy()
    supports = df[df.low == df.low.rolling(5, center=True).min()].low
    return supports

def find_resistance(df):
    df = df.copy()
    resistance = df[df.high == df.high.rolling(5, center=True).max()].high
    return resistance
```

Fig 2.6: Computing support and resistance levels

```
def find_sp_resistance_levels(supports,resistances):
    levels = pd.concat([supports,resistances])
    levels = levels.dropna().sort_index()
    return levels
```

Fig 2.7: Combining the support and resistance levels to a single dataframe

```
s = np.mean(signals_df['high'] - signals_df['low'])

def is_far_from_level(l,levels,s):
    return np.sum([abs(l - levels[i]) < s for i in range(len(levels))]) == 0

def masking_filter(levels):
    levels = levels.copy()
    filtered_levels = []
    corresponding_dates = []
    for date,level in levels.items():
        if is_far_from_level(level,filtered_levels,s):
            filtered_levels.append(level)
            corresponding_dates.append(date)
    filtered_levels_df = pd.DataFrame(filtered_levels, columns=['level'], index=corresponding_dates)
    filtered_levels_df['type'] = ['support' if level in supports.values else 'resistance' for level in filtered_levels_df['level']]
    return filtered_levels_df
```

Fig 2.8: Applying Masking filter to avoid cluttering of lines

	level	type
2023-01-06 00:00:00+05:30	1446.50	support
2023-01-18 00:00:00+05:30	1551.85	resistance
2023-01-20 00:00:00+05:30	1524.00	support
2023-02-03 00:00:00+05:30	1604.65	resistance
2023-03-23 00:00:00+05:30	1370.00	support
2023-04-17 00:00:00+05:30	1185.30	support
2023-04-25 00:00:00+05:30	1215.00	support
2023-05-02 00:00:00+05:30	1279.75	resistance
2023-05-05 00:00:00+05:30	1254.25	support
2023-05-23 00:00:00+05:30	1314.15	resistance
2023-07-19 00:00:00+05:30	1498.80	resistance
2023-08-17 00:00:00+05:30	1420.00	support
2024-01-15 00:00:00+05:30	1664.95	resistance
2024-01-23 00:00:00+05:30	1637.70	support
2024-01-30 00:00:00+05:30	1690.00	resistance
2024-02-06 00:00:00+05:30	1733.00	resistance

Fig 2.9: Support-resistance levels after filtering

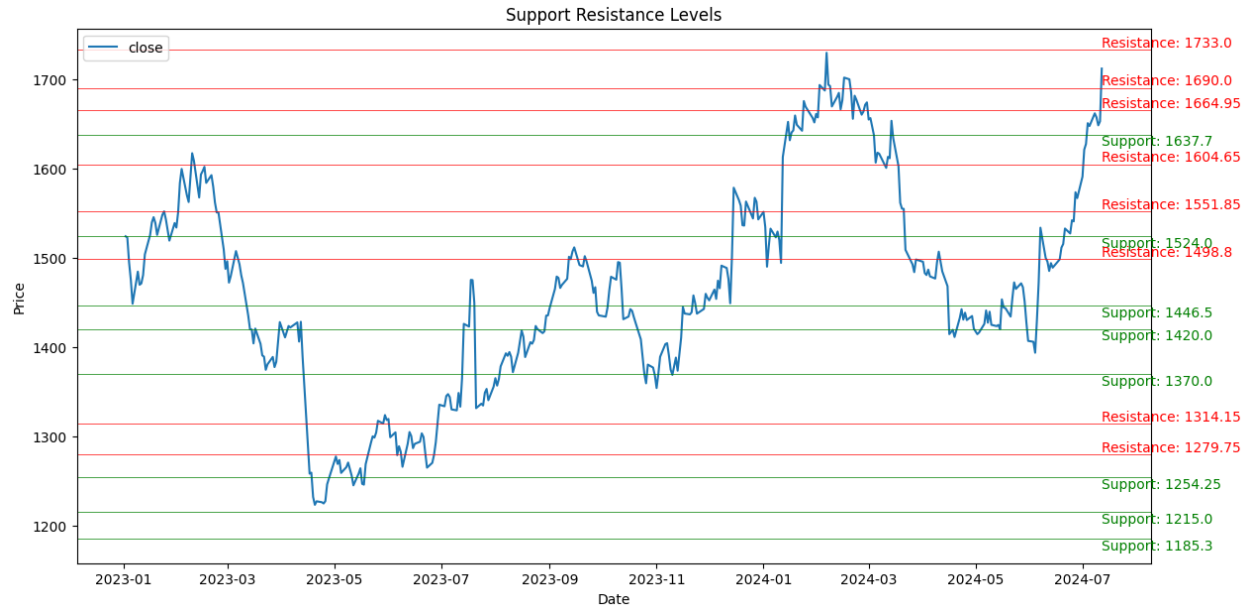


Fig 3.0: Visualizing Support-resistance levels

Support Resistance Levels

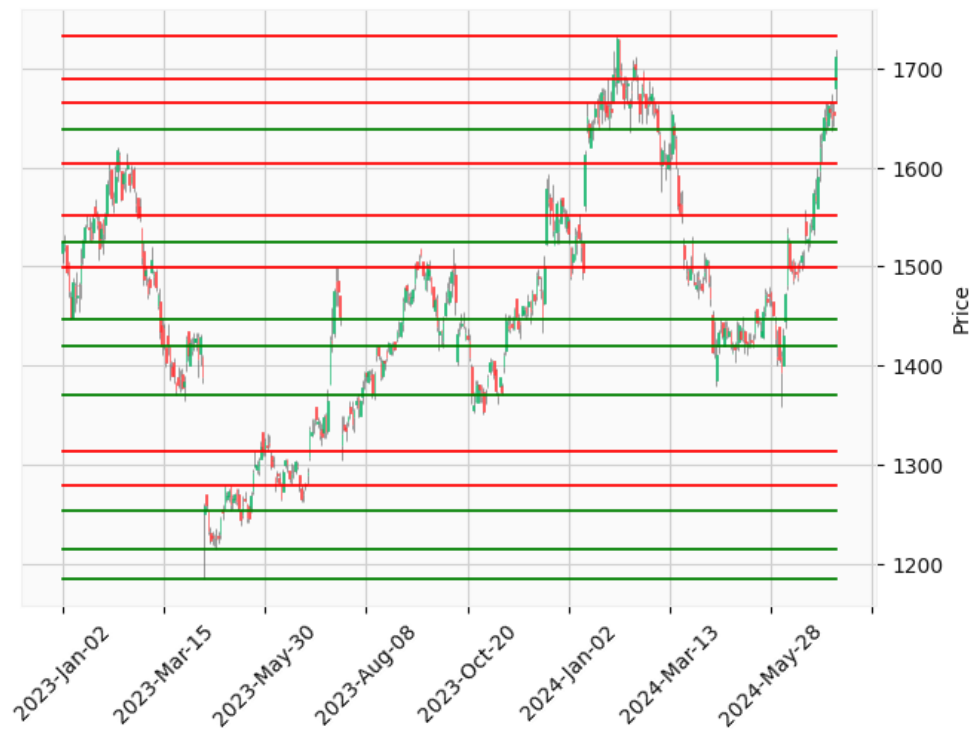


Fig 3.1: Visualizing Support-resistance levels on a candlestick chart

5.1.5 Simulating orders on historic data

As mentioned previously to identify potential buy signals, the stock price should be near a support level and a bullish pattern (either bullish engulfing or hammer) is detected. If conditions are met and there is sufficient capital, the stock is purchased, setting a stop-loss and take-profit level, as well as a trailing stop-loss. While holding the stock, the function continually updates the trailing stop-loss if the price increases beyond the profit target. A sell signal is triggered if the price hits the stop-loss, take-profit, trailing stop-loss, or a bearish pattern (either bearish engulfing or shooting star) is detected near a resistance level.

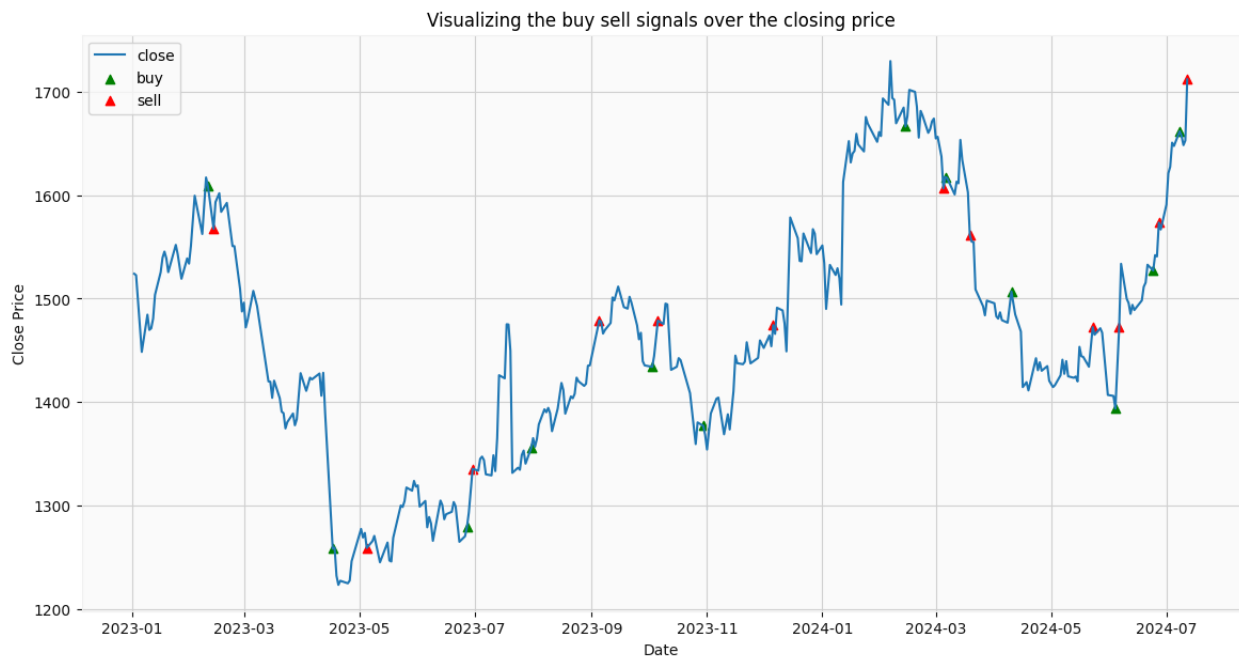


Fig 3.2: Visualizing the buy-sell signals

5 trades_df[trades_df['profit_loss'] > 0]													Python
8													
	buy_date	buy_price	target_profit	stop_loss	total_buy_price	quantity	sell_date	sell_price	total_selling_price	profit_loss	total_profit_loss	available_capital	
1	2023-04-17 00:00:00+05:30	1258.30	1296.0490	1233.134	8808.10	7.0	2023-05-05 00:00:00+05:30	1259.10	8813.70	0.80	5.60	9759.00	
2	2023-06-27 00:00:00+05:30	1279.15	1317.5245	1253.567	8954.05	7.0	2023-06-30 00:00:00+05:30	1335.50	9348.50	56.35	394.45	10153.45	
3	2023-07-31 00:00:00+05:30	1355.70	1396.3710	1328.586	9489.90	7.0	2023-09-05 00:00:00+05:30	1478.90	10352.30	123.20	862.40	11015.85	
4	2023-10-03 00:00:00+05:30	1434.00	1477.0200	1405.320	10038.00	7.0	2023-10-06 00:00:00+05:30	1478.70	10350.90	44.70	312.90	11328.75	
5	2023-10-30 00:00:00+05:30	1377.05	1418.3615	1349.509	11016.40	8.0	2023-12-06 00:00:00+05:30	1474.30	11794.40	97.25	778.00	12106.75	
9	2024-06-04 00:00:00+05:30	1393.65	1435.4595	1365.777	9755.55	7.0	2024-06-06 00:00:00+05:30	1472.25	10305.75	78.60	550.20	11605.55	
10	2024-06-24 00:00:00+05:30	1527.15	1572.9645	1496.607	10690.05	7.0	2024-06-27 00:00:00+05:30	1573.35	11013.45	46.20	323.40	11928.95	
11	2024-07-08 00:00:00+05:30	1661.65	1711.4995	1628.417	11631.55	7.0	2024-07-12 00:00:00+05:30	1711.75	11982.25	50.10	350.70	12279.65	
1 trades_df[trades_df['profit_loss'] < 0]													Python
	buy_date	buy_price	target_profit	stop_loss	total_buy_price	quantity	sell_date	sell_price	total_selling_price	profit_loss	total_profit_loss	available_capital	
0	2023-02-10 00:00:00+05:30	1608.55	1656.8065	1576.379	9651.30	6.0	2023-02-13 00:00:00+05:30	1567.45	9404.70	-41.1	-246.6	9753.40	
6	2024-02-14 00:00:00+05:30	1666.20	1716.1860	1632.876	11663.40	7.0	2024-03-05 00:00:00+05:30	1606.50	11245.50	-59.7	-417.9	11688.85	
7	2024-03-06 00:00:00+05:30	1617.55	1666.0765	1585.199	11322.85	7.0	2024-03-19 00:00:00+05:30	1561.45	10930.15	-56.1	-392.7	11296.15	
8	2024-04-10 00:00:00+05:30	1506.80	1552.0040	1476.664	10547.60	7.0	2024-05-23 00:00:00+05:30	1472.40	10306.80	-34.4	-240.8	11055.35	

Fig 3.3: Examining wining losing trades

5.1.6 Performing the back-test simulation using Backtesting.py

Back-testing is a common technique used to evaluate the performance of a trading strategy or model retrospectively. It determines the effectiveness of a trading strategy by simulating its application on historical data. Successful back-testing can provide traders and analysts with the confidence to implement the strategy in future trading. Since the development of back-testing framework is a tedious process, I used backtesting.py to perform the backtest on my trading algorithm. The procedure on how this was done is similar to how the simulation on trades was done in the previous step.



Fig 3.4: Visualizing the backtest result

```
Start                2023-01-02 00:00...
End                  2024-07-12 00:00...
Duration              557 days 00:00:00
Exposure Time [%]    25.065274
Equity Final [$]     11028.7452
Equity Peak [$]      11028.7452
Return [%]           10.287452
Buy & Hold Return [%] 12.319554
Return (Ann.) [%]    6.7274
Volatility (Ann.) [%] 8.905952
Sharpe Ratio         0.755382
Sortino Ratio        1.379016
Calmar Ratio         0.744977
Max. Drawdown [%]    -9.030347
Avg. Drawdown [%]    -2.368474
Max. Drawdown Duration 233 days 00:00:00
Avg. Drawdown Duration 49 days 00:00:00
# Trades              16
Win Rate [%]         62.5
Best Trade [%]        3.443639
Worst Trade [%]       -3.092826
Avg. Trade [%]        0.646541
Max. Trade Duration   19 days 00:00:00
Avg. Trade Duration    7 days 00:00:00
Profit Factor         1.750252
Expectancy [%]        0.676483
SQN                   1.05838
_strategy             MyStrategy
_equity_curve          ...
_trades                Size EntryB...
dtype: object
```

Fig 3.5: Backtesting Result

The backtesting results for the trading strategy indicate the performance over the period from January 2, 2023, to July 12, 2024. The strategy achieved a total return of 10.29%, which, although slightly lower than the buy-and-hold return of 12.32%, reflects a cautious and controlled approach to trading. The annualized return of 6.73%, combined with an annualized volatility of 8.91%, resulted in a Sharpe ratio of 0.76, indicating that the strategy delivered moderate risk-adjusted returns. The Sortino ratio of 1.38, which focuses on downside risk, suggests that the strategy was effective in managing negative price movements. Additionally, the Calmar ratio of 0.74 highlights the strategy's ability to maintain a balance between risk and return, despite a maximum drawdown of 9.03%.

The strategy executed 16 trades with a win rate of 62.5%, indicating a relatively high probability of successful trades. The average trade returned 0.65%, with the best trade yielding a 3.44% profit and the worst trade incurring a 3.09% loss. The average duration of trades was 7 days, with the longest trade lasting 19 days. The profit factor of 1.75 suggests that the strategy generated significantly more profit from winning trades than it lost from losing trades. Furthermore, the positive expectancy of 0.68% per trade indicates that the strategy is likely to be profitable over the long run. Overall, the results suggest that is a robust trading approach that manages risk effectively while delivering consistent returns, making it a viable option for traders seeking steady growth with controlled risk exposure.

5.1.6 Placing orders on live streaming data using web sockets

KiteTicker is used for live streaming data for real-time trading by connecting to the Kite Connect WebSocket API. It subscribes to market tick data for specified instruments, receiving continuous updates on price movements and other market information. The `on_ticks` callback function processes this incoming data, executing trading logic based on predefined conditions. The `on_connect` callback manages the initial connection setup, subscribing to relevant instrument tokens, while `on_close` handles disconnection scenarios.

To automate generating orders in the stock market. I defined the `placeOrder` function that places a buy order and on top of that it places two sell orders – one for selling at the stop loss level and the other for selling at the target profit level. This ensures that stock orders are placed appropriately. Additionally, the `modifyOrder` function is defined to update the buy orders stoploss with the trailing stop-loss, to preserve the profits. For example, if a stock was bought at Rs. 1700 and stop-loss was set to 5% (Rs. 5) so, if the closing price hits Rs. 1695 the stock would be sold to prevent further loss. A trailing stop aims to safeguard profits by allowing a trade to stay open and benefit as long as the price moves favorably. The trade is closed if the price shifts direction by a predetermined percentage or dollar amount. For example, if a stock was bought at Rs. 1700 and trailing stop-loss was set to Rs. 5 if the close price of the stock increases the trailing stop moves

up simultaneously maintaining the Rs. 5 gap, if the close price drops by Rs. 5 then the stock gets sold.

To execute the trading algorithm, I replicated the same procedure as I did while testing with historic data. This function (`main()`) is the core of the trading strategy. It starts by attempting to fetch current position and order data from the Kite API, retrying up to 10 times if necessary. The function then generates trading signals using various candlestick patterns, support, and resistance levels. Based on these signals, the function determines the nearest support and resistance levels relative to the current price. If a buy signal is detected and a position is not already held, the function calculates the buy price, quantity, stop loss, and take profit levels, and places a buy order through the `placeOrder` function. It also adjusts the available capital after placing the buy order. If a sell signal is detected and a position is held, the function modifies the order to implement a trailing stop loss if the current price exceeds the buy price by a certain percentage. It then executes a sell order and records the trade details, updating the total profit/loss and capital. Using the `time` library this code runs in loop continuously, checking the current time to ensure that trading operations are only performed during market hours (starting at 9 AM). Within this loop, the `WebSocket` connection is managed to receive real-time data and trigger trading actions accordingly. If the current time is after-market hours (2:30 PM), the script exits.

6. Conclusion

In conclusion, the primary aim of this project was to design and implement an algorithmic trading strategy using Zerodha's Kite API, using price action concepts and support-resistance lines to generate effective buy and sell signals. I was able develop a simple and robust system capable of automating trades based on historical and live market data. The backtesting results demonstrated the strategy's ability to manage risk effectively while delivering consistent returns, confirming its viability for real-world application. Furthermore, by automating the trading process, I was able mitigate the influence of emotional decision-making, which is crucial in the highly volatile stock market environment. This project has not only enhanced my technical skills in algorithmic trading but also provided valuable insights into the practical application of financial theories and programming techniques. The successful testing of this trading algorithm underscores the potential of automated trading systems to achieve steady growth with controlled risk, laying a strong foundation for future advancements in this field.

Further improvements for the project include usage of incorporating additional technical indicators such as MACD, RSI, and Bollinger Bands, and implementing machine learning models such as LSTM for better signal accuracy. Optimize parameter tuning using techniques like Grid Search, expand back testing to cover diverse market conditions, and enhance risk management with strategies like volatility-based position sizing. Develop a real-time performance dashboard, integrate multiple data sources, and create a user-friendly interface. Implement a paper trading environment for validation, automate data preprocessing, improve error handling and logging, and diversify trading strategies. These enhancements will improve the system's performance, reliability, and robustness, ensuring consistent returns and effective risk management.

6.1 Environmental and Societal Impact

The development of an algorithmic trading system has minimal direct environmental impact since it primarily relies on digital resources and computing power. However, optimizing the code for energy efficiency and using cloud services with green certifications can further reduce its carbon footprint. Societally, this project promotes financial inclusivity by enabling retail traders to access sophisticated trading tools previously available only to institutional investors. By automating trading decisions, it reduces human error and emotional bias, potentially leading to more stable financial markets. Furthermore, the project emphasizes the importance of education and technological proficiency, contributing to the advancement of financial literacy and technical skills in society.

7. Ethics in Engineering Certification



8. NBA/IET Mapping

NBA - PROGRAM OUTCOMES (PO) & PROGRAM SPECIFIC OUTCOMES (PSO)

Engineering Graduates will be able to:

PO1:Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2:Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3:Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4:Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5:Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6:The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7:Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8:Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10:Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PSO1: Analyse and solve real world problems by applying a combination of hardware and software. PSO2: Formulate & build optimised solutions for systems level software & computationally intensive applications.

PSO3: Design & model applications for various domains using standard software engineering practices.
 PSO4: Design & develop solutions for distributed processing & communication.

NBA CO PO Mapping

CSE 4298-ITR	CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CSE 4298.1	Understand the functioning of the industry	1	0	0	0	0	0	0	2	2	0	0	0	1	0	0	0
CSE 4298.2	Understand the requirements of real world applications	2	1	3	1	1	1	1	0	3	2	0	1	2	1	1	0
CSE 4298.3	Demonstrate skills to use modern engineering tools, software and equipment to analyze problems	0	0	2	2	3	1	1	0	3	2	0	1	2	1	1	0
CSE 4298.4	Demonstrate an ability to envisage and work on laboratory and multidisciplinary tasks	0	0	2	1	3	1	1	0	3	2	0	1	2	1	1	0
CSE 4298 (Avg. correlation level)		0.75	0.25	1.75	1	1.75	0.75	0.75	0.5	2.75	1.5	0	0.75	1.75	0.75	0.75	0

NBA Program articulation Matrix

Course Code	Course Title	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CSE 4298	Industrial Training	0.75	0.25	1.75	1	1.75	0.75	0.75	0.5	2.75	1.5	0	0.75	1.75	0.75	0.75	0

IET-AHEP Learning outcome statements

<i>Code</i>	<i>Learning outcome (LO)</i>
C1.	Apply knowledge of mathematics, statistics, natural science and engineering principles to the solution of complex problems. Some of the knowledge will be at the forefront of the particular subject of study
C2.	Analyse complex problems to reach substantiated conclusions using first principles of mathematics, statistics, natural science and engineering principles
C3.	Select and apply appropriate computational and analytical techniques to model complex problems, recognising the limitations of the techniques employed
C4.	Select and evaluate technical literature and other sources of information to address complex problems
C5.	Design solutions for complex problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
C6.	Apply an integrated or systems approach to the solution of complex problems
C7.	Evaluate the environmental and societal impact of solutions to complex problems and minimise adverse impacts
C8.	Identify and analyse ethical concerns and make reasoned ethical choices informed by professional codes of conduct
C9.	Use a risk management process to identify, evaluate and mitigate risks (the effects of uncertainty) associated with a particular project or activity
C10.	Adopt a holistic and proportionate approach to the mitigation of security risks
C11.	Adopt an inclusive approach to engineering practice and recognise the responsibilities, benefits and importance of supporting equality, diversity and inclusion
C12.	Use practical laboratory and workshop skills to investigate complex problems
C13.	Select and apply appropriate materials, equipment, engineering technologies and processes, recognising their limitations

C14.	Discuss the role of quality management systems and continuous improvement in the context of complex problems
C15.	Apply knowledge of engineering management principles, commercial context, project and change management, and relevant legal matters including intellectual property rights
C16.	Function effectively as an individual, and as a member or leader of a team
C17.	Communicate effectively on complex engineering matters with technical and non-technical audiences
C18.	Plan and record self-learning and development as the foundation for lifelong learning/CPD

VII SEMESTER - CSE 4298 – Industrial Training - CLO - AHEP LO Mapping

Course Learning Outcome	Statement	C1	C2	C3	C4	C6	C7	C8	C11	C15	C18
CLO 4298.1	Apply mathematics, science and engineering skills to identify, formulate, synthesize and solve the problems from various areas of computer science engineering.	✓		✓							
CLO 4298.2	Have knowledge of new trends in engineering/technology by developing programmable coding in various computer programming languages.				✓						
CLO 4298.3	Use the industry standard tools to analyze, design, develop and test software engineering based applications.			✓							

CLO 4298.4	Apply theoretical knowledge to real-world engineering problems and manage complex engineering projects.		✓			✓					
CLO 4298.5	Understand the adverse societal impacts of the solutions to complex problems during project development.						✓				
CLO 4298.6	Identify and analyze ethical concerns to be followed during the practice school/research project internships and make reasoned moral choices guided by the internal and external supervisors during the tenure.							✓			
CLO 4298.7	Recognize the responsibilities deemed by the external and internal supervisors and understand the importance of supporting equality, diversity, and inclusion between peers.								✓		
CLO 4298.8	Apply knowledge of engineering management principles and understand why project and change management may be required during practice school and project work.									✓	
CLO 4298.9	Indicate the future direction of the project development and appreciate how it can be realized with collaborative, lifelong learning, and self-learning.										✓

Declaration: Through this Industrial Training, I have accomplished the above stated program articulation and IET learning outcomes.

9. References

- [1] Udemy, "Algorithmic Trading on Zerodha KiteConnect Platform," Udemy. [Online]. Available: <https://www.udemy.com/course/algorithmic-trading-on-zerodha-kiteconnect-platform>. [Accessed: 10-May-2024].
- [2] J. D. Schwager, "Getting Started in Technical Analysis". Illustrated, Paperback ed. New York, NY, USA: Wiley, 1999.
- [3] Investopedia, "Hammer," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/h/hammer.asp>. [Accessed: 10-Jun-2024].
- [4] Investopedia, "Bullish Engulfing Pattern," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/b/bullishengulfingpattern.asp>. [Accessed: 12-Jun-2024].
- [5] Investopedia, "Shooting Star," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/s/shootingstar.asp>. [Accessed: 11-Jun-2024].
- [6] Investopedia, "Bearish Engulfing Pattern," Investopedia. [Online]. Available: <https://www.investopedia.com/terms/b/bearishengulfingp.asp>. [Accessed: 14-Jun-2024].
- [7] M. Konrad, "Strategies Library," Backtesting.py Documentation. [Online]. Available: <https://kernc.github.io/backtesting.py/doc/examples/Strategies%20Library.html>. [Accessed: 20-Jun-2024].
- [8] Zerodha, "Kite Connect API Documentation," Kite Connect. [Online]. Available: <https://kite.trade/docs/pykiteconnect/v4/>. [Accessed: 20-May-2024].
- [9] Zerodha Tech Pvt Ltd. "Kite Connect API." [Online]. Available: <https://kite.trade/>. [Accessed: 24-May-2024].