Upstox Options Trading

Introduction :

The Upstox Options Trading Project is designed to simplify and automate interactions with the Indian stock market, specifically focusing on options trading using the Upstox API. This project provides an efficient and structured way to perform essential trading operations, including authenticating with Upstox, retrieving options data, and calculating the necessary margin and potential premiums for selected trades.

Key components of this project include:

1. **Authentication**: Utilizing OAuth2 to securely connect to the Upstox API and obtain an access token for subsequent requests.
2. **Option Chain Data Retrieval**: Fetching detailed option chain data for specified instruments, such as NIFTY or BANKNIFTY, for a chosen expiry date and option type (Call or Put).
3. **Margin and Premium Calculations**: Calculating the required trading margin and potential premiums based on the bid or ask price and lot size, using user holdings and market data.

With an interactive, modular design, the project enables both automatic data processing and user-driven selections for dynamic analysis. This structure is particularly valuable for traders looking to quickly assess market opportunities and potential earnings based on real-time data.

Project Overview :

The Upstox Options Trading Project leverages the Upstox API to streamline and automate options trading tasks for the Indian stock market. Designed as an interactive and modular Python-based solution, this project provides traders with an efficient way to access, analyze, and calculate essential trading information.
Core Functionalities

Authentication:

Uses OAuth2 to securely authenticate with the Upstox API.
Retrieves an access token required for authorized API requests, enabling secure data access and operations.

Option Chain Data Retrieval:

Allows users to fetch option chain data for specified instruments (e.g., NIFTY or BANKNIFTY), expiry dates, and option types (Call or Put).
Retrieves key information like strike prices and bid/ask prices, offering insights into market depth and pricing.

Margin and Premium Calculations:

Calculates the trading margin required for specific options and computes the premium that could be earned.
Uses Upstox's margin calculation API along with user portfolio data to provide accurate estimates for each option.

Interactive Workflow
The project's interactive design enables users to input or randomly select trading parameters. Users can repeatedly fetch option data, calculate trading margins, and view real-time premiums in an iterative loop, allowing for responsive and adaptable trading decisions.

Code Structure :

The code is organized into several main parts, each addressing a specific function:

1.Imports and Dependencies

Libraries:
requests: Used for making HTTP requests to the Upstox API.
pandas: Utilized for data manipulation, specifically to organize API responses into structured DataFrames for easier analysis and display.
random: Allows for random selection of inputs when a user does not provide specific values.

Code :

```
import requests
import random
import pandas as pd
```

2.User Authentication and Access Token Generation

Purpose: This section authenticates the user with Upstox via OAuth2, enabling secure access to API endpoints.
Process:

- Prompts the user for api_key, secret_key, and redirect_uri.
- Generates a login URL for user authorization.
- Exchanges the authorization code provided by Upstox for an access token.
- Saves the token in a text file (access_token.txt) for future use, allowing for persistent authentication.

Code :

```
# Authentication Code
api_key = input('\tapi_key\t')
secret_key = input('\tsecret_key\t')
redirect_uri = input('\tredirect_uri\t')
login_url =
f'https://api.upstox.com/v2/login/authorization/dialog?client_id=
{api_key}&redirect_uri={redirect_uri}'
code = input('\tcode\t')
api__auth__url =
'https://api.upstox.com/v2/login/authorization/token'
# Access Token Handling
access_token = auth__req_res.json()['access_token']
```

4.Instrument Data and Portfolio Holdings Retrieval

Instrument Data Retrieval:

- The code fetches complete instrument data from Upstox's market quote endpoint.
- Filters the data to focus on Call (CE) and Put (PE) options, which are relevant to options trading.
- Parses the expiry column for cleaner, more user-friendly date formats.

Portfolio Holdings Retrieval:

- Retrieves long-term holdings data from the user's Upstox portfolio, specifically filtering for relevant fields (instrument_token, product, and quantity).
- Merges instrument and portfolio data on instrument_token, creating a DataFrame that consolidates user-specific information with market data.

Code :

```
# Fetch Instrument and Portfolio Data
api__mtqe_ints_exage_cmplte__url = "https://assets.upstox.com/market-quote/instruments/exchange/complete.json.gz"
mtqe_ints_exage_cmplte__df = pd.read_json(api__mtqe_ints_exage_cmplte__url)
# Filter for Options (PE and CE)
prtflo_lgtmhs__df = pd.DataFrame(prtflo_lgtmhs__req_res['data'])[['instrument_token','product','quantity']]
# Merge Data
mtqe_ints_exage_cmplte_prtflo_lgtmhs__df = pd.merge(mtqe_ints_exage_cmplte__df, prtflo_lgtmhs__df, left_on='underlying_key', right_on='instrument_token')
```

Logic and Workflow :

Option Chain Data Retrieval (func__get_option_chain_data())

Purpose: Retrieves options data based on the instrument, expiry date, and option type (Call or Put).

Steps:

- Input Filtering: Based on the user's selection, filters the instrument DataFrame for relevant options.
- Key Selection: Retrieves the instrument_key specific to the chosen instrument, which is necessary for the API request.
- API Request: Sends a GET request to Upstox's option chain API, using the instrument_key and expiry_date as parameters.
- Data Parsing: The response data is parsed to retrieve the highest bid_price for Put options or the highest ask_price for Call options, which are essential for assessing the option's value.

Code :

```
def func__get_option_chain_data(args__instrument_name: str, args__expiry_date: str, args__side: str) -> pd.DataFrame:

                                    instrument_key__df = mtqe_ints_exage_cmplte__df[(mtqe_ints_exage_cmplte__df['name'] == args__instrument_name) & (mtqe_ints_exage_cmplte__df['expiry'] == args__expiry_date) & (mtqe_ints_exage_cmplte__df['instrument_type'] == args__side)]['underlying_key'].to_list()

   print(instrument_key__df)

            instrument_key = input('\tinstrument_key\t') or random.choice(instrument_key__df)



   api__opt_chn__url = "https://api.upstox.com/v2/option/chain"

   api__opt_chn__hdrs = {

      'Authorization' : f'Bearer {access_token}',

      'Accept': 'application/json'

   }

   api__opt_chn__prms = {

      'instrument_key': instrument_key,

                                                'expiry_date': mtqe_ints_exage_cmplte__df[mtqe_ints_exage_cmplte__df['underlying_key'] == instrument_key]['expiry']

   }
```

```
        opt_chn__req_res = requests.request("GET", api__opt_chn__url, headers=api__opt_chn__hdrs, params=api__opt_chn__prms).json()

    ittn_prdct_qunty__df = pd.DataFrame({

        'instrument_name': [args__instrument_name],

        'strike_price': [mtqe_ints_exage_cmplte__df[mtqe_ints_exage_cmplte__df['underlying_key'] == instrument_key]['strike_price'].mean()],

        'side': [args__side],

        'bid/ask': [max(itr__data["put_options"]["market_data"]["bid_price"] for itr__data in opt_chn__req_res['data']) if args__side == 'PE' else max(itr__data["call_options"]["market_data"]["ask_price"] for itr__data in opt_chn__req_res['data'])]

    })

    return ittn_prdct_qunty__df
```

Margin and Premium Calculations (func__calculate_margin_and_premium())

Purpose: This function calculates the required margin and potential premium for trading options.

Steps:

- Payload Preparation: Builds a JSON payload containing the selected instrument_key, quantity, and product.
- API Request: Sends a POST request to Upstox's margin calculation endpoint.
- Data Parsing and Calculation:
    - Extracts required_margin from the API response.

- Calculates premium_earned based on lot_size and bid/ask price for the selected option.

◆ Output: Returns a DataFrame with margin_required and premium_earned columns.

Code :

```
def func__calculate_margin_and_premium(args__data: pd.DataFrame) -> pd.DataFrame:

    api__crgs_mrgn__url = "https://api.upstox.com/v2/charges/margin"

    api__crgs_mrgn__hdrs = {

        "accept": "application/json",

        "Authorization": f"Bearer {access_token}",

        "Content-Type": "application/json"

    }

    api__crgs_mrgn__json = {

        "instruments": [

            {

                "instrument_key": prtflo_lgtmhs__df['instrument_token'].values[0],

                "product": prtflo_lgtmhs__df['product'].values[0],

                "quantity": int(prtflo_lgtmhs__df['quantity'].values[0]),

                "transaction_type": "SELL",

            }
```

```
        ]
    }
    crgs_mrgn__req_res = requests.request("POST",
api__crgs_mrgn__url, headers=api__crgs_mrgn__hdrs,
json=api__crgs_mrgn__json)
    mnrd_pmed__lst =
[crgs_mrgn__req_res.json()['data']['required_margin'],
args__data['bid/ask'].values[0] *
mtqe_ints_exage_cmplte__df[mtqe_ints_exage_cmplte__df['un
derlying_key'] ==
prtflo_lgtmhs__df['instrument_token'].values[0]]['lot_size'].valu
es[0]]
    ittn_prdct_qunty_mnrd_pmed__df =
args__data.assign(**{'margin_required':mnrd_pmed__lst[0],'pre
mium_earned':mnrd_pmed__lst[1]})
    return ittn_prdct_qunty_mnrd_pmed__df
```

Approach :

Modular and Interactive Design

- Each function has a single responsibility, following the Single Responsibility Principle. For example, func__get_option_chain_data() handles option chain retrieval only, while func__calculate_margin_and_premium() focuses on calculations.
- The modular design improves readability, maintainability, and debugging.

Error Handling and Token Persistence

- The access token is saved to a text file, making it reusable across sessions, thus simplifying re-authentication.
- Error handling is built around API requests to ensure the user receives immediate feedback if issues arise, like expired tokens or incorrect parameters.

Automated and Manual Input Options

- The code uses random.choice() to select default options if the user doesn't provide specific inputs, creating flexibility for both manual and automated testing.

API Integration and Data Consolidation

- This project consolidates data from multiple sources (instruments, long-term holdings, and margin calculations) and merges them effectively into unified DataFrames, making analysis and reporting easier for users.
- The workflow is streamlined by merging instrument and portfolio data upfront, allowing the code to directly access consolidated information for downstream processing.

AI Tools and Assistance :

1. Function Design and Structuring

AI assistance was instrumental in designing the overall structure of the code. Specifically, ChatGPT was used to:

- **Generate Function Skeletons:** AI generated initial structures for core functions like func__get_option_chain_data() and func__calculate_margin_and_premium(), providing a well-organized and modular framework for handling API requests and calculations.
- **Ensure Modular Design:** AI recommended breaking down the project into modular components to simplify testing and maintenance, aligning each function with a single responsibility, such as data retrieval or calculation.

2. API Research and Documentation Support

AI assistance was valuable in understanding Upstox API specifications, especially around:

- **OAuth2 Authentication**: AI provided an overview of OAuth2 requirements, ensuring the authentication steps aligned with Upstox's OAuth2 flow for token retrieval.
- **API Endpoint and Payload Structure**: ChatGPT offered insights into creating API requests, formatting payloads, and specifying headers, which were essential for handling Upstox's margin calculation and option chain endpoints.

3. Error Handling and Debugging :

AI tools were used to debug common errors and optimize data handling:

- **Error Debugging in API Requests:** ChatGPT provided tips on handling common errors in API requests, such as incorrect payload formats or token validation errors, helping ensure robust error handling.
- **Optimized Data Merging:** AI recommended strategies for merging instrument and portfolio data, allowing for efficient data consolidation and retrieval.

Code Optimization and Interactive Design

AI enhanced the project's usability and efficiency by suggesting:

- **Random Input Options:** Using random.choice() as a fallback for user input, which made testing and interactive usage more flexible, was a suggestion from AI.
- **Data Processing with Pandas:** ChatGPT recommended using pandas to handle, filter, and display data, ensuring that the code was efficient and compatible with large data sets.

Conclusion :

The Upstox Options Trading Project successfully demonstrates how automation and API integration can enhance options trading in the Indian stock market. By leveraging the Upstox API, this project simplifies complex trading operations, from secure authentication to detailed data retrieval and margin calculations, empowering users to make informed decisions based on real-time data.

Key Achievements :

1. Seamless Authentication with OAuth2

The project implements secure, token-based authentication using OAuth2, which is crucial for accessing Upstox’s protected endpoints. By obtaining an access token and saving it for reuse, the project provides a secure, scalable method for user authorization, enhancing both security and user convenience. This foundational step allows all other functionalities to operate within Upstox's secure framework, meeting modern security standards.

2. Modular and Flexible Design

The code is built on a modular structure, with each function having a clear, single responsibility. For instance, func__get_option_chain_data() exclusively retrieves option chain data, while func__calculate_margin_and_premium() focuses solely on calculating margin requirements and potential premiums. This modularity not only makes the code easier to maintain and debug but also allows users to adapt individual components for further expansion or integration with other systems, such as additional trading strategies or market analysis tools.

3. Dynamic and Interactive Workflow

The project provides an interactive, user-driven workflow that enables traders to either manually input their preferences or let the code select options automatically. This interactive design supports both experienced traders, who may want to specify exact parameters, and casual users, who can explore market data more flexibly. Through this, the project enhances usability, making it adaptable to various user skill levels and trading styles.

4. Real-Time Data Integration and Analysis

The integration with Upstox's API allows the project to retrieve real-time data for options trading. By accessing up-to-date information on strike prices, bid/ask values, and instrument details, the project helps traders respond promptly to market changes. Additionally, margin and premium calculations enable users to assess potential earnings and costs immediately, facilitating a more informed approach to trading strategies and risk management.

5. AI-Assisted Development for Efficiency and Innovation

AI tools, specifically ChatGPT, significantly enhanced the project's development process. AI-assisted:

- Function Design and Structuring: By providing initial skeletons and modularity suggestions.
- API Request Optimization: Ensuring payload and parameter formatting aligned with Upstox's requirements.
- Error Debugging and Handling: Improving resilience through robust error handling and reusable helper functions.

This AI-driven approach not only accelerated development but also improved the code's efficiency, reliability, and ease of use.

Impact and Future Potential :

Enhanced Trading Decisions

This project empowers traders by simplifying complex calculations and delivering real-time insights into margin requirements and premiums. With easy access to the Upstox option chain data, users can make quicker, more informed trading decisions, potentially improving their profitability and reducing risks associated with options trading.

Scalability and Adaptability

The modular design ensures that this project can be expanded to accommodate additional functionalities, such as integrating more financial instruments, expanding into new markets, or incorporating advanced analytics and machine learning models for predictive trading. As the project's foundation is solid and flexible, future development efforts can focus on adding value through new features rather than reworking existing functionality.

Educational Value and Accessibility

This project also serves as an educational tool for those learning about options trading and API integration. By offering both manual and automated options, users at different experience levels can interact with the code and gain a better understanding of options trading mechanics, market data, and financial calculations. Additionally, the use of AI tools in development demonstrates a modern approach to coding, illustrating the benefits of AI-assisted programming for educational and professional purposes.