Polymarket Copy Trading Bot Documentation

Overview

The Polymarket Copy Trading Bot is designed to **monitor** the trading activity of a **target user** and **replicate** their trades on a **proxy wallet** using the Polymarket Central Limit Order Book (CLOB) API. The bot is built with a focus on **efficiency**, **accuracy**, and **risk management**. Below is a detailed explanation of the bot's **strategies** and **implementation**.

Key Components and Workflow

1. Initialization (`index.ts`)

The bot starts by initializing the database connection and setting up the CLOB client. It then begins monitoring the target user's trades and executing them on the proxy wallet.

Key Steps:

- **Database Connection**: Connects to the database to store and retrieve trade data.
- **CLOB Client Setup**: Creates a client to interact with the Polymarket API.
- **Trade Monitoring**: Starts monitoring the target user's trades.
- **Trade Execution**: Executes trades on the proxy wallet based on the monitored activity.

2. Trade Monitoring (`tradeMonitor.ts`)

The bot continuously monitors the target user's trading activity and updates the database with the latest trades.

Key Strategies:

- **Fetching User Activity**:
    - The bot fetches the target user's **positions** and **activities** from the Polymarket site.
    - It filters out **old trades** (older than 1 hour) to ensure it only acts on recent activity. (Consider server down time)
- **Database Updates**:
    - The bot stores the fetched trades in the database for further processing.

- It ensures that **duplicate trades** are not processed again.

Logical Flow:

1. Fetch the target user's positions and activities.
2. Filter out old trades and duplicates.
3. Store the new trades in the database.

3. Trade Execution (`tradeExecutor.ts`)

The bot executes trades on the proxy wallet based on the target user's activity. It uses **three main strategies**: **Buy**, **Sell**, and **Merge**.

Logical Flow:

1. Fetch the target user's trades from the database.
2. For each trade:
    - If the trade is a **BUY**, execute the **Buy Strategy**.
    - If the trade is a **SELL**, execute the **Sell Strategy**.
    - If the bot has an existing position, execute the **Merge Strategy**.
3. Update the database to mark the trade as executed.

4. CLOB Client Creation (`createClobClient.ts`)

The bot creates a CLOB client to interact with the Polymarket API. This client is used to fetch market data and execute trades.

Key Steps:

- **Wallet Setup**: Initializes the wallet using the private key and proxy wallet address.
- **API Key Creation**: Creates or derives an API key for secure communication with the Polymarket API.
- **CLOB Client Initialization**: Sets up the CLOB client with the necessary credentials.

5. Data Fetching Utilities

The bot uses utility functions to fetch data from APIs and manage user balances.

Key Functions:

- **fetchData(url)**: Fetches data from a given URL using Axios.
- **getMyBalance(address)**: Retrieves the USDC balance of the specified wallet address.

Strategy:

These utilities ensure that the bot has **real-time data** on user balances and market conditions, which is crucial for making informed trading decisions.

6. Order Posting (`postOrder.ts`)

This file contains the logic for posting orders to the Polymarket CLOB. It handles **Buy**, **Sell**, and **Merge** orders.

Key Strategies:

Buy Order:

- The bot places a **BUY** order for the same asset as the target user, scaled by the balance ratio.
- It ensures that the **price difference** is within an acceptable range.

Sell Order:

- The bot places a **SELL** order for the same asset as the target user, scaled by the position ratio.
- It ensures that there are sufficient assets to sell.

Merge Order:

- If the bot has an existing position, it merges the new trade with the existing position.
- This strategy reduces the number of transactions and optimizes the bot's trading.

Logical Flow:

1. Determine the type of order (Buy, Sell, or Merge).
2. Fetch the order book to find the best price.
3. Place the order and handle retries if it fails.
4. Update the database to mark the trade as executed.

Detailed Strategies

1. Buy Strategy

- **Objective**: Replicate the target user's **BUY** orders on the proxy wallet.
- **Steps**:

i. Calculate the **ratio** of the proxy wallet's balance to the target user's balance.
ii. Fetch the **order book** to find the best **ask price**.
iii. Place a **BUY** order for the same asset, scaled by the calculated ratio.
iv. Ensure the **price difference** is within an acceptable range.

2. Sell Strategy

- **Objective**: Replicate the target user's **SELL** orders on the proxy wallet.
- **Steps**:

i. Calculate the **ratio** of the proxy wallet's position to the target user's position.
ii. Fetch the **order book** to find the best **bid price**.
iii. Place a **SELL** order for the same asset, scaled by the calculated ratio.
iv. Ensure there are sufficient assets to sell.

3. Merge Strategy

- **Objective**: Optimize trading by merging new trades with existing positions.
- **Steps**:

i. Check if the bot has an existing position in the same asset.
ii. If yes, merge the new trade with the existing position.
iii. Place a **SELL** order to adjust the position if necessary.

Conclusion

The Polymarket Copy Trading Bot is a sophisticated tool designed to **automate** the replication of trades from a target user. It uses **three core strategies** (Buy, Sell, and Merge) to ensure efficient and accurate trading. The bot continuously monitors the target user's activity, fetches real-time market data, and executes trades on the proxy wallet.