Binance Market Data WebSocket Implementation Project

Project Overview

This project implements a real-time candlestick chart visualization for multiple cryptocurrencies using Binance's WebSocket API. The chart updates dynamically, reflecting live market data and enabling users to toggle between different cryptocurrencies and time intervals.

Project Objectives

1. **Real-Time Market Data**: Fetch real-time market data from Binance's WebSocket API.
2. **Cryptocurrency Toggle**: Users can switch between ETH/USDT, BNB/USDT, and DOT/USDT markets.
3. **Candlestick Visualization**: The project visualizes data using a candlestick chart with selectable time intervals (1 minute, 3 minutes, 5 minutes).
4. **Data Persistence**: Maintain previously fetched data for each cryptocurrency even when toggling between coins and intervals.
5. **Responsive UI**: A clean and user-friendly interface for real-time interaction and chart display.

Technologies Used

- **JavaScript**: For client-side scripting.
- **Lightweight Charts**: To display the candlestick charts.
- **Binance WebSocket API**: For receiving real-time market data.
- **HTML/CSS**: To structure and style the user interface.
- **Local Storage**: To persist the chart data locally when switching between coins and intervals.

Core Features

1. WebSocket Connection:

- The app connects to the Binance WebSocket API for real-time market data.
- The connection is dynamic, adjusting based on the selected cryptocurrency and interval.

WebSocket Endpoint:

```
bash
Copy code
wss://stream.binance.com:9443/ws/<symbol>@kline_<interval>
```

Supported Symbols:

- `ETH/USDT`: Ethereum to Tether.
- `BNB/USDT`: Binance Coin to Tether.
- `DOT/USDT`: Polkadot to Tether.

Supported Intervals:

- `1m`: 1 minute
- `3m`: 3 minutes
- `5m`: 5 minutes

2. Dynamic Cryptocurrency and Interval Toggle:

- Users can select between `ETH/USDT`, `BNB/USDT`, and `DOT/USDT` from a dropdown menu.
- Users can switch between `1m`, `3m`, and `5m` intervals, and the candlestick chart updates accordingly.

Persistence:

The chart data for each cryptocurrency and interval is stored in local storage. This ensures that when the user switches between coins or intervals, previously received data is not lost.

3. Charting and Visualization:

- The candlestick chart displays live market updates.
- The chart updates without flickering, providing a smooth user experience when new data is received.
- **Lightweight Charts** is used to plot the candlestick data.

Features of the Chart:

- Real-time candlestick updates.
- Responsive resizing based on the window size.
- Local storage is used to cache previous data for faster switching between coins and intervals.

4. Data Management:

- The app stores the fetched candlestick data for each coin and interval in local storage.
- When a user switches back to a previously selected coin or interval, the saved data is restored without requiring a new WebSocket connection.
- If no data is saved, a new WebSocket connection is established to fetch fresh data.

5. User Interface (UI):

- A clean and responsive UI allows users to:
    - Select a cryptocurrency from a dropdown (`ETH/USDT`, `BNB/USDT`, or `DOT/USDT`).
    - Select a time interval (`1m`, `3m`, or `5m`).
    - View real-time candlestick charts.

Code Walkthrough

1. **WebSocket Connection**:
    a. When a user selects a coin and interval, the app establishes a WebSocket connection using the Binance WebSocket API.
    b. Example WebSocket connection for `ETH/USDT` with a `1-minute` interval:

bash

```
Copy code
wss://stream.binance.com:9443/ws/ethusdt@kline_1m
```

2. **Chart Initialization**:
    a. The candlestick chart is initialized using the `Lightweight Charts` library.
    b. Data from the WebSocket is processed and fed into the chart in real-time.
3. **Handling Coin and Interval Changes**:
    a. When the user switches between coins or intervals, the app checks if data for the selected combination exists in local storage.
    b. If data exists, it loads the chart with the saved data. If not, a new WebSocket connection is made to fetch fresh data.
4. **Data Persistence**:
    a. The app saves the candlestick data in local storage, using the coin and interval as keys.
    b. For example, data for `ETH/USDT` at a `1-minute` interval is saved under the key `ethusdt_1m`.

Instructions to Run the Project

1. **Clone the Repository**:
    a. Clone or download the project files from the repository.
2. **Open `index.html`**:
    a. Open the `index.html` file in your browser.
    b. Ensure you have an active internet connection for accessing the Binance WebSocket API.
3. **Testing the Features**:
    a. Select a cryptocurrency from the dropdown.
    b. Choose a time interval (`1m`, `3m`, or `5m`).
    c. Observe the candlestick chart updating in real-time.
4. **Debugging**:
    a. Open the browser console (F12) to see WebSocket connection logs or error messages.

Potential Extensions

- Add support for more cryptocurrencies by extending the symbol map and WebSocket connection logic.
- Implement more advanced charting features like indicators (moving averages, volume).
- Integrate backend support for historical data storage and analysis.

Conclusion

This project successfully implements a real-time cryptocurrency market data visualization tool using Binance's WebSocket API. With a responsive UI, real-time updates, and persistent data handling, users can toggle between multiple coins and intervals without losing historical chart data.