Hybrid Trading Simulator Frontend

This project is the React-based frontend for a Hybrid Trading Simulator, providing a user interface to interact with a cryptocurrency exchange backend. It displays a live order book, allows users to place buy and sell orders, and connects to MetaMask for wallet integration.

Project Structure

- App.tsx: The main React component that orchestrates the entire application. It manages wallet connection, WebSocket communication for real-time order book updates, and renders the OrderBook and TradeForm components.
- components/OrderBook.tsx: A React component responsible for displaying the live order book data (bids and asks). It visualizes market depth and updates in real-time as data streams from the backend WebSocket.
- components/TradeForm.tsx: A React component that provides the user interface for placing trade orders. Users can select between buy/sell, limit/market order types, and input amount and price (for limit orders). It sends order data to the backend API.
- index.html: The main HTML file that serves as the entry point for the React application. It contains the root div where the React app is mounted.
- main.tsx: The entry point for the React application, responsible for rendering the App component into the index.html file.
- index.css: The primary CSS file, utilizing Tailwind CSS directives (@tailwind). It also includes custom scrollbar styles and imports the 'Inter' font for typography.
- package.json: Manages the frontend project's dependencies (React, Ethers, etc.), development scripts (start, build), and other metadata.
- tailwind.config.js: The configuration file for Tailwind CSS, specifying content paths for JIT compilation and allowing for theme extensions and plugin integration.

Features

- **Wallet Connection**: Integrates with MetaMask, allowing users to connect their Ethereum wallet.
- **Real-time Order Book**: Displays live bid and ask data, providing market depth updates via WebSocket.
- **Order Placement**: Enables users to submit buy and sell orders, supporting both limit and market order types.
- **Responsive UI**: Built with Tailwind CSS for a modern, responsive, and mobile-friendly design.
- **Type-Safe Development**: Developed with TypeScript for improved code quality and maintainability.

Setup and Installation

Prerequisites

- **Node.js & npm/yarn**: For managing project dependencies and running the development server.
- **MetaMask**: Browser extension for connecting to an Ethereum wallet.
- **Backend Running**: This frontend requires the associated backend service (from the api.py project) to be running and accessible at http://localhost:8000.

Installation

1. **Clone the repository** (if you haven't already):
   ```
   git clone <repository-url>
   cd hybrid-trading-simulator-frontend
   ```

2. **Install Node.js dependencies**:
   ```
   npm install
   # or
   yarn install
   ```

Running the Application

1. **Start the development server**:
   ```
   npm start
   # or
   yarn start
   ```

   This will typically start the application on http://localhost:5173 (or another available port). The browser should automatically open.

Usage

1. **Connect Wallet**: Click the "Connect Wallet" button in the header to connect your MetaMask wallet.
2. **View Order Book**: The "Order Book (BTC-USD)" section will display real-time bids and asks streaming from the backend.
3. **Place Orders**:
   - Select "Buy" or "Sell".
   - Choose "Limit Order" or "Market Order".
   - Enter the "Amount (BTC)".
   - If "Limit Order" is selected, enter the "Price (USD)".
   - Click "Place Buy Order" or "Place Sell Order".
   - A confirmation message will appear upon successful order submission.

Technologies Used

- **React**: JavaScript library for building user interfaces.
- **TypeScript**: Strongly typed superset of JavaScript.
- **Tailwind CSS**: A utility-first CSS framework for rapid UI development.
- **Ethers.js**: A comprehensive JavaScript library for interacting with the Ethereum blockchain and its ecosystem.
- **Vite**: A fast frontend development tool that provides a quick development server and build process.

Development Notes

- **Backend Integration**: The TradeForm.tsx sends order data to http://localhost:8000/api/v1/order, and App.tsx connects to ws://localhost:8000/ws/market_data. Ensure your backend is running on these ports.
- **Error Handling**: The application includes basic error handling for wallet connection and API calls. For a production-grade application, more robust error displays (e.g., toast notifications, dedicated error messages) would be beneficial instead of alert() calls.
- **MetaMask Events**: App.tsx listens for accountsChanged and chainChanged events from MetaMask to ensure the UI stays synchronized with the user's wallet state.
- **Styling**: Tailwind CSS is used extensively for styling, allowing for highly customizable and responsive components. Custom scrollbar styles are also included in index.css.