Step-by-Step Guide to Run the Trade Simulator

1. Prerequisites:

- **Python:** Ensure you have Python 3.7 or newer installed. If not, download it from python.org.
- **pip:** Python's package installer. It usually comes with Python.
- **Web Browser:** A modern browser like Chrome, Firefox, Edge, or Safari.
- **Text Editor:** Any text editor to save the code files (e.g., VS Code, Sublime Text, Notepad++, or even basic Notepad/TextEdit).
- **VPN (Potentially for OKX):** You've mentioned that a VPN might be needed for OKX access depending on your region. If so, ensure your VPN is active and correctly configured *before* running the Python backend.

2. Save the Code Files:

- **Backend (Python):**
  - Take the Python code from the artifact trade_simulator_backend_combined_v1.
  - Create a new folder on your computer (e.g., MyTradeSimulator).
  - Save the Python code in this folder as a file named trade_backend.py.
- **Frontend (HTML):**
  - I will provide the corresponding HTML code below. Copy it.
  - Save this HTML code in the **same folder** (e.g., MyTradeSimulator) as a file named trade_frontend.html.

(The HTML frontend code that connects to port 8000 will be provided in the next immersive block.)

3. Install Required Python Libraries:

- Open your computer's terminal or command prompt:
  - **Windows:** Search for "cmd" or "PowerShell".
  - **macOS:** Search for "Terminal" in Spotlight or Applications.
  - **Linux:** Usually Ctrl+Alt+T or find "Terminal" in your applications menu.
- Navigate to the folder where you saved the Python and HTML files. For example, if you saved them in MyTradeSimulator on your Desktop:
  - Windows: cd Desktop\MyTradeSimulator
  - macOS/Linux: cd Desktop/MyTradeSimulator
- Once in the correct directory, install the necessary Python libraries by typing the following command and pressing Enter:
  pip install websockets numpy

  Wait for the installation to complete. You should see messages indicating successful installation.

4. Run the Python Backend Server:

- In the same terminal window (which should still be in the MyTradeSimulator directory), run

the Python backend script:
python trade_backend.py

- **What to Expect in the Terminal:**
  - You'll see log messages from the script, starting with something like:
    INFO:root:Starting Trade Simulator Backend (Port 8000 v2.2)...
    INFO:websockets.server:server listening on ws://localhost:8000
    INFO:root:UI WebSocket server started on ws://localhost:8000
    INFO:root:Attempting to connect to OKX L2 feed: wss://ws.gomarket-cpp.goquant.io/ws/l2-orderbook/okx/BTC-USDT-SWAP
  - If the backend successfully connects to the OKX WebSocket, you'll see further logs related to that.
  - If there are issues connecting to OKX (e.g., VPN problems, network restrictions, OKX server issues), you'll see error messages in this terminal.
  - **Crucially, keep this terminal window open.** Closing it will stop the backend server, and your frontend will not be able to connect or receive data.
  - Monitor this terminal for any errors or important status updates.

5. Run the HTML Frontend in Your Browser:

- Go to the MyTradeSimulator folder using your computer's file explorer.
- Find the trade_frontend.html file.
- Open this file with your web browser. You can usually do this by:
  - Double-clicking the trade_frontend.html file.
  - Right-clicking the file, selecting "Open with," and choosing your preferred browser.
  - Dragging the trade_frontend.html file into an open browser window.
  - Typing the full file path into your browser's address bar (e.g., file:///C:/Users/YourName/Desktop/MyTradeSimulator/trade_frontend.html).

6. Observe the "Final Outlook" in Your Browser:

- **Browser Developer Console (Important for Debugging):**
  - As soon as the trade_frontend.html page loads in your browser, open the Developer Console.
    - You can usually do this by pressing F12.
    - Navigate to the "Console" tab within the developer tools.
  - You should see log messages from the frontend's JavaScript, starting with something like:
    WebSocket: Attempting to connect to backend at ws://localhost:8000

- **UI Connection Status Bar:**
  - At the top of the webpage, there's a connection status bar.
  - It should initially say "Connecting to Backend..." (orange).
  - If the connection to your running Python backend is successful, it will change to

"Connected to Backend" (green).

- **Data Flow and UI Updates:**
  - If the Python backend is running, connected to OKX, and receiving market data, it will process this data and send it to your frontend via the WebSocket connection.
  - You should then see the "Processed Output Values" section in the UI (Best Bid, Best Ask, Expected Slippage, Expected Fees, etc.) start to update with values.
  - The "Live L2 Order Book" tables (Asks and Bids) should also populate with data.
  - The "Raw Message from Backend" box at the bottom of the right panel will show the raw JSON data being received by the frontend from the backend. This is very useful for debugging.
- **Interacting with Input Parameters:**
  - In the left "Input Parameters" panel, try changing values like "Quantity (USD Equivalent)" or "Annualized Volatility (%)".
  - Click the "Send Parameters to Backend" button.
  - **Check the Python backend's terminal:** You should see a log message indicating that it received new parameters from the UI.
  - The output values displayed in the UI should then update based on these new parameters and the latest market data tick processed by the backend.

7. Troubleshooting Common Issues:

- **Python Backend Error: "Address already in use" (for port 8000):**
  - This means another program on your computer (possibly a previous, unstopped instance of your trade_backend.py script) is already using port 8000.
  - **Solution:** Stop the other program. If you can't find it, you can change the UI_WEBSOCKET_PORT in *both* the trade_backend.py script AND the backendUrl variable in the trade_frontend.html script to a different number (e.g., 8767 or 8760). Then, restart the backend and refresh the frontend.
- **Frontend UI shows "Disconnected" or "Connection Error":**
  - **Is the Python backend script (trade_backend.py) running?** Check its terminal window.
  - **Are there errors in the Python backend terminal?** It might have crashed or failed to start its local WebSocket server.
  - **Did the backend fail to connect to OKX?** (Check backend logs for OKX connection errors; VPN might be needed).
  - **Check the Browser Console (F12):** Look for WebSocket connection errors. It might say "Connection refused" if the backend server isn't running or isn't accessible at ws://localhost:8000.
- **Backend seems to run and connect to OKX, but no data appears in the UI:**
  - **Check the Browser Console (F12):** Are there any JavaScript errors listed that might be preventing the UI from updating after receiving data?
  - **Check the Python Backend Logs:** Is it actually processing ticks and attempting to

broadcast messages to UI clients? Look for "Error sending to a UI client" messages.
    - **Firewall:** While less common for localhost connections, ensure no personal firewall software on your computer is blocking connections between your browser and localhost:8000.
- **OKX Connection Issues (seen in Python backend logs):**
    - **VPN:** If OKX requires a VPN from your location, ensure it's active and working correctly for the Python environment.
    - **OKX Endpoint/API:** The specific WebSocket endpoint wss://ws.gomarket-cpp.goquant.io/ws/l2-orderbook/okx/BTC-USDT-SWAP might be temporarily unavailable, or its API requirements (like specific subscription messages) might have changed. The current script assumes it streams data directly or that a generic subscription (if you uncomment and modify that part) would work. If problems persist here, you might need to consult the precise API documentation for this particular OKX endpoint.

By carefully following these steps and checking the logs in both the backend terminal and the browser console, you should be able to get the system running and diagnose any issues that arise. Good luck!