Quant Developer Assessment: Market-Making + Hedging Simulation

Objective

Design and implement a simplified market-making strategy that:

- **Quotes on Exchange 1** (providing tight spreads and liquidity)
- **Hedges on Exchange 2** (to stay delta-neutral)
- Tracks **inventory**, **risk**, and **PnL** in real time
- Simulates **realistic order flow** and **market data**
- Provides a minimal API to interact with the system

Instructions

1. Core Strategy Logic

Implement the following in Golang/Python:

Exchange Simulation

- Simulate two exchanges (Exchange 1 & Exchange 2), each with:
    - Independent order books (limit & market order support)
    - Price feeds that evolve over time
    - Basic matching engine or instant fill logic

Market-Making Logic

- Quote **both bid and ask** prices on Exchange 1 based on mid-price and desired spread
- Adjust quotes dynamically based on:
    - Inventory skew
    - Volatility or price drift
- Maintain **order queue awareness** if simulating realistic matching

Hedging Logic

- Whenever a quote on Exchange 1 is filled:
    - Place a corresponding hedge order on Exchange 2 (instant or slippage-aware)
    - Ensure delta-neutrality is maintained
- Log latency between quote-fill and hedge-execution (simulate delay)

2. Portfolio & Risk Management

- Track net inventory per asset
- Compute and display:
    - **Delta** exposure
    - **Realized / Unrealized PnL**
    - **Inventory imbalances / thresholds**
    - **Spread captured on Exchange 1 vs hedge cost on Exchange 2**

Risk Controls

- Implement basic risk throttles:
    - Max position size
    - Max loss over rolling window (e.g., 5-minute PnL drop triggers quote pause)
    - Kill-switch if hedge fails or latency spikes

3. Simulation Environment

- Simulate realistic price movement (e.g., random walk with volatility)
- Simulate client order flow hitting the Exchange 1 quotes
- Run a time-stepped simulation over a configurable interval (e.g., 1000 ticks)

4. REST API

Expose endpoints to:

- Fetch current quotes, positions, PnL
- Submit a manual test fill on Exchange 1
- Pause/Resume quoting
- Query risk status and throttle state
- Reset simulation

5. Documentation & Tests

- **README.md describing:**
    - Strategy logic
    - Assumptions (e.g., latency model, pricing model)
    - Risk control methods
    - Observations about spread capture & hedging costs
- **Unit tests for:**
    - Order matching
    - Inventory updates
    - Hedging logic
- **Makefile with:**
    - make run — start API
    - make test — run test suite

Evaluation Criteria

Category	Expectations
Strategy Logic	Quoting & hedging linked correctly; spreads managed dynamically
Market Simulation	Reasonable assumptions for price updates, order matching, fill latency
Risk Awareness	Inventory limits, kill switches, delta management implemented clearly
Code Quality	Modular, well-documented, easy to read and test
API Design	Simple and useful interface for triggering fills, inspecting state
Reporting	Output or logs that show trade decisions, PnL evolution, risk exposure

Optional Bonus Features

- Hedging delay sensitivity model (latency-aware execution slippage)
- Partial fills and order queue simulation
- Adaptive spread width based on volatility
- A simple CLI or dashboard to visualize:
    - Live PnL
    - Quote position
    - Inventory imbalance

Note

- keep it simple stupid - **KISS** principle
- Try to finish the task within 4 hours. We do not ask for a perfect production ready service.