Assignment - Simple Binance Order Book

Backend (Node.js, Express)

- Fetch initial snapshot
  Call
  GET https://api.binance.com/api/v3/depth?symbol=BTCUSDT&limit=1000
  which returns lastUpdateId, bids, asks.
  Docs: Binance REST Order Book

- Subscribe to Binance WebSocket
  Connect to
  wss://stream.binance.com:9443/ws/btcusdt@depth@100ms
- Initial sync (follow Binance's guide)
  1. Start the WebSocket and buffer events.
  2. Fetch the REST snapshot.
  3. Discard any events with u <= lastUpdateId (older than snapshot).
  4. The first event you apply must satisfy:
     U <= lastUpdateId + 1 <= u
     (this ensures it "bridges" the snapshot).
  5. After that, update your book with each event, set lastUpdateId = u.
- Maintain an in-memory order book (two Maps for bids/asks).
- Apply updates as absolute quantities (qty = "0" means delete that price level).
- Expose your own WebSocket (e.g., ws://localhost:3001/orderbook).
  - Send to connected frontend clients a ready-to-render payload with:
    - Top 5 bids (sorted descending).
    - Top 5 asks (sorted ascending).
    - Timestamp.
    - Example:

```
{
  "type": "orderbook",
  "symbol": "BTCUSDT",
  "ts": 1710000000000,
  "bids": [["60250.0","1.10"], ...],
  "asks": [["60251.0","0.90"], ...],
}
```

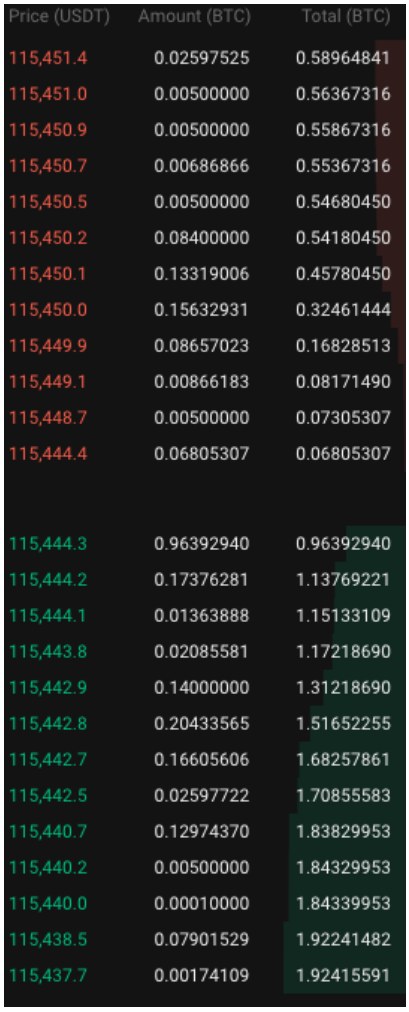
- Resilience
  - Reconnect to Binance if the WS closes (retry every 1-2 seconds).
  - Send a heartbeat to your frontend clients every ~5 seconds: { "type": "ping" }.
    If they reply with { "type": "pong" }, that's enough.

Frontend (Next.js)

- Connect only to your backend WebSocket (ws://localhost:3001/orderbook).
- Handle
    - { "type": "ping" } reply { "type": "pong" }.
    - { "type": "orderbook", ... } update the UI.
- UI requirements
    - Show Best Bid, Best Ask, Spread.
    - Show Top 5 bids (descending by price) and Top 5 asks (ascending).
    - Columns: Price, Quantity, Total (Cumulative quantity of bids or asks)

What's in Scope

- One symbol only BTCUSDT.
- No need for grouping, multiple symbols, or REST endpoints for clients.
- sorting rules (bids = highest first, asks = lowest first)
- Focus is on
    - Correct snapshot + diff sync.
    - Correct book updates (absolute quantity, remove on zero).
    - Basic resilience.



Price (USDT)	Amount (BTC)	Total (BTC)
115,451.4	0.02597525	0.58964841
115,451.0	0.00500000	0.56367316
115,450.9	0.00500000	0.55867316
115,450.7	0.00686866	0.55367316
115,450.5	0.00500000	0.54680450
115,450.2	0.08400000	0.54180450
115,450.1	0.13319006	0.45780450
115,450.0	0.15632931	0.32461444
115,449.9	0.08657023	0.16828513
115,449.1	0.00866183	0.08171490
115,448.7	0.00500000	0.07305307
115,444.4	0.06805307	0.06805307
115,444.3	0.96392940	0.96392940
115,444.2	0.17376281	1.13769221
115,444.1	0.01363888	1.15133109
115,443.8	0.02085581	1.17218690
115,442.9	0.14000000	1.31218690
115,442.8	0.20433565	1.51652255
115,442.7	0.16605606	1.68257861
115,442.5	0.02597722	1.70855583
115,440.7	0.12974370	1.83829953
115,440.2	0.00500000	1.84329953
115,440.0	0.00010000	1.84339953
115,438.5	0.07901529	1.92241482
115,437.7	0.00174109	1.92415591

Run Expectations

- Backend: runs Express + WS on port 3001.
- Frontend: runs Next.js app on port 3000.
- Visit http://localhost:3000 see order book updating in real time.