Order Status Notification via WebSocket (Sequence)

Participants: User, Frontend / UI, WebSocket Server, Order/Trade Service, Database

Phase A: Connect & Authenticate

- User → Frontend / UI: open trading page
- Frontend / UI → WebSocket Server: connect()
- WebSocket Server → Frontend / UI: auth request (token?)
- Frontend / UI → WebSocket Server: send token (JWT)

alt [token invalid]
- WebSocket Server → Frontend / UI: disconnect / unauthorized

[token ok]
- WebSocket Server → WebSocket Server: joinRoom(userId)
- WebSocket Server → Frontend / UI: connected

Phase B: Notify After Status Change

- Order/Trade Service → Order/Trade Service: order status changed
- Order/Trade Service → Database: insert NotificationLog(eventId, userId, payload, delivered=false)
- Order/Trade Service → WebSocket Server: emitToUser(userId, payload)
- WebSocket Server → Frontend / UI: push(payload)
- Frontend / UI → User: update UI (order/trade status)

opt [ACK (bonus)]
- Frontend / UI → WebSocket Server: ACK(eventId)
- WebSocket Server → Order/Trade Service: ackReceived(eventId)
- Order/Trade Service → Database: mark delivered=true (eventId)

alt [disconnected (reconnect + replay)]
- Frontend / UI → WebSocket Server: reconnect()
- WebSocket Server → Frontend / UI: auth request (token?)
- Frontend / UI → WebSocket Server: send token (JWT)
- WebSocket Server → WebSocket Server: joinRoom(userId)
- Frontend / UI → WebSocket Server: sync(lastEventId)
- WebSocket Server → Database: fetch missed notifications(since lastEventId)
- Database → WebSocket Server: missed payloads
- WebSocket Server → Frontend / UI: replay(missed payloads)
- Frontend / UI → User: update UI