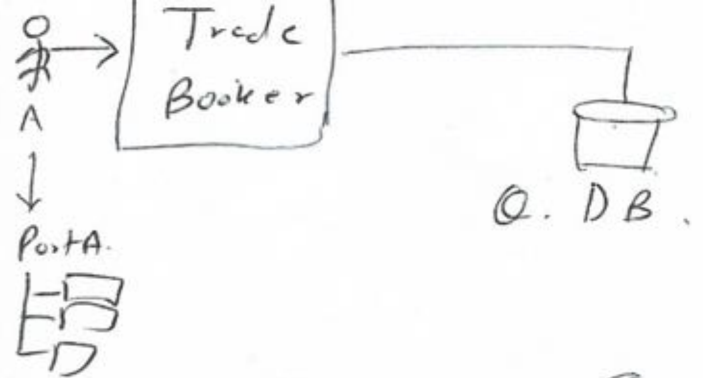
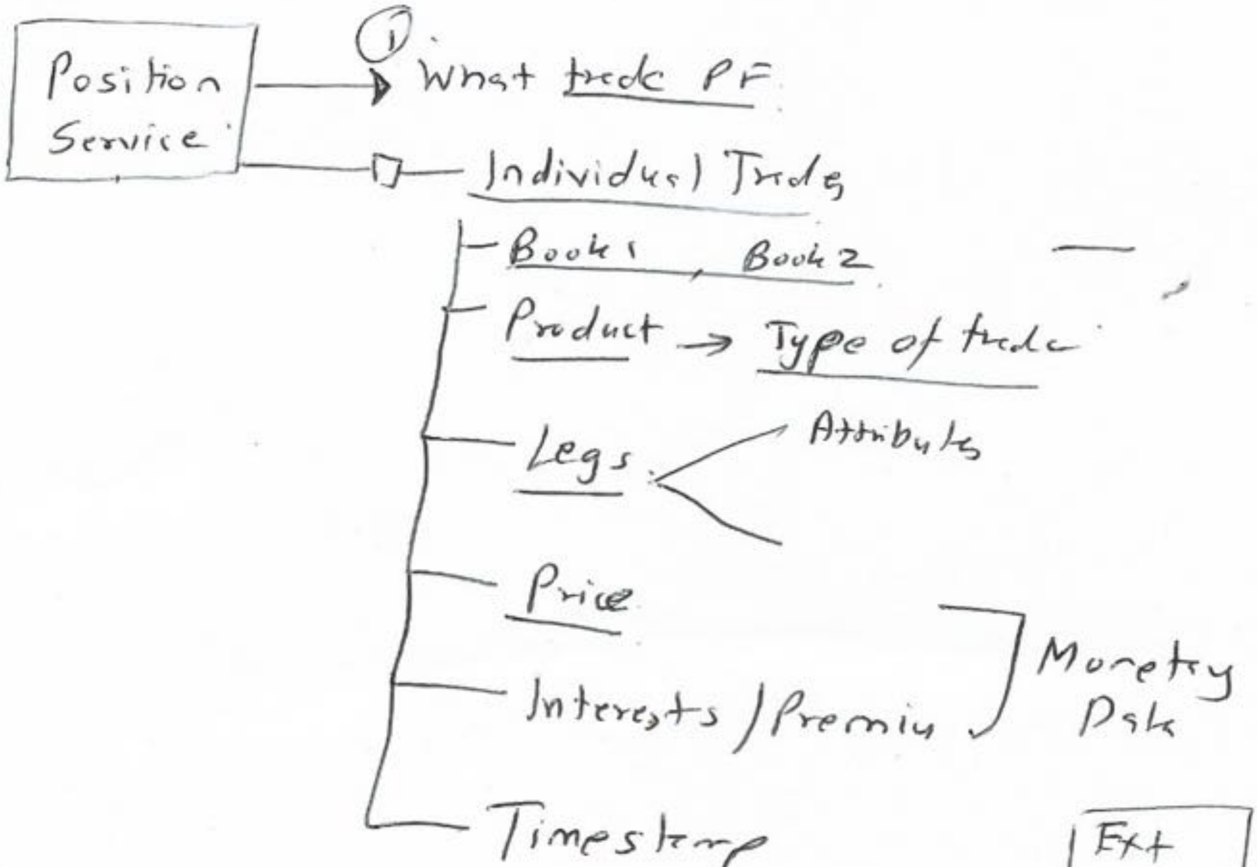
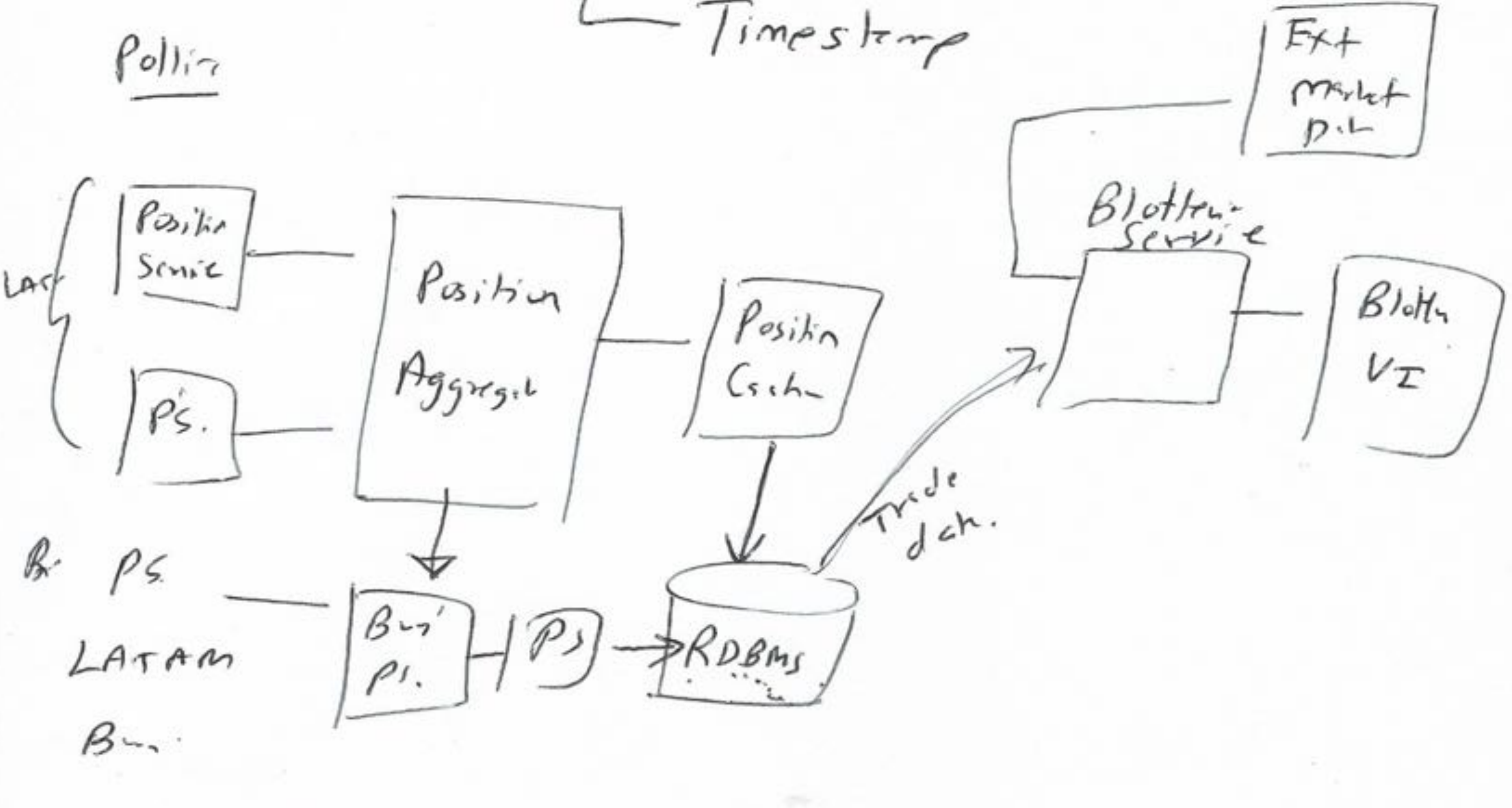


Trade Booker
A
O. DB.
PortA.

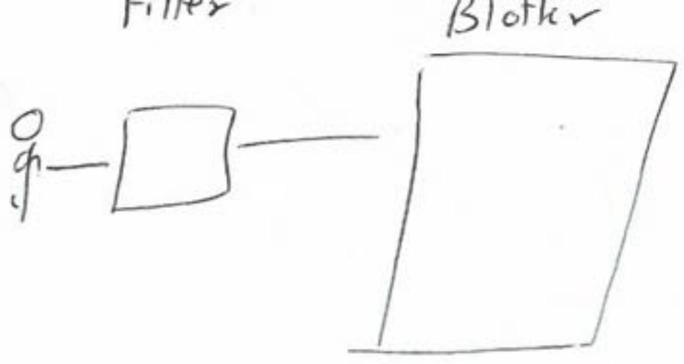


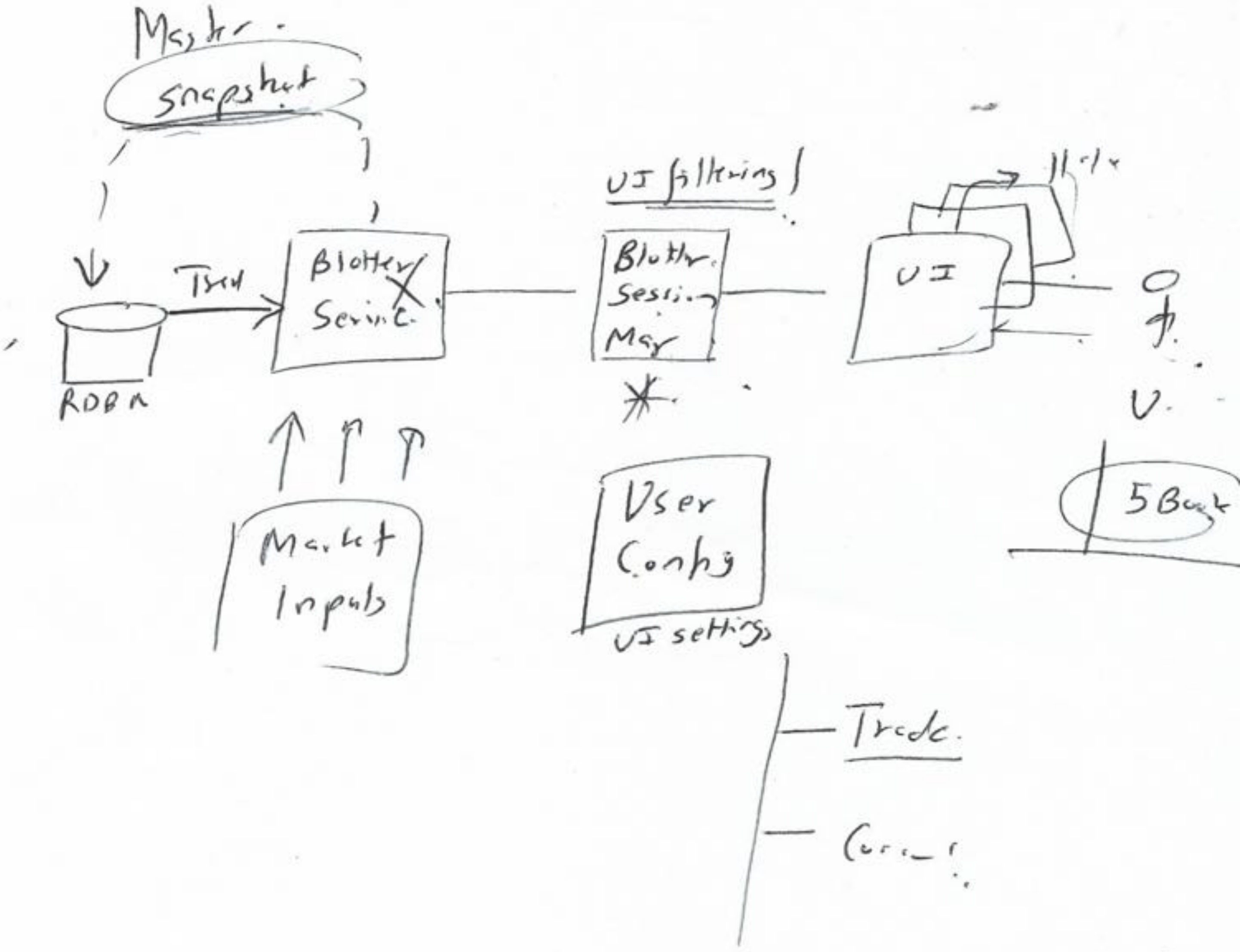
Position Service
1. What trade PF
Individual Trade
Book1, Book2
Product → Type of trade
Legs
Attributes
Price
Interests / Premium
Monetary Data
Timestamp

Pollin



Ext Market Data
LAC
Position Service
PS.
Position Aggregat
Position Cache
Blotter Service
Blotter UI
Trade data.
PS
LATAM
Bus
Bus Pl.
PS
RDBMS


Filter
Blotter


Master
snapshot
RDBR
Trd
Blotter Service
UI filtering
Blotter Session Mgr
UI
User Config
UI settings
Market Inputs
Trade
5 Bank

```
    6. _fetch_from_api() - Simulate API fetch (placeholder)
    """

    def __init__(self, queue: Optional[queue.Queue] = None):
        # TODO: Initialize service components
      self.portfolio = ______ # Portfolio that we are interested in
                              → Can be a static config for
                                this ingestion service.
      self.~~lastEven~~
        pass cutoffTS / → ~~current~~ current time → # This will have last processed
                                                      trade Ts.
    def start(self):
        """Start the ingestion service"""
        # TODO: Start producer thread

          # Fetch Last Processed Ts.




        pass

    def stop(self):
        """Gracefully stop the service"""
        # TODO: Stop threads and cleanup

          # Store Last Processed Timestamp.
            or publish


        pass


    def _producer_loop(self):
        """Main producer loop - fetch, validate, enqueue trades"""
        # TODO: Implement producer-consumer logic

          processedTS = 0

          While not test_queue.empty():

# Fetch           trade_data = test_queue.get()

                  for trade in trade_data:

② Validate.           if validate_trade(trade):

                            # Write trade as processed.

                            ~~or put on~~

③ Publish                   or publish to a Message Queue

        pass ④ Update       # update processed Ts
                 state      # update processed

    def _validate_trade(self, trade_data: Dict[str, Any]) -> bool:
        """Validate trade data before enqueuing"""
        # TODO: Implement validation rules
        # Rules:
        # - quantity > 0
        # - price > 0
        # - symbol in VALID_SYMBOLS
        # - side in [TradeSide.BUY, TradeSide.SELL]
        # - status in [TradeStatus.FILLED, TradeStatus.PARTIAL, TradeStatus.CANCELED]

      if trade cancelled → Invalid

      if not in our PF → Invalid.



        pass

    def _fetch_from_api(self) -> Optional[Dict[str, Any]]:
        """Fetch trade from external API (placeholder)"""
        # TODO: Simulate API call, process the result
        # Return None if no trade available











        pass


# TEST HARNESS (DO NOT MODIFY)
def run_test():
    """Test your implementation"""
    print("Starting TradeIngestionService Test...")

    # Create service with shared queue
    test_queue = queue.Queue()
    service = TradeIngestionService(test_queue)

    # Start service
    service.start()

    # Run for 10 seconds
    time.sleep(10)

    # Stop service
    service.stop()

    # Process remaining trades
    processed = 0
    while not test_queue.empty():
        trade_data = test_queue.get()
        print(f"Trade {trade_data['trade_id']}: {trade_data['symbol']} - {trade_data['quantity']} shares")
        processed += 1

    print(f"\nTest Complete: {processed} trades processed")
    assert processed >= 5, "Not enough trades generated!"
```