Code prompts

- Rectified the implementation of createReservation method to use Redis cache and not directly DB (DB writes) Prompt - "Based on the changes above, we also need to update the createReservation method in the ReservationService class to use the kafka approach rather than incrementReservedCount directly. Can you make these updates in this class and also update the corresponding tests as applicable"
- Corrected poll Interval(100ms → 10ms) to wait for reservation creation while implementing async reservation creations

  Prompt - "If RESERVATION_POLL_INTERVAL_MS is set to 100ms, then this will add atleast a 100ms latency even in the happy cases where the reservation request was processed within 10ms, but we are waiting additional 90ms to poll. Should we consider reducing the poll interval to a smaller value, like 10ms? What are the implications? Can Redis handle this frequent polling?"

  Claude proposed an initial smaller poll interval(5ms for 5 times) with progressive exponential backoff upto 100ms , which seemed like a reasonable approach to optimize happy path scenarios, while avoiding retry storms for failure cases.
- ConfimReservation method to put load on DB?

  Prompt - "In the confirmReservation method, we have directly updated the reservation in the database. I was curious about the implications of this pattern, because this can also generate a lot Write transactions to the database. My understanding is that since we are updating separate rows per reservation, identified by it's reservation id, we do not have any concurrency challenges here. However, if we are producing a lot of WPS on the same database table, even if it's for different rows, will that cause stress on the database"

  Reasoning - Because orders will be confirmed at different time for different users, the expected traffic for this will be around 2k WPS which seems reasonable for the Db to handle.
- Kafka queue topic configuration

  "The consumer configuration of kafka defined in application.yml doesn't seem to be correctly fine tuned. I see several issues with the current configuration

  1. max-poll-interval-ms is set to 30s. What does this configuration signify? From our design, we expect the batch to be processed within 10 milliseconds.
  2. fetch-max-wait-ms is set to 500, which is high. My understanding from this configuration is that it will wait 500ms to accumulate a batch of 250 requests. For a latency sensitive application, this seems high and should be in a few milliseconds.
  3. We have the same consumer configuration for all the topics. That doesn't seem ideal because the throughput expectations would be different. If we fine tune the consumer to the highest throughput we require in our system, it should be fine but what are your thoughts on it?
  4. We wanted to create separate topic partitions for every sku in the reservation-requests topic. Where is that configuration?"

- Using a single DB transaction to check for and reserve available inventory

  Prompt - ”why are we using inventoryRepository to get available count and increment reserved count separately, according to our design these checks should be happening in a single transaction in the DB(postgreSQL)”

  “also, restrict request.getQuantity() to 1”

- Edge case handling for reservation allotment where request size for reservation requests is lesser than batch size (Partial allotment)

  Prompt - “In this query, we are trying to allocate the totalQuantity, which is a sum of all the quantities requested in validated requests. As an edge case, this could result in an under-allocation where we had an opportunity to do partial allocation. For example, if we had available quanitity as 240 and the sum of all validated requests quantity was 250, we would end up not allocating any quantity. Instead, with partial allocation, we could have allocated 240 and rejected only 10rejectAllRequests method is simply updating the ValidatedRequest object. How will the ReservationService get to know that the request is rejected? Since the ReservationService is polling redis cache for reservation, should we also create an entry in redis for rejected requests so that the ReservationService createReservation api can get the status of even rejected requests and respond back to the client?”

  "This change will result in a getAvailableCount everytime. Instead, I just want to make a getAvailableCount in the partial allocation case. Fist we can try to do full allocation without getting the available count, if rowsUpdated is 0, then that means i am in the partial allocation stage and in that stage i can first read the available count and then only allocate that many" (rejecting always-read approach)

- Intimidated the ReservationService about the rejected reservation requests.

  Prompt - “rejectAllRequests method is simply updating the ValidatedRequest object. How will the ReservationService get to know that the request is rejected? Since the ReservationService is polling redis cache for reservation, should we also create an entry in redis for rejected requests so that the ReservationService createReservation api can get the status of even rejected requests and respond back to the client?”

- Corrected product lookup from DB to cache to reduce DB load. findproductbySkuId was directly attempting to fetch from DB(PostgreSQL) first in the code and not Redis cache data, made it to look for product info from cache.

- Re-implemented 3 layer redundancy for 120s hold - ”search for 3 layers of **Three-Layer Redundancy System** mentioned in SYSTEM ARCHITECTURE ULTRA V2 and see if it has been implemented in our app”

- Implemented Rate Limiting and fair queueing wherever it went missing in the architecture "according to the design doc SYSTEM_ARCHITECTURE_ULTRA_V2, Decision 5: Rate Limiting & Fair Queuing has implementation of rate limiting, please read from there and implement rate limiting on this app”

- **Missing Rejection Caching for Validation Failures** Prompt - "cache these rejections in Redis, so that the polling ReservationService does not timeout waiting for these rejected requests instead of getting immediate failure notification"

- **Missing Exception Handler for ReservationFailedException** - Prompt - "please handle this exception appropriately"
- **Duplicate Inventory Decrement Operations** - Prompt - "Event-driven updates: ReservationService publishes Kafka events expecting consumers to update inventory, do this one instead of directly calling inventoryRepository.confirmReservation()"
- **"No Event Consumer for ReservationEvents"** - "we will add consumer for these topics eventually, you can ignore it for now”
- we also setup aws infrastructure (redis,PostgreSQL instances) using aws cli and handed over the credentials to claude for it to setup the infra
- Also generated **ARCHITECTURE_FLOWCHARTS.md** file (can be read via mermaid extension for markdown files) from the code implementation, which has the following charts for a better overview

  1. High-Level Architecture
  2. Reservation Flow
  3. Checkout Flow
  4. Inventory Management Flow
  5. Caching Strategy
  6. Event-Driven Architecture
  7. Database Schema
  8. Component Interaction