Technical Requirements – Order Queue Oupost

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/g98LzeMNxdeqdhGDiYMn/h0XHb5luoXpGbnVGWmlz/details

🎭 Scenario

This challenge simulates a real-time order queue system. A producer sends orders into a channel with a fixed delay, while a consumer processes them with a random delay. If the consumer is paused or slower than the producer, the queue starts to grow — eventually reaching and exceeding its capacity.

🎨 Figma Mockups

> https://www.figma.com/design/SAX4hR2rhXFUZgQjm1iZXF/Async-Adventures?node-id=3-589

🎯 Feature Goal

Build a UI that visually reflects how the queue grows or shrinks depending on the system state. The user can toggle between **Pause** and **Start** to observe queue behavior in real time.

📌 Requirements

- Initial screen:
  - Title: Order Queue Outpost
  - Description: "Press Play to start processing orders"
  - Button: ▶️ Start

The key idea of this challenge is to simulate the interaction between a **producer** and a **consumer** in real time.

- **When "Start" is pressed:**
  - The **producer** begins sending orders into the queue with a fixed delay (**250 ms** between each new order).
  - The **consumer** starts processing orders with a random delay (**100–250 ms**).
  - Since the consumer’s speed is usually greater or equal to the producer’s speed, the queue either remains stable or grows slowly.
- **When "Pause" is pressed:**
  - The **consumer** stops completely and no longer processes orders.
  - The **producer** continues to push new orders at the same rate (250 ms).

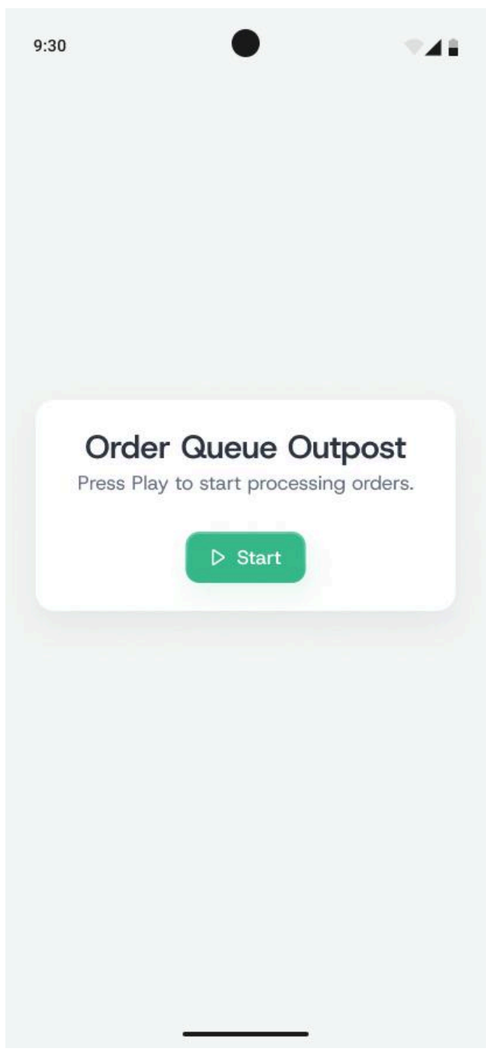
    - The **queue starts growing**, visually reflected by:
        - A queue counter (e.g., Queue: 10/25)
        - A progress bar changing color:
            - 0–33% → green
            - 34–66% → yellow
            - 67–100% → red
            - 100% → overflow state (with animation as shown in the Figma mockup)
- **When "Start" is pressed again:**
    - The **consumer** resumes processing orders.
    - Since its speed is higher than the producer's, the **queue starts decreasing**.
    - The progress bar gradually transitions back in color and returns to a value below 100%.

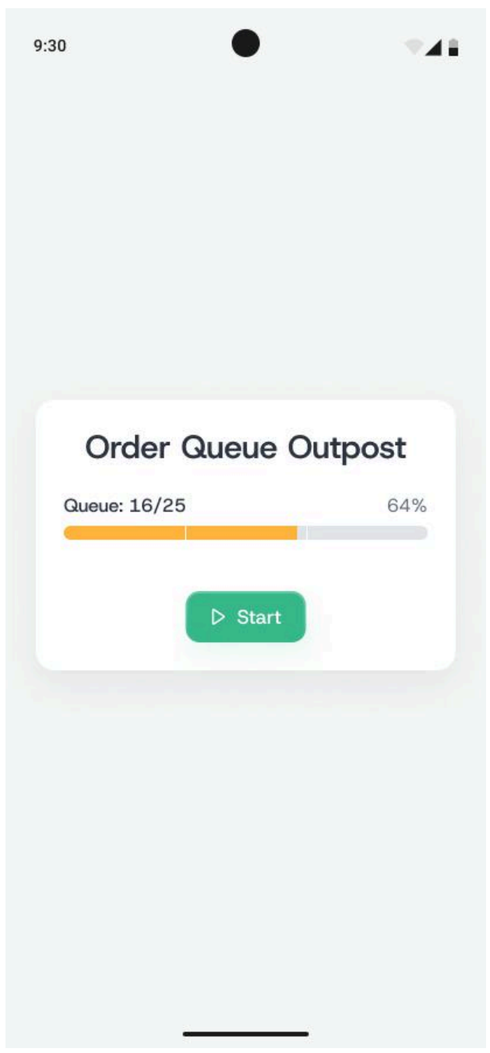
The main visual effect of this challenge is **queue growth** during pause and **queue shrinkage** after the consumer resumes.

🔢 Initial Data for Testing

```
val orders = (1..50).toList()
val orderChannel = Channel<Int>(capacity = 25)
```

- The producer emits one orderId every **250ms** into **orderChannel**
- The consumer reads from the channel with a random delay: delay(Random.nextLong(100, 250))
- In **Pause** mode, the consumer stops completely
- In **Start** mode, the consumer resumes
- If the queue is full, the producer continues to emit orders (overflow state)
- The UI must reflect all queue changes numerically and visually





🤔 What's Allowed?

- Standard Android/Jetpack libraries
- No 3rd party libraries are allowed or would be required to complete this challenge

⚠️ What's not important

- Full screen size or orientation adaptability
- Displaying actual order IDs is optional
- Any stylistic or visual effects outside what's clearly shown in the Figma mockup

🏆 Submission & Rewards

- Successfully submitting this challenge via the /submit-challenge command on Discord grants you **100 XP**.
- Your submission must include:
  a. A link to a Gist with your implementation
  b. A screen recording (max 20 seconds) showing:
    - Tapping **Start** to begin queue processing
    - Tapping **Pause** and waiting for the queue to fill
    - Visual overflow once the queue exceeds capacity
    - Tapping **Start** again to resume processing
    - The progress bar returning below 100%