In-Memory Queue

Problem Statement:

Design an efficient in-memory queueing system with low latency requirements and also write producer and consumer using multi threading with following requirements:

Requirements:

1. There should be a queue that can receive the message from the producer, and send the message to the consumer.
2. The queue should be bounded in size and completely held in-memory. Size should be configurable.
3. The queue should only hold JSON messages.
4. The queue will have at least one producer and multiple consumers.
5. Consumers register callbacks that will be invoked whenever there is a new message
6. Allow subscription of consumers to messages that match a particular expression
7. Consumers might have dependency relationships between them.
   For ex :
   if there are three consumers A, B and C. One dependency relationship can be that C cannot consume a particular message before A and B have consumed it.
   C -> (A,B) (-> means must process after).
8. Handle concurrent writes and reads consistently between producer and consumers.
9. Provide retry mechanisms to handle failures in message processing. It could be a failure in publishing or consumption.
10. Handle the message TTL, means the message could expire after some time T. If a message is expired, it should not be delivered to the consumer.
11. Implementation of sideline (Dead-Letter) queue: move to sideline after retries exhausted.

Sample Executions :

Example 1:

New message {"messageId": "abc"} added to the queue, Queue size: 1
New message {"messageId": "def"} added to the queue, Queue size: 2
Consumer A consumed messageId abc
New message {"messageId": "xyz"} added to the queue, Queue size: 2
Consumer B consumed messageId def

Example 2: Consider queue size is 3

New message {"messageId": "abc"} added to the queue, Queue size: 1
New message {"messageId": "def"} added to the queue, Queue size: 2
Consumer A consumed messageId abc
New message {"messageId": "xyz"} added to the queue, Queue size: 2

Consumer B consumed messageId def
New message {"messageId": "fgh"} added to the queue, Queue size: 3
Queue is full, cannot add more messages to the queue

Example 3: Consider 2 consumers: Consumer A consumes messages with httpCode 200 and consumer B consumes messages with all other httpCodes
New message {"messageId": "abc", "httpCode": "200"} added to the queue, Queue size: 1
New message {"messageId": "def", "httpCode": "200"} added to the queue, Queue size: 2
New message {"messageId": "xyz", "httpCode": "400"} added to the queue, Queue size: 3
Consumer A consumed messageId abc
New message {"messageId": "hgj", "httpCode": "200"} added to the queue, Queue size: 3
Consumer B consumed messageId xyz
Consumer A consumed messageId def
Consumer B consumed messageId hgj

Example 4:
New message {"messageId": "abc", "httpCode": "200"} added to the queue, Queue size: 1
New message {"messageId": "def", "httpCode": "200"} added to the queue, Queue size: 2
New message {"messageId": "xyz", "httpCode": "200"} added to the queue, Queue size: 3
Consumer A consumed messageId abc
Consumer A consumed messageId def
messageId xyz is expired

Example 5: Consider after 3 retries the message is moved to sideline
New message {"messageId": "xyz", "httpCode": "200"} added to the queue, Queue size: 1
New message {"messageId": "abc", "httpCode": "200"} added to the queue, Queue size: 2
Consumer A consumed messageId abc
Message processing failed - messageId: abc, consumer: A, Number of remaining retries: 2
Message processing failed - messageId: abc, consumer: A, Number of remaining retries: 1
Message processing failed - messageId: abc, consumer: A, Number of remaining retries: 0
Retries exhausted, moving messageId abc to sideline

Guidelines :

- Try completing the tasks one by one, run it, test it, then move on to the next. Pick the task in any order that you want.
- Think about the extension of the problem before choosing your LLD. You might be asked to add some new features in this problem during evaluation time.
- You are not allowed to use the in-built queue data structure provided by any language. Expected to implement the queue.
- You can use library for JSON

How will the candidate be evaluated:

- Code should be working
- Separation of concerns
- Abstractions
- Application of OO design principles
- Testability
- Code readability
- Language proficiency