CS6650 Assignment 3
Siyue Li

1. GitHub Repo

https://github.com/ZetaL0519/CS6650Assignment3

2. Description of your server design, include major classes, packages, relationships, how messages get sent/received, etc.

The server architecture I chose for this assignment is Tomcat/Java Servlets and Amazon RDS MySQL database. Aside from the servlet itself, since we are utilizing RabbitMQ for asynchronous messaging for review api, I added another RabbitMQ server to process and consume the message queue.

Major classes and database

1. AlbumServlet

AlbumServlet is mainly designed for handling incoming POST requests related to album creation. This servlet manages the reception and processing of requests, and the requests are processed synchronously. AlbumServlet will process and validate the image file and album profile information, generate a unique albumID for each album and put each post call into the database.

2. ReviewServlet

ReviewServlet is to handle incoming POST requests related to album reviews. Calls to this servlet are processed asynchronously with the use of RabbitMQ. Before processing requests, this servlet will establish a connection to a RabbitMQ message broker. Then it will process the incoming requests by extracting albumID and reviewType from the url and then constructs a JSON message containing this data and publish it to RabbitMQ channel. When the message is consumed, this will also be updated to the SQL database.

3. DatabasePool

I choose to use AWS RDS MySQL database. The process of creating database and table is like the following:

```
CREATE DATABASE album_store;
USE album_store;

CREATE TABLE albumInfo (
    AlbumID VARCHAR(255) PRIMARY KEY,
    ImageData LONGBLOB,
    AlbumProfile VARCHAR(255),
    NumberOfLikes INT DEFAULT NULL,
    NumberOfDislikes INT DEFAULT NULL
);
```

In summary, the schema of my database table only contains one table. Album id is auto incremented by one each time we post item.

For image, I used the image given by the professor, which is 3KB. I stored the image in the form of LONGBLOB. Blob storage is a type of cloud storage for unstructured data. A "blob," which is short for Binary Large Object, is a mass of data in binary form that does not necessarily conform to any file format.

For this DatabasePool class, a Hikari connection pool is initialized in the constructor, providing an efficient way to handle connections. The class also configures url, database username and password and connection pool parameters. It encapsulates the functionality for establishing and closing connections to MySQL database.

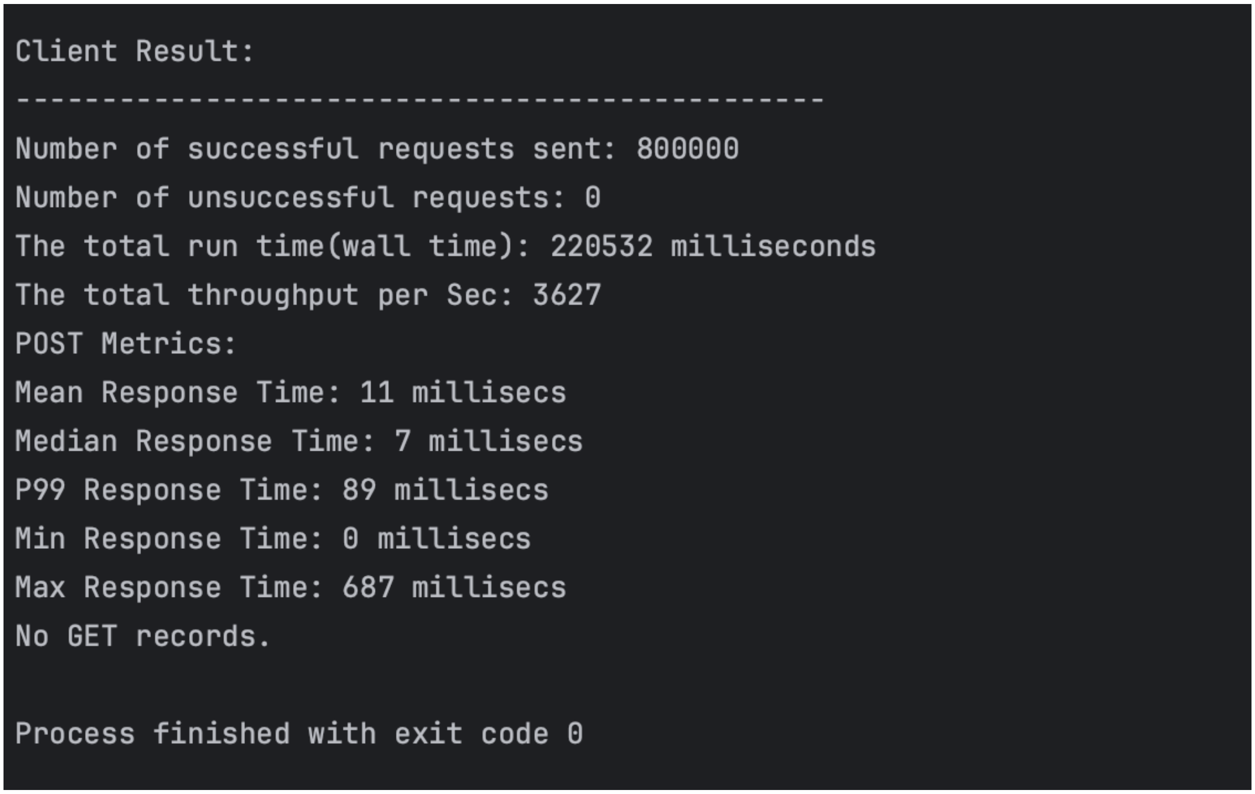
4. RabbitMQService

The RabbitMQService class serves to manage a channel pool with a size of 120 channels to establish a connection to RabbitmQ using the provided host information (exchange name and exchange type). It then adds each channel to the ConcurrentLinkedDeque, forming a pool. Also, this class included a close method that gracefully closes each channel within the provided pool, handling potential exceptions.

In the RabbitMQService package, the RabbitRunnable will match each message with each queue, consume and process the review message, then incrementing the number of reviews in database.

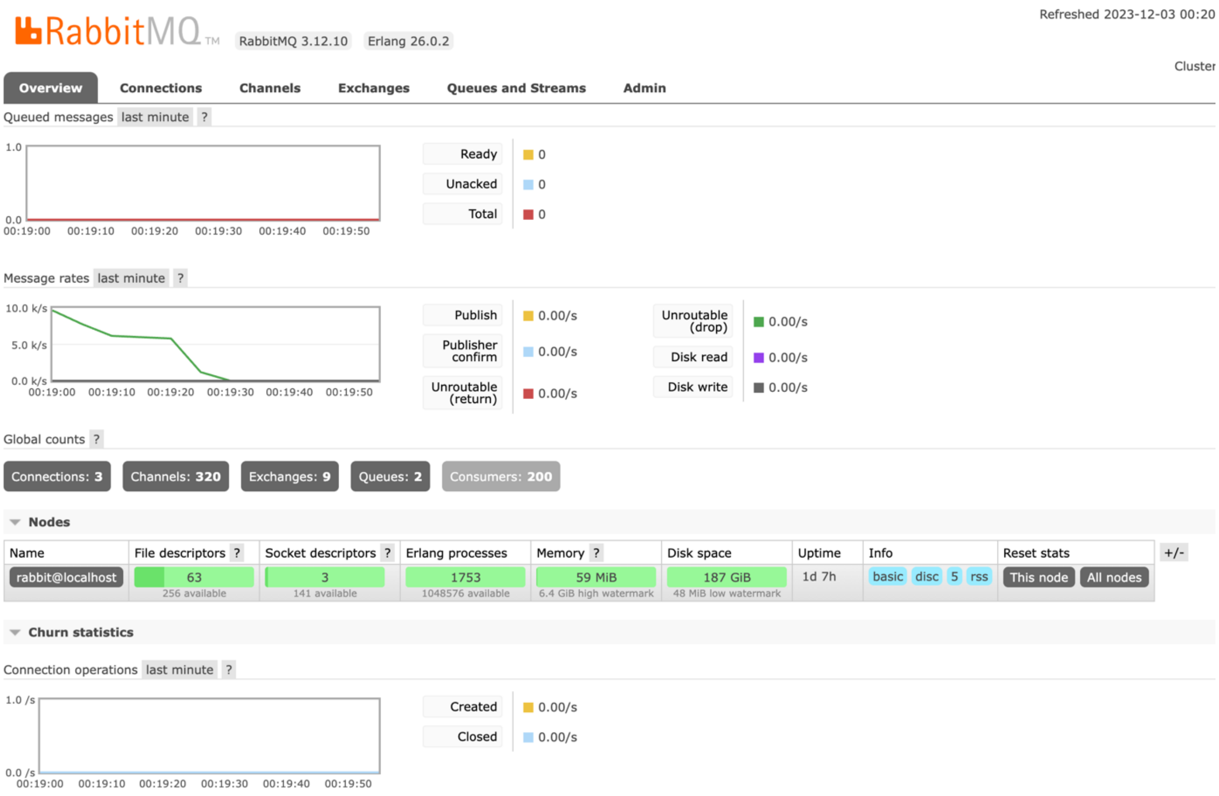
3. Output windows for the 3 client configuration tests run against a single server/DB

Input: 10 10 2



```
Client Result:
----------------------------------------------
Number of successful requests sent: 800000
Number of unsuccessful requests: 0
The total run time(wall time): 220532 milliseconds
The total throughput per Sec: 3627
POST Metrics:
Mean Response Time: 11 millisecs
Median Response Time: 7 millisecs
P99 Response Time: 89 millisecs
Min Response Time: 0 millisecs
Max Response Time: 687 millisecs
No GET records.

Process finished with exit code 0
```



Ready	🟨 0
Unacked	🟦 0
Total	🟥 0

Publish	🟨 0.00/s	Unroutable (drop)	🟩 0.00/s
Publisher confirm	🟦 0.00/s	Disk read	🟪 0.00/s
Unroutable (return)	🟥 0.00/s	Disk write	⬛ 0.00/s

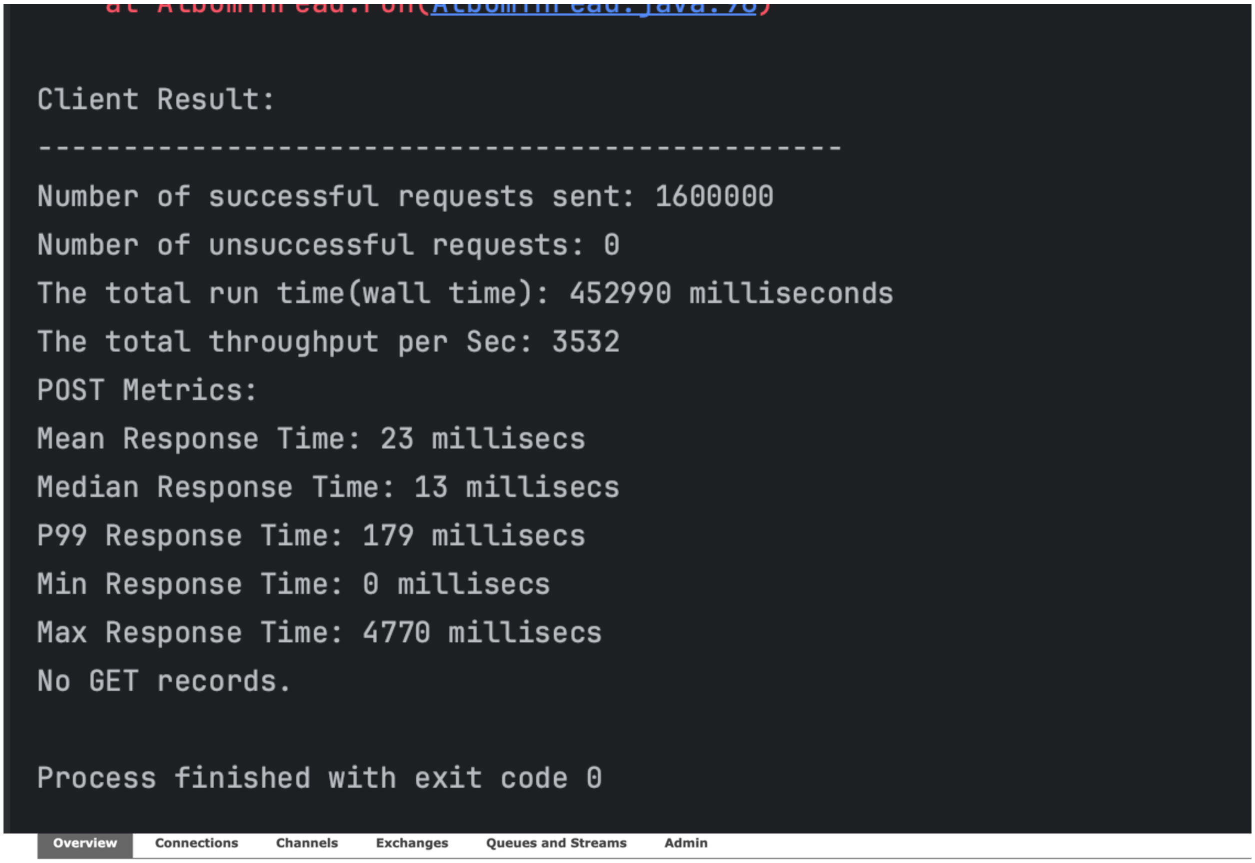
Connections: 3 Channels: 320 Exchanges: 9 Queues: 2 Consumers: 200

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@localhost	63 256 available	3 141 available	1753 1048576 available	59 MiB 6.4 GiB high watermark	187 GiB 48 MiB low watermark	1d 7h	basic disc 5 rss	This node All nodes

Created	🟨 0.00/s
Closed	🟦 0.00/s

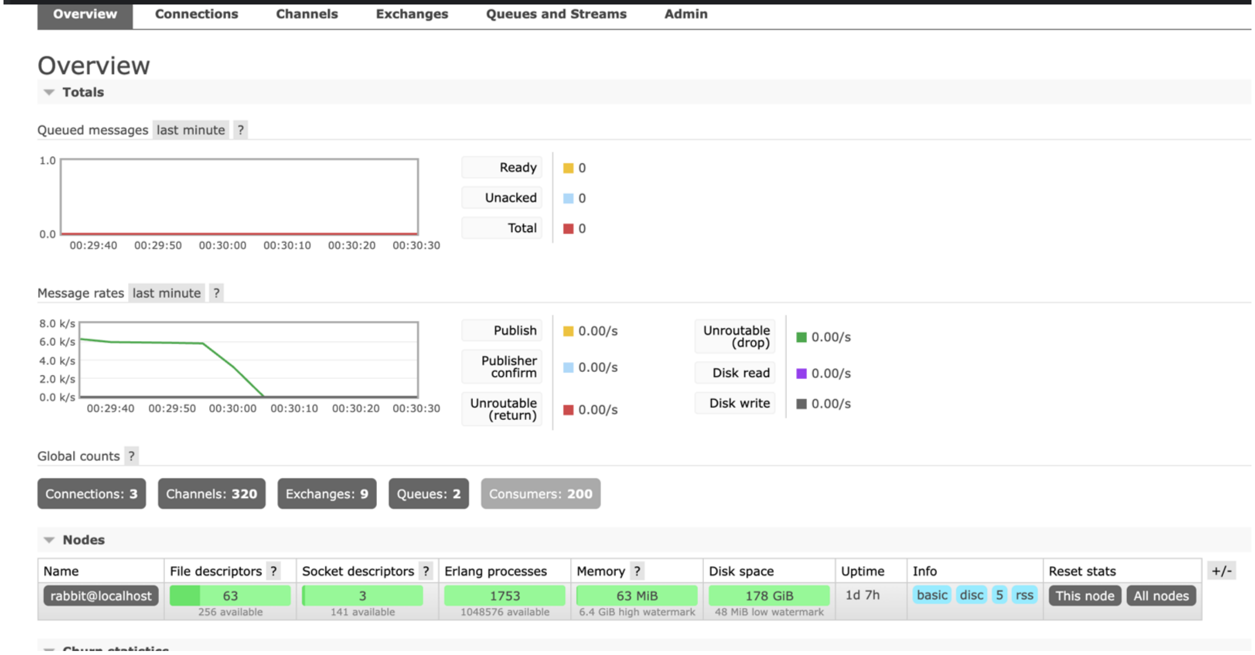
Input: 10 20 2



```
    at AlbumThread.run(AlbumThread.java:78)

Client Result:
-----------------------------------------------
Number of successful requests sent: 1600000
Number of unsuccessful requests: 0
The total run time(wall time): 452990 milliseconds
The total throughput per Sec: 3532
POST Metrics:
Mean Response Time: 23 millisecs
Median Response Time: 13 millisecs
P99 Response Time: 179 millisecs
Min Response Time: 0 millisecs
Max Response Time: 4770 millisecs
No GET records.

Process finished with exit code 0
```



Overview

Totals

Queued messages last minute ?

Ready	🟨 0
Unacked	🟦 0
Total	🟥 0

Message rates last minute ?

Publish	🟨 0.00/s	Unroutable (drop)	🟩 0.00/s
Publisher confirm	🟦 0.00/s	Disk read	🟪 0.00/s
Unroutable (return)	🟥 0.00/s	Disk write	■ 0.00/s

Global counts ?

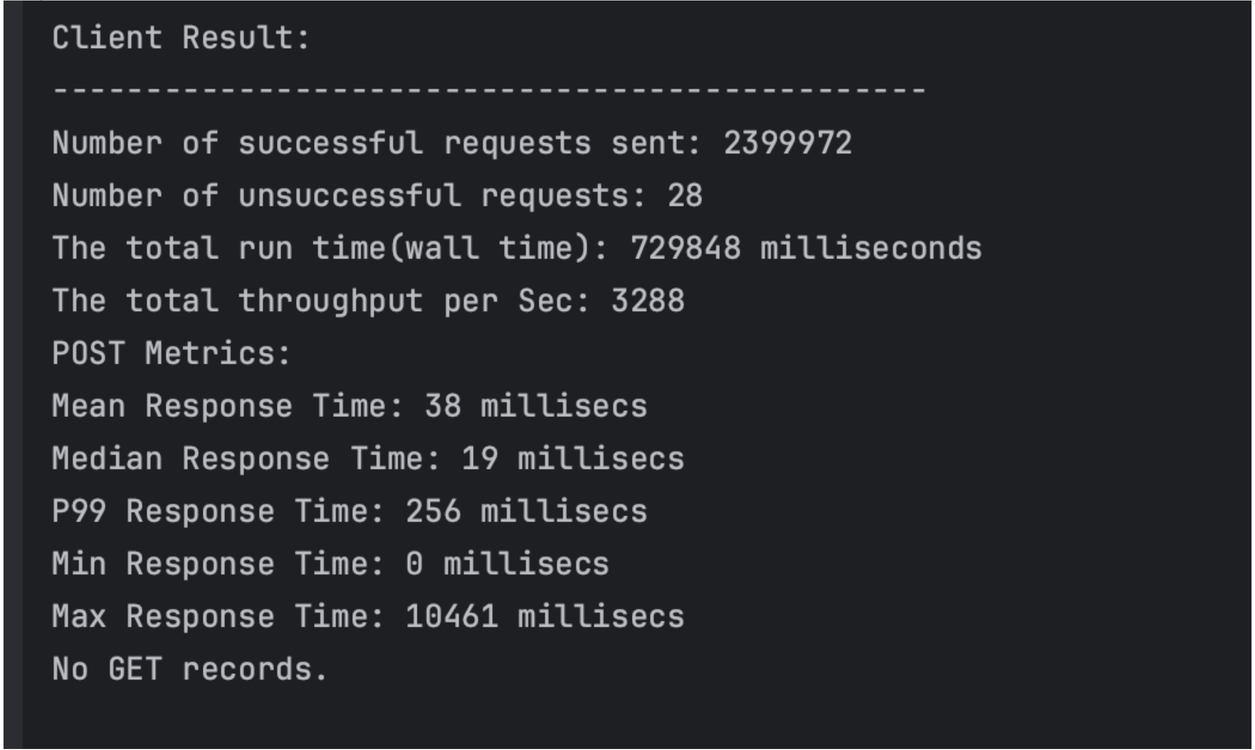
Connections: **3** Channels: **320** Exchanges: **9** Queues: **2** Consumers: **200**

Nodes

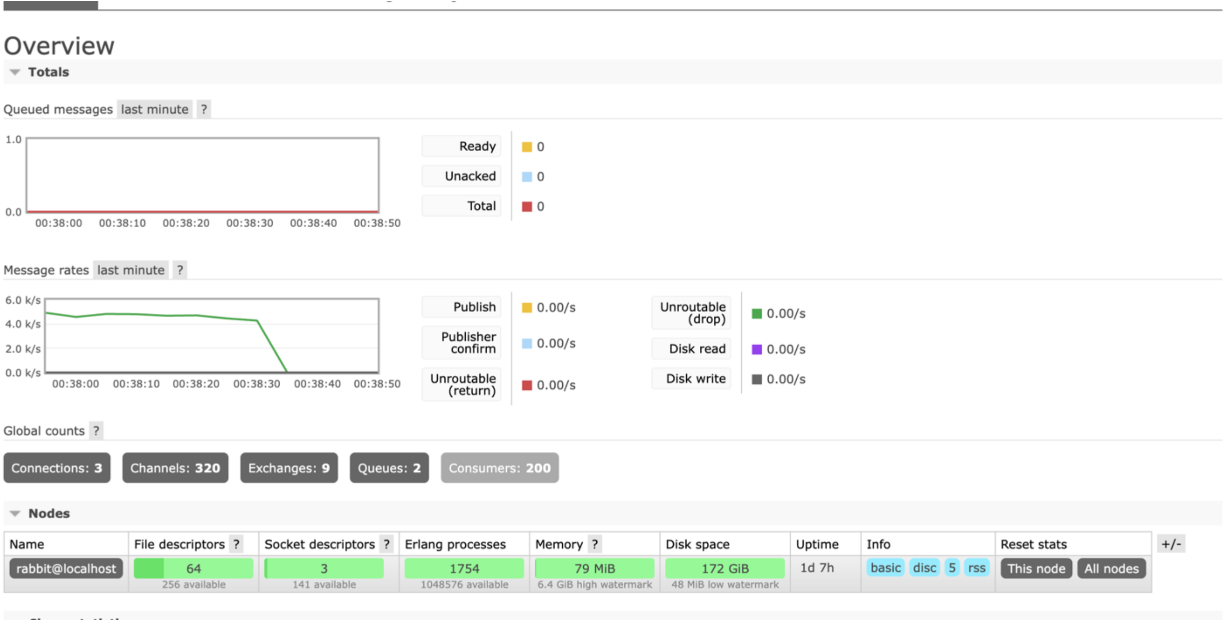
Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@localhost	63 256 available	3 141 available	1753 1048576 available	63 MiB 6.4 GiB high watermark	178 GiB 48 MiB low watermark	1d 7h	basic disc 5 rss	This node All nodes	

Churn statistics

```
Input: 10 30 2
```



```
Client Result:
-----------------------------------------------
Number of successful requests sent: 2399972
Number of unsuccessful requests: 28
The total run time(wall time): 729848 milliseconds
The total throughput per Sec: 3288
POST Metrics:
Mean Response Time: 38 millisecs
Median Response Time: 19 millisecs
P99 Response Time: 256 millisecs
Min Response Time: 0 millisecs
Max Response Time: 10461 millisecs
No GET records.
```



Overview

▾ Totals

Queued messages last minute ?

Ready	🟨 0
Unacked	🟦 0
Total	🟥 0

Message rates last minute ?

Publish	🟨 0.00/s	Unroutable (drop)	🟩 0.00/s
Publisher confirm	🟦 0.00/s	Disk read	🟪 0.00/s
Unroutable (return)	🟥 0.00/s	Disk write	⬛ 0.00/s

Global counts ?

Connections: 3 Channels: 320 Exchanges: 9 Queues: 2 Consumers: 200

▾ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@localhost	64 256 available	3 141 available	1754 1048576 available	79 MiB 6.4 GiB high watermark	172 GiB 48 MiB low watermark	1d 7h	basic disc 5 rss	This node All nodes