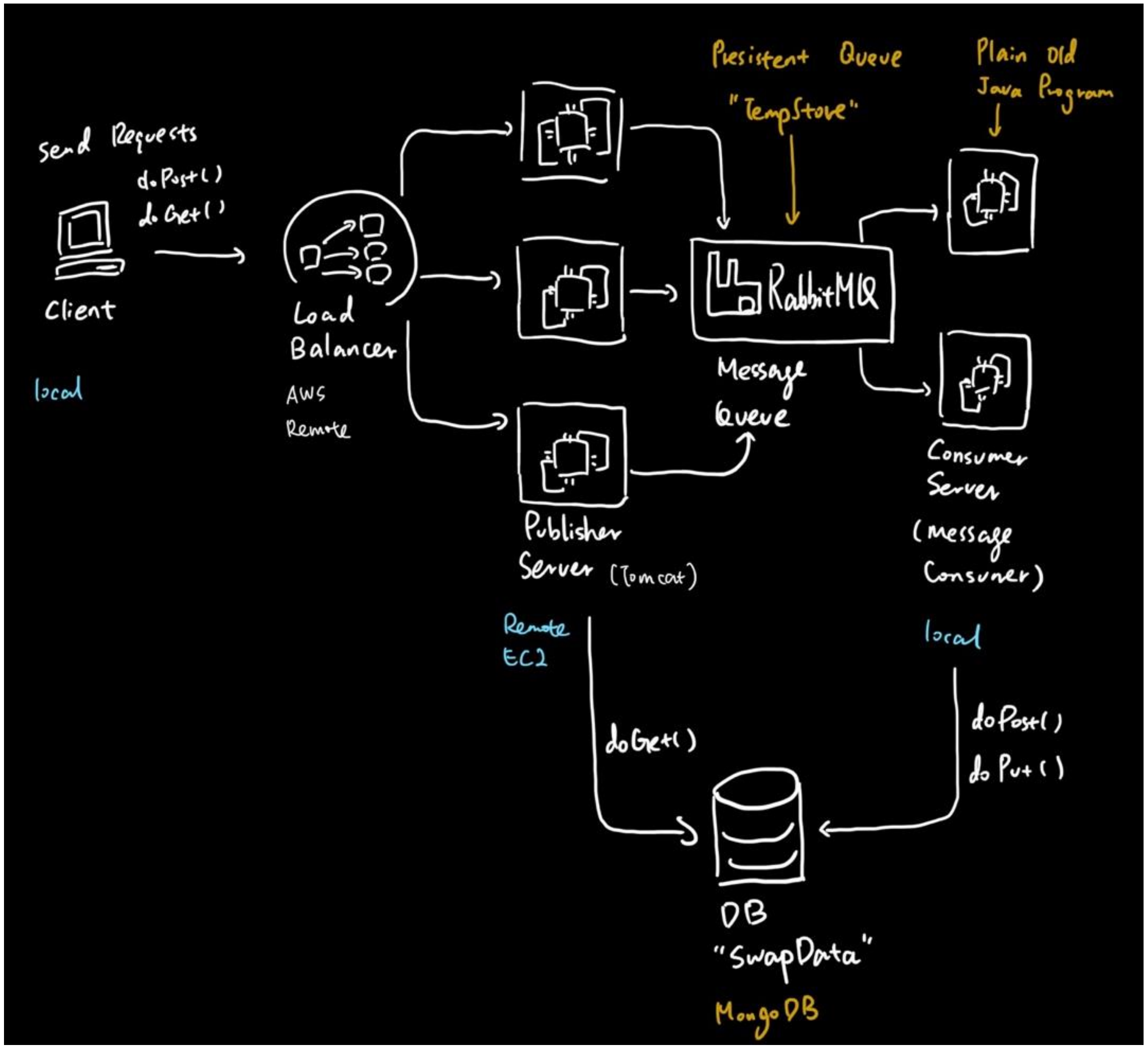
Assignment 3 Report: Twinder Adding Persistence and Reads

CS6650 Distributed System
Zidi Xia

The URL of git repo

https://github.com/sharonzidi/cs6650_twinder/tree/main/Assignment3

Twinder Architecture Design



Client (local): The client generates Swipe events and send to server through load balancer.

Load balancer: The load balancer used to distribute requests among multiple web servers and ensure that all users receive timely and efficient responses.

Publisher Server (Tomcat): The publisher server receiving request from LB and send messages to message queue. It implements the GET requests and retrieves results from the database directly.

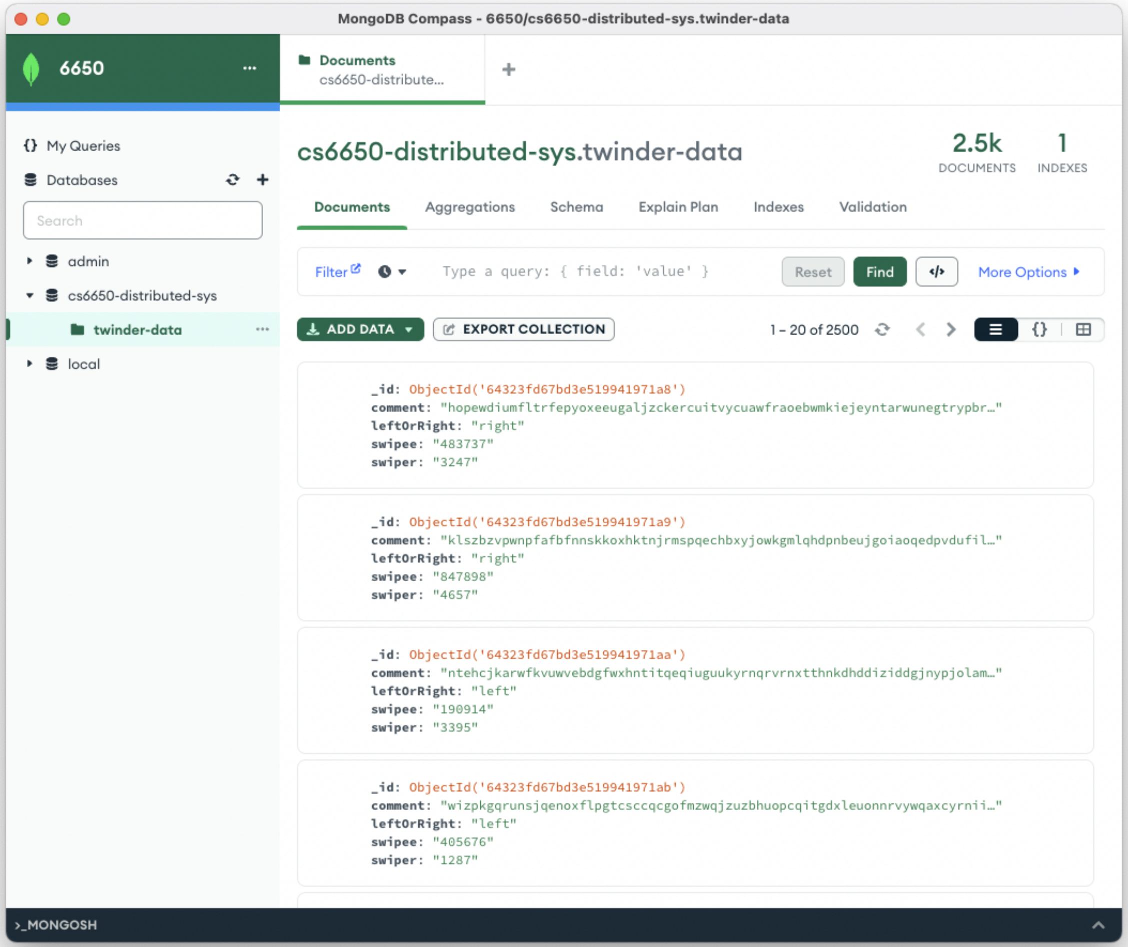
Message Queue (RabbitMQ): Using RMQ to manage message queues since it provides a flexible and scalable way to implement messaging between distributed systems.

Consumer Server (Plain Old Java Program): The consumer reads new Swipe events from RMQ and updates the database that stores information about users and swipe events.

Database (MongoDB Atlas): The project uses MongoDB Atlas to stores information about users and swipe events. MongoDB's flexibility, scalability, and performance make it a powerful choice for this project, and works great with large and complex data requirements.

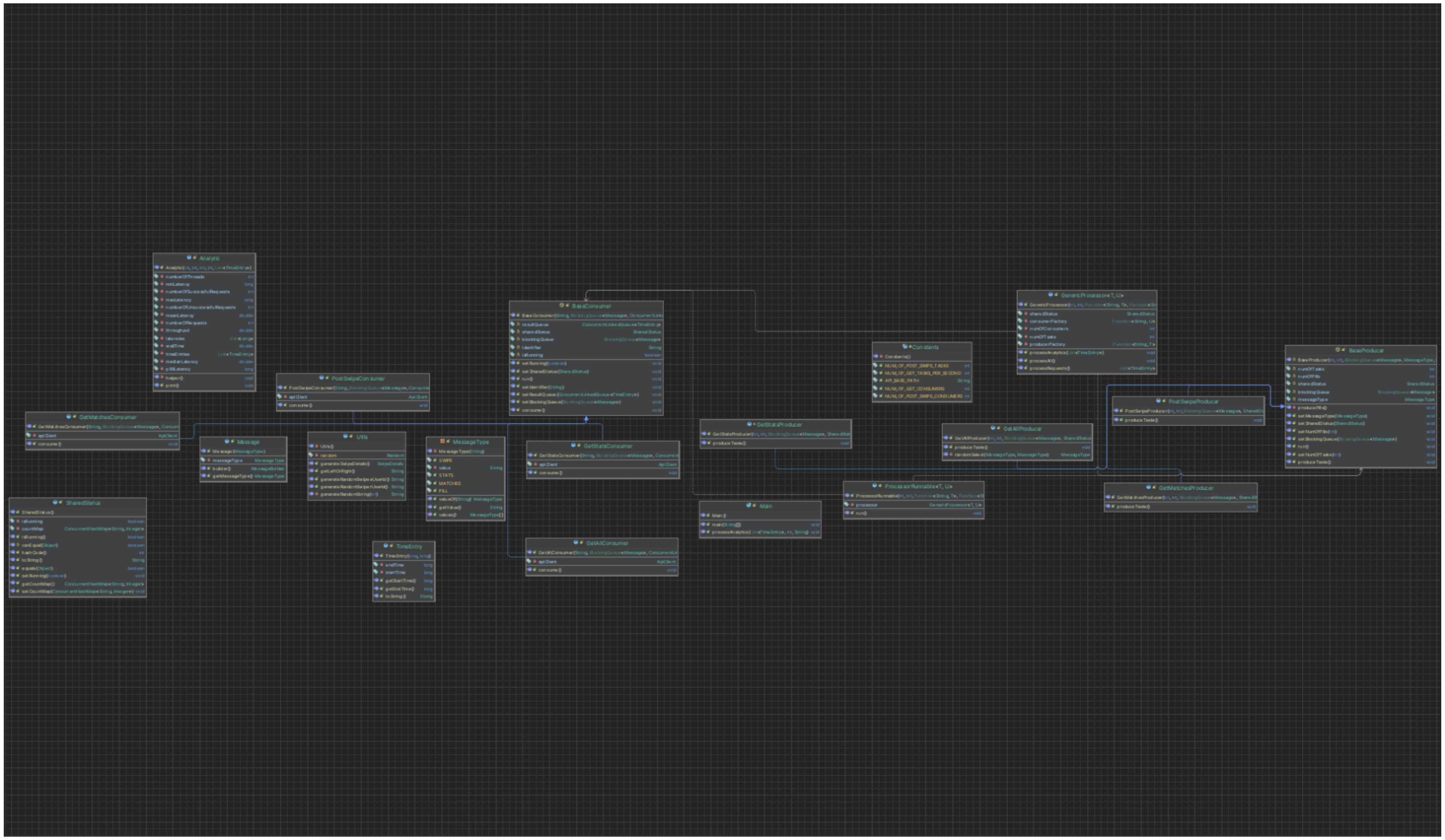
Database Design

This project uses MongoDB to store data. MongoDB contains cs6650-disxtributed-sys database. In the cs6650-disxtributed-sys database, there is a collection called twinder-data which contains a group of documents. A document is a set of key-value pairs that represent a single instance of swipe data. MongoDB documents are stored in BSON (Binary JSON) format, which is a binary representation of JSON documents.



```
_id: ObjectId('64323fd67bd3e519941971a8')
comment: "hopewdiumfltrfepyoxeeugaljzckercuitvycuawfraoebwmkiejeyntarwunegtrypbr…"
leftOrRight: "right"
swipee: "483737"
swiper: "3247"

_id: ObjectId('64323fd67bd3e519941971a9')
comment: "klszbzvpwnpfafbfnnskkoxhktnjrmspqechbxyjowkgmlqhdpnbeujgoiaoqedpvdufil…"
leftOrRight: "right"
swipee: "847898"
swiper: "4657"

_id: ObjectId('64323fd67bd3e519941971aa')
comment: "ntehcjkarwfkvuwvebdgfwxhntitqeqiuguukyrnqrvrnxtthnkdhddiziddgjnypjolam…"
leftOrRight: "left"
swipee: "190914"
swiper: "3395"

_id: ObjectId('64323fd67bd3e519941971ab')
comment: "wizpkgqrunsjqenoxflpgtcsccqcgofmzwqjzuzbhuopcqitgdxleuonnrvywqaxcyrnii…"
leftOrRight: "left"
swipee: "405676"
swiper: "1287"
```

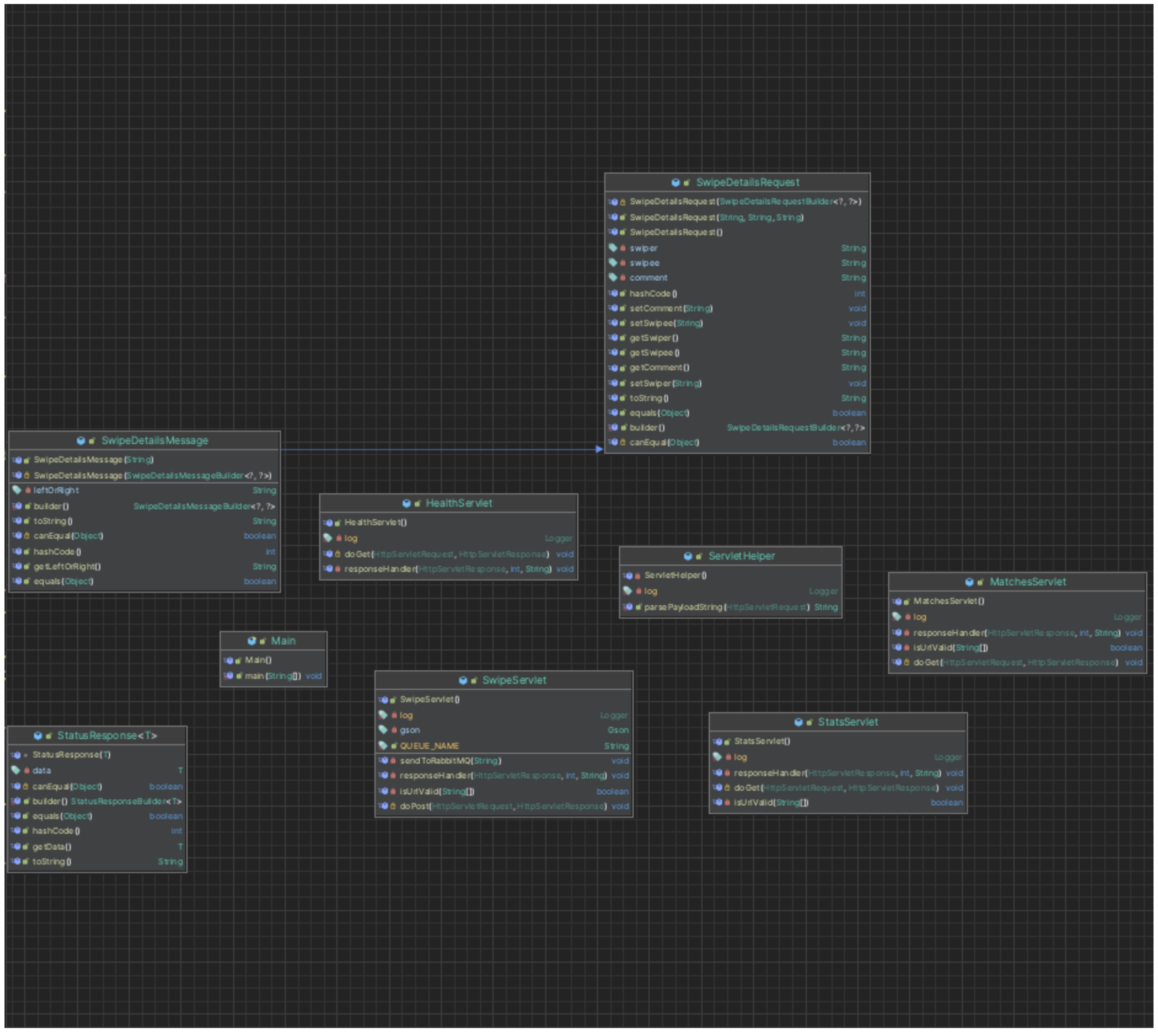
Twinder Client UML



Twinder Publisher Server UML

Send Swipe data as payload to remote queue and return success to the client.

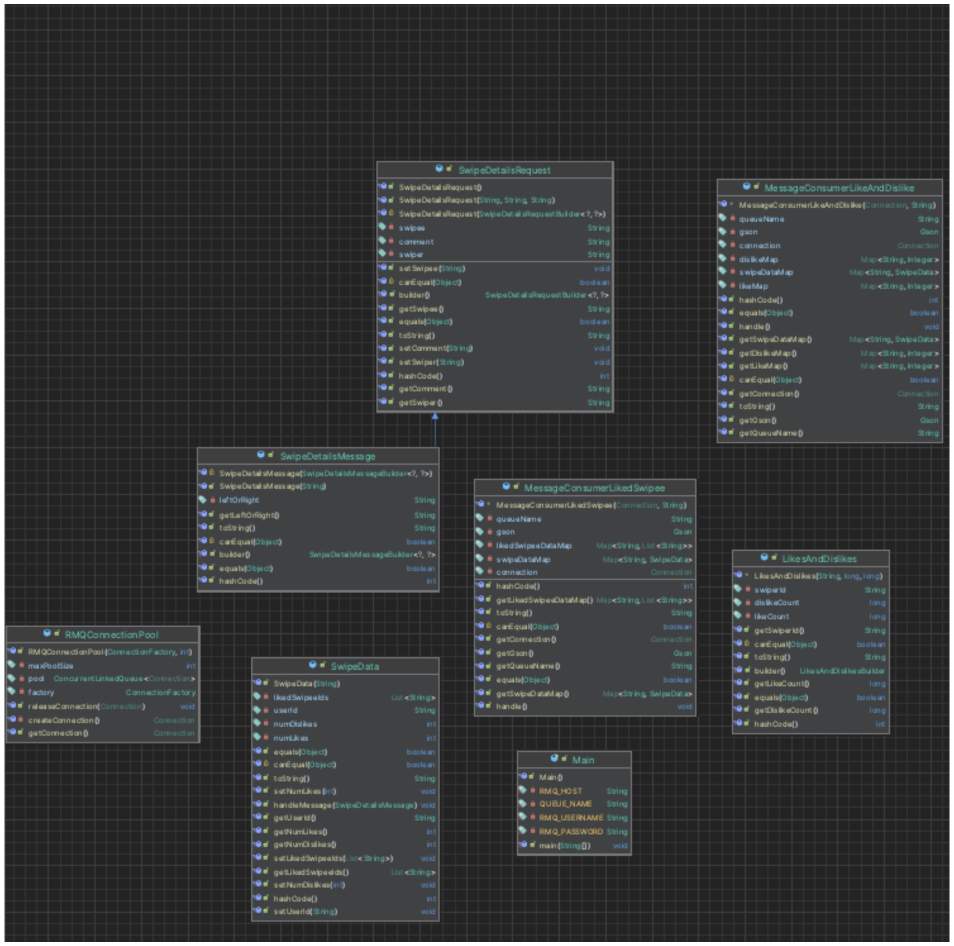
Implements the GET requests and retrieves results from the *SwipeData* database directly.


SwipeDetailsRequest
SwipeDetailsRequest(SwipeDetailsRequestBuilder<?, ?>)
SwipeDetailsRequest(String, String, String)
SwipeDetailsRequest()
swiper String
swipee String
comment String
hashCode() int
setComment(String) void
setSwipee(String) void
getSwiper() String
getSwipee() String
getComment() String
setSwiper(String) void
toString() String
equals(Object) boolean
builder() SwipeDetailsRequestBuilder<?, ?>
canEqual(Object) boolean
SwipeDetailsMessage
SwipeDetailsMessage(String)
SwipeDetailsMessage(SwipeDetailsMessageBuilder<?, ?>)
leftOrRight String
builder() SwipeDetailsMessageBuilder<?, ?>
toString() String
canEqual(Object) boolean
hashCode() int
getLeftOrRight() String
equals(Object) boolean
HealthServlet
HealthServlet()
log Logger
doGet(HttpServletRequest, HttpServletResponse) void
responseHandler(HttpServletResponse, int, String) void
ServletHelper
ServletHelper()
log Logger
parsePayloadString(HttpServletRequest) String
MatchesServlet
MatchesServlet()
log Logger
responseHandler(HttpServletResponse, int, String) void
isUrlValid(String[]) boolean
doGet(HttpServletRequest, HttpServletResponse) void
Main
Main()
main(String[]) void
SwipeServlet
SwipeServlet()
log Logger
gson Gson
QUEUE_NAME String
sendToRabbitMQ(String) void
responseHandler(HttpServletResponse, int, String) void
isUrlValid(String[]) boolean
doPost(HttpServletRequest, HttpServletResponse) void
StatsServlet
StatsServlet()
log Logger
responseHandler(HttpServletResponse, int, String) void
doGet(HttpServletRequest, HttpServletResponse) void
isUrlValid(String[]) boolean
StatusResponse<T>
StatusResponse(T)
data T
canEqual(Object) boolean
builder() StatusResponseBuilder<T>
equals(Object) boolean
hashCode() int
getData() T
toString() String

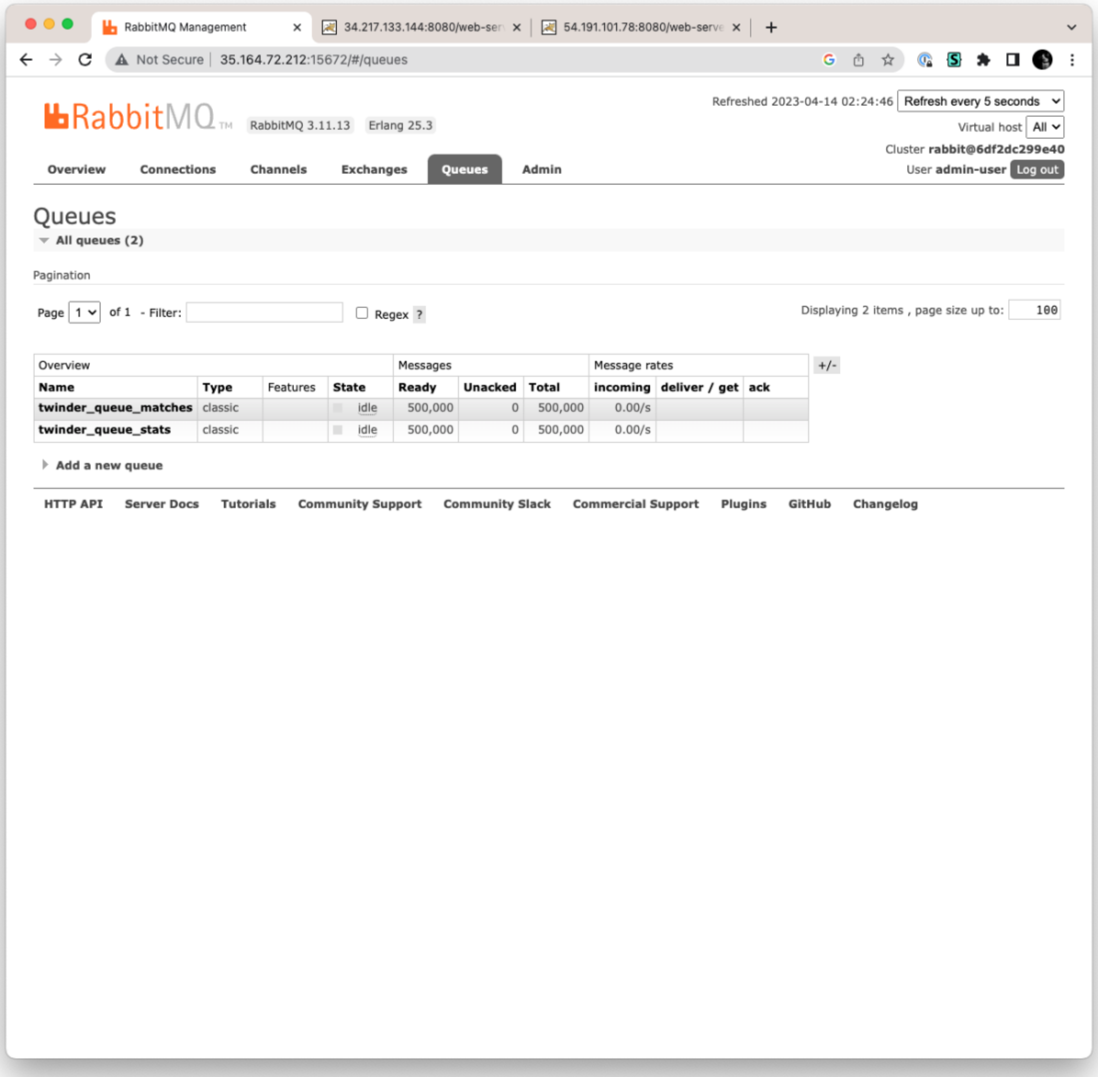
Twinder Consumer Client UML

Pull messages off the queue and consume each message.

Reads new Swipe events from *TempStore* and updates a database that stores information about users and swipe events.



A3 RMQ Screenshots with Queue Size



Refreshed 2023-04-14 02:24:46

Virtual host All

Cluster rabbit@6df2dc299e40

User admin-user

Queues

All queues (2)

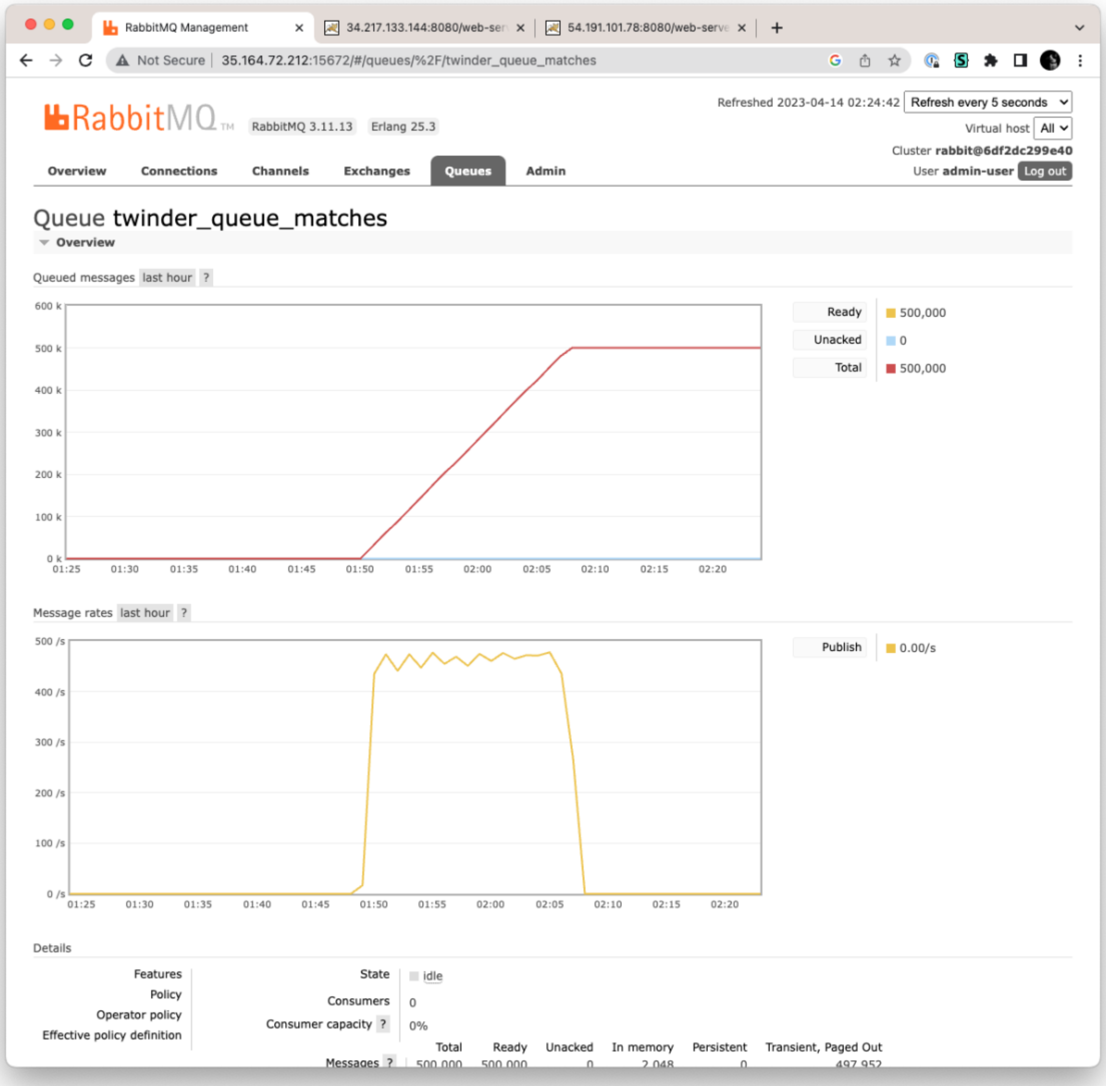
Pagination

Page 1 of 1 - Filter:

Displaying 2 items , page size up to: 100

Overview				Messages			Message rates		
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
twinder_queue_matches	classic		idle	500,000	0	500,000	0.00/s		
twinder_queue_stats	classic		idle	500,000	0	500,000	0.00/s		

A3 RMQ Screenshots with Flat Line Profile
Twinder_queue_matches queue



RabbitMQ 3.11.13 Erlang 25.3

Refreshed 2023-04-14 02:24:42 Refresh every 5 seconds

Virtual host All

Cluster **rabbit@6df2dc299e40**

User **admin-user**

Queue **twinder_queue_matches**

▾ Overview

Queued messages last hour ?

Ready	🟨 500,000
Unacked	🟦 0
Total	🟥 500,000

Message rates last hour ?

Publish	🟨 0.00/s

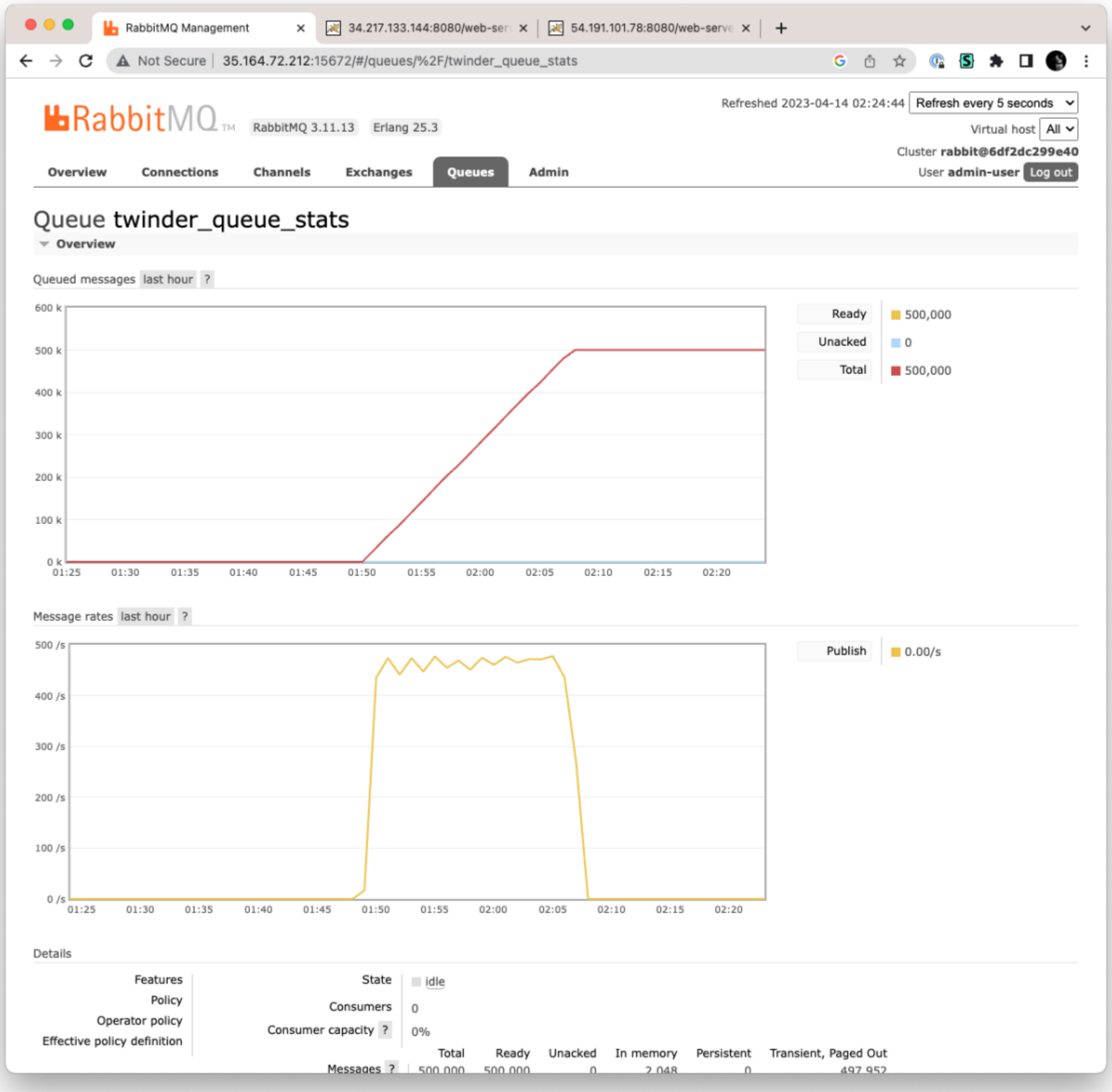
Details

Features	
Policy	
Operator policy	
Effective policy definition	

State	idle
Consumers	0
Consumer capacity ?	0%

	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Messages ?	500,000	500,000	0	2,048	0	497,952

Twinder_queue_stats queue



Refreshed 2023-04-14 02:24:44 Refresh every 5 seconds

Virtual host All

Cluster **rabbit@6df2dc299e40**

User **admin-user**

Queue **twinder_queue_stats**

Queued messages last hour

Ready	🟨 500,000
Unacked	🟦 0
Total	🟥 500,000

Message rates last hour

Publish	🟨 0.00/s

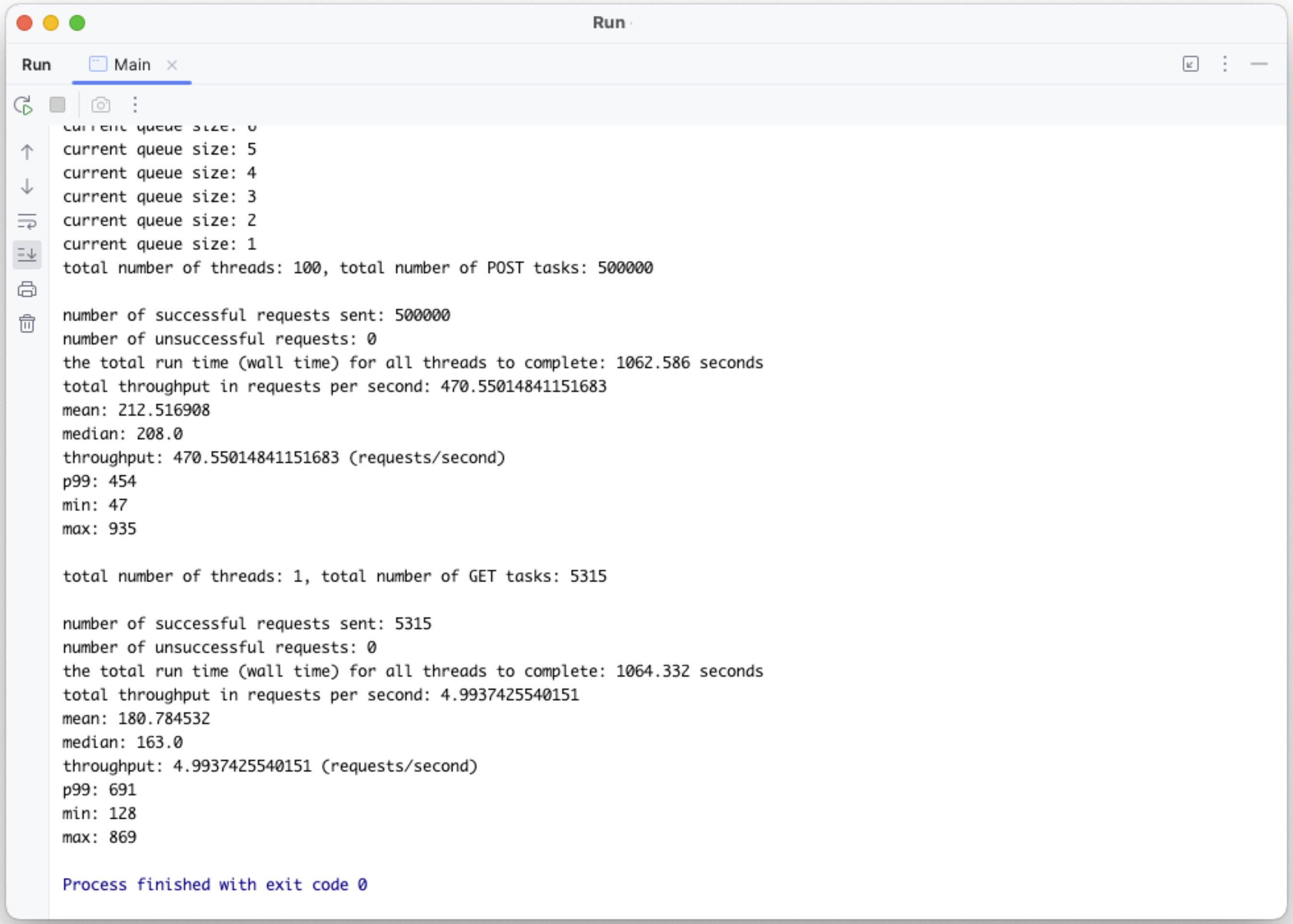
Details

Features	
Policy	
Operator policy	
Effective policy definition	

State	idle
Consumers	0
Consumer capacity ?	0%

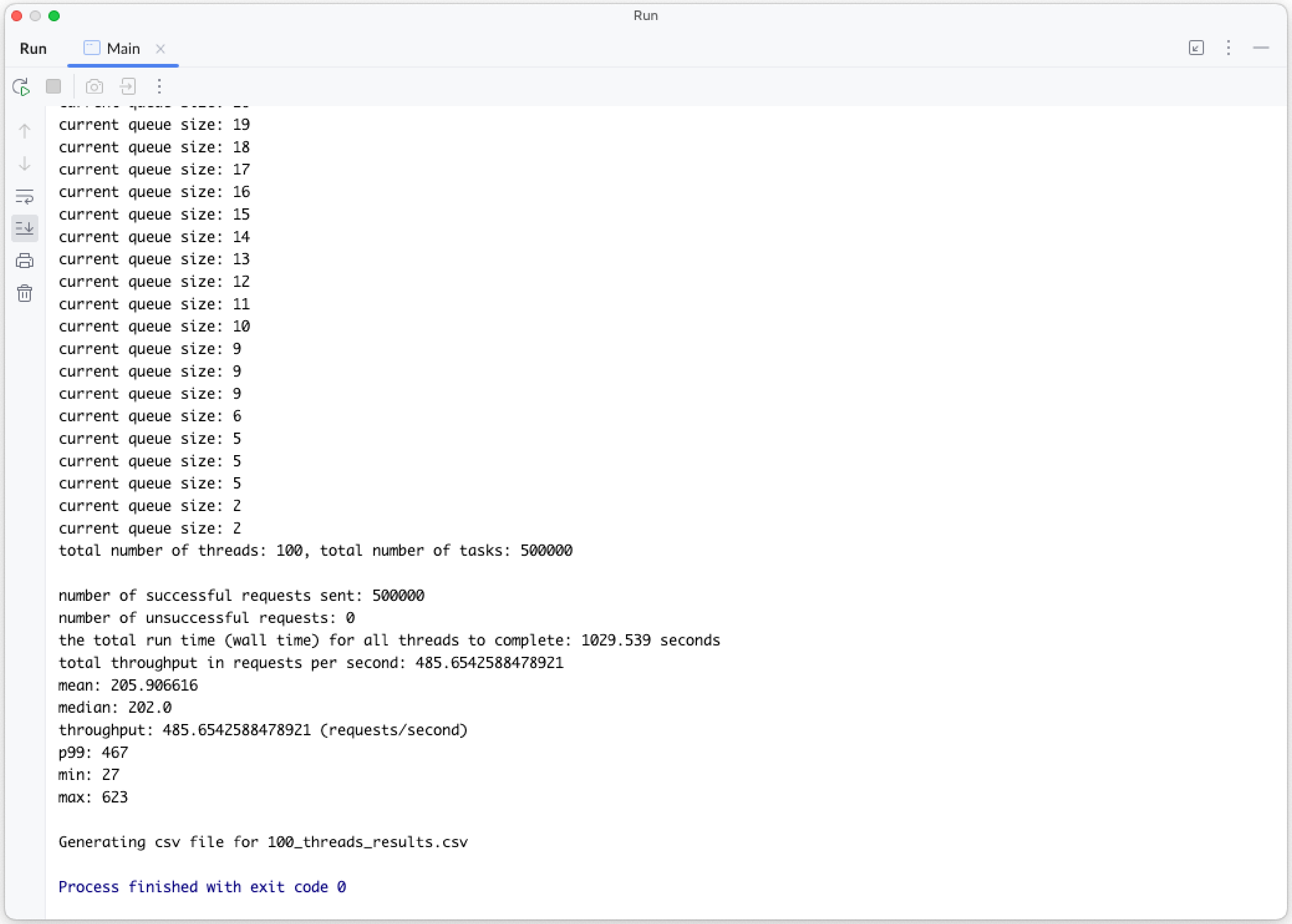
	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Messages ?	500,000	500,000	0	2,048	0	497,952

A3 Client test run screenshots showing results for GET and POST requests and best performance:



```
current queue size: 6
current queue size: 5
current queue size: 4
current queue size: 3
current queue size: 2
current queue size: 1
total number of threads: 100, total number of POST tasks: 500000

number of successful requests sent: 500000
number of unsuccessful requests: 0
the total run time (wall time) for all threads to complete: 1062.586 seconds
total throughput in requests per second: 470.55014841151683
mean: 212.516908
median: 208.0
throughput: 470.55014841151683 (requests/second)
p99: 454
min: 47
max: 935

total number of threads: 1, total number of GET tasks: 5315

number of successful requests sent: 5315
number of unsuccessful requests: 0
the total run time (wall time) for all threads to complete: 1064.332 seconds
total throughput in requests per second: 4.9937425540151
mean: 180.784532
median: 163.0
throughput: 4.9937425540151 (requests/second)
p99: 691
min: 128
max: 869

Process finished with exit code 0
```

A2 performance screenshots:



```
current queue size: 19
current queue size: 18
current queue size: 17
current queue size: 16
current queue size: 15
current queue size: 14
current queue size: 13
current queue size: 12
current queue size: 11
current queue size: 10
current queue size: 9
current queue size: 9
current queue size: 9
current queue size: 6
current queue size: 5
current queue size: 5
current queue size: 5
current queue size: 2
current queue size: 2
total number of threads: 100, total number of tasks: 500000

number of successful requests sent: 500000
number of unsuccessful requests: 0
the total run time (wall time) for all threads to complete: 1029.539 seconds
total throughput in requests per second: 485.6542588478921
mean: 205.906616
median: 202.0
throughput: 485.6542588478921 (requests/second)
p99: 467
min: 27
max: 623

Generating csv file for 100_threads_results.csv

Process finished with exit code 0
```

A2 & A3 performance Comparisons

A2

Total number of threads: 100
Total number of tasks: 500k
Wall time for all threads to complete: **1029.539 seconds**
Total throughput in requests per second: **485.654**
Mean: 205.906
Median: 202
P99: 467
Min: 27
Max: 623

A3

Total number of threads: 100
Total number of tasks: 500k
Wall time for all threads to complete: **1062.586 seconds**
Total throughput in requests per second: **470.550**
Mean: 212.516
Median: 208.0
P99: 454
Min: 47
Max: 935

As we can see from the above screenshots and data, the throughput and latencies at the client in Assignment 3 is within 10% of Assignment 2. The GET request mean latency(180) no longer than the mean POST request latency(212).