



Programming Assignment

Priority Queue

Corporate Head Office

Expertflow LLC, Jägerweg 18, 3014 Bern,
Switzerland

www.ExpertFlow.com

Contents

[Assignment Details](#)

[Goal](#)

[Technology](#)

[Development & Submission guidelines](#)

[Problem](#)

[Arrival Process](#)

[Service Mechanism](#)

[Queue Characteristics](#)

[Data Structure](#)

[Customer Request](#)

[Priority Queue](#)

[Console API](#)

[List Customers in Queue](#)

[List Customers details in Queue](#)

[Service Customer](#)

[Enqueue Customer Request](#)

[Reneg Customer Request](#)

[System Information](#)

[System Memory Dump \(Optional\)](#)

[REST API](#)

[List Customers in Queue](#)

[List Customers Details in Queue](#)

[Service Customer](#)

[Enqueue Customer Request](#)

[Reneg Customer Request](#)

[System Information](#)

[System Memory Dump \(Optional\)](#)

Assignment Details

Goal

You are required to develop a priority queue which must:

1. Prioritize Customer Requests based on their integer priority weight. It can range from 1 - 10 while 1 being the lowest and 10 being the highest priority.
2. The queue should serve the highest priority customer requests first.
3. The queue must accommodate the same priority customer requests in a first come first serve fashion.

Technology

Choose a language from any of the following:

- a. Java/CSharp
- b. C++,
- c. Go

You can use any language functions or available classes (such as Java Queue interface). You don't need to develop basic constructs yourself.

Development & Submission guidelines

Before submission, please consider the following

1. That the code is clean and readable
2. Use meaningful variable names
3. Use camel case for naming conventions
4. Add public method docs/comments in code
5. Implement unit tests for PriorityQueue class/structure.
6. Implement file logging in the form of stories, logs should be generated at /var/log/priorityqueue/ directory in Linux based systems or at C://PriorityQueue/ in Windows-based systems.
7. Do not submit executable files
8. Create a document explaining the code build cycle, it should also state build tools which you choose and build files must be present in the code directory e.g. makefile / Maven files / Gradle files, etc. If you are using .NET for dev and managing dependencies using NUGET, do not forget to add a guide for these dependencies resolution.
9. Create a zip file of your code directory which should contain all the necessary files to build and run code. Do not include dependencies binaries in zip file, instead add steps to resolve dependencies in the said document.
10. Create a class diagram of your implementation. The class diagram file should be zipped in the code zip file and it must be mentioned in the doc. The class diagram can be created using draw.io or use any tool of your choice.
11. Upload the resulting zip file to your Google Drive
12. Submit this assignment in response to the email that originally assigned the assignment. Attach documentation and link to the zip file. Do not forget to provide zip file access to the invigilator.

Problem

You are required to develop a priority queue which must adhere to the following queuing characteristics i.e.

Arrival Process

Arrival process can be:

1. A customer can arrive singly
2. Bulk/Batch of customers can arrive
3. The queue must limit the number of customer requests i.e. 50000. It must gracefully deny all new service requests.

Service Mechanism

Service mechanism must adhere to:

1. The system should provide a functionality to dequeue a customer request from the queue for service. This function needs to be exposed using REST API and via Console.
2. During service (dequeue process), the system must display the time a customer request is waiting in the queue.

Queue Characteristics

The queue should have:

1. Reneging: Customers can leave the queue if they have waited for too long. The system must expose a mechanism for reneging.
2. The queue should prioritize customers based on priority weight. See the description for details.
3. Customer requests with the same priority weightage must be served in an FCFS/FIFO fashion.
4. Lower priority tasks must starve in favor of higher priority tasks. No aging mechanism is required.

Data Structure

Customer Request

Customer Request data structure should contain at least the following attributes. You can add additional attributes based on requirements.

```
class CustomerRequest {  
    private int id;  
    private String customerName;  
    private String description;  
    private int priorityWeight;  
    private Date enqueueTime;  
}
```

Priority Queue

Priority Queue class should contain the following attributes:

```
class PriorityQueue {  
    private String queueName;  
    private String queueDescription;  
    /*  
     * Add data structure for queue implementation which can enqueue CustomerRequest objects  
     * and prioritize them based on CustomerRequest.priorityWeight  
     */  
}
```

Console API

The system should expose the following functionality over the console. The following screenshot is just an example of how it will look. You can modify/beautify the menu.

```
*****  
* Welcome to Priority Queue, please select from following *  
*****  
1. List Customers in Queue  
2. List Customer Details in Queue  
3. Service Customer  
4. Enqueue Customer Request  
5. Renege Customer Request  
6. System Information  
7. System Memory Dump  
  
Enter Selection:
```

List Customers in Queue

This should list all customer requests in the queue. The list should only contain customer request IDs.

```
*****
* Welcome to Priority Queue, please select from following *
*****
1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection: 1
{
  queueName: 'DefaultQueue',
  queueDescription: 'This queue is for demonstration of Priority Queue implementation',
  size: 10,
  oldestTaskId: 8,
  customerRequests: [
    { id: 1 }, { id: 10 },
    { id: 7 }, { id: 3 },
    { id: 6 }, { id: 5 },
    { id: 2 }, { id: 4 },
    { id: 9 }, { id: 8 }
  ]
}

1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```

List Customers details in Queue

This list should show all customer requests in the queue.

```
Enter Selection: 2
Printing List of Customer Details in Queue
{
  queueName: 'DefaultQueue',
  queueDescription: 'This queue is for demonstration of Priority Queue implementation',
  size: 10,
  oldestTaskId: 8,
  customerRequests: [
    {
      id: 1,
      customerName: 'one',
      description: '',
      priorityWeight: 10,
      enqueueTime: '2020-06-27T06:45:13.223Z'
    },
    {
      id: 10,
      customerName: 'ten',
      description: '',
      priorityWeight: 10,
      enqueueTime: '2020-06-27T06:54:13.363Z'
    },
    {
      id: 7,
      customerName: 'seven',
      description: '',
      priorityWeight: 9,
      enqueueTime: 2020-06-27T09:24:51.290Z
    },
    {
      id: 3,
      customerName: 'three',
      description: '',
      priorityWeight: 8,
      enqueueTime: 2020-06-27T09:24:51.290Z
    },
    {
      id: 6,
      customerName: 'six',

```

Service Customer

Service Customer is used to dequeuing a customer request. The system should print dequeued customer requests along-with wait time for that customer request in seconds.

```
Enter Selection: 3
Dequeuing Customer Request
{
  id: 1,
  customerName: 'one',
  description: '',
  priorityWeight: 10,
  enqueueTime: '2020-06-27T06:45:13.223Z',
  waitTimeInSec: 9672.441
}

1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Reneg Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```

Enqueue Customer Request

This API is used to enqueue customer request. It should ask for customer name, description, and priority weight and assign additional attributes i.e. id and enqueue time. The system should print enqueued customer request along-with its position in the queue.

```
Enter Selection: 4
Please enter following information:
Customer Name: eleven
Description: eleven
Priority Weight: 1

Customer Request is enqueued with following information:
{
  customerName: 'eleven',
  description: 'eleven',
  priorityWeight: 1,
  id: 11,
  enqueueTime: '2020-06-27T07:32:06.276Z',
  positionInQueue: 11
}
1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```

Renege Customer Request

If a customer thinks it is taking too long to service a request, he may renege his customer request from the queue. It should be removed from the queue and the customer request should be printed on console along-with wait time in seconds and a success/failure message.

```
Enter Selection: 5
Please enter customer id: 11
Reneged following customer request successfully:
{
  customerName: 'eleven',
  id: 11,
  enqueueTime: '2020-06-27T06:45:13.223Z',
  waitTimeInSec: 7684.487,
  message: 'Request reneged successfully'
}

1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```


System Information

This API should return system information at any point of time. It can return IN_SERVICE or MAX_CAPACITY_REACHED along with queue details.

```
Enter Selection: 6
{
  status: 'IN_SERVICE',
  queue: {
    name: 'default',
    size: '30000',
    oldestCustomerRequestTimeInSec: 301234
  }
}

1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```

System Memory Dump (Optional)

This is an optional API. The following screenshot is just an example of returned data. It should return runtime statistics of the application/system. Statistics may include but not limited to:

1. Memory Usage
2. Heap Dump
3. CPU usage
4. Number of threads
5. Network activity

Its results can vary depending on the library used for taking system statistics.

```
Enter Selection: 7
{
  tstamp: 1593244962340,
  uptime: 182480,
  load: [ 0.25634765625, 0.12255859375, 0.0830078125 ],
  mem: { used: 842268, free: 5318140, swapused: 776, swapfree: 4062452 }
}

1. List Customers in Queue
2. List Customer Details in Queue
3. Service Customer
4. Enqueue Customer Request
5. Renege Customer Request
6. System Information
7. System Memory Dump

Enter Selection:
```

REST API

The system should expose the following REST APIs

List Customers in Queue

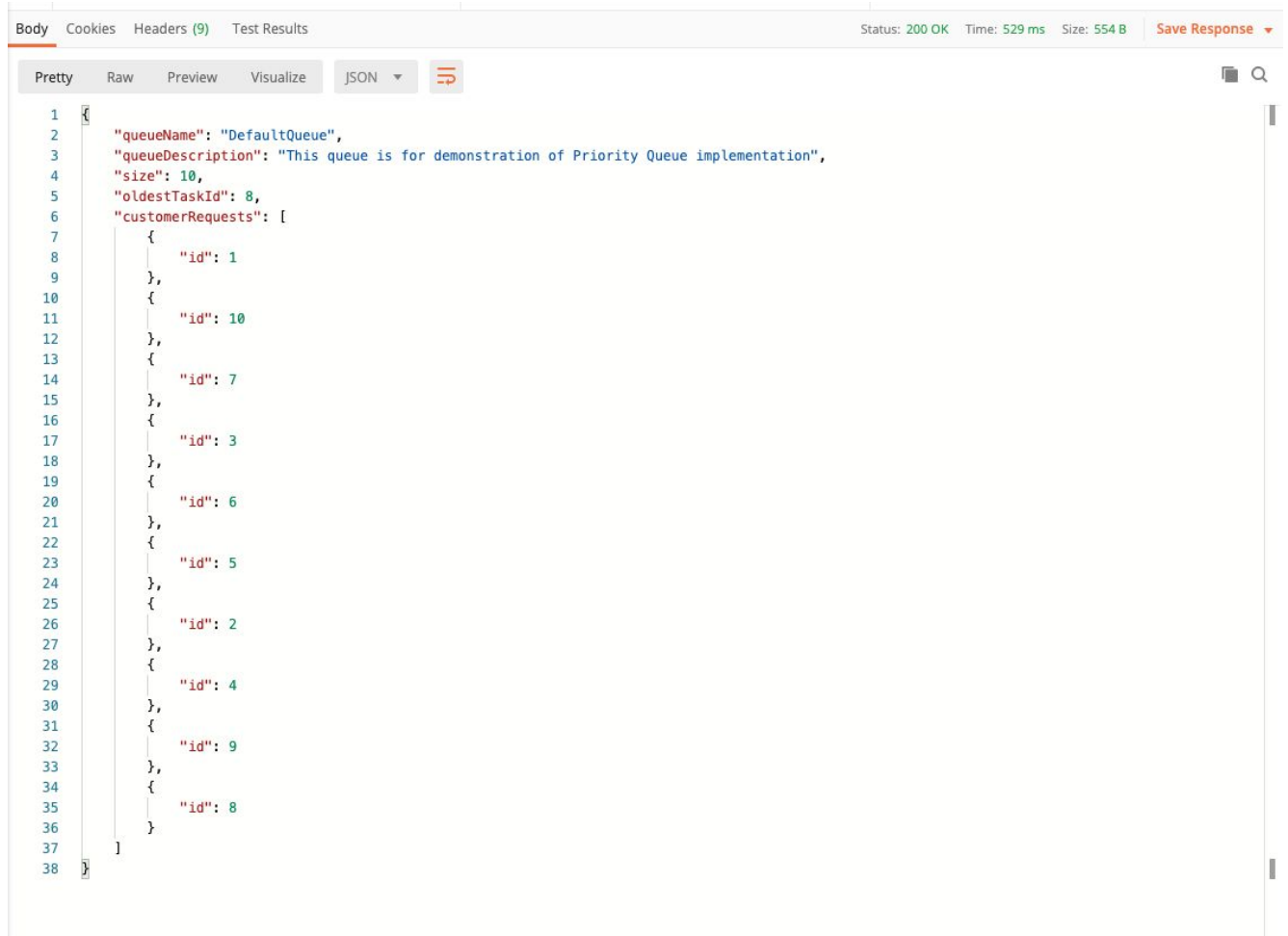
API Route: /api/v1.0/queue/list

Request Type: GET

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: This API request should return enqueued customer requests ids. The response body should comply with the following. However, you can add any additional parameters which you deem fit.



The screenshot shows a REST client interface with the following details:

- Body** tab is selected.
- Status:** 200 OK
- Time:** 529 ms
- Size:** 554 B
- Save Response** button is visible.
- JSON** format is selected.
- The response body is a JSON object with the following structure:


```

{
  "queueName": "DefaultQueue",
  "queueDescription": "This queue is for demonstration of Priority Queue implementation",
  "size": 10,
  "oldestTaskId": 8,
  "customerRequests": [
    { "id": 1 },
    { "id": 10 },
    { "id": 7 },
    { "id": 3 },
    { "id": 6 },
    { "id": 5 },
    { "id": 2 },
    { "id": 4 },
    { "id": 9 },
    { "id": 8 }
  ]
}
```

List Customers Details in Queue

API Route: `/api/v1.0/queue/detail`

Request Type: GET

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: This API request should return enqueued customer requests with all of their details. The response body should comply with the following. However, you can add any additional parameters which you deem fit.

Body Cookies Headers (10) Test Results Status: 200 OK Time: 746 ms Size: 1.54 KB Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "queueName": "DefaultQueue",
3   "queueDescription": "This queue is for demonstration of Priority Queue implementation",
4   "size": 10,
5   "oldestTaskId": 8,
6   "customerRequests": [
7     {
8       "id": 1,
9       "customerName": "one",
10      "description": "",
11      "priorityWeight": 10,
12      "enqueueTime": "2020-06-27T06:45:13.223Z"
13    },
14    {
15      "id": 10,
16      "customerName": "ten",
17      "description": "",
18      "priorityWeight": 10,
19      "enqueueTime": "2020-06-27T06:54:13.363Z"
20    },
21    {
22      "id": 7,
23      "customerName": "seven",
24      "description": "",
25      "priorityWeight": 9,
26      "enqueueTime": "2020-06-27T07:15:46.058Z"
27    },
28    {
29      "id": 3,
30      "customerName": "three",
31      "description": "",
32      "priorityWeight": 8,
33      "enqueueTime": "2020-06-27T07:15:46.058Z"
34    },
35    {
36      "id": 6,
37      "customerName": "six",
38      "description": "",
39      "priorityWeight": 7,
40      "enqueueTime": "2020-06-27T07:15:46.058Z"
41    }
42  ]
43 }
  
```

Service Customer

API Route: /api/v1.0/queue/service

Request Type: GET

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: Service customer should dequeue one task upon every request and return customer details i.e.

Body Cookies Headers (9) Test Results Status: 200 OK Time: 281 ms Size: 431 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 1,
3   "customerName": "one",
4   "description": "",
5   "priorityWeight": 10,
6   "enqueueTime": "2020-06-27T06:45:13.223Z",
7   "waitTimeInSec": 2138.7
8 }
  
```

Enqueue Customer Request

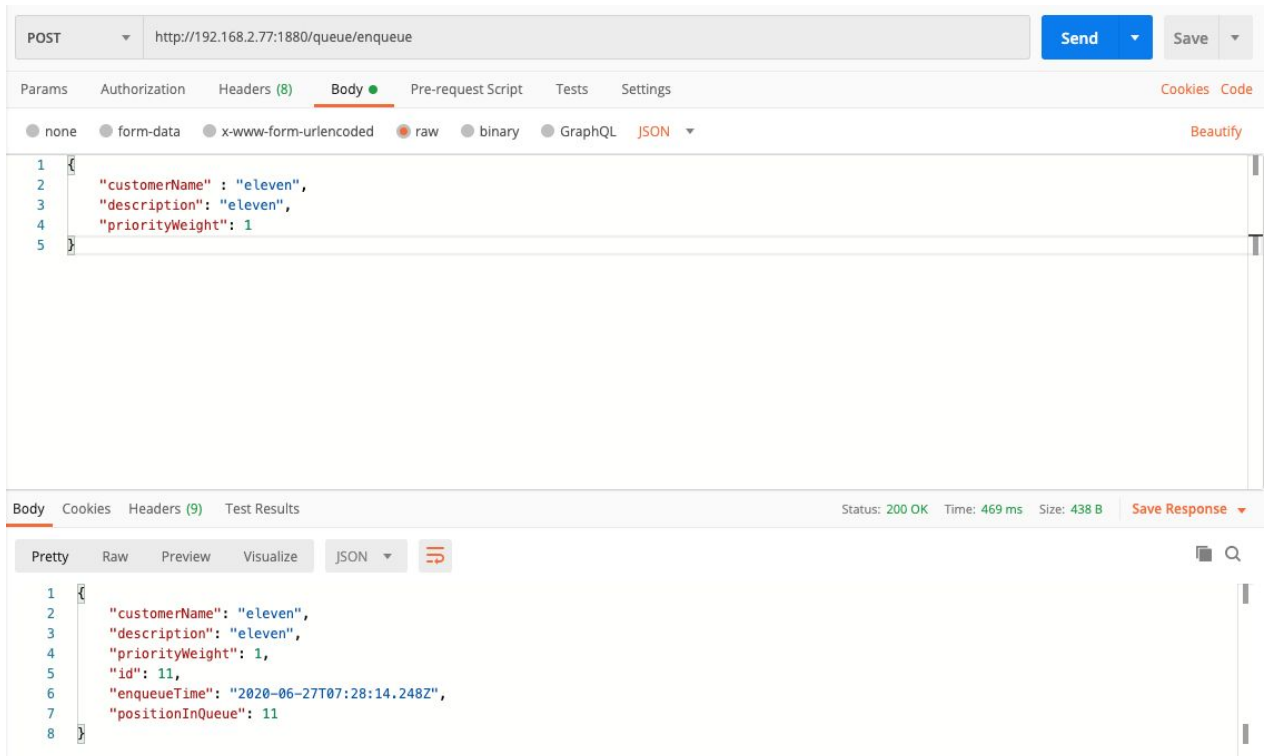
API Route: /api/v1.0/queue/enqueue

Request Type: POST

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: A new customer can enqueue customer requests using this API. It should contain the following request and response body.



POST ▼ http://192.168.2.77:1880/queue/enqueue Send Save ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies Code


● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL ● JSON ▼ Beautify

```

1 {
2   "customerName": "eleven",
3   "description": "eleven",
4   "priorityWeight": 1
5 }

```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 469 ms Size: 438 B Save Response ▼

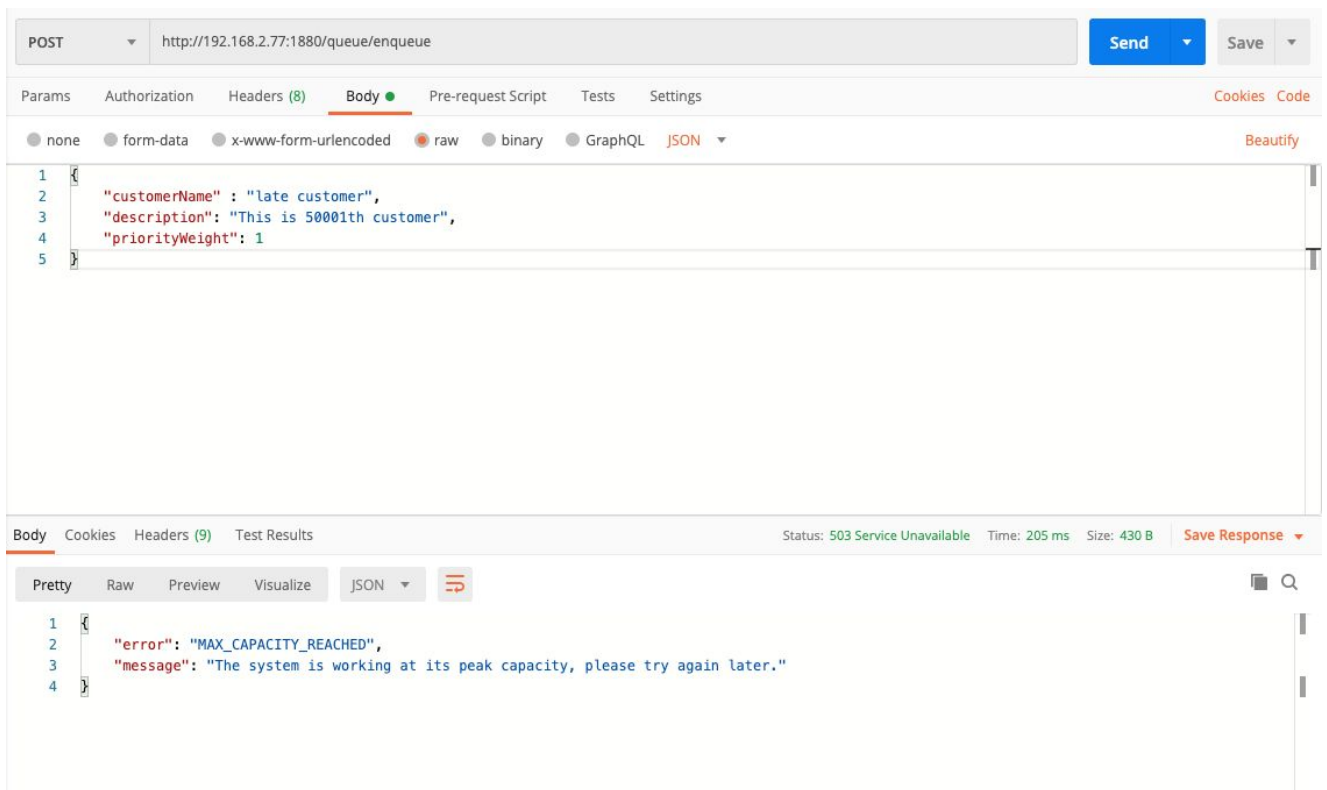
Pretty Raw Preview Visualize JSON ▼ 

```

1 {
2   "customerName": "eleven",
3   "description": "eleven",
4   "priorityWeight": 1,
5   "id": 11,
6   "enqueueTime": "2020-06-27T07:28:14.248Z",
7   "positionInQueue": 11
8 }

```

When the queue is serving at its maximum capacity, it should return the following error message.



POST ▼ http://192.168.2.77:1880/queue/enqueue Send Save ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies Code


● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL ● JSON ▼ Beautify

```

1 {
2   "customerName": "late customer",
3   "description": "This is 50001th customer",
4   "priorityWeight": 1
5 }

```

Body Cookies Headers (9) Test Results Status: 503 Service Unavailable Time: 205 ms Size: 430 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 

```

1 {
2   "error": "MAX_CAPACITY_REACHED",
3   "message": "The system is working at its peak capacity, please try again later."
4 }

```

Note

Screenshots might contain different API URLs. You must use URL mentioned in API details.

Renege Customer Request

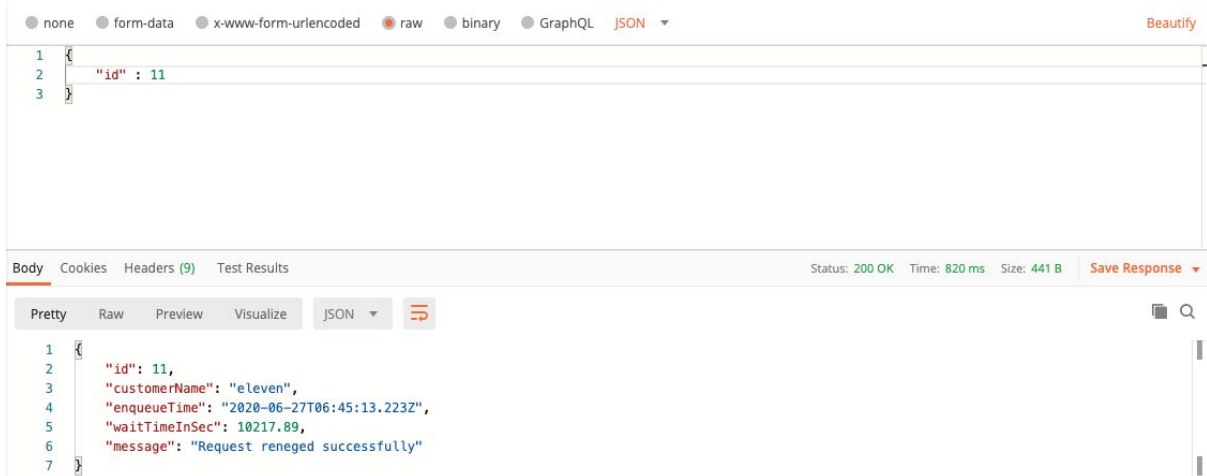
API Route: /api/v1.0/queue/renege

Request Type: DELETE

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: If a customer is tired of waiting for service, he/she can renege his request. The request should return something like as follows:



```

1 {
2   "id": 11,
3 }
  
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 820 ms Size: 441 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 11,
3   "customerName": "eleven",
4   "enqueueTime": "2020-06-27T06:45:13.223Z",
5   "waitTimeInSec": 10217.89,
6   "message": "Request reneged successfully"
7 }
  
```

System Information

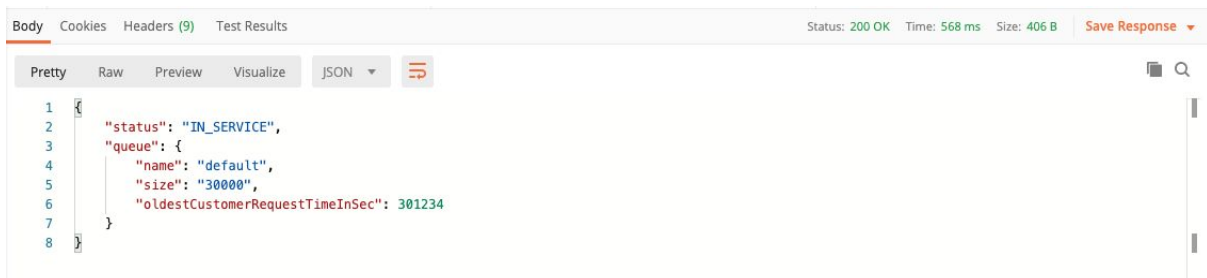
API Route: /api/v1.0/SystemInfo

Request Type: GET

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: This API should return the overall system information i.e.



```

1 {
2   "status": "IN_SERVICE",
3   "queue": {
4     "name": "default",
5     "size": "30000",
6     "oldestCustomerRequestTimeInSec": 301234
7   }
8 }
  
```

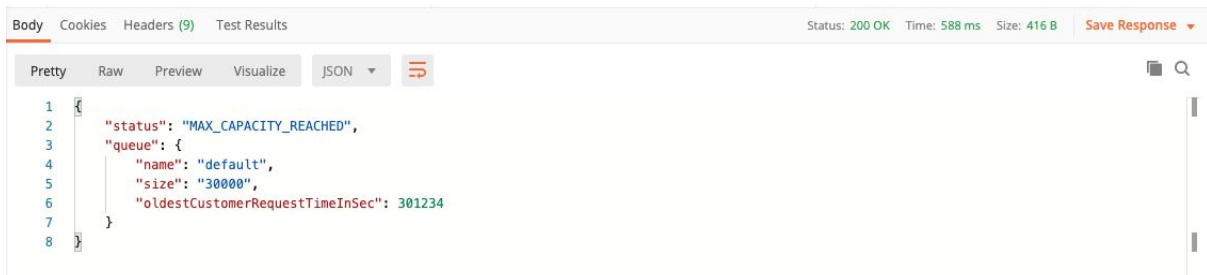
Body Cookies Headers (9) Test Results Status: 200 OK Time: 568 ms Size: 406 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "status": "IN_SERVICE",
3   "queue": {
4     "name": "default",
5     "size": "30000",
6     "oldestCustomerRequestTimeInSec": 301234
7   }
8 }
  
```

Or in a case when the queue has max customer requests.



```

1 {
2   "status": "MAX_CAPACITY_REACHED",
3   "queue": {
4     "name": "default",
5     "size": "30000",
6     "oldestCustomerRequestTimeInSec": 301234
7   }
8 }
  
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 588 ms Size: 416 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "status": "MAX_CAPACITY_REACHED",
3   "queue": {
4     "name": "default",
5     "size": "30000",
6     "oldestCustomerRequestTimeInSec": 301234
7   }
8 }
  
```

System Memory Dump (Optional)

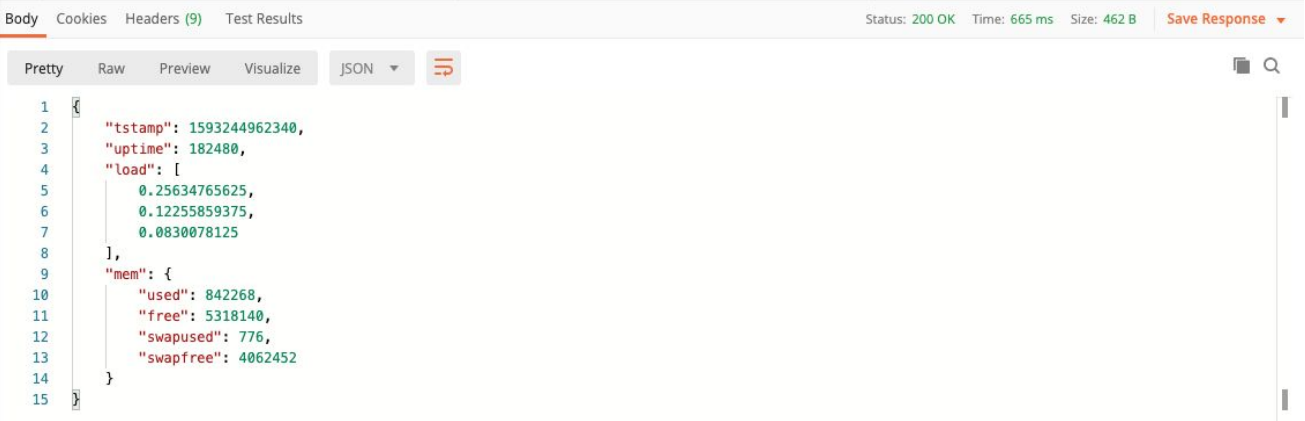
API Route: /api/v1.0/statistics

Request Type: GET

Status Code: Return status codes should comply with the [Http Status Codes](#).

Return Content-type: [application/json](#)

Description: This API should return runtime matrices of application. The returned result may include different attributes based on the libraries used to capture the system status. For example:



The screenshot shows a web browser's developer tools interface. The 'Body' tab is selected, displaying a JSON response. The status bar at the top indicates 'Status: 200 OK', 'Time: 665 ms', and 'Size: 462 B'. The JSON data is as follows:

```

{
  "tstamp": 1593244962340,
  "uptime": 182480,
  "load": [
    0.25634765625,
    0.12255859375,
    0.0830078125
  ],
  "mem": {
    "used": 842268,
    "free": 5318140,
    "swapused": 776,
    "swapfree": 4062452
  }
}
  
```