Test cases

Done by :

Teo Lian Mao P1926705

Mahia Clare Cassandra Dayrit P1936166

Tools used:Postman

Table of contents

|  |
| --- |
| Customer  1.Check Queue  1.1 Check Queue-200  1.2 Check Queue-404  1.3 Check Queue-200(customer in queue)  2.Join Queue  2.1 Join Queue-201  2.2 Join Queue-422  2.3 JoinQueue-422(queue is inactive)  Company  3.Create Queue  3.1 Create Queue-201  3.2 Create Queue-400  3.3Create Queue-422  4.Update Queue  4.1 Update Queue-200  4.2 Update Queue-400  4.3 Update Queue-404  5.Server Available  5.1 Server Available-200  5.2 Server Available-200(no customer in queue)  5.3 Server Available-404  6.Arrival Rate  6.1 Arrival Rate-200  6.2 Arrival Rate-404  6.3 Arrival Rate-400 |

Customer

1. Check Queue

**1.1 Check Queue-200**

|  |
| --- |
| **Title:** Check Queue API - Successful request checks the total number of people in the queue,people ahead, and status of the customer  **Description:** A successful request will display the total number of people in the queue that are not served and the total number of people ahead of the given customer\_id.  **Precondition:** Custom table should be created, with 3 records where queue\_id=’QUEUE42069’, queue\_id is the correct type.    **Test Steps:**   1. Start the Backend Server 2. Send a Check Queue GET request to the backend with  queue\_id=’QUEUE42069’ and customer\_id=1234567890 in the query params 3. Receive a 200 OK response 4. Check the response body to check if results are correct   **Expected Result:** Based on the image above, that particular customer id is next in the queue by the time created and its isn’t served,so there should be no customers ahead.Since only one customer in this queue has been served, the total should be 2  **Example Evidence:**    All expectations met- **PASS** |

**1.2 Check Queue-404**

|  |
| --- |
| **Title:** Check Queue API - Successful request sends a 404 error indicating that the queue cannot be found  **Description:** A successful request will display an error message showing that the given queue id does not exist  **Precondition:** Custom table should be created in the company table    **Test Steps:**   1. Start the Backend Server 2. Send a Check Queue GET request to the backend with  queue\_id=’QUEUE53412’ and customer\_id=1234567890 in the query params, this queue id does not exist in the database 3. Receive a 404 not found response 4. Check the response body to check if results are correct   **Expected Result:** The error message states that the queue id of QUEUE53412 cannot be found, with an error code of “unknown queue”  **Example Evidence:**    All expectations met- **PASS** |

**1.3 Check Queue-200(customer not in queue)**

|  |
| --- |
| **Title:** Check Queue API - Successful request sends a 200 response indicating that the customer is not in the queue  **Description:** A successful request will display the total number of people in the queue QUEUE96420 and an inactive status, the customer\_id=’1265286542’ will be used, since it is not in QUEUE69420  **Precondition:** Custom table should be created in the customer table with multiple rows for queue\_id=”QUEUE69420”    **Expected Result:** 200 ok response showing the total number of people in the queue, a status of inactive and the ahead value being -1  **Test Steps:**   1. Start the Backend Server 2. Send a Check Queue GET request to the backend with  queue\_id=’QUEUE69420’ and customer\_id=1265286542 in the query params, with this customer id being not in the queue 3. Receive a 200 success response 4. Check the response body to check if results are correct   **Expected Result:** 200 ok response showing the total number of people in the queue being 1 (excluding served customers), a status of inactive and the ahead value being -1  **Example Evidence:**    All expectations met- **PASS** |

**2. Join Queue**

**2.1 Join Queue-201**

|  |
| --- |
| **Title:** join Queue API - Successful request allows the customer to join an existing queue using customer\_id and queue\_id  **Description:** A successful request will give a 201 response, indicating that the customer has joined the queue  **Precondition:** Table created with a few customers in queue 69420 that are not served yet, customer\_id and queue\_id in the body are the correct types.    **Test Steps:**   1. Start the Backend Server 2. Send a join Queue POST request to the backend with the body comprising of queue\_id=’’QUEUE69420” and customer\_id=1234567891 3. Receive a 201 OK response 4. Go to elephant SQL , run a select statement to check the response body to see if the new field is added   **Expected Result:** New queue is created for the customer with a 201 ok response status  **Example Evidence:**      All expectations met- **PASS** |

**2.2 Join Queue-422**

|  |
| --- |
| **Title:** join Queue API - Successful request shows that that customer is already in the queue  **Description:** A successful request will give a 422 response, indicating that the customer has been the queue  **Precondition:** Table created with a few customers in queue 69420    **Test Steps:**   1. Start the Backend Server 2. Send a join Queue POST request to the backend with the body comprising of queue\_id=’’QUEUE69420” and customer\_id=1234567890, with this customer id being already in the queue 3. Receive a 422 Unprocessable Entity   response   1. Check the response body in postman   **Expected Result:** 422 error message shows that customerid is 1234567890 is already in QUEUE69420, code being already in queue with code being “ALREADY\_IN\_QUEUE”  **Example Evidence:**    All expectations met- **PASS** |

**2.3 Join Queue-422(inactive)**

|  |
| --- |
| **Title:** join Queue API - Successful request shows that that the queue\_id given is inactive  **Description:** A successful request will give a 422 response, indicating that the response cannot continue due to the queue id in the company database being inactive  **Precondition:** Table created with a queue\_id of QUEUE67536 being inactive    **Test Steps:**   1. Start the Backend Server 2. Send a join queue POST request to the backend with the body comprising of queue\_id=’’QUEUE67536” and customer\_id= 1234567890 3. Receive a 422 Unprocessable Entity response 4. Check response body   **Expected Result:** 422 error message states that QUEUE67536 is inactive, with the code showing ‘INACTIVE QUEUE’  **Example Evidence:**    All expectations met- **PASS** |

Company

**3.Create Queue**

**3.1 Create Queue-201**

|  |
| --- |
| **Title:** Create Queue API - Successful request allows the company to create a new queue with its own queue id, company and create as many queues as they like.  **Description:** A successful request will give a 201 response, indicating that the company has created the queue  **Precondition:** Custom table should be created, with multiple records and their respective status    **Test Steps:**   1. Start the Backend Server 2. Send a Create Queue POST request to the backend with the body comprising of queue\_id=’’QUEUE73651” and company\_id= 1234567890 3. Receive a 201 OK response 4. Go to elephant SQL , run a select statement to check the response body to see if the new field is added   **Expected Result:** New queue is created for the company with a 201 ok response status, with the status set to ‘inactive’ as default  **Example Evidence:**    All expectations met- **PASS** |

**3.2 Create Queue-400**

|  |
| --- |
| **Title:** Create Queue API - Successful request gives a error 400 response , indicating that the response body is invalid  **Description:** A successful request will give a 400 response, in which the company\_id would be invalid.  **Precondition:** Custom Company table should be created.    **Test Steps:**   1. Start the Backend Server 2. Send a Create Queue POST request to the backend with the body comprising of queue\_id=’’QUEUE73651” and company\_id= 123457890 3. Receive a 400 Bad Request response 4. Check the response body to see if the error message is correct   **Expected Result:** Error 400 message would be sent with it being about the company id being invalid and the code being “inavlid json body”  **Example Evidence:**    All expectations met- **PASS** |

**3.3 Create Queue-422**

|  |
| --- |
| **Title:** Create Queue API - Successful request gives a error 422 response , indicating that the queue\_id given in the response body already exists  **Description:** A successful request will give a 422 response, showing that the queue\_id has been in the company table and cannot be duplicated  **Precondition:** Custom Company table should be created with multiple records inserted    **Test Steps:**   1. Start the Backend Server 2. Send a Create Queue POST request to the backend with the body comprising of queue\_id=’’QUEUE69420” and company\_id= 1234567890 3. Receive a 422 Unprocessable Entity  response 4. Check the response body to see if the error message is correct   **Expected Result:** Error 422 message would be sent with it being about the queue\_id already existing and the code being “QUEUE\_EXISTS”  **Example Evidence:**    All expectations met- **PASS** |

**4.Update Queue**

**4.1 Update Queue-200**

|  |
| --- |
| **Title:** Update Queue API - Successful request allows the company to update the status an existing queue with its own queue id,.  **Description:** A successful request will give a 200 response, indicating that the company has updated the queue  **Precondition:** Custom table should be created, with multiple records and their respective status    **Test Steps:**   1. Start the Backend Server 2. Send a Update Queue POST request to the backend with the request params comprising of queue\_id=’’QUEUE73651” 3. Receive a 200 OK response 4. Go to elephant SQL , run a select statement to check the response body to see if the status of that queue is changed   **Expected Result:** Queue is updated in the company table with a 200 ok response status, with the status set to ‘active’ from ‘inactive’.  **Example Evidence:**    **Queue\_id ‘QUEUE73651’ changed its status to ACTIVE**  All expectations met- **PASS** |

**4.2 Update Queue-400**

|  |
| --- |
| **Title:** Update Queue API - Successful request gives an error message that the queue\_id is invalid .  **Description:**A successful request will give a 400 response, indicating that the queue\_id is not of proper structure, which is 10 alphanumeric characters long.  **Precondition:** Custom table should be created, with multiple records and their respective status  **Test Steps:**   1. Start the Backend Server 2. Send a Update Queue POST request to the backend with the request params comprising of queue\_id=’’QUEE73651” and status=”DEACTIVATE” 3. Receive a 400 Bad Request response 4. Check the response body to see if expectations are met   **Expected Result:** Error 400 message mentioning the company\_id must be 10 digits and the code being “INVALID\_QUERY\_STRING”  **Example Evidence:**    All expectations met- **PASS** |

**4.3 Update Queue-404**

|  |
| --- |
| **Title:** Update Queue API - Successful request gives an error message showing that the queue\_id given cannot be found .  **Description:**A successful request will give a 404 response, indicating that the queue\_id does not exist.  **Precondition:** Custom table should be created, with multiple records and their respective status    **Test Steps:**   1. Start the Backend Server 2. Send a Update Queue POST request to the backend with the request params comprising of queue\_id=’’QUEUE23465” and status=”DEACTIVATE”, where “QUEUE23465” is not present in the table 3. Receive a 404 Not Found response 4. Check the response body to see if expectations are met   **Expected Result:** Error 404 message mentioning the queue\_id is not found and the code being “INVALID\_QUERY\_STRING”  **Example Evidence:**    All expectations met- **PASS** |

**5. Server Available**

**5.1 Server Available-200**

|  |
| --- |
| **Title:** Server Available API - Successful request allows the company to check which customer is next to be served, if any  **Description:** A successful request will give a 200 response, indicating that the company has an indication of the next customer in the queue, if any.  **Precondition:** Custom table should be created, with multiple records and their statuses as ‘not\_served’, to be served later on, and a record where customer \_id 12345676891    **Test Steps:**   1. Start the Backend Server 2. Send a SERVER AVALIABLE PUT request to the backend with the request params comprising of queue\_id=’’QUEUE42069” 3. Receive a 200 OK response 4. Go to elephant SQL , run a select statement to check the response body to see if the status of that queue is changed   **Expected Result:** Id of the first customer would be shown in the response, with the served status being changed to served. Customers that are already served are not to be included  **Example Evidence:**      **Customer\_id 123678790 served status has changed from not\_served to served, since he is the first in line (time\_created is earlier)**  All expectations met- **PASS** |

**5.2 Server Available-200(no customers in queue)**

|  |
| --- |
| **Title:** Server Available API - Access request allows the company to check which customer is next to be served , but in this case there are no customers in the queue  **Description:** A successful request will give a 200 response, indicating that the company has an indication of the next customer in the queue, which should be none.  **Precondition:** we would use the queue\_id of QUEUE12345, where there are no customers in that queue    **Test Steps:**   1. Start the Backend Server 2. Send a Server Available GET request to the backend with the request body comprising of queue\_id=’’QUEUE12345” 3. Receive a 200 OK response 4. Check response body in postman   **Expected Result:** customer id will be equal to 0 as there are no customers in the queue  **Example Evidence:**    All expectations met- **PASS** |

**5.3 Server Available-404**

|  |
| --- |
| **Title:** Server Available API - Access request allows the company to check which customer is next to be served ,but the queue id is not found  **Description:** A successful request will give a 404 response, indicating that the queue id does not exist in the company table.  **Precondition:** we would use the queue\_id of QUEUE34527, where it does not exist at all  **Test Steps:**   1. Start the Backend Server 2. Send a SERVER AVAILABLE GET request to the backend with the request params comprising of queue\_id=’’QUEUE34527”, 3. Receive a 404 not found response 4. Check response body in postman   **Expected Result:** Response body of 404 is given showing the error that queue\_id QUEUE34527 is not found and code being “UNKNOWN\_QUEUE”    QUEUE34527 does not exist  **Example Evidence:**    All expectations met- **PASS** |

**6. Arrival Rate**

**6.1 Arrival Rate-200**

|  |
| --- |
| **Title:** Arrival Rate API - Successful request allows the company to know the number of customers in a queue over a period of time every second.  **Description:** A successful request will give a 200 response, showing the number of customers and the specific timestamp .  **Precondition:** Custom table should be created, with multiple records and their respective status    **Test Steps:**   1. Start the Backend Server 2. Send a ARRIVAL RATE GET request to the backend with the request params comprising of queue\_id=’’QUEUE42069”,from=’ 2020-11-16 07:09:18’ ,duration =’2’ (2 minutes) 3. Receive a 200 OK response 4. Check response body in web browser   **Expected Result:** Long response body from 2020-11-16 07:09:18 to 2020-11-16 07:11:18’ , count should start from zero then increment to one , followed by 2 as more fields get added over that period of time  **Example Evidence:**    **\*zoom in**    All expectations met- **PASS** |

**6.2 Arrival Rate-404**

|  |
| --- |
| **Title:** Arrival Rate API -queue\_id is not found.  **Description:** A successful request will give a 404 response, indicating that the queue\_id given does not exist in the company database.  **Precondition:** Custom table should be created, with multiple records and use queue\_id QUEUE34527 which does not exist    **Test Steps:**   1. Start the Backend Server 2. Send a ARRIVAL RATE GET request to the backend with the request params comprising of queue\_id=’’QUEUE34527”,from=’ 2020-11-16 07:09:18’ ,duration =’2’ (2 minutes) 3. Receive a 404 not found response 4. Check response body in postman   **Expected Result:** A 404 response indicating that queue id of QUEUE34527 not found and code being ”UNKNOWN\_QUEUE”  **Example Evidence:**    All expectations met- **PASS** |

**6.3 Arrival Rate-400**

|  |
| --- |
| **Title:** Arrival Rate API -invalid json.  **Description:** A successful request will give a 400 response, indicating that the queue\_id is not of proper structure, which is 10 alphanumeric characters long.  **Precondition:** Custom table should be created, with multiple records added.  **Test Steps:**   1. Start the Backend Server 2. Send a ARRIVAL RATE GET request to the backend with the request params comprising of queue\_id=’’SfG56”,from=’ 2020-11-16 07:09:18’ ,duration =’2’ (2 minutes), this queue id is less then 10 digits long 3. Receive a 400 bad request response 4. Check response body in postman   **Expected Result:** A 400 response indicating that queue id is less than 10 digits and code being =’INVALID\_QUERY\_STRING”  **Example Evidence:**    All expectations met- **PASS** |