**Table Of Content**

[**1. Create Queue API**](#_oxj4ugv1wlxy) **2**

[1.1 Successful Request](#_872ab7a678y5) 2

[1.2 Successful Request - Not Case Sensitive](#_gcb3u1gxhwq6) 6

[1.3 Queue Exists](#_o6u1tx1l613h) 10

[1.4 Queue Id Validation Failed](#_g0fxf1621mxe) 12

[1.5 Company Id Validation Failed](#_2xxf77p6loje) 14

[1.6 Unexpected Server Error](#_wu678bsd45iv) 16

[**2. Update Queue API**](#_kpltdjeowecl) **18**

[2.1 Successful request](#_iryvxf12gt9w) 18

[2.2 Non-existence Queue Id](#_9g8dh8ac2rqg) 21

[2.3 Queue Id Validation Failed](#_ivxe7bt4obek) 23

[2.4 Status Validation Failed](#_qokc6x6pf3qo) 25

[2.5 Unexpected Server Error](#_bbc0ufgkx2ml) 27

[**3. Server Available API**](#_2z26ltu0rw63) **29**

[3.1 Successful request](#_62ndfg1nn0dp) 29

[3.2 Successful request](#_cpwh88tlbgvi) 34

[3.3 Non-existence Queue Id](#_xitpy733ecpw) 36

[3.4 Queue Id Validation Failed](#_gbpn267tqqk) 38

[3.5 Unexpected Server Error](#_8uc3ymmi6dlf) 40

[**4. Reset API**](#_m7w10tqh68rw) **42**

[4.1 Successful request](#_nln4fe3180sw) 42

[4.2 Unexpected Server Error](#_nh699ounjc39) 46

[**5. Join Queue API**](#_omu47ce3qs4m) **49**

[5.1 Successful request - Queue Id Not Case Sensitive](#_jhgun0ed9gig) 49

[5.2 Non-existence Queue Id](#_ti4fsar2k7eo) 52

[5.3 Queue Id Validation Failed](#_wg2hlm5fete2) 54

[5.4 Queue Id is Inactive](#_pehfl0indkit) 56

[5.5 Customer Already In Queue](#_m3zt464wsxmf) 58

[**6. Check Queue API**](#_gspemkas8nwr) **60**

[6.1 Successful Request](#_kb145ck479fz) 60

[6.2 Successful Request - Customer to be served next](#_7elw064n0x1o) 62

[6.3 Successful Request - Customer Not in Queue](#_c0w059hbyl7x) 64

[6.4 Queue Id Not Found](#_mkgaret1pyqj) 66

[6.5 Invalid Customer Id](#_qokikniwuiir) 68

[**7. Arrival Rate API**](#_lgpkn13m6zls) **70**

[7.1 Successful response](#_49o6asxqlofb) 70

[7.2 Error Response - Arrival Rate Range Does Not Exist](#_wsxwnf4lalff) 72

[7.3 Error Response - Queue Id Invalid](#_umirqr4ja0d8) 74

[7.4 Error Response - Queue Id Not Found](#_wb5unia7im30) 76

[7.5 Error Response - Date and Time Format is Incorrect](#_dqygdw94vln3) 78

[7.6 Error Response - Duration Invalid](#_ixz2hhrem47e) 80

# 

# 1. Create Queue API

## 1.1 Successful Request

|  |  |
| --- | --- |
| **Title** | Create Queue API - Successful request creates queue in database. |
| **Description** | A successful request made to the server should create a new entry in the queue\_id with the correct parameters. |
| **Precondition** | Table should be created, queue\_id (“QUEUE33333”) does not already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 1234567890,  “queue\_id”: “QUEUE33333”  }   1. Receive a 201 Created response from Postman. 2. Go to elephantsql to inspect the queue\_tab. |
| **Expected Result** | A new row with the queue\_id (“QUEUE33333”) is created. |
| **Evidence** | 201 Created response from Postman.    queue\_id (“QUEUE33333”) created in table Queue on Elephant SQL. |

## 

## 1.2 Successful Request - Not Case Sensitive

|  |  |
| --- | --- |
| **Title** | Create Queue API - Not Case Sensitive - Successful request creates queue in database. |
| **Description** | A successful request made to the server should create a new entry in the queue\_id with the correct parameters. |
| **Precondition** | Table should be created, queue\_id (“QUEUE22222”) does not already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 1234567890,  “queue\_id”: “QueuE22222”  }   1. Receive a 201 Created response from Postman. 2. Go to elephantsql to inspect the queue\_tab. |
| **Expected Result** | A new row with the queue\_id (“QUEUE22222”) is created. |
| **Evidence** | 201 Created response from Postman.    queue\_id (QUEUE22222) created in table Queue on Elephant SQL. |

## 

## 

## 1.3 Queue Exists

|  |  |
| --- | --- |
| **Title** | Create Queue API - Queue Exists - Unsuccessful request will not create Queue in Database. |
| **Description** | An unsuccessful request made to the server due to a queue already existing should return an error that indicates queue id already exists. |
| **Precondition** | Table should be created, queue\_id (“QUEUE12340”) already exists. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 1234567686,  “queue\_id”: “queue12340”  }   1. Receive a 422 Unprocessable Entity response from Postman that has an error message indicating queue id already exists. |
| **Expected Result** | Receive a 422 Unprocessable Entity response from Postman that has an error message with error code that indicates that queue id(“QUEUE12340”) already exists in the table. |
| **Evidence** |  |

## 

## 

## 

## 

## 

## 

## 1.4 Queue Id Validation Failed

|  |  |
| --- | --- |
| **Title** | Create Queue API - Queue Id Validation Failed - Unsuccessful request will not create Queue in Database. |
| **Description** | An unsuccessful request made to the server due to invalid queue\_id should return an error that indicates the queue\_id is invalid. |
| **Precondition** | Table should be created. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 1234567686,  “queue\_id”: “queue1234”  }   1. Receive a 400 Bad Request response from Postman. |
| **Expected Result** | Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue\_id is invalid. |
| **Evidence** |  |

## 

## 1.5 Company Id Validation Failed

|  |  |
| --- | --- |
| **Title** | Create Queue API - Company Id Validation Failed - Unsuccessful request will not create Queue in Database. |
| **Description** | An unsuccessful request made to the server due to invalid company\_id should return an error that indicates the company\_id is invalid. |
| **Precondition** | Table should be created. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 123456789,  “queue\_id”: “queue12340”  }   1. Receive a 400 Bad Request response from Postman. |
| **Expected Result** | Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the company\_id is invalid. |
| **Evidence** |  |

## 1.6 Unexpected Server Error

|  |  |
| --- | --- |
| **Title** | Create Queue API - Unexpected Server Error - Unsuccessful request will not create Queue in Database. |
| **Description** | An unsuccessful request made to the server due to server error should return an error indicating what happened. |
| **Precondition** | Tables should be created, wrong database credentials in database.js. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Create Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/company/queue  Body:  {  “company\_id”: 1234567890,  “queue\_id”: “queue12340”  }   1. Receive a 500 Internal Server response from Postman. |
| **Expected Result** | Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened. |
| **Evidence** |  |

# 2. Update Queue API

## 2.1 Successful request

|  |  |
| --- | --- |
| **Title** | Update Queue API - Successful request update information of a queue. |
| **Description** | A successful request made to the server should update information of a queue with the correct parameters. |
| **Precondition** | Table should be created, queue\_id (QUEUE12345) already exists with default status ‘false’. |
| **Test Steps** | 1. Start the Backend Server 2. Send an Update Queue request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/queue?queue\_id=QUEUE12345  Body:  {  “status”: “ACTIVATE”  }   1. Receive a 200 OK response from Postman. 2. Go to elephantsql to inspect the queue\_tab. |
| **Expected Result** | Status of queue\_id (“QUEUE12345”) should be updated to ‘true’. |
| **Evidence** | 200 OK responses from Postman.    Status of queue\_id (“QUEUE12345”) updated to ‘true’ in table Queue on Elephant SQL. |

## 

## 2.2 Non-existence Queue Id

|  |  |
| --- | --- |
| **Title** | Update Queue API - Non-existence Queue Id. |
| **Description** | An unsuccessful request made to the server due to non-existence queue\_id should return an error that indicates that the queue\_id does not exist. |
| **Precondition** | Table should be created, queue\_id (“QUEUE12440”) does not already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Update Queue request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/queue?queue\_id=QUEUE12440  Body:  {  “status”: “ACTIVATE”  }   1. Receive a 404 Not Found response from Postman. |
| **Expected Result** | Receive a 404 Not Found response from Postman that has an error message with error code that indicates the queue\_id (“QUEUE12440”) not found in the table. |
| **Evidence** |  |

## 

## 2.3 Queue Id Validation Failed

|  |  |
| --- | --- |
| **Title** | Update Queue API - Queue Id Validation Failed - Unsuccessful request will not update the table Queue. |
| **Description** | An unsuccessful request made to the server due to invalid queue\_id should return an error that indicates the queue\_id is invalid. |
| **Precondition** | Table should be created. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Update Queue request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/queue?queue\_id=QUEUE1234  Body:  {  “status”: “ACTIVATE”  }   1. Receive a 400 Bad Request response from Postman that has an error message indicating the queue\_id is invalid. |
| **Expected Result** | Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue\_id is invalid. |
| **Evidence** |  |

## 

## 

## 

## 2.4 Status Validation Failed

|  |  |
| --- | --- |
| **Title** | Update Queue API - Status Validation Failed - Unsuccessful request will not update status of a queue. |
| **Description** | An unsuccessful request made to the server due to invalid status should return an error that indicates the status is invalid. |
| **Precondition** | Table should be created, queue\_id (“QUEUE12340”) already exists in the table Queue. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Update Queue request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/queue?queue\_id=QUEUE12340  Body:  {  “status”: “ACTIVA”  }   1. Receive a 400 Bad Request response from Postman that has an error message indicating the status is invalid. |
| **Expected Result** | Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the status is invalid. |
| **Evidence** |  |

## 

## 

## 

## 

## 

## 2.5 Unexpected Server Error

|  |  |
| --- | --- |
| **Title** | Update Queue API - Unexpected Server Error - Unsuccessful request will not update Queue in Database. |
| **Description** | An unsuccessful request made to the server due to server error should return an error indicating what happened. |
| **Precondition** | Tables should be created, wrong database credentials in database.js. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Update Queue request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/queue?queue\_id=QUEUE12340  Body:  {  “status”: “ACTIVATE”  }   1. Receive a 500 Internal Server response from Postman. |
| **Expected Result** | Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened. |
| **Evidence** |  |

# 3. Server Available API

## 3.1 Successful request

|  |  |
| --- | --- |
| **Title** | Server Available API - Successful request. |
| **Description** | A successful request made to the server should allow the company to indicate that they have a server that is available to serve the next customer in the queue if any. |
| **Precondition** | Tables should be created with meaningful records( Queue table and CustomerQueueNumber table). |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Server Available request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/server  Body:  {  “queue\_id”: “QUEUE12343”  }   1. Receive a 200 OK response from Postman. 2. Go to elephantsql to check the Queue and CustomerQueueNumber table. |
| **Expected Result** | Show the customer\_id of the next customer in the queue to the server if any. |
| **Evidence** | Initially, the current queue number of QUEUE12343 is 0.    After sending a Server Available request to the backend with a valid queue\_id (“QUEUE12343”) as body by using Postman, will then receive 200 OK responses from Postman, it shows the customer\_id of the next customer to be served.    For queue\_id (“QUEUE12343”) in table Queue, the current\_queue\_number updated from 0 to 1.      In table CustomerQueueNumber, the customer\_id that holds queue\_number 1 in QUEUE12343 is 1334567899 which is shown in Postman. |

## 

## 3.2 Successful request

|  |  |
| --- | --- |
| **Title** | Server Available API - Successful request but no customer in Queue. |
| **Description** | A successful request made to the server should allow the company to indicate that they have a server that is available to serve the next customer in the queue if any. |
| **Precondition** | Tables should be created, queue\_id (“QUEUE12340”) with no customer in the queue. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Server Available request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/server  Body:  {  “queue\_id”: “QUEUE12340”  }   1. Receive a 200 OK response from Postman. |
| **Expected Result** | queue\_id (“QUEUE12340”) should show customer\_id=0 in Postman when don't have the next customer in the queue to the server. |
| **Evidence** | Since there is no next customer to serve in QUEUE12340, ‘customer\_id’:0 will be shown in Postman |

## 3.3 Non-existence Queue Id

|  |  |
| --- | --- |
| **Title** | Server Available Queue API - Non-existence Queue Id |
| **Description** | An unsuccessful request made to the server due to non-existence queue\_id should return an error that indicates that the queue id does not exist. |
| **Precondition** | Table should be created, queue\_id (“QUEUE12440”) does not already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Server Available request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/server  Body:  {  “queue\_id”: “QUEUE12440”  }   1. Receive a 404 Not Found response from Postman. |
| **Expected Result** | Receive a 404 Not Found response from Postman that has an error message with error code that indicates the queue\_id (“QUEUE12440”) not found in the table. |
| **Evidence** |  |

## 

## 

## 

## 

## 3.4 Queue Id Validation Failed

|  |  |
| --- | --- |
| **Title** | Server Available API - Queue Id Validation Failed. |
| **Description** | An unsuccessful request made to the server due to invalid queue\_id should return an error that indicates the queue\_id is invalid. |
| **Precondition** | Table should be created. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Server Available request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/server  Body:  {  “queue\_id”: “QUEUE1244”  }   1. Receive a 400 Bad Request response from Postman that has an error message indicating the queue\_id is invalid. |
| **Expected Result** | Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue\_id is invalid. |
| **Evidence** |  |

## 

## 3.5 Unexpected Server Error

|  |  |
| --- | --- |
| **Title** | Server Available API - Unexpected Server Error. |
| **Description** | An unsuccessful request made to the server due to server error should return an error indicating what happened. |
| **Precondition** | Tables should be created, wrong database credentials in database.js. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Server Available request to the backend by using Postman:   Method: PUT  Path: http://localhost:3000/company/server  Body:  {  “queue\_id”: “QUEUE12440”  }   1. Receive a 500 Internal Server response from Postman. |
| **Expected Result** | Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened. |
| **Evidence** |  |

# 

# 

# 4. Reset API

## 4.1 Successful request

|  |  |
| --- | --- |
| **Title** | Reset API - Successful request reset table in database. |
| **Description** | A successful request made to the server should reset the database to its fresh state with no data inserted. |
| **Precondition** | Tables should be created with data inserted.  Table - Queue    Table - CustomerQueueNumber |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Reset request to the backend by using Postman.   Method: POST  Path: http://localhost:3000/reset   1. Receive a 200 OK response from Postman. 2. Go to elephantsql to check the Queue and CustomerQueueNumber table. |
| **Expected Result** | Data in the tables are reset to its fresh state with no data inserted. |
| **Evidence** | 200 OK response from Postman.    Table Queue in ElephantSQL    Table CustomerQueueNumber in ElephantSQL |

## 

## 

## 

## 

## 

## 

## 4.2 Unexpected Server Error

|  |  |
| --- | --- |
| **Title** | Reset API - Unexpected Server Error -Unsuccessful requests will not reset tables in the database. |
| **Description** | An unsuccessful request made to the server due to server error should return an error indicating what happened. |
| **Precondition** | Tables should be created, wrong database credentials in database.js. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Reset request to the backend by using Postman.   Method: POST  Path: http://localhost:3000/reset   1. Receive a 500 Internal Server response from Postman. 2. Go to elephantsql to check the Queue and CustomerQueueNumber table. |
| **Expected Result** | Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened. Unsuccessful requests won't reset tables in the database. |
| **Evidence** | 500 Internal Server response from Postman.    Below is the table in ElephantSQL. Unsuccessful requests won't reset tables in the database. |

# 5. Join Queue API

## 5.1 Successful request - Queue Id Not Case Sensitive

|  |  |
| --- | --- |
| **Title** | Join Queue API - Queue Id not case sensitive. |
| **Description** | Successful request indicating that the customer has joined the queue. |
| **Precondition** | queue\_id(“QUEUE12345”) must exist in table CustomerQueueNumber and customer\_id(1134567899) must not be inside. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Join Queue request to the backend by using Postman:   Method: POST  Path: http://localhost:3000/customer/queue  Body:  {  “customer\_id”: 1134567899,  “queue\_id”: “queue12345”  }   1. Receive a 201 Created response code from Postman. |
| **Expected Result** | Receive a 201 Created response indicating that the customer joined the queue successfully. |
| **Evidence** | The customer with customer\_id(1134567899) joined the queue with queue\_id(“QUEUE12345) successfully. |

## 5.2 Non-existence Queue Id

|  |  |
| --- | --- |
| **Title** | Join Queue API - Non-existence Queue Id. |
| **Description** | Failed request indicating that the queue id doesn't exist. |
| **Precondition** | queue\_id(“QUEUE12355”) must not be in the table. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Join Queue request to the backend by using Postman:   Method: POST    Path: <http://localhost:3000/customer/queue>  Body:  {  “customer\_id”: 1134567899,  “queue\_id”: “queue12355”  }   1. Receive a 404 Not Found response from Postman. |
| **Expected Result** | Receive a 404 Not Found response indicating that the queue does not exist. |
| **Evidence** |  |

## 5.3 Queue Id Validation Failed

|  |  |
| --- | --- |
| **Title** | Join Queue API - Queue Id Validation Failed. |
| **Description** | Failed request indicating that the queue id is not valid. |
| **Precondition** | Tables must be created. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Join Queue request to the backend by using Postman:     Method: POST  Path: http://localhost:3000/customer/queue  Body:  {  “customer\_id”: 1134567899,  “queue\_id”: “queue123456”  }   1. Receive a 400 Bad request from Postman. |
| **Expected Result** | Receive a 400 Bad Request indicating that the queue is invalid. |
| **Evidence** |  |

## 

## 

## 

## 5.4 Queue Id is Inactive

|  |  |
| --- | --- |
| **Title** | Join Queue API - Queue Id is Inactive. |
| **Description** | Failed request indicating that the queue id is not active. |
| **Precondition** | Status of queue\_id(“111111111”) is false. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Join Queue request to the backend by using Postman:   Method: POST  Path: <http://localhost:3000/customer/queue>  Body:  {  “customer\_id”: 1134567899,  “queue\_id”: “1111111111”  }   1. Receive a 422 Unprocessable Entity response from Postman. |
| **Expected Result** | Receive a 422 Unprocessable Entity response indicating that the queue is inactive. |
| **Evidence** |  |

## 

## 

## 5.5 Customer Already In Queue

|  |  |
| --- | --- |
| **Title** | Join Queue API - Customer already in queue. |
| **Description** | Failed request indicating that the customer was already in the queue. |
| **Precondition** | customer\_id(1134567899) inside the queue already. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Join Queue request to the backend by using Postman:   Method: POST  Path: <http://localhost:3000/customer/queue>  Body:  {  “customer\_id”: 1134567899,  “queue\_id”: “queue123456”  }   1. Receive a 404 Not Found response from Postman. |
| **Expected Result** | Receive a 404 Not Found response indicating that the customer was already in the queue from Postman. |
| **Evidence** |  |

# 

# 6. Check Queue API

## 6.1 Successful Request

|  |  |
| --- | --- |
| **Title** | Check Queue API - Successful Request. |
| **Description** | Success request showing the customer position in the queue. |
| **Precondition** | customer\_id(1234567894) and queue\_id(“QUEUE12345”) already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Check Queue request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/customer/queue?customer\_id=1234567894&queue\_id=queue12345   1. Receive a 200 OK response from Postman and json body. |
| **Expected Result** | Receive a 200 OK response and show the position of the customer in the queue. |
| **Evidence** |  |

## 

## 

## 6.2 Successful Request - Customer to be served next

|  |  |
| --- | --- |
| **Title** | Check Queue API - Successful Request - Customer to be served next. |
| **Description** | Success request showing the customer position in the queue. |
| **Precondition** | customer\_id(1234567890) and queue\_id(“QUEUE12345”) already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send a Check Queue request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/customer/queue?customer\_id=1234567890&queue\_id=queue12345   1. Receive a 200 OK code and json body from Postman. |
| **Expected Result** | Receive a 200 OK response and json body that show the position of the customer in the queue from Postman. |
| **Evidence** |  |

## 6.3 Successful Request - Customer Not in Queue

|  |  |
| --- | --- |
| **Title** | Check Queue API - Successful Request - Customer Not in Queue. |
| **Description** | Success request showing the customer position in the queue. |
| **Precondition** | customer\_id(1234567895) and queue\_id(“QUEUE12345”) already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/customer/queue?customer\_id=1234567895&queue\_id=queue12345   1. Receive a 200 OK code and json body from Postman . |
| **Expected Result** | Receive a 200 OK response and json body that show the position of the customer in the queue from Postman. |
| **Evidence** |  |

## 

## 6.4 Queue Id Not Found

|  |  |
| --- | --- |
| **Title** | Check Queue API - Queue Id Not Found |
| **Description** | Error response indicating queue id not found. |
| **Precondition** | queue\_id(“QUEUE12344”) does not already exist in the table. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/customer/queue?customer\_id=1234567895&queue\_id=queue12344   1. Receive a 404 Not Found code and json body from Postman. |
| **Expected Result** | Receive a 404 Not Found code and the error message from Postman. |
| **Evidence** |  |

## 6.5 Invalid Customer Id

|  |  |
| --- | --- |
| **Title** | Check Queue API - Invalid Customer Id. |
| **Description** | Error response indicating customer id is invalid. |
| **Precondition** | Invalid customer\_id(12345678955). |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/customer/queue?customer\_id=12345678955&queue\_id=queue12345   1. Receive a 400 Bad Request and json body from Postman. |
| **Expected Result** | Receive a 400 Bad Request and the error message from Postman. |
| **Evidence** |  |

# 7. Arrival Rate API

## 7.1 Successful response

|  |  |
| --- | --- |
| **Title** | Arrival Rate API . |
| **Description** | Success Response should show the number of arrivals within a certain period of time. |
| **Precondition** | Tables should be created with meaningful records. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path:http://localhost:3000/company/arrival\_rate?queue\_id=QUEUE12345&from=2020-11-27T14%3A36%3A00%2B08%3A00&duration=1   1. Receive a 200 OK from Postman. |
| **Expected Result** | Receive a 200 OK from Postman and json body that show the number of arrivals within a certain period of time. |
| **Evidence** |  |

## 

## 7.2 Error Response - Arrival Rate Range Does Not Exist

|  |  |
| --- | --- |
| **Title** | Arrival Rate API - Arrival Rate Range Does Not Exist. |
| **Description** | A request sent containing an arrival rate range that does not exist should return an error response. |
| **Precondition** | Tables should be created with meaningful records. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/company/arrival\_rate?queue\_id=QUEUE12345&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1   1. Receive a 404 Not Found from Postman. |
| **Expected Result** | Receive a 404 Not Found from Postman and error message indicating the arrival rate does not exist. |
| **Evidence** |  |

## 

## 7.3 Error Response - Queue Id Invalid

|  |  |
| --- | --- |
| **Title** | Arrival Rate API - Queue Id Invalid |
| **Description** | A request sent containing an invalid Queue Id should return an error response. |
| **Precondition** | Tables should be created with meaningful records. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/company/arrival\_rate?queue\_id=QUEUE123455&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1   1. Receive a 400 Bad Request from Postman. |
| **Expected Result** | Receive a 400 Bad Request from Postman and error message indicating the Queue Id is invalid. |
| **Evidence** |  |

## 

## 

## 

## 

## 7.4 Error Response - Queue Id Not Found

|  |  |
| --- | --- |
| **Title** | Arrival Rate API - Queue Id Not Found. |
| **Description** | An unsuccessful request due to non-existence Queue Id should return an error response. |
| **Precondition** | Tables should be created with meaningful records, queue\_id(“QUEUE12344”) does not already exist. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/company/arrival\_rate?queue\_id=QUEUE12344&from=2020-11-26T15%3A25%3A58%2B08%3A00&duration=1   1. Receive a 404 Not Found from Postman. |
| **Expected Result** | Receive a 404 Not Found from Postman and error message indicating the Queue Id is not found. |
| **Evidence** |  |

## 

## 7.5 Error Response - Date and Time Format is Incorrect

|  |  |
| --- | --- |
| **Title** | Arrival Rate API - Date and Time format is Incorrect. |
| **Description** | An unsuccessful request due to incorrect datetime format should return an error response. |
| **Precondition** | Tables should be created with meaningful records. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path: <http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-31T15%3A25%3A58%2B08%3A00&duration=1>   1. Receive a 400 Bad Request from Postman. |
| **Expected Result** | Receive a 400 Bad Request from Postman and error message indicating the date and time format is invalid.(2020/11/31 is not a valid date) |
| **Evidence** |  |

## 7.6 Error Response - Duration Invalid

|  |  |
| --- | --- |
| **Title** | Arrival Rate API - Duration Invalid. |
| **Description** | A request sent containing an invalid duration should return an error response. |
| **Precondition** | Tables should be created with meaningful records. |
| **Test Steps** | 1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman:   Method: GET  Path: http://localhost:3000/company/arrival\_rate?queue\_id=QUEUE12345&from=2020-11-27T21%3A00%3A00%2B08%3A00&duration=1441   1. Receive a 400 Bad Request from Postman. |
| **Expected Result** | Receive a 400 Bad Request from Postman and error message indicating the duration is invalid. |
| **Evidence** |  |