**ADES CA1 Test Cases (Group 4)**

**DIT/2B/02**

**Members:** Alastair Tan (p1936096), Yu Dong En (p1936348), Guan Shao Jun (p1925545)

**Important Note (Precondition):**

Before executing the test-steps, user must:

1. Install and open Postman
2. Refer to our Developer Guide to set up the application
3. Run the server in the command prompt/terminal in Visual Studio Code to start the server, using the command:

**“node server”**

\* If user is using SP Wi-Fi, he/she must:

1. Install Cisco VPN
2. Connect to [vpn.sp.edu.sg](http://vpn.sp.edu.sg/)

# **Table of Contents**

[**Table of Contents**](#_i516dk6lw40c) **2**

[**Unit Tests**](#_s6v3r7zfe22) **3**

[API 1 (Check Queue API)](#_ciai9p9dw9xi) 3

[API 2 (Join Queue API)](#_uaeqrbcxpjbj) 10

[API 3 (Create Queue API)](#_1nfnwnvs8e9y) 16

[API 4 (Update Queue API)](#_ba5bmtx16klt) 21

[API 5 (Server Available API)](#_31hk9pqkf85m) 26

[API 6 (Arrival Rate API)](#_wa2pgojeg34a) 30

[**Acceptance Tests**](#_q20fpw1timhn) **37**

[Situation 1](#_jwrgcsrpaz6v) 37

[Situation 2](#_2lavkrydjjax) 41

[Acceptance Test 1](#_76oyl98dy12w) 44

[Acceptance Test 2](#_jy4eqc9jctyt) 47

[Acceptance Test 3](#_idhu5asz0e0u) 50

[Acceptance Test 4](#_1il983yvng8e) 54

# Unit Tests

## API 1 (Check Queue API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Check Queue API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-001 |  | **Test Designed by:** Alastair Tan Choon Wei |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Check Queue API |  | **Test Execution date:** |
| **Description:** Tester should be able to check the number of people in the queue. |  | **Evidence:** [**https://imgur.com/a/Gs5gBYy**](https://imgur.com/a/Gs5gBYy) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “GET” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | Customer not in Queue | Queue ID provided must be present in the queue table. Customer ID provided must not be in the given Queue. | **URL:** http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12345”  } | **Status**: 200 OK  **Response JSON will return:**   * Total number of people in the queue * Ahead = -1 * Status of the queue   **Expected output:**  {  “total”: 5,  “ahead”: -1,  “status”: “ACTIVE”  } | **Attached in imgur  Link:**  [**https://imgur.com/a/Gs5gBYy**](https://imgur.com/a/Gs5gBYy) |
| 2 | Customer in Queue | Queue ID provided must be present in the database.  Customer ID provided must be in the given Queue and the customer involved must be behind at least 1 other customer. | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567893  queue\_id = “QUEUE12345”  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Number of people ahead of the customer * Status of the queue   **Expected output:**  {  “total”: 5,  “ahead”: 3,  “status: “ACTIVE”  } |
| 3 | Customer at the start of the Queue | Queue ID provided must be present in the database.  Customer ID provided must be in the given Queue and the customer involved must be at the head of the queue. | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567890  queue\_id = “QUEUE12345”  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Number of people ahead of the customer * Status of the queue   **Expected output:**  {  “total”: 5,  “ahead”: 0,  “status: “ACTIVE”  } |
| 4 | Queue without  Customer ID | Queue ID provided must be present in the database. | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  queue\_id = “QUEUE10000”  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Ahead = -1 * Status of the queue   **Expected output:**  {  “total”: 1,  “ahead”: -1,  “status: “ACTIVE”  } |
| 5 | Queue ID is not case sensitive | Queue ID provided must be present in the database.  Customer ID provided must be in the given Queue. | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567893  queue\_id = “QUEuE12345”  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Number of people ahead of the customer * Status of the queue   **Expected output:**  {  “total”: 5,  “ahead”: 3,  “status: “ACTIVE”  } |
| 6 | Queue does not exist (Non existent Queue ID) | Queue ID provided must be present in the database. | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12346”  } | **Status:** 404 Not Found  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Queue ID QUEUE12346 Not Found”,  “code”: “UNKNOWN\_QUEUE”  } |
| 7 | Customer ID provided is too small | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999  queue\_id = “QUEUE12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “customer\_id is below 10 digits”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 8 | Customer ID provided is too long | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 99999999999  queue\_id = “QUEUE12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “customer\_id is above 10 digits”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 9 | Queue ID provided is too short | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999999  queue\_id = “QUEUE1234”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 10 | Queue ID provided is too long | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999999  queue\_id = “QUEUE123456”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 11 | Queue ID provided contains invalid characters | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999999  queue\_id = “QUEUE(234^”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id has invalid characters!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 12 | A required param is not present in the request. | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999999  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is not present!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 13 | Unexpected Server Error | Modify the code by adding a throw(“error”) to the function that runs to simulate an error | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 9999999999  queue\_id = “QUEUE12345”  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

## API 2 (Join Queue API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Join Queue API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-002 |  | **Test Designed by:** Alastair Tan Choon Wei |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Join Queue API |  | **Test Execution date:** |
| **Description:** Tester should be able to join the queue as a customer. |  | **Evidence:** [**https://imgur.com/a/veb92Zu**](https://imgur.com/a/veb92Zu) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “POST” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | Customer is able to join a queue successfully | Customer ID provided should not be present in the given Queue and the status of the queue in the database must be “ACTIVE”. | **URL:** http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12345”  } | A new row will be created in the customers table. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | **Attached in imgur  Link:**  [**https://imgur.com/a/veb92Zu**](https://imgur.com/a/veb92Zu) |
| 2 | Customer ID provided is already in given queue | Customer ID provided should be present in the given Queue and the status of the queue in the database must be “ACTIVE”. | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12345”  } | **Status:** 422 Unprocessable Entity  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Customer ‘1234567899’ is already in queue ‘QUEUE12345’!”,  “code”: “ALREADY\_IN\_QUEUE”  } |
| 3 | Customer joins queue that does not exist | Queue ID must not be present in the database. | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12346”  } | **Status:** 404 Not Found  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  "error": "Queue ID QUEUE12346 not found.",  "code": "UNKNOWN\_QUEUE"  } |
| 4 | Customer ID provided is not an integer | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = “a234567899”  queue\_id = “QUEUE12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “customer\_id is not an integer!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 5 | Customer ID provided is too small | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 123456789  queue\_id = “QUEUE12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “customer\_id is below 10 digits!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 6 | Customer ID provided is too long | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 12345678999  queue\_id = “QUEUE12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “customer\_id is above 10 digits!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 7 | Queue ID provided is too short | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE1234”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 8 | Queue ID provided is too long | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE123456”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 9 | Queue ID provided contains invalid characters. | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE!@#$5”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id has invalid characters!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 10 | Queue ID provided is not a String | - | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = 1234567899  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is not a String!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 11 | Status of queue provided is not active | The status of the queue in the database needs to be “INACTIVE” | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “1111111111”  } | **Status:** 422 Processable Entity  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Queue 111111111 is inactive.”,  “code”: “INACTIVE\_QUEUE”  } |
| 12 | A required param is not present in the request. | - | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  customer\_id = 1234567899  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message indicating which param is not present * Error code   **Expected output:**  {  “error”: “queue\_id is not present!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 13 | Unexpected Server Error | Modify the code by adding a throw(“error”) to the function that runs to simulate an error | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567899  queue\_id = “QUEUE12345”  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

## API 3 (Create Queue API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Create Queue API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-003 |  | **Test Designed by:** Alastair Tan Choon Wei |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Create Queue API |  | **Test Execution date:** |
| **Description:** Tester should be able to create a queue. |  | **Evidence:** [**https://imgur.com/a/Y303UPy**](https://imgur.com/a/Y303UPy) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “POST” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | Queue is created successfully | The ID of the customer should not be present in the database and the queue id must not already exist in the database. | **URL:** http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = "QUEUE12348"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | **Attached in imgur  Link:**  [**https://imgur.com/a/Y303UPy**](https://imgur.com/a/Y303UPy) |
| 2 | Queue already exist | The ID of the customer needs to be present in the database and the queue id must already exist | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = "QUEUE12345"  } | **Status:** 422 Unprocessable Entity  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: ”Queue ID ‘QUEUE12345’ already exists”,  “code”: “QUEUE\_EXISTS”  } |
| 3 | Company ID provided contains invalid characters or is a string | The ID of the queue must not be present in the database | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = “one4567\*()”  queue\_id = "QUEUE12345"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “company\_id is not an integer!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 4 | Company ID provided is too small | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 123456789  queue\_id = "QUEUE12345"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “company\_id is too short!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 5 | Company ID provided is too long | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 12345678999  queue\_id = "QUEUE12345"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “company\_id is too long!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 6 | Queue ID provided is too small | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = "QUEUE1234"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 7 | Queue ID provided is too long | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = "QUEUE123456"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 8 | Queue ID provided is not a String | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = 1234567899  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is not a String!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 9 | Queue ID provided contains invalid characters | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567890  queue\_id = "QUEUE123$%"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id has invalid characters!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 10 | A required param is not present in the request. | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  company\_id = 1234567899  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message indicating which param is not present * Error code   **Expected output:**  {  “error”: “queue\_id is not present!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 11 | Unexpected Server Error | Modify the code by adding a throw(“error”) to the function that runs to simulate an error | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567899  queue\_id = "QUEUE12346"  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

## API 4 (Update Queue API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Update Queue API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-004 |  | **Test Designed by:** Alastair Tan Choon Wei |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Update Queue API |  | **Test Execution date:** |
| **Description:** Tester should be able to update a queue. |  | **Evidence:** [**https://imgur.com/a/313Bz54**](https://imgur.com/a/313Bz54) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “PUT” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | Queue is updated successfully | The queue must be present in the database. | **URL:** http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE12345”  }  **Body Data:**  {  status = “ACTIVATE”  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | **Attached in imgur  Link:**  [**https://imgur.com/a/313Bz54**](https://imgur.com/a/313Bz54) |
| 2 | Queue does not exist | The queue must not be in the database. | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE12346”  }  **Body Data:**  {  status = “ACTIVATE”  } | **Status:** 422 Unprocessable Entity  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: ”Queue ID ‘QUEUE12346’ cannot be found”,  “code”: “UNKNOWN\_QUEUE”  } |
| 3 | Status provided in the body is not “DEACTIVATE” OR “ACTIVE” | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE12346”  }  **Body Data:**  {  status = “INACTIVATE”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Status must be either ‘ACTIVATE’ or ‘DEACTIVATE’”,  “code”: “INVALID\_JSON\_BODY”  } |
| 4 | Queue ID provided contains invalid character | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEU!12346”  }  **Body Data:**  {  status = “ACTIVATE”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id has invalid characters!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 5 | Queue ID provided is too small | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE1234”  }  **Body Data:**  {  status = “ACTIVATE”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 6 | Queue ID provided is too long | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE123456”  }  **Body Data:**  {  status = “ACTIVATE”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 7 | queue\_id is not present in the request | - | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  status = "ACTIVATE"  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message indicating which param is not present * Error code   **Expected output:**  {  “error”: “queue\_id is not present!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 8 | status is not defined in the body | - | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE12346”  }  **Body Data:**  {  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Status must be either ‘ACTIVATE’ or ‘DEACTIVATE’”,  “code”: “INVALID\_JSON\_BODY”  } |
| 9 | Unexpected Server Error | Modify the code by adding a throw(“error”) to the function that runs to simulate an error | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE12346”  }  **Body Data:**  {  status = "ACTIVATE"  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

## API 5 (Server Available API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Server Available API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-005 |  | **Test Designed by:** Yu Dong En, Alastair Tan Choon Wei |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Server Available API |  | **Test Execution date:** |
| **Description:** Tester should be able to retrieve the customer ID and update the customer ID served. |  | **Evidence:** [**https://imgur.com/a/vgGjiA7**](https://imgur.com/a/vgGjiA7) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “PUT” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | Notify the Server to serve the next available customer. | A queue by this ID must be present in the database .  The queue must contain customers that have not been served yet. | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue12345”  } | **Status**: 200 OK  Customer that is next in line will have its status changed to TRUE in the database and be returned in the JSON object.  **Response JSON will return:**   * Next customer ID   **Expected output:**  {  “customer\_id”: 1234567890  } | **Attached in imgur  Link:**  [**https://imgur.com/a/vgGjiA7**](https://imgur.com/a/vgGjiA7) |
| 2 | No one is left in the queue | A queue by this ID must be present in the database.  The queue must either contain no customers or only contain customers that have been served already | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue12346”  } | **Status**: 200 OK  **Response JSON will return:**   * Customer ID as 0   **Expected output:**  {  “customer\_id”: 0  } |
| 3 | Queue does not exist. | A queue by this ID must not be present in the database. | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue12347”  } | **Status:** 404 Not Found  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Queue ID ‘queue12347’ cannot be found!”,  “code”: “UNKNOWN\_QUEUE”  } |
| 4 | Queue ID provided is too long | - | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue123456”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 5 | Queue ID provided is too short | - | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue1234”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 6 | Queue ID provided contains invalid characters | - | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “QUEU!12345”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id has invalid characters!”,  “code”: “INVALID\_JSON\_BODY”  } |
| 7 | Queue ID is not provided in the body. | - | **URL:** http://localhost:3000/company/server | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message indicating which param is not present * Error code   **Expected output:**  {  “error”: “queue\_id is not present”,  “code”: “INVALID\_JSON\_BODY”  } |
| 8 | Unexpected Server Error | - | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “queue12346”  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

## API 6 (Arrival Rate API)

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Cases for Arrival Rate API**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** A-006 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Test Arrival Rate API |  | **Test Execution date:** |
| **Description:** Tester should be able to retrieve the number of customers that arrive within a given period of time. |  | **Evidence:** [**https://imgur.com/a/MD9bIbc**](https://imgur.com/a/MD9bIbc) |

|  |
| --- |
| **Pre-Conditions:** Change the Postman request to “GET” to test the API. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Title** | **Precondition** | **Test Data** | **Expected Results** | **Evidence** |
| 1 | No customers joining in the given time period | A queue with no customers in it is present in the database. | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-12T07:06:33%2B00:00”  duration = 180  } | **Status**: 200 OK  **Response JSON will return:**  An Array containing JSON Objects which contain:   * Timestamp * Count   The count will be the number of customers that joined during that timestamp.  **e.g** If 2 customers join during that timestamp, the count would be 2 for that timestamp.  The Length of the Array would be Duration Specified \* 60  **Expected output:**  An Array containing JSON Objects like:  {  “timestamp”: 1605164794,  “count”: 0  }  The array’s length would be 10800. | **Attached in imgur  Link:**  [**https://imgur.com/a/MD9bIbc**](https://imgur.com/a/MD9bIbc) |
| 2 | 1 or more customer joins in the given time period | A queue with 1 or more customers in it  is present in the database.  The period which is derived from the from and duration parameters must include the recorded timestamp.  **eg**. if the customer joined at 2020-11-30T10:00:01,  the from must be before that,  2020-11-30T10:00:00  and the duration must encompass that time,  180 | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-21T06:00:00%2B00:00”  duration = 180  } | **Status**: 200 OK  **Response JSON will return:**  An Array containing JSON Objects which contain:   * Timestamp * Count   The count will be that amount of customers that joined during that timestamp.  **e.g** If 2 customers join during that timestamp, the count would be 2 for that timestamp.  The Length of the Array would be Duration Specified \* 60  **Expected output:**  An Array containing JSON Objects like:  {  “timestamp”: 1605164794,  “count”: 0  }  With 1 Object containing  {  “timestamp”: 1605941033,  “count”: 1  }  The array’s length would be 10800 |
| 3 | Queue does not exist | A queue\_id that is the same as the value to be inserted into the query must not be present in the database. | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12346”  from = “2020-11-30T10:00:00%2B08:00”  duration = 180  } | **Status:** 404 Not Found  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Queue ID QUEUE12346 Not Found”,  “code”: “UNKNOWN\_QUEUE”  } |
| 4 | Request is missing a required Query String | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-21T06:00:00%2B00:00”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message indicating which param is not present * Error code   **Expected output:**  {  “error”: “duration is not present!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 5 | Queue ID provided is too long | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE123456”  from = “2020-11-30T10:00:00%2B08:00”  duration = 180  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too long!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 6 | Queue ID provided is too short | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE1234  from = “2020-11-30T10:00:00%2B08:00”  duration = 180  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “queue\_id is too short!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 7 | from provided is not in the correct format | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020/11/30T10:00:00%2B08:00”  duration = 180  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “from is not in a correct format!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 8 | duration provided is too low | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-30T10:00:00%2B08:00”  duration = “0”  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “duration is too low!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 9 | duration provided is too high | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-30T10:00:00%2B08:00”  duration = 1441  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “duration is too high!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 10 | duration provided is not a Integer | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-30T10:00:00%2B08:00”  duration = -1.5  } | **Status:** 400 Bad Request  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “duration is not a integer!”,  “code”: “INVALID\_QUERY\_STRING”  } |
| 11 | Unexpected Server Error | - | **URL:** http://localhost:3000/company/arrival\_rate  **Query Data:**  {  queue\_id = “QUEUE12345”  from = “2020-11-30T10:00:00%2B08:00”  duration = 180  } | **Status:** 500 Internal Server Error  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Unable to establish connection with database”,  “code”: “UNEXPECTED\_ERROR”  } |

|  |
| --- |
| **Post Conditions:** |

# Acceptance Tests

## Situation 1

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-001 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Common Situation 1 (Start and End) |  | **Test Execution date:** 20 November 2020 |
| **Description:** Create a queue, activate the queue, add a customer to the queue, serve the customer, make sure the queue is empty, then deactivate the queue. |  | **Evidence:** [**https://imgur.com/a/Li41lJ8**](https://imgur.com/a/Li41lJ8) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test.  A Queue with an ID of ‘QUEUE11111’ should not exist in the queue table. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a POST request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  company\_id = 1234567890  queue\_id = "QUEUE11111"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | Creating a Queue |
| 2 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11111”  }  **Body Data:**  {  status = "ACTIVATE"  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | Activating a Queue |
| 3 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object specified in the test data column | **URL:** http://localhost:3000/customer/queue  **Body Data:**  {  customer\_id = 1234567890  queue\_id = "QUEUE11111"  } | A new row will be created in the database. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | Adding a Customer to the queue |
| 4 | Send a PUT request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = "QUEUE11111"  } | Customer that is next in line will have its status changed to TRUE in the database and be returned in the JSON object.  **Status**: 200 OK  **Response JSON will return:**   * Next customer ID   **Expected output:**  {  “customer\_id”: 1234567890  } | Serving the Customer |
| 5 | Send a PUT request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:** http://localhost:3000/company/server  **Body Data:**  {  queue\_id = "QUEUE11111"  } | **Status**: 200 OK  **Response JSON will return:**   * Customer ID as 0   **Expected output:**  {  “customer\_id”: 0  } | Making sure queue is empty |
| 6 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:** http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11111”  **Body Data:**  {  status = "DEACTIVATE"  } | The status of the queue in the queue table will be updated to ‘INACTIVE’.  **Status**: 200 OK | Deactivating Queue |

|  |
| --- |
| **Post Conditions:** |

## Situation 2

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-002 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Common Situation 2 (Getting Information) |  | **Test Execution date:** 20 November 2020 |
| **Description:** Get the a certain queue’s information and its arrival rate for the day, then reset the database. |  | **Evidence:** [**https://imgur.com/a/Vf7IYit**](https://imgur.com/a/Vf7IYit) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test.  An active Queue with an ID of ‘QUEUE12345’ should exist in the queue table. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a GET request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:** http://localhost:3000/customer/queue  **Query Data:**  {  queue\_id = “QUEUE12345”  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Number of people ahead of the customer * Status of the queue   **Expected output:**  {  "total": 1234,  "ahead": 0,  "status": "ACTIVE"  } | Getting the Data of the queue |
| 2 | Send a GET request to  <http://localhost:3000/company/arrival_rate>  with a body of a JSON object and the query params specified in the test data column | **Query Data:**  {  queue\_id = “QUEUE12346"  from = “2020-11-19T07:06:33%2B00:00”  duration = 1440  } | **Status**: 200 OK  **Response JSON will return:**  An Array containing JSON Objects which contain:   * Timestamp * Count   The count will be the number of customers that joined during that timestamp.  **Expected output:**  An Array containing JSON Objects  {  "timestamp": "1605164794",  "count": 0  }  The Array’s Length would be 86400. | Getting the Arrival Rate for the day. |
| 3 | Send a POST request to  <http://localhost:3000/reset> | - | **Status:** 200 OK | Resetting Database |

|  |
| --- |
| **Post Conditions:** |

## 

## Acceptance Test 1

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-003 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Customer is able to join queue |  | **Test Execution date:** 26 November 2020 |
| **Description:**  **Given** that a queue is active AND a customer is not in the queue  **When** the customer joins a queue  **Then** the customer should be in the queue AND is able to view the number of people ahead of itself in the queue. |  | **Evidence:** [**https://imgur.com/a/f0y2FeN**](https://imgur.com/a/f0y2FeN) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a POST request to  <http://localhost:3000/company/queue>  with a body of a JSON object specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Body Data:**  {  company\_id = 1234567890  queue\_id = "QUEUE11112"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | Creating a Queue |
| 2 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11112”  }  **Body Data:**  {  status = "ACTIVATE"  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | Activating a Queue |
| 3 | Send a GET request to  <http://localhost:3000/customer/queue>  with the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  queue\_id = “QUEUE11112”  } | **Status**: 200 OK  **Response JSON will return:**   * Total number of people in the queue * Ahead = -1 * Status of the queue   **Expected output:**  {  "total": 0,  "ahead": -1,  "status": "ACTIVE"  } | Making sure no customers are in the queue |
| 4 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:** http://localhost:3000/customer/queue  **Body Data:**  {  queue\_id = “QUEUE11112”  customer\_id = 1234567890  } | A new row will be created in the customers table. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | Adding a customer to the queue |
| 5 | Send a GET request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  queue\_id = “QUEUE11112”  customer\_id = 1234567890  } | **Status:** 200 OK  **Response JSON will return:**   * Total number of people in the queue * Number of people ahead of the customer * Status of the queue   **Expected output:**  {  "total": 1,  "ahead": 0,  "status": "ACTIVE"  } | View number of people ahead of itself. |

|  |
| --- |
| **Post Conditions:** |

## Acceptance Test 2

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-004 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Companyis able to stop Customers from joining queues |  | **Test Execution date:** 26 November 2020 |
| **Description:**  **Given** that a queue is active AND customers are constantly joining the queue  **When** the company deactivate the queue  **Then** the customers would eventually no longer be able to join the queue AND arrival rate drops to zero. |  | **Evidence:** [**https://imgur.com/a/Ndfcu21**](https://imgur.com/a/Ndfcu21) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a POST request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  company\_id = 1234567890,  queue\_id = "QUEUE11113"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | Creating a Queue |
| 2 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11113”  }  **Body Data:**  {  status = "ACTIVATE"  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | Activating a Queue |
| 3 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  queue\_id=”QUEUE11113”,  customer\_id =1234567890  } | A new row will be created in the customers table. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | Adding a customer to the queue |
| 4 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11113”  }  **Body Data:**  {  status = “DEACTIVATE”  } | The status of the queue in the queue table will be updated to ‘INACTIVE’.  **Status**: 200 OK | Deactivating Queue |
| 5 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  queue\_id = “QUEUE11113”,  customer\_id = 1234567891  } | **Status:** 422 Unprocessable Entity  **Response JSON will return:**   * Error message * Error code   **Expected output:**  {  “error”: “Queue ‘QUEUE11113’ is inactive.”,  “code”: “INACTIVE\_QUEUE”  } | Attempting to add a customer to the queue |

|  |
| --- |
| **Post Conditions:** |

## Acceptance Test 3

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-005 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Company is able to allocate a customer to an available server even after queue is deactivated |  | **Test Execution date:** 26 November 2020 |
| **Description:**  **Given** that there are customers waiting in the queue  **When** the company indicate an available server (1 server)  **Then** the company will get the customer at the top of the queue to be served  **Given** that there are still customers waiting in the queue  **When** the company deactivate the queue AND indicate an available server  **Then** the company will get the next customer at the top of the queue to be served. |  | **Evidence:** [**https://imgur.com/a/7PPazEt**](https://imgur.com/a/7PPazEt) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a POST request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  company\_id = 1234567890,  queue\_id = "QUEUE11114"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | Creating a Queue |
| 2 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11114”  }  **Body Data:**  {  status = "ACTIVATE"  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | Activating a Queue |
| 3 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  queue\_id = “QUEUE11114”,  customer\_id = 1234567890  } | A new row will be created in the customers table. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | Adding a first customer to the queue |
| 4 | Send a POST request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Body Data:**  {  queue\_id = “QUEUE11114”,  customer\_id = 1234567891  } | A new row will be created in the database. The customer should be added into the queue using the Queue ID provided.  **Status**: 201 Created | Adding a second customer to the queue |
| 5 | Send a PUT request to  <http://localhost:3000/company/server>  with a body of a JSON object in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “QUEUE11114”  } | Customer that is first in line will have its status changed to TRUE in the database and be returned in the JSON object.  **Status**: 200 OK  **Response JSON will return:**   * First in line customer ID   **Expected output:**  {  “customer\_id”: 1234567890  } | Serve the first customer |
| 6 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = ”QUEUE11114”  }  **Body Data:**  {  status = "DEACTIVATE"  } | The status of the queue in the queue table will be updated to ‘INACTIVE’.  **Status**: 200 OK | Deactivating the Queue |
| 7 | Send a PUT request to  <http://localhost:3000/company/server>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  queue\_id = “QUEUE11114”  } | Customer that is first in line will have its status changed to TRUE in the database and be returned in the JSON object.  **Status**: 200 OK  **Response JSON will return:**   * First in line customer ID   **Expected output:**  {  “customer\_id”: 1234567891  } | Serve the next customer in the newly deactivated queue |

|  |
| --- |
| **Post Conditions:** |

## Acceptance Test 4

|  |
| --- |
| **Project Name:** ADES CA1 |

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case ID:** B-006 |  | **Test Designed by:** Yu Dong En |
| **Module Name:** ADES |  | **Test Executed by:** Yu Dong En |
| **Test Title:** Multiple customers are able to join an active queue concurrently |  | **Test Execution date:** 26 November 2020 |
| **Description:**  **Given** that there is an active queue  **When** more than 1 customer try to join the queue at the same time  **Then** all the customers will see success and are eventually in the queue AND the arrival rate would reflect a spike around this time. |  | **Evidence:** [**https://imgur.com/a/POmRVqJ**](https://imgur.com/a/POmRVqJ) |

|  |
| --- |
| **Pre-Conditions:**  Postman is required to run this test. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Results** | **Notes** |
| 1 | Send a POST request to  <http://localhost:3000/company/server>  with a body of a JSON object specified in the test data column | **URL:**  http://localhost:3000/company/server  **Body Data:**  {  company\_id = 1234567890,  queue\_id = "QUEUE11115"  } | A new row will be created in the queue table. A new queue will be added using the Queue ID provided.  **Status**: 201 Created | Creating a Queue |
| 2 | Send a PUT request to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/queue  **Query Data:**  {  queue\_id = “QUEUE11115”  }  **Body Data:**  {  status = "ACTIVATE"  } | The status of the queue in the queue table will be updated to ‘ACTIVE’.  **Status**: 200 OK | Activating a Queue |
| 3 | Send 2 PUT requests to  <http://localhost:3000/company/queue>  with a body of a JSON object and the query params specified in the test data column  At the same time, using Postman’s collection runner to queue up 2 requests and run them at the same time. | **URL (For both Requests):**  http://localhost:3000/company/queue  **Query Data (For both Requests):**  {  queue\_id = “QUEUE11115”  }  **Body Data (For Request 1):**  {  queue\_id = "QUEUE11115"  customer\_id = 1234567890  }  **Body Data (For Request 2):**  {  queue\_id = "QUEUE11115”  customer\_id = 1234567891  } | **Status:** 201 Created (for both Requests) | Simultaneously joining a queue |
| 4 | Send a GET request to  <http://localhost:3000/customer/queue>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/customer/queue  **Query Data:**  {  queue\_id = "QUEUE11115"  } | **Status**: 200 OK  **Response JSON will return:**   * Total number of people in the queue * Ahead = -1 * Status of the queue   **Expected output:**  {  "total": 2,  "ahead": -1,  "status": "ACTIVE"  } | Checking the queue |
| 5 | Send a GET request to  <http://localhost:3000/company/arrival_rate>  with a body of a JSON object and the query params specified in the test data column | **URL:**  http://localhost:3000/company/arrival\_rate  **Body Data:**  {  queue\_id = “QUEUE11115”  from = “2020-11-26T21:00:00%2B80:00”  duration = 60  } | **Status**: 200 OK  **Response JSON will return:**  An Array containing JSON Objects which contain:   * Timestamp * Count   The count will be the number of customers that joined during that timestamp.  **Expected output:**  An Array containing JSON Objects similar to this:  {  “timestamp”: 1605164794,  “count”: 0  }  Including 1 JSON object inside displaying:  {  “timestamp”: 1606398302,  “count”: 2  }  The Array’s Length would be 360. | Check the arrival rate |

|  |
| --- |
| **Post Conditions:** |