E2E API Tests – In Progress

Test Objective: Ensuring all API endpoints function as expected

Scope of Testing: This plan focuses on API functional testing, including Create, Read, Update, and Delete (CRUD) operations for various endpoints.

Test Strategy:

Testing Types:

- **Functional Testing**: Validating that API endpoints behave as expected.
- **End-to-End Testing**: Ensuring complete workflows function correctly.

Test tools:

- Mocha/Chai for automated tests and assertions in JavaScript

Test environment:

- Environment: Staging
- Dependencies: Installed app on the company used for testing, in draft mode with configured webhooks
- All data (created resources, users, companies, branches) required for the tests, is deleted after the execution of the tests

Test Scope and Coverage:

Tested API Endpoints:

-Get Token

-Get All Customers

-Create new customer

-Get Customer By ID

-Update Customer

-Delete customer

-Retrieve Company Data

-Retrieve Branch Data

-Retrieve Service Availability

-Retrieve Service Avail By External ID

-Reserve Booking

-Confirm Booking

-Delete Booking

-List all services

-Create a Service

-Retrieve a Service

-Negative Test Retrieve a Service With Wrong service ID

-Negative Test Create a Service - Missing Body

-Retrieve Service Booking Limit

-Reset Service

-Update a Service

-Negative Test Update Service with wrong ID

-Delete a Service

-List all Service Categories

-Create Service Category

-Retrieve Data For Service Category

-Update Service Category

-Delete Service Category

-List all resources

-Create Resource

-Update resource

-Delete a Resource By ID

-List all appointments

-Create Appointment

-Retrieve data for an appointment

-Update Appointment

-Delete an appointment

-List all enterprise customers

-Create enterprise customer

-Get enterprise customer

-Update enterprise customer

-Delete enterprise customer

-Get all enterprise custom field categories

-Create enterprise custom field category

-GetEnterpriseCustomFieldCategory

-Update enterprise custom field category

-Delete enterprise custom field category

-List all time shifts for a certain resource

-Create time shift for a resource

-Get single Time Shift By ID

-Update a time shift for a resource

-Delete a time shift for a resource

-Get information for a company

-Change online status of a company

-List all customer custom fields

-List all custom field categories

-Retrieve data of a custom field category

-Create customer custom field category

-Update custom field category name

-Delete custom field category

-List all group services

-List all group service categories

-Create group service

-Update group service

-Retrieve data for group service by ID

-Delete a group service

-Create group service category

-Update group service category

-Delete a group service category

Test Cases:

Positive test cases: Ensuring the entire flow of the endpoints with valid data is running with correct status code and no errors.

TC01 Validate successful token retrieval and expiration time

This test verifies the functionality of the "Get Token" API endpoint, which is used to retrieve access and refresh tokens for authorization purposes. The test checks that the API responds with a status code of 200 and validates that the access token expiration time is set correctly. Additionally, the test stores the retrieved access token, refresh token, and expiration time in global variables for use in future tests.

Preconditions:

- Ensure that environment variables APP_ID and APP_SECRET are correctly configured in the .env file.
- Ensure that the base URL for the API is properly set in the Globals.js file.

Test Steps:

1. Send a POST request to the /v1/auth/token endpoint with the following details:
    a. Headers:
        i. Content-Type: application/json
    b. Body:

```
{
  "appid": "{{APP_ID}}",
  "appsecret": "{{APP_SECRET}}"
}
```

2. Receive the response from the API containing the access token, refresh token, and expiration time.
3. Store the accessToken, refreshToken, and expires values in global variables using helper functions.

Assertions:

1. Verify that the response status code is 200.
2. Verify that the expires field in the response body is equal to 7200 (seconds).

Postconditions:

- The access token, refresh token, and expiration time are stored in global variables for future use.

CUSTOMERS ENDPOINT

TC02 - Validate the retrieval of all customers with pagination and sorting

Description:

This test suite verifies the functionality of the "Get All Customers" API endpoint. It checks if the API correctly returns a list of customers when queried with pagination, sorting, and filtering parameters. The suite performs various checks, including status code validation, ensuring that the response body is in the correct format, and confirming that the customer data is returned as expected.

Preconditions:

- Ensure that valid access tokens and company IDs are stored in global variables (getAccessToken() and getCompanyId()).
- The API base URL is properly configured in Globals.js.

Test Steps:

1. Send a GET request to the /v1/customers endpoint with the following query parameters:
    a. page=1: The page of results to return.
    b. limit=20: The number of customers to return per page.
    c. sort=name: The sorting field for the customers, sorted by their name.
    d. sort_type=asc: The sorting order (ascending).

Headers:

    e. Authorization: Bearer token from getAccessToken().
    f. company-id: Company ID from getCompanyId().
    g. Content-Type: application/json.
2. Receive the response from the API, which includes a list of customers.

Assertions:

1. **Status Code:** Verify that the response status code is 200, indicating a successful request.
2. **Response Format:** Verify that the response is an object.
3. **Customer Data:** Verify that the response contains a data field, which should be an array.
4. **Non-null Data:** Verify that the data array is not null and contains customer records.

Postconditions:

- The test confirms that the "Get All Customers" API endpoint functions correctly, returning the expected list of customers based on the provided pagination and sorting parameters.

TC03 - Validate the creation of a new customer with custom fields

Description

This test suite verifies the functionality of the "Create New Customer" API endpoint. It checks if the API correctly creates a new customer when provided with valid custom fields. The suite performs various checks, including status code validation, ensuring that the response body is in the correct format, and confirming that the customer data is returned as expected.

Preconditions

- Ensure that valid access tokens and company IDs are stored in global variables (getAccessToken() and getCompanyId()).

- The API base URL is properly configured in Globals.js.

Test Steps

1. **Configure Request Options**:
    a. Set the request method to POST.
    b. Set the headers to include Authorization with the access token, Content-Type as application/json;charset=utf-8, and company-id with the company ID.
    c. Set the request body to include custom fields for the new customer:{
    "customfields": [
    {
    "id": "67458a745fd82371b8033d39",
    "value": "TestVSC3"
    },
    {
    "id": "67458a745fd82371b8033d3f",
    "value": "LastNameVSC2"
    },
    {
    "id": "67458a745fd82371b8033d38",
    "value": "testapi@mail.com"
    }
    ]
    }

2. **Send POST Request**:
    a. Send a POST request to ${getBaseUrl()}v1/customers with the configured options.
3. **Parse Response**:
    a. Await the response and parse the response body to extract the new customer details.
    b. Store the customer ID using setUserId(customerId).
4. **Validate Response Status**:
    a. Check that the response status code is 200 using expect(response.status).to.equal(200).

5. **Validate Response Structure**:
    a. Ensure the response body is an object using expect(customer).to.be.an('object').
    b. Verify that the customer object is not null using expect(customer).is.not.null.
    c. Check that the firstName field is "TestVSC3" using expect(customer.data.firstName).to.equal("TestVSC3").
    d. Check that the lastName field is "LastNameVSC2" using expect(customer.data.lastName).to.equal("LastNameVSC2").
    e. Verify that the email field is "testapi@mail.com" using expect(customer.data.email).to.equal("testapi@mail.com").

Postconditions

- The test confirms that the "Create New Customer" API endpoint functions correctly, returning the expected customer details based on the provided custom fields.
- The new customer ID is stored for further use.

TC04 - Validate the retrieval of a customer by ID

Description

This test suite verifies the functionality of the "Get Customer By ID" API endpoint. It checks if the API correctly returns the details of a customer when queried with a specific customer ID. The suite performs various checks, including status code validation, ensuring that the response body is in the correct format, and confirming that the customer data is returned as expected.

Preconditions

- Ensure that valid access tokens and company IDs are stored in global variables (getAccessToken() and getCompanyId()).
- The API base URL is properly configured in Globals.js.
- A valid customer ID is stored using getUserId().

Test Steps

1. **Configure Request Options**:
    a. Set the request method to GET.
    b. Set the headers to include Authorization with the access token, Content-Type as application/json;charset=utf-8, and company-id with the company ID.
2. **Send GET Request**:
    a. Send a GET request to ${getBaseUrl()}v1/customers/${getUserId()} with the configured options.
3. **Parse Response**:
    a. Await the response and parse the response body to extract the customer details.
    b. Store the customer ID using customerIdToCheck = customer.data.id.
4. **Validate Response Status**:
    a. Check that the response status code is 200 using expect(response.status).to.equal(200).
5. **Validate Response Structure**:
    a. Ensure the response body is an object using expect(customer).to.be.an('object').
    b. Verify that the customer object is not null using expect(customer).to.not.be.null.
    c. Check that the id field in the response matches the expected customer ID using expect(customer.data.id).to.equal(getUserId()).

Postconditions

- The test confirms that the "Get Customer By ID" API endpoint functions correctly, returning the expected customer details based on the provided customer ID.
- The customer ID is validated to match the expected ID.

Endpoint	Expected Result	Actual Result
Verify that the /v1/auth/token endpoint returns an access token with status code 200 and correct expiration time.	Returns accessToken with status 200, expire time correct	✓ Returns accessToken, status 200, expire time correct (626ms)
Get All Customers	Status 200, Returns object, data array of customers, data not null	✓ Status 200, Returns object, data array of customers, data not null
Create New Customer	Status 200, Returns object, Customer object not null, Correct first name, last name, and email	✓ Status 200, object returned, first name, last name, and email are correct
Get Customer By ID	Status 200, Returns object, Customer not null, correct customer ID	✓ Status 200, Returns object, correct customer ID, customer not null
Update Customer	Status 200, Returns object, Customer object not null, First name updated	✓ Status 200, object returned, First name updated correctly
Delete Customer	Status 200, Verify customer is deleted	✓ Status 200, customer deleted (126ms)
Retrieve Company Data	Status 200, Returns object, data object of company data not null	✓ Status 200, Returns company data object, data not null
Retrieve Branch Data	Status 200, Returns object, correct Enterprise ID	✓ Status 200, Returns object, correct Enterprise ID
Retrieve Service Availability	Status 200, Correct response type, Correct service and resource ID	✓ Status 200, correct service and resource ID
Retrieve Service Avail by External ID	Status 200, Correct response type, Correct service and external resource ID	✓ Status 200, correct service and resource ID
Reserve Booking	Status 200, Correct booking data format, Booking ID and secret not null	✓ Status 200, booking ID and secret not null
Confirm Booking	Status 200, Correct booking format, Booking ID as expected	✓ Status 200, booking ID correct, data not null
Delete Booking	Status 200, Verify response is true	✓ Status 200, response true
List All Services	Status 200, Data array of services, data not null	✓ Status 200, data array returned, not null

Create a Service	Status 200, Service created successfully, Correct service ID, name	✓ Status 200, Service ID and name correct
Retrieve a Service	Status 200, Correct response type, Service ID as expected	✓ Status 200, Service ID as expected
Negative Test Retrieve Service with Wrong ID	Status 404, Response name correct - Not Found	✓ Status 404, Response correct - Not Found
Negative Test Create Service - Missing Body	Status 400, Response message correct	✓ Status 400, message correct
Retrieve Service Booking Limit	Status 200, Response not null	✓ Status 200, response not null
Reset Service	Status 200, Price property reset successfully	✓ Status 200, price reset successfully
Update a Service	Status 200, Correct object type, Updated name correct	✓ Status 200, name updated correctly
Negative Test Update Service with Wrong ID	Status 404, Response message correct - Not Found	✓ Status 404, Not Found
Delete a Service	Status 200, Verify service deleted	✓ Status 200, service deleted
List All Service Categories	Status 200, Data array returned, Response object correct	✓ Status 200, data array returned
Create Service Category	Status 200, Service category created, Name correct	✓ Status 200, Name correct
Retrieve Data for Service Category	Status 200, Correct object type, Correct service category ID	✓ Status 200, Service category ID correct
Update Service Category	Status 200, Name property exists, Name updated successfully	✓ Status 200, Name updated
Delete Service Category	Status 200, Verify service category deleted	✓ Status 200, service category deleted
List All Resources	Status 200, Correct data array, Data not null	✓ Status 200, data array returned, not null
Create Resource	Status 200, Resource created, Name and email correct	✓ Status 200, Resource name and email correct
Update Resource	Status 200, Resource updated, Correct name	✓ Status 200, Resource name updated
Delete a Resource By ID	Status 200, Resource deleted successfully	✓ Status 200, Resource deleted

List All Appointments	Status 200, Data array of appointments, Response not null	✓ Status 200, Data array returned, not null
Create Appointment	Status 200, Appointment created, ID exists	✓ Status 200, Appointment ID exists
Retrieve Appointment Data	Status 200, Appointment data retrieved, ID exists	✓ Status 200, ID exists
Update Appointment	Status 200, Appointment updated, ID exists, Title updated	✓ Status 200, Appointment updated
Delete Appointment	Status 200, Appointment deleted	✓ Status 200, Appointment deleted
List All Enterprise Customers	Status 200, Data array exists, Data not null	✓ Status 200, Data array exists
Create Enterprise Customer	Status 200, Customer created, First and last names correct	✓ Status 200, Customer names correct
Get Enterprise Customer	Status 200, Customer data retrieved, Fields exist	✓ Status 200, Fields exist
Update Enterprise Customer	Status 200, Customer updated, First and last names correct	✓ Status 200, Customer names updated
Delete Enterprise Customer	Status 200, Customer deleted	✓ Status 200, Customer deleted
Get All Enterprise Custom Field Categories	Status 200, Data array present	✓ Status 200, Data array exists
Create Enterprise Custom Field Category	Status 200, Category created, Name correct	✓ Status 200, Category name correct
Get Enterprise Custom Field Category	Status 200, Custom field category ID exists	✓ Status 200, ID exists
Update Enterprise Custom Field Category	Status 200, Name updated successfully	✓ Status 200, Name updated
Delete Enterprise Custom Field Category	Status 200, Category deleted successfully	✓ Status 200, Category deleted
List All Time Shifts for a Resource	Status 200, Data array exists, No timeshifts found	✓ Status 200, No timeshifts found
Create Time Shift for a Resource	Status 200, Timeshift created, ID and interval exist	✓ Status 200, Timeshift created
Get Single Time Shift By ID	Status 200, Time shift ID exists	✓ Status 200, ID exists
Update Time Shift	Status 200, Time shift updated, Working times updated	✓ Status 200, Times updated

Delete Time Shift	Status 200, Time shift deleted	✓ Status 200, Time shift deleted
Get Company Information	Status 200, Company ID exists	✓ Status 200, Company ID exists
Change Company Online Status	Status 200, IsOnline property exists	✓ Status 200, IsOnline exists
List All Customer Custom Fields	Status 200, Data array exists, Not empty	✓ Status 200, Data array exists
List All Custom Field Categories	Status 200, Data array exists, Not empty	✓ Status 200, Data array exists
Retrieve Custom Field Category Data	Status 200, Correct object type, Category ID exists	✓ Status 200, Category ID exists
Create Customer Custom Field Category	Status 200, Category created, Custom field category ID exists	✓ Status 200, Category ID exists
Update Custom Field Category Name	Status 200, Name updated successfully	✓ Status 200, Name updated
Delete Custom Field Category	Status 200, Category deleted	✓ Status 200, Category deleted
List All Group Services	Status 200, Data array exists, Group Service ID checked	✓ Status 200, Group Service ID exists
List All Group Service Categories	Status 200, Data array exists, Group Service Category ID checked	✓ Status 200, Group Service Category ID exists
Create Group Service	Status 200, Group service created, Group service name correct	✓ Status 200, Group service name correct
Update Group Service	Status 200, Group service updated, Name updated	✓ Status 200, Group service name updated
Delete Group Service	Status 200, Group service deleted successfully	✓ Status 200, Group service deleted
Retrieve data for group service by ID	Status 200, External service ID exists	✓ Status 200, **Group Service Category** ID exists
Create Group Service Category	Status 200, External service created, ID exists	✓ Status 200, **Group Service Category** ID exists
Update Group Service Category	Status 200, External service updated, Name updated	✓ Status 200, **Group Service Category** name updated
Delete Group Service Category	Status 200, External service deleted successfully	✓ Status 200, External service deleted

Exit Criteria

- **Pass Criteria**: All API functionalities for the entire flow must pass with no errors.
- **Fail Criteria:** 1 or more tests fail.