# [AutomationExercise.com](http://automationexercise.com/) API Testing

1. **Project Overview**

I tested a demo e-commerce website to simulate user-related operations including user registration, user login, delete user, update user, get users/products, search products/brands, etc. The goal of the API testing was to validate functionality, status codes, and error responses using Postman.

# API Endpoints Tested

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /api/productsList | GET | Get all products list |
| /api/brandsList | GET | Get all brands list |
| /api/searchProduct | POST | Search product |
| /api/verifyLogin | POST | Login with valid credentials |
| /api/createAccount | POST | Create a new user account |
| /api/deleteAccount | DELETE | Delete user account |
| /api/updateAccount | PUT | Update user account |
| /api/getUserDetailByEmail | GET | Get user account detail by email |

1. **Test Scenarios**
   1. **Get All Products (/api/productsList)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expecte d Result** | **Actual Result** | **Status** | **Notes** |
| TC001 | Valid request all products list | GET | 200 OK | 200 OK | Passed |  |
| TC002 | Invalid method request products list (POST instead of GET) | POST | 405  Method Not Allowed | 405  Method Not Allowed | Passed with Notes | Actual result is shown in the response body |

* 1. **Get All Brands (/api/brandsList)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expecte d Result** | **Actual Result** | **Status** | **Notes** |
| TC003 | Valid request all brands list | GET | 200 OK | 200 OK | Passed |  |
| TC004 | Invalid method request brands list (PUT instead of GET) | PUT | 405  Method Not Allowed | 405  Method Not Allowed | Passed with Notes | Actual result is shown in the response body |

* 1. **Get All Products (/api/searchProduct)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expect ed Result** | **Actual Result** | **Status** | **Notes** |
| TC005 | Search products list | POST  Search\_product = Polo | 200 OK | 200 OK | Passed |  |
| TC006 | Search products list without parameter | POST | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |
| TC007 | Search product with empty string input | POST | 200 OK | 200 OK | Passed |  |
| TC008 | Invalid Method request search product | GET | 405  Method Not Allowed | 405  Method Not Allowed | Passed with Notes | Actual result is shown in the response body |

* 1. **Create New User Account (/api/createAccount)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expect ed Result** | **Actual Result** | **Status** | **Notes** |
| TC009 | Valid user registration | POST | 201  Created | 201  Created | Passed with Notes | Actual result is shown in the response body |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TC010 | User registration without complete data | POST | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |
| TC011 | User registration with existing email | POST | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |
| TC012 | User registration with invalid email format | POST | 400 Bad Request | 201  Created | Failed |  |

* 1. **User Login (/api/verifyLogin)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expect ed Result** | **Actual Result** | **Status** | **Notes** |
| TC013 | Valid login | POST | 200 OK | 200 OK | Passed |  |
| TC014 | Login with missing email or password | POST | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |
| TC015 | Invalid email login | POST | 404 Not Found | 404 Not Found | Passed with Notes | Actual result is shown in the response body |
| TC016 | Invalid password login | POST | 404 Not Found | 404 Not Found | Passed with | Actual result is |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Notes | shown in the response body |
| TC017 | Invalid method request user login | DELETE | 405  Method Not Allowed | 405  Method Not Allowed | Passed with Notes | Actual result is shown in the response body |

* 1. **Get User Account Detail by Email (/api/getUserDetailByEmail)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expect ed Result** | **Actual Result** | **Status** | **Notes** |
| TC018 | Valid email | GET | 200 OK | 200 OK | Passed |  |
| TC019 | Non-existing email | GET | 404 Not Found | 404 Not Found | Passed with Notes | Actual result is shown in the response body |
| TC020 | Missing email parameter | GET | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |

* 1. **Delete User Account (/api//api/deleteAccount)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expect ed Result** | **Actual Result** | **Status** | **Notes** |
| TC021 | Valid account delete | DELETE | 200 OK | 200 OK | Passed |  |
| TC022 | Wrong email or password | DELETE | 404 Not Found | 404 Not Found | Passed with Notes | Actual result is shown in the response body |
| TC023 | Missing email and password parameter | DELETE | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the response body |

* 1. **Update User Account (/api//api/updateAccount)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Method &**  **Input** | **Expecte d Result** | **Actual Result** | **Status** | **Notes** |
| TC024 | Update user account | PUT | 200 OK | 200 OK | Passed |  |
| TC025 | Update user account with only email and password but no details to update | PUT | 400 OK | 200 OK | Failed |  |
| TC026 | Missing email or password | PUT | 400 Bad Request | 400 Bad Request | Passed with Notes | Actual result is shown in the |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | response body |
| TC027 | Wrong email or password | PUT | 404 Not Found | 404 Not Found | Passed with Notes | Actual result is shown in the response body |

# Postman Test Scripts

* 1. **Status Code Check**

Because the actual status code is shown in response body, I write the scripts to check in response body, not HTTP status.

let response = pm.response.json();

pm.test("Response code is 200", function(){

pm.expect(response.responseCode).to.eql(200);

});

pm.test("Response code is 201", function(){

pm.expect(response.responseCode).to.eql(201);

});

pm.test("Response code is 400", function(){

pm.expect(response.responseCode).to.eql(400);

});

pm.test("Response code is 404", function(){

pm.expect(response.responseCode).to.eql(404);

});

pm.test("Response code is 405", function(){

pm.expect(response.responseCode).to.eql(405);

});

* 1. **Response Time Check**

**pm.test("Response time is less than 1000ms", function () { pm.expect(pm.response.responseTime).to.be.below(1000);**

**});**

* 1. **Response Body Check**
     + GET all products list

pm.test("All products have a name", function(){ response.products.forEach(p => {

pm.expect(p.name).to.be.a('string').to.not.be.empty;

})

});

* + - GET all brands list

pm.test("All brands name are not empty", function(){ response.brands.forEach(p => {

pm.expect(p.brand).to.be.a('string').to.not.be.empty;

})

});

* + - POST Search Product
      * Check every response body contains same value as the search keyword

let response = pm.response.json(); let searchValue = 'Tops';

pm.test("All response categories match the keyword", function(){ response.products.forEach(p => {

pm.expect(p.category.category.toLowerCase()).to.include(searchValue.toLow erCase());

});

});

* + - GET User Detail
      * Check returned response is Valid

let response = pm.response.json(); let search = ['alif123@gmail.com';](mailto:%27alif123@gmail.com)

pm.test("Account detail returned is Valid", function(){ pm.expect(response.user.email).to.eql(search);

});

* 1. **Check Response is An Array/Object or Not Empty**
     + GET all products list

pm.test("Products list is an array", function(){

pm.expect(response.products).to.be.an('Array');

});

pm.test("Products list is not empty", function(){

pm.expect(response.products.length).to.be.above(0);

});

* + - GET all brands list

pm.test("Brands list is an array", function(){

pm.expect(response.brands).to.be.an('Array');

});

pm.test("Brands list is not empty", function(){

pm.expect(response.brands.length).to.be.above(0);

});

* + - POST Search Product

pm.test("Products list is an array", function(){

pm.expect(response.products).to.be.an('Array');

});

pm.test("Products list is not empty", function(){

pm.expect(response.products.length).to.be.above(0);

});

* + - GET User Detail

pm.test("User detail is an object", function(){ pm.expect(response.user).to.be.an('Object');

});

pm.test("User detail is not empty", function(){

pm.expect(Object.keys(response.user).length).to.be.above(0);

});

* 1. **Check Error Message**
     + Invalid Method Request for Brands List

pm.test("Response message is correct", function(){

pm.expect(response.message).to.eql('This request method is not supported.');

});

* + - Valid Login

pm.test("Response message is correct", function(){

pm.expect(response.message).to.eql('User exists!');

});

* + - Invalid Login with Missing Email or Password

pm.test("Response message is correct", function(){

pm.expect(response.message).to.eql('Bad request, email or password parameter is missing in POST request.');

});

* + - DELETE Account with Invalid Password

pm.test("Response message is correct", function(){

pm.expect(response.message).to.eql('Account not found!')

});

* + - UPDATE User Account

pm.test("Response message is correct", function(){

pm.expect(response.message).to.eql('User updated!')

});

# Bugs and Observations

* 1. **User Registration Accepts Invalid Email Format**

## Steps to Reproduce:

1. Send POST request to /api/createAccount
2. Provide invalid email format in request body (e.g. alif@ or alif@gmail)

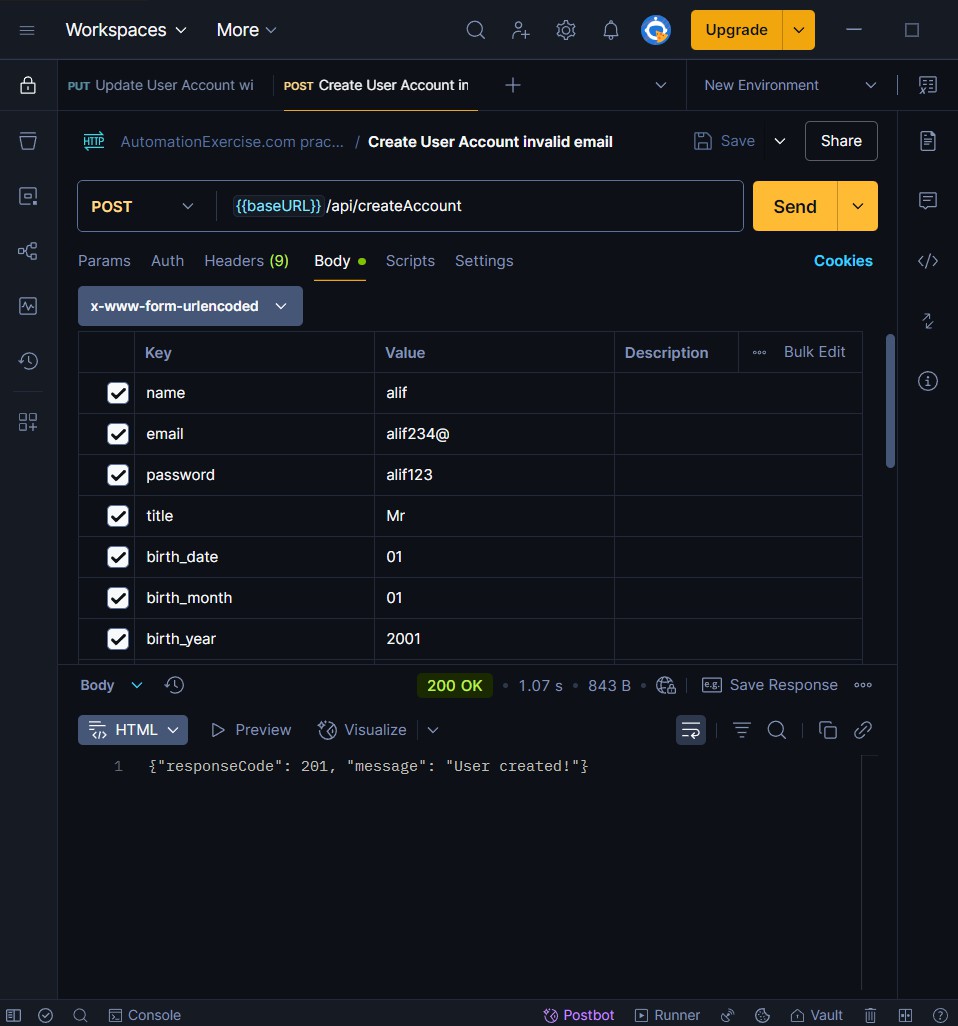
## Expected Result:

1. Should return 400 Bad Request
2. Error message: “Invalid email format”

## Actual Result:

1. Returns 200 OK
2. Message: “User created!”

## Screenshot:

****

* + - **Severity:** High
    - **Suggestion:** Add server side validation for required fields
  1. **User Update Success with Only Email and Password and No Detail to Update**

## Steps to Reproduce:

1. Send PUT request to /api/updateAccount
2. Provide only email and password in request body

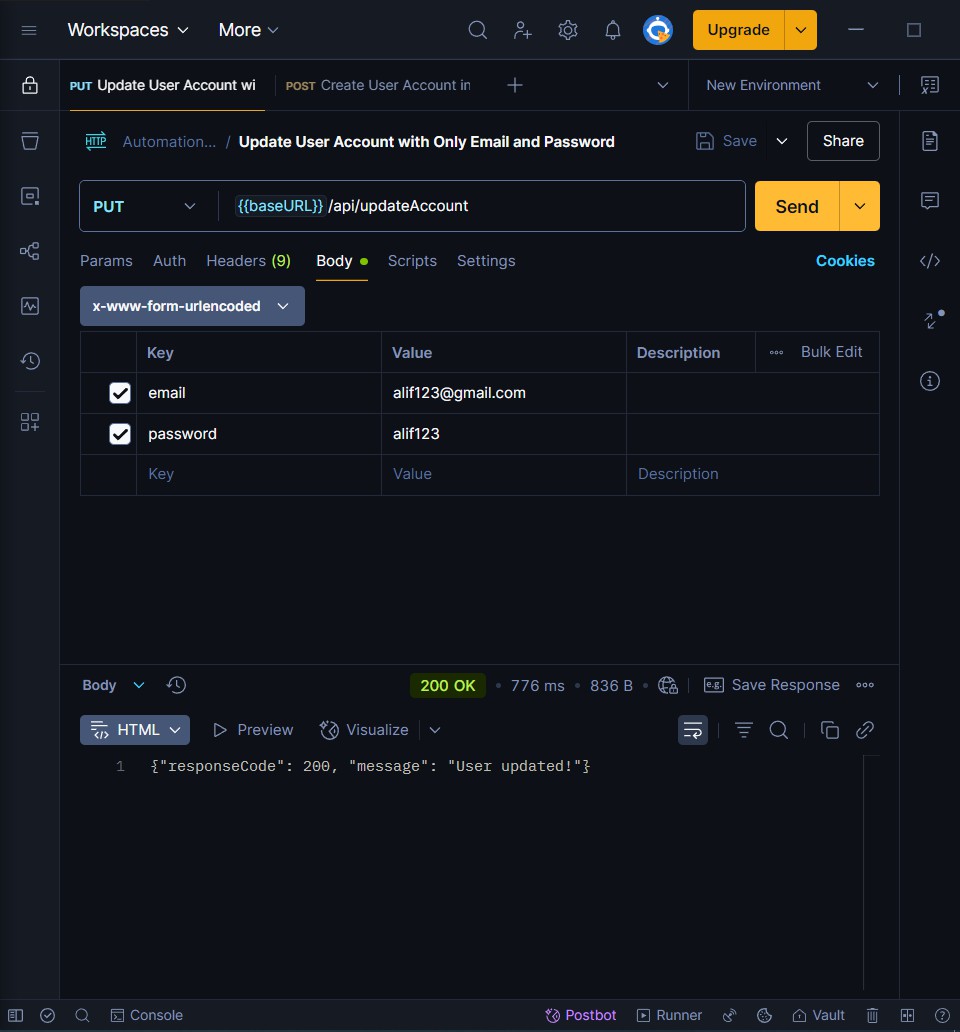
## Expected Result:

1. Should return 400 Bad Request
2. Error message: “Provide detail to update.”

## Actual Result:

1. Returns 200 OK
2. Message: “User updated!”

## Screenshot:

****

* + - **Severity:** Low
    - **Suggestion:** Add server side validation for required fields