Test Case Results Overview

Test Case ID	Test Name	Status	Priority	Defects Logged	Elapsed Time
TC1	Valid User Login	**FAIL**	Medium	API-101	3m
TC2	Invalid User Login	**FAIL**	Medium	API-102	2m
TC3	Fetch All Products	**PASS**	Medium	None	1m
TC4	Fetch a Single Product	**PASS**	Medium	None	2m
TC5	Limit Product Results	**PASS**	Medium	None	1m
TC6	Sort Product Results	**PASS**	Medium	None	2m
TC7	Add a New Cart	**FAIL**	Medium	API-103	6m
TC8	Fetch a User’s Cart	**FAIL**	Medium	API-103	2m
TC9	Delete a Cart	**FAIL**	Medium	API-104	2m

Detailed Results

Test Case ID: TC1
Test Name: Valid User Login
Preconditions: A valid username and password exist in the database.
Steps to Execute:

1. Send a POST request to `/auth/login` with valid credentials.
2. Verify the response contains a valid token.

Expected Result:

- Status Code: 200 OK
- Response Body: `{ "token": "sample-token" }`

Actual Result:

- Status Code: 401 Unauthorized
- Response Body: `{ "error": "username or password is incorrect" }`

Notes:

- Authentication failed with valid credentials.
- **Defect ID:** API-101

Test Case ID: TC2
Test Name: Invalid User Login
Preconditions: None (invalid credentials are used).
Steps to Execute:

1. Send a POST request to `/auth/login` with invalid credentials.
2. Verify the response contains an error message.

Expected Result:

- Status Code: 401 Unauthorized
- Response Body: `{ "error": "Invalid email or password" }`

Actual Result:

- Status Code: 404 Not Found
- Response Body: `Cannot POST /users/signin`

Notes:

- Endpoint appears misconfigured or undocumented.
- **Defect ID:** API-102

Test Case ID: TC3
Test Name: Fetch All Products
Preconditions: The database contains multiple products.
Steps to Execute:

1. Send a GET request to `/products`.
2. Verify all products are returned.

Expected Result:

- Status Code: 200
- Response Body: JSON array of products.

Actual Result:

- **PASS**: List of products successfully retrieved with a matching response.

Test Case ID: TC4
Test Name: Fetch a Single Product
Preconditions: A valid product ID exists in the database.
Steps to Execute:

1. Send a GET request to `/products/{id}` with a valid product ID.
2. Verify product details match expected data.

Expected Result:

- Status Code: 200
- Response Body: Details of the specified product.

Actual Result:

- **PASS**: Retrieved product details matched expected values.

Test Case ID: TC5
Test Name: Limit Product Results
Preconditions: At least 5 products exist in the database.
Steps to Execute:

1. Send a GET request to `/products?limit=5`.
2. Verify only 5 products are returned.

Expected Result:

- Status Code: 200
- Response Body: Array of 5 products.

Actual Result:

- **PASS**: Only 5 products were returned as expected.

Test Case ID: TC6
Test Name: Sort Product Results
Preconditions: Multiple products exist in the database.
Steps to Execute:

1. Send a GET request to `/products?sort=desc`.
2. Verify products are sorted in descending order.

Expected Result:

- Status Code: 200
- Response Body: Products sorted in descending order.

Actual Result:

- **PASS**: Products were correctly sorted in descending order.

Test Case ID: TC7
Test Name: Add a New Cart
Preconditions: None.
Steps to Execute:

1. Send a POST request to `/carts` with a valid payload.
2. Verify the response contains the new cart details.

Expected Result:

- Status Code: 201 Created
- Response Body: New cart details.

Actual Result:

- Status Code: 200
- SyntaxError: Unexpected end of JSON input.

Notes:

- Payload processing failed.
- **Defect ID:** API-103

Test Case ID: TC8
Test Name: Fetch a User's Cart
Preconditions: The specified user ID has an existing cart.
Steps to Execute:

1. Send a GET request to `/carts/user/{userId}` with a valid user ID.
2. Verify the response contains the user's cart details.

Expected Result:

- Status Code: 200
- Response Body: User's cart details.

Actual Result:

- SyntaxError: Unexpected end of JSON input.

Notes:

- Payload response error.
- **Defect ID:** API-103

Test Case ID: TC9
Test Name: Delete a Cart
Preconditions: A valid cart ID exists in the database.
Steps to Execute:

1. Send a DELETE request to `/carts/{id}` with a valid cart ID.
2. Verify the cart is deleted.

Expected Result:

- Status Code: 200
- Response Body: `{ "status": "Cart deleted" }`

Actual Result:

- SyntaxError: Unexpected end of JSON input.

Notes:

- The issue with DELETE operation response handling.
- **Defect ID:** API-104