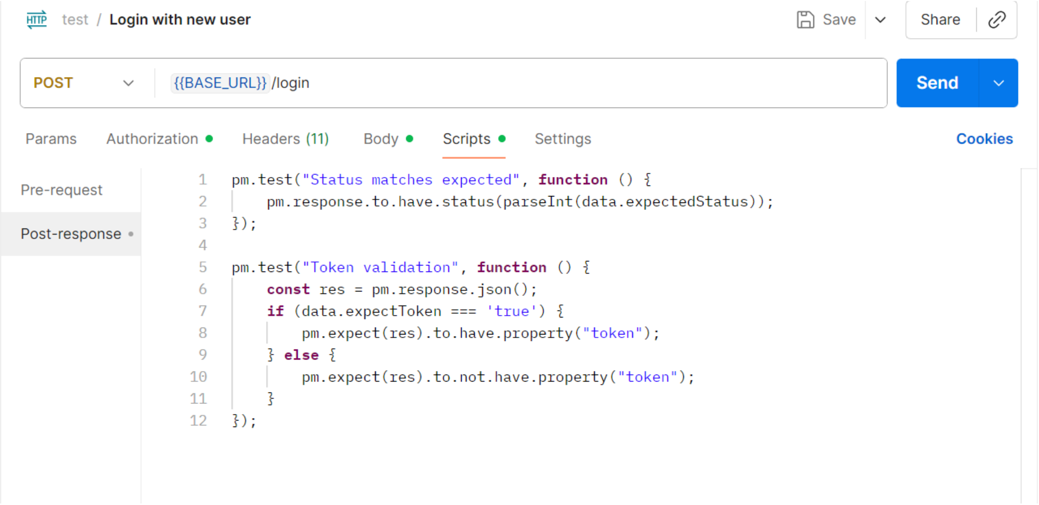
API test Automation using Postman

Made use of environment variables for this project. Also tests are written for every request i.e positive and negative tests.

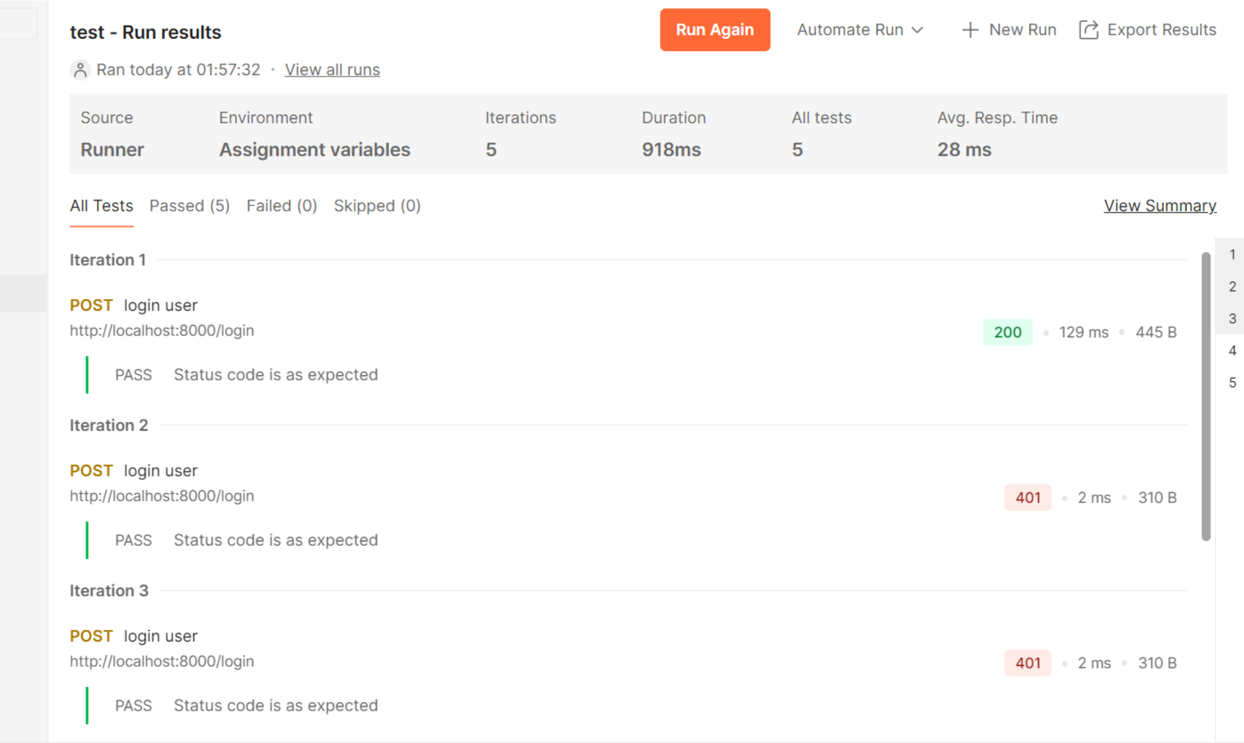
- Login testflow with positive and negative testcases
    - Made the testcase parameterize with giving .csv file
    - Csv file data
        - username,password,expectedStatus,expectToken
        - validuser,correctpass,200,true
        - invalid_user,wrong_password,401,false
        - existing_user,anotherpass,409,false
        - ,blankpass,400,false
        - Userwithnpass,,400,false



```
pm.test("Status matches expected", function () {
    pm.response.to.have.status(parseInt(data.expectedStatus));
});

pm.test("Token validation", function () {
    const res = pm.response.json();
    if (data.expectToken === 'true') {
        pm.expect(res).to.have.property("token");
    } else {
        pm.expect(res).to.not.have.property("token");
    }
});
```

    -



test - Run results

Ran today at 01:57:32 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Assignment variables	5	918ms	5	28 ms

All Tests Passed (5) Failed (0) Skipped (0)

Iteration 1

POST login user
http://localhost:8000/login — 200 • 129 ms • 445 B
PASS Status code is as expected

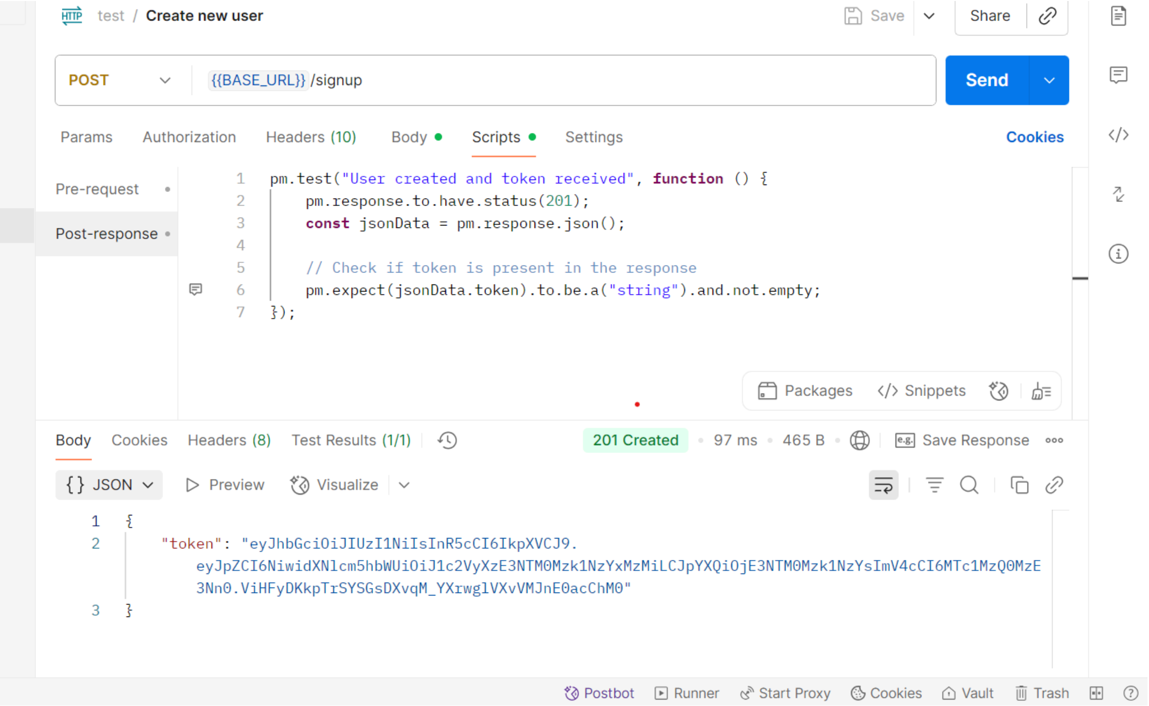
Iteration 2

POST login user
http://localhost:8000/login — 401 • 2 ms • 310 B
PASS Status code is as expected

Iteration 3

POST login user
http://localhost:8000/login — 401 • 2 ms • 310 B
PASS Status code is as expected

- Create new user testcase
    - Wrote this without parameterize by sending the request twice so as it can be executed directly by clicking on Send
    - First is positive request



test / Create new user

POST {{BASE_URL}} /signup

Scripts — Post-response

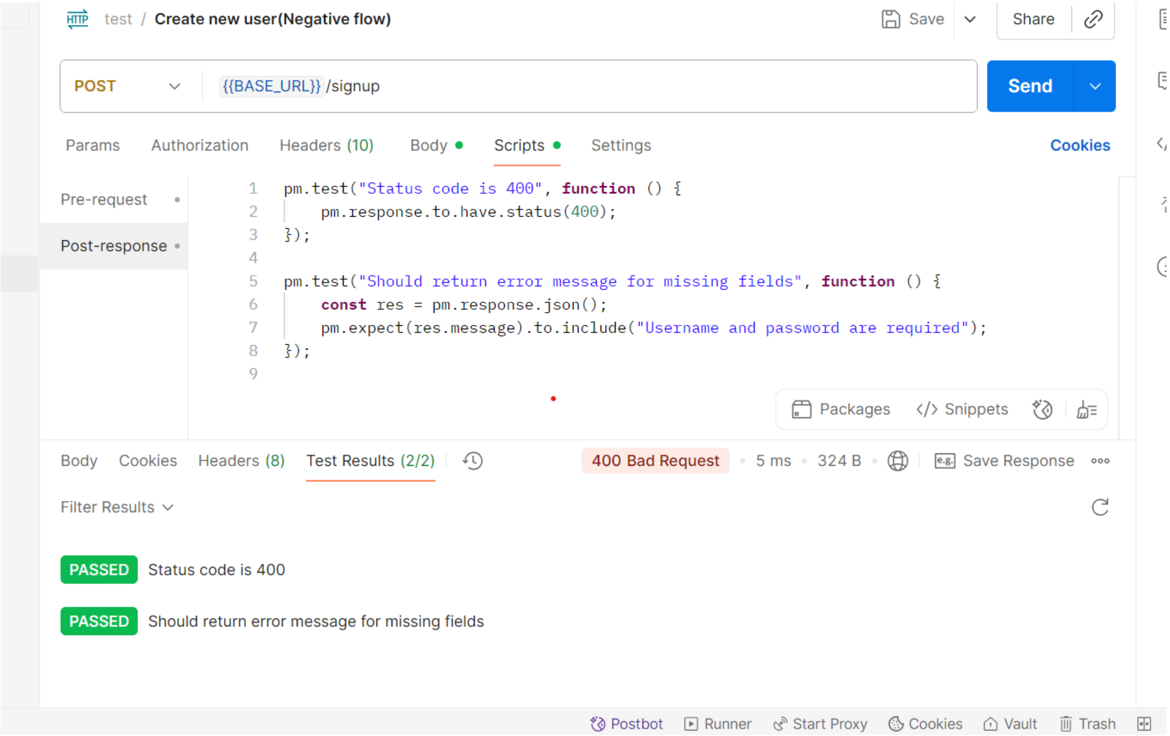
```
pm.test("User created and token received", function () {
    pm.response.to.have.status(201);
    const jsonData = pm.response.json();

    // Check if token is present in the response
    pm.expect(jsonData.token).to.be.a("string").and.not.empty;
});
```

201 Created • 97 ms • 465 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6NiwidXNlcm5hbWUiOiJ1c2VyXzE3NTM0Mzk1NzYxMzMiLCJpYXQiOjE3NTM0Mzk1NzYsImV4cCI6MTc1MzQ0MzE
        3Nn0.ViHFyDKkpTrSYSGsDXvqM_YXrwglVXvVMJnE0acChM0"
}
```

- Second is the negative request where username is empty



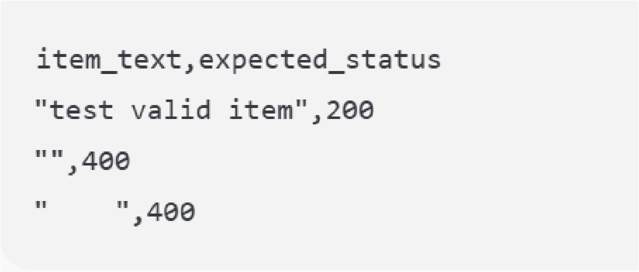
```
pm.test("Status code is 400", function () {
    pm.response.to.have.status(400);
});

pm.test("Should return error message for missing fields", function () {
    const res = pm.response.json();
    pm.expect(res.message).to.include("Username and password are required");
});
```

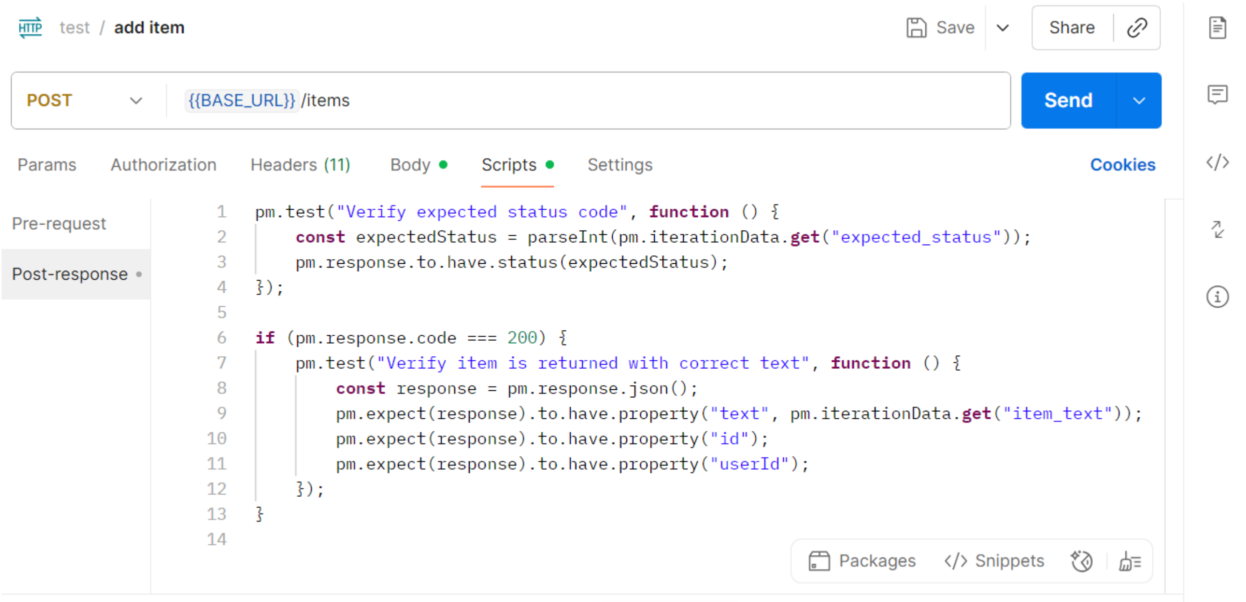
PASSED Status code is 400

PASSED Should return error message for missing fields

  - Multiple more combinations can be done where username is empty , password is empty , both are empty , providing user that does not exist etc.

- Add item for existing user (Includes positive and negative tests)
  - Test data
    - 

```
item_text,expected_status
"test valid item",200
"",400
"    ",400
```

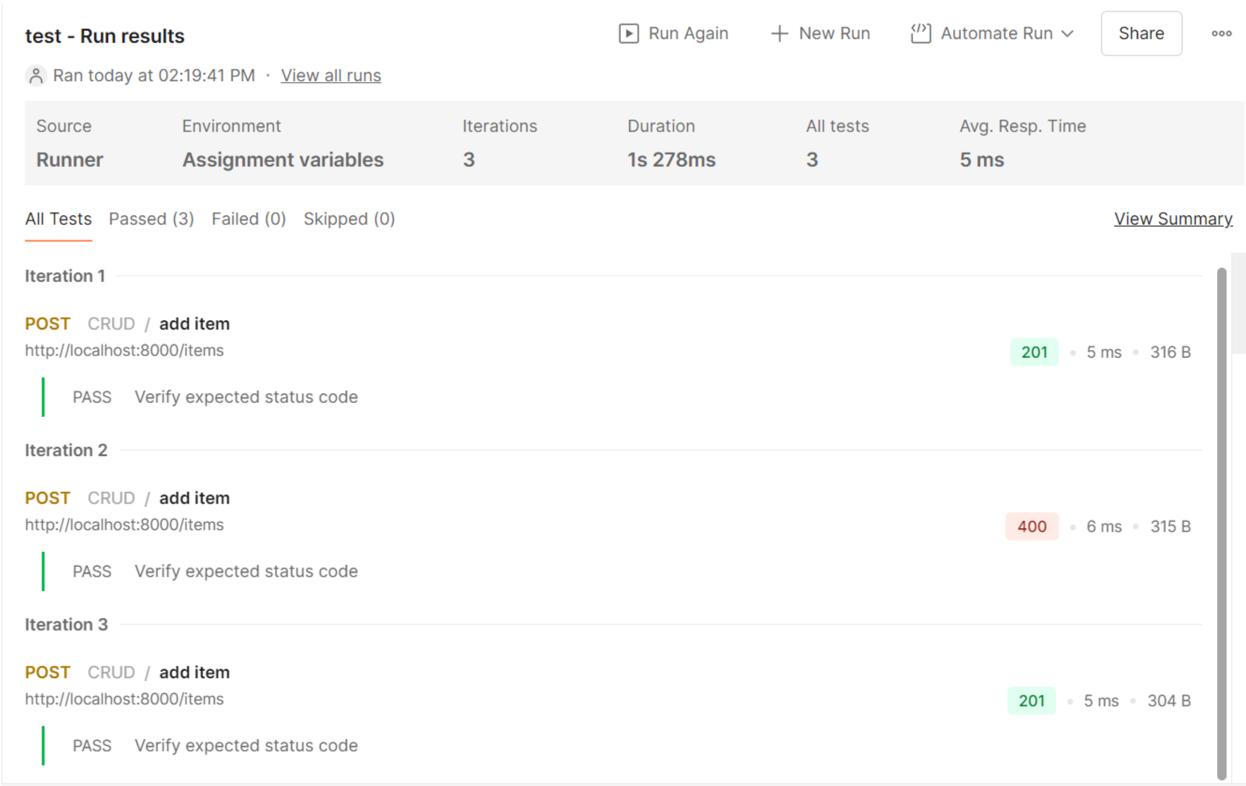


`POST` `{{BASE_URL}} /items`

```
pm.test("Verify expected status code", function () {
    const expectedStatus = parseInt(pm.iterationData.get("expected_status"));
    pm.response.to.have.status(expectedStatus);
});

if (pm.response.code === 200) {
    pm.test("Verify item is returned with correct text", function () {
        const response = pm.response.json();
        pm.expect(response).to.have.property("text", pm.iterationData.get("item_text"));
        pm.expect(response).to.have.property("id");
        pm.expect(response).to.have.property("userId");
    });
}

```

- ○ 

test - Run results

Ran today at 02:19:41 PM · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Assignment variables	3	1s 278ms	3	5 ms

All Tests   Passed (3)   Failed (0)   Skipped (0)

Iteration 1

POST CRUD / add item
http://localhost:8000/items   201 • 5 ms • 316 B

PASS   Verify expected status code

Iteration 2

POST CRUD / add item
http://localhost:8000/items   400 • 6 ms • 315 B

PASS   Verify expected status code

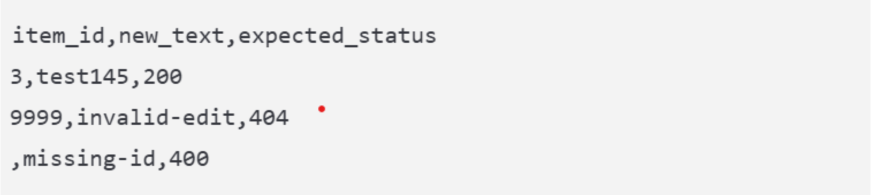
Iteration 3

POST CRUD / add item
http://localhost:8000/items   201 • 5 ms • 304 B

PASS   Verify expected status code

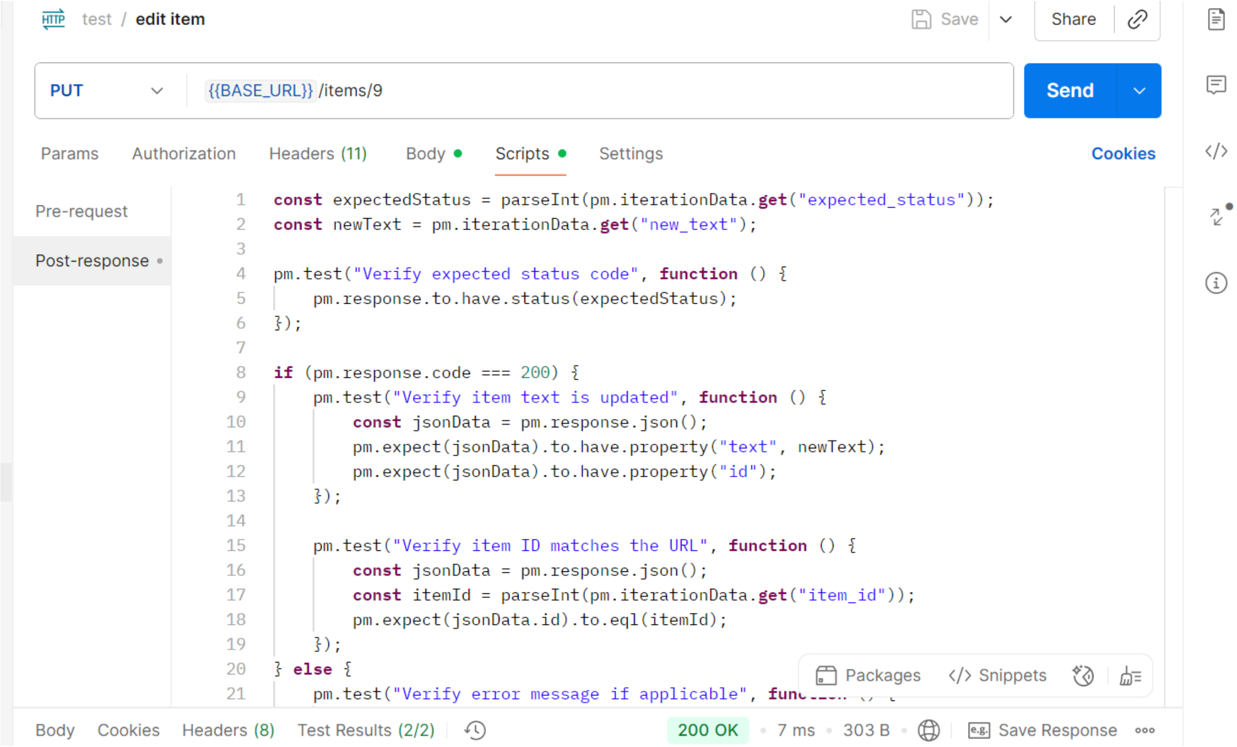
- ○

- Edit item (Added positive flows screenshots) (The collection has positive and negative testcases included)
  - Test data used



```
item_id,new_text,expected_status
3,test145,200
9999,invalid-edit,404
,missing-id,400
```

■



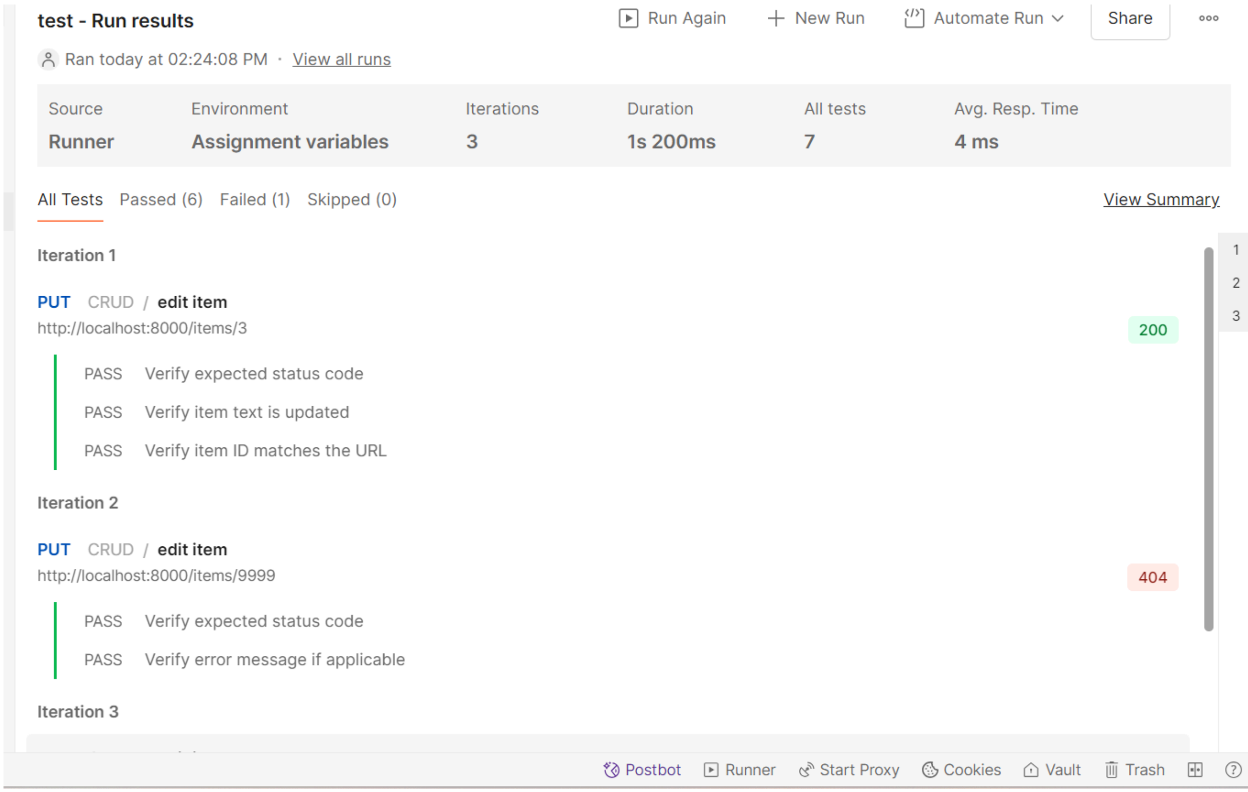
```
const expectedStatus = parseInt(pm.iterationData.get("expected_status"));
const newText = pm.iterationData.get("new_text");

pm.test("Verify expected status code", function () {
    pm.response.to.have.status(expectedStatus);
});

if (pm.response.code === 200) {
    pm.test("Verify item text is updated", function () {
        const jsonData = pm.response.json();
        pm.expect(jsonData).to.have.property("text", newText);
        pm.expect(jsonData).to.have.property("id");
    });

    pm.test("Verify item ID matches the URL", function () {
        const jsonData = pm.response.json();
        const itemId = parseInt(pm.iterationData.get("item_id"));
        pm.expect(jsonData.id).to.eql(itemId);
    });
} else {
    pm.test("Verify error message if applicable", func
```

○



test - Run results

Ran today at 02:24:08 PM · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Assignment variables	3	1s 200ms	7	4 ms

All Tests Passed (6) Failed (1) Skipped (0) View Summary

Iteration 1

PUT CRUD / edit item
http://localhost:8000/items/3 200

PASS Verify expected status code
PASS Verify item text is updated
PASS Verify item ID matches the URL

Iteration 2

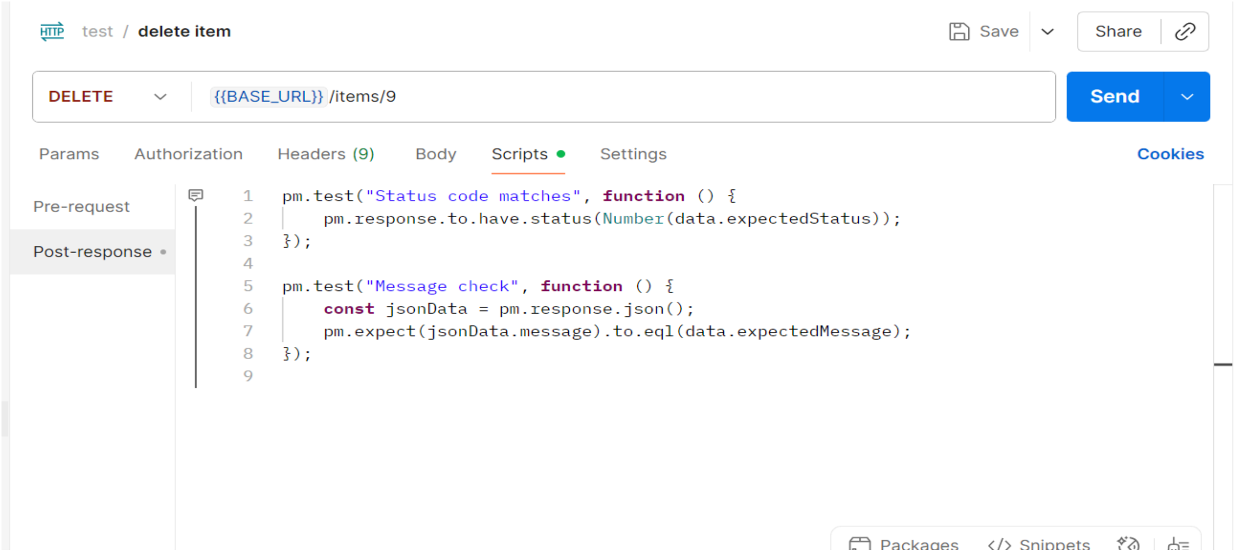
PUT CRUD / edit item
http://localhost:8000/items/9999 404

PASS Verify expected status code
PASS Verify error message if applicable

Iteration 3

- Delete item
  - The testdata file looks like

```
item_id,expectedStatus,expectedMessage
3,200,Item deleted successfully
9999,404,Item not found
```

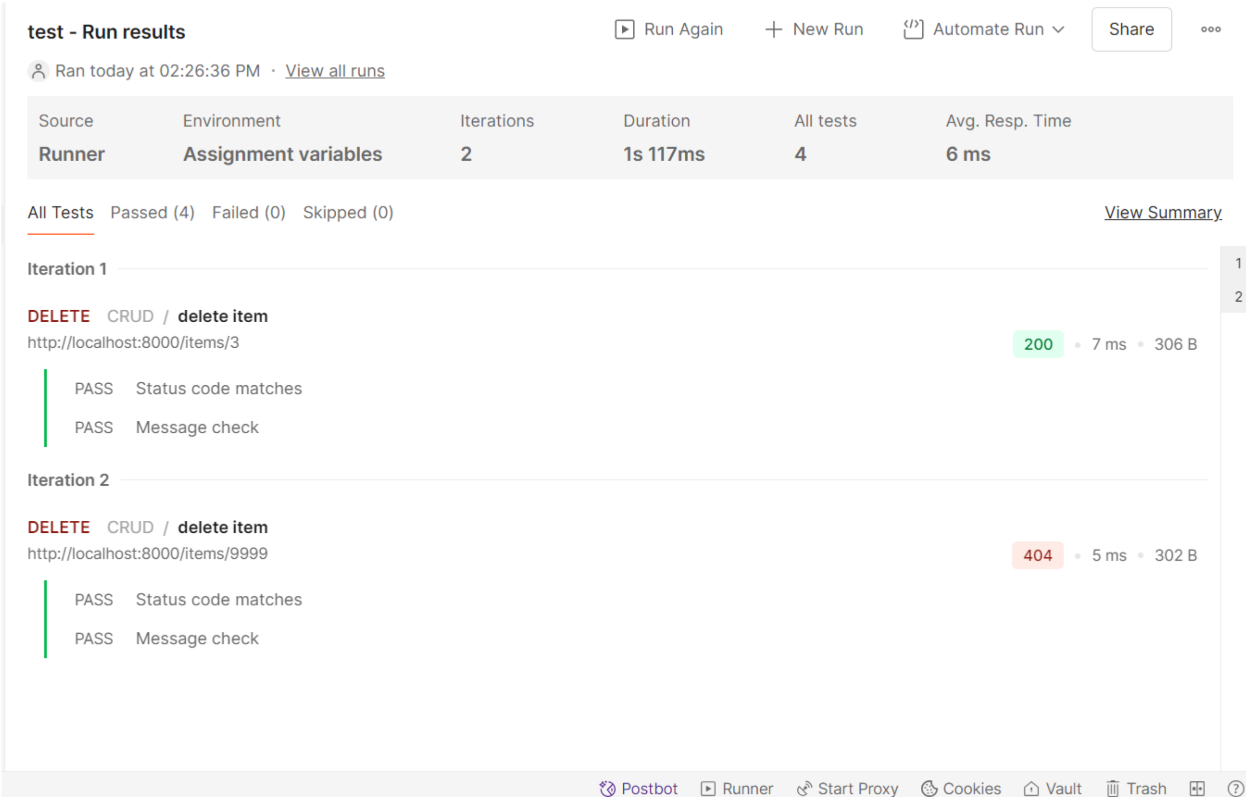


test / delete item

DELETE {{BASE_URL}} /items/9

```
pm.test("Status code matches", function () {
    pm.response.to.have.status(Number(data.expectedStatus));
});

pm.test("Message check", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql(data.expectedMessage);
});
```



test - Run results

Ran today at 02:26:36 PM · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Assignment variables	2	1s 117ms	4	6 ms

All Tests Passed (4) Failed (0) Skipped (0) View Summary

Iteration 1

DELETE CRUD / delete item
http://localhost:8000/items/3 — 200 • 7 ms • 306 B

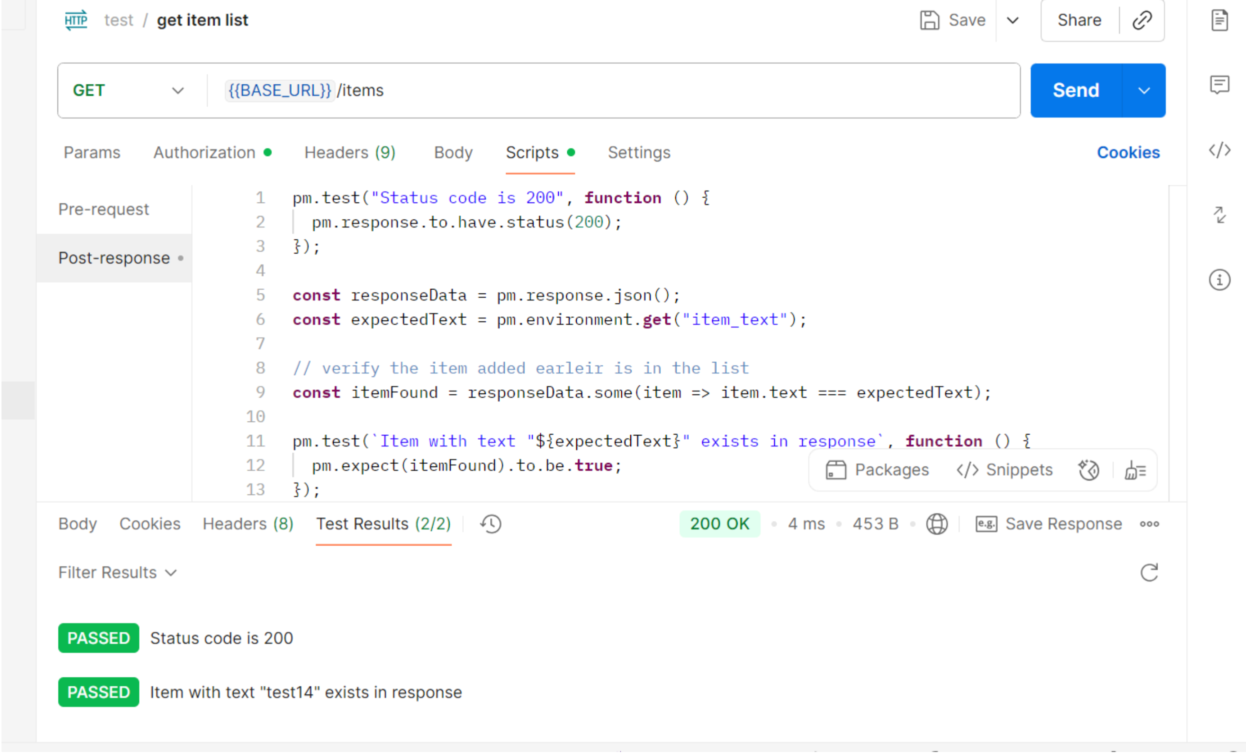
PASS Status code matches
PASS Message check

Iteration 2

DELETE CRUD / delete item
http://localhost:8000/items/9999 — 404 • 5 ms • 302 B

PASS Status code matches
PASS Message check

- Get item list



test / get item list

GET {{BASE_URL}} /items

Params Authorization Headers (9) Body Scripts Settings Cookies

Post-response

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});

const responseData = pm.response.json();
const expectedText = pm.environment.get("item_text");

// verify the item added earleir is in the list
const itemFound = responseData.some(item => item.text === expectedText);

pm.test(`Item with text "${expectedText}" exists in response`, function () {
  pm.expect(itemFound).to.be.true;
});
```

Body Cookies Headers (8) Test Results (2/2) — 200 OK • 4 ms • 453 B

Filter Results

PASSED Status code is 200

PASSED Item with text "test14" exists in response