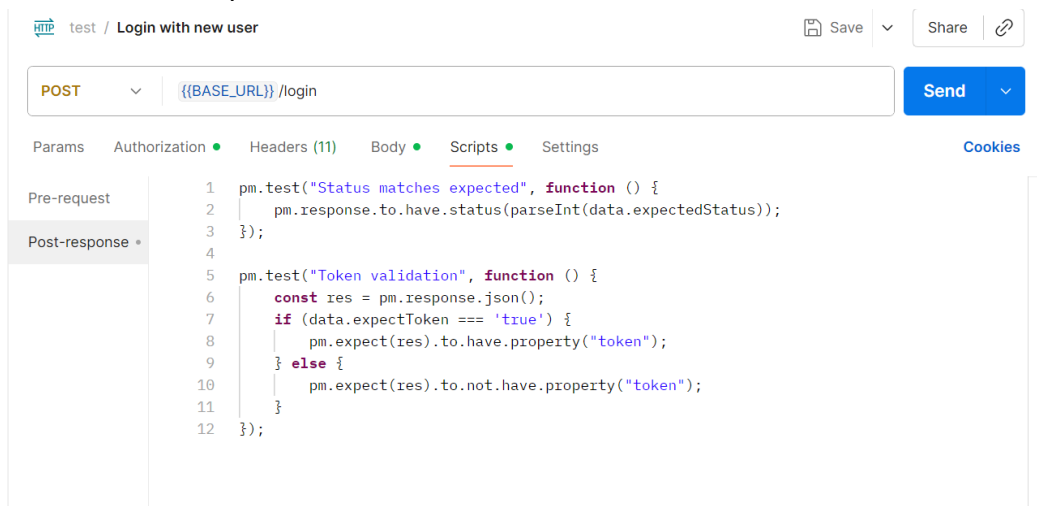


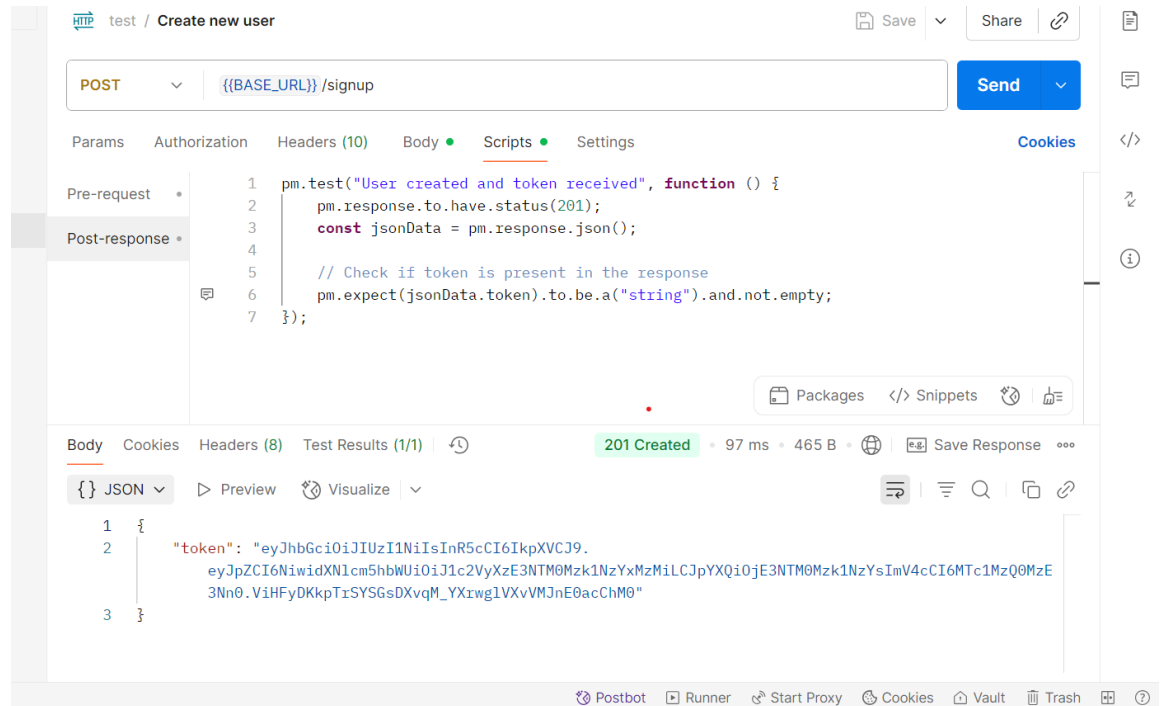
## API test Automation using Postman

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

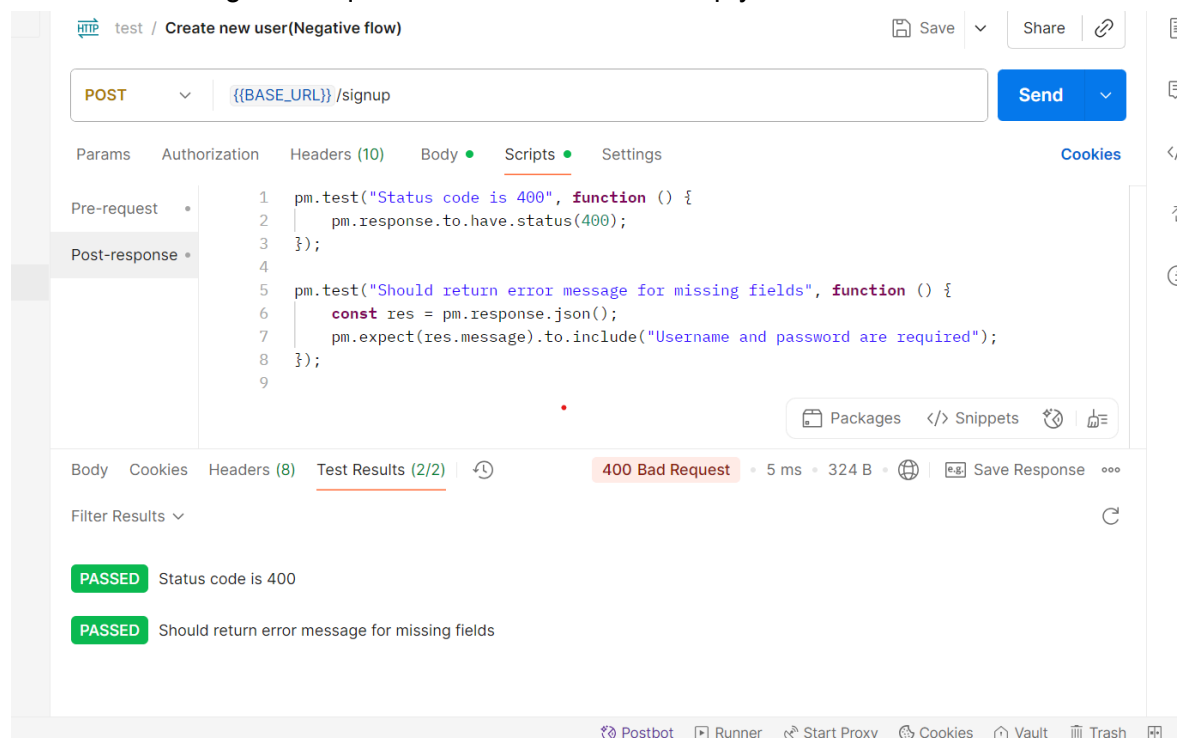
- Login testflow with positive and negative testcases
  - Made the testcase parameterize with giving .csv file
  - Csv file data
    - username,password,expectedStatus,expectToken
    - validuser,correctpass,200,true
    - invalid\_user,wrong\_password,401,false
    - existing\_user,anotherpass,409,false
    - ,blankpass,400,false
    - Userwithnpass,,400,false



- 
- 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



- 
- Second is the negative request where username is empty



- 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
```

test / add item

POST {{BASE\_URL}}/items

Send

Params Authorization Headers (11) Body Scripts Settings Cookies

Pre-request

Post-response

```
1 pm.test("Verify expected status code", function () {
2   const expectedStatus = parseInt(pm.iterationData.get("expected_status"));
3   pm.response.to.have.status(expectedStatus);
4 });
5
6 if (pm.response.code === 200) {
7   pm.test("Verify item is returned with correct text", function () {
8     const response = pm.response.json();
9     pm.expect(response).to.have.property("text", pm.iterationData.get("item_text"));
10    pm.expect(response).to.have.property("id");
11    pm.expect(response).to.have.property("userId");
12  });
13 }
14
```

Packages Snippets

- 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
```

test / edit item

PUT {{BASE\_URL}}/items/9

Send

Params Authorization Headers (11) Body Scripts Settings Cookies

Pre-request

Post-response \*

```

1  const expectedStatus = parseInt(pm.iterationData.get("expected_status"));
2  const newText = pm.iterationData.get("new_text");
3
4  pm.test("Verify expected status code", function () {
5      pm.response.to.have.status(expectedStatus);
6  });
7
8  if (pm.response.code === 200) {
9      pm.test("Verify item text is updated", function () {
10         const jsonData = pm.response.json();
11         pm.expect(jsonData).to.have.property("text", newText);
12         pm.expect(jsonData).to.have.property("id");
13     });
14
15     pm.test("Verify item ID matches the URL", function () {
16         const jsonData = pm.response.json();
17         const itemId = parseInt(pm.iterationData.get("item_id"));
18         pm.expect(jsonData.id).to.eql(itemId);
19     });
20 } else {
21     pm.test("Verify error message if applicable", function () {

```

200 OK • 7 ms • 303 B • Save Response

- 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

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Pre-request

Post-response \*

```

1  pm.test("Status code matches", function () {
2      pm.response.to.have.status(Number(data.expectedStatus));
3  });
4
5  pm.test("Message check", function () {
6      const jsonData = pm.response.json();
7      pm.expect(jsonData.message).to.eql(data.expectedMessage);
8  });
9

```

Packages Snippets

- Get item list

HTTPtest / get item list

Save

Share

GET

{{BASE\_URL}}/items

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

Cookies

Pre-request

Post-response

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 const responseData = pm.response.json();
6 const expectedText = pm.environment.get("item_text");
7
8 // verify the item added earleir is in the list
9 const itemFound = responseData.some(item => item.text === expectedText);
10
11 pm.test('Item with text "${expectedText}" exists in response', function () {
12   pm.expect(itemFound).to.be.true;
13 });
```

Body

Cookies

Headers (8)

Test Results (2/2)

200 OK

4 ms

453 B

Save Response

Filter Results

PASSED

Status code is 200

PASSED

Item with text "test14" exists in response

O