

Unit Testing

Frameworks used: *Jest and Supertest*
Postman

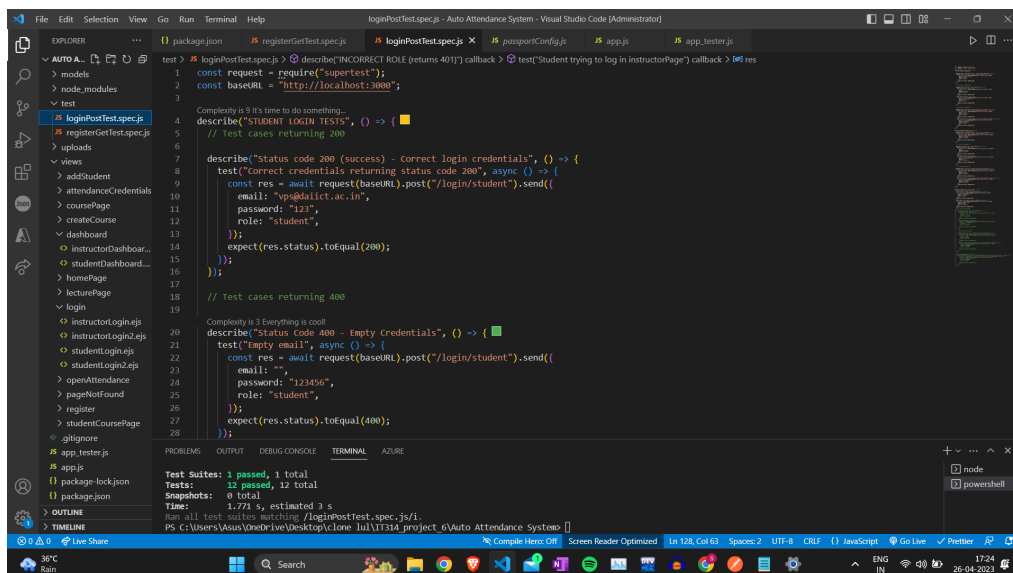
Login Unit Testing

The following testing module contains different scenarios for:

1. Instructor Login
2. Student Login

Code is saved as *loginPostTest.spec.js*

Code snippet:



```
loginPostTest.spec.js - Auto Attendance System - Visual Studio Code (Administrator)

1 const request = require('supertest');
2 const baseUrl = 'http://localhost:3000';
3
4 // Test cases returning 200
5
6 describe('STUDENT LOGIN TESTS', () => {
7   describe('Status code 200 (success) - correct login credentials', () => {
8     test('correct credentials returning status code 200', async () => {
9       const res = await request(baseUrl).post('/login/student').send({
10         email: 'vp@dsilict.ac.in',
11         password: '123',
12         role: 'student',
13       });
14       expect(res.status).toEqual(200);
15     });
16   });
17
18   // Test cases returning 400
19
20   describe('Status code 400 - empty credentials', () => {
21     test('empty email', async () => {
22       const res = await request(baseUrl).post('/login/student').send({
23         email: '',
24         password: '123456',
25         role: 'student',
26       });
27       expect(res.status).toEqual(400);
28     });
29   });
30 });

Test Suites: 1 passed, 1 total
Tests: 12 passed, 12 total
Snapshots: 0 total
Time: 1.771 s, estimated 3 s
Run all test suites watching /loginPostTest.spec.js/1
PS C:\Users\Asha\OneDrive\Desktop\clone_lu\IT314_project_6\Auto Attendance System >
```

Results:

```
PS C:\Users\Asus\OneDrive\Desktop\clone_lu\IT314_project_6\Auto Attendance System> npm test -- loginPostTest.spec.js

> express@1.0.0 test
> jest loginPostTest.spec.js

PASS test/loginPostTest.spec.js
  STUDENT LOGIN TESTS
    Status code 200 (success) - Correct login credentials
      ✓ Correct credentials returning status code 200 (178 ms)
    Status Code 400 - Empty Credentials
      ✓ Empty email (7 ms)
      ✓ Empty password (4 ms)
    Status code 401 - Unauthorized status code scenarios
      ✓ Invalid email format returns 401 unauthorized code (69 ms)
      ✓ Valid email but wrong password returns 401 unauthorized code (158 ms)
  INSTRUCTOR LOGIN TESTS
    Status code 200 (success) - Correct login credentials
      ✓ Correct credentials returning status code 200 (134 ms)
    Status Code 400 - Empty Credentials
      ✓ Empty email (4 ms)
      ✓ Empty password (5 ms)
    Status code 401 - Unauthorized status code scenarios
      ✓ Invalid email format returns 401 unauthorized code (59 ms)
      ✓ Valid email but wrong password returns 401 unauthorized code (153 ms)
  INCORRECT ROLE (returns 401)
    ✓ Instructor trying to log in studentPage (151 ms)
    ✓ Student trying to log in instructorPage (63 ms)

Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        1.771 s, estimated 3 s
Ran all test suites matching /loginPostTest.spec.js/i.
PS C:\Users\Asus\OneDrive\Desktop\clone_lu\IT314_project_6\Auto Attendance System>
```

The same is done for the register module.

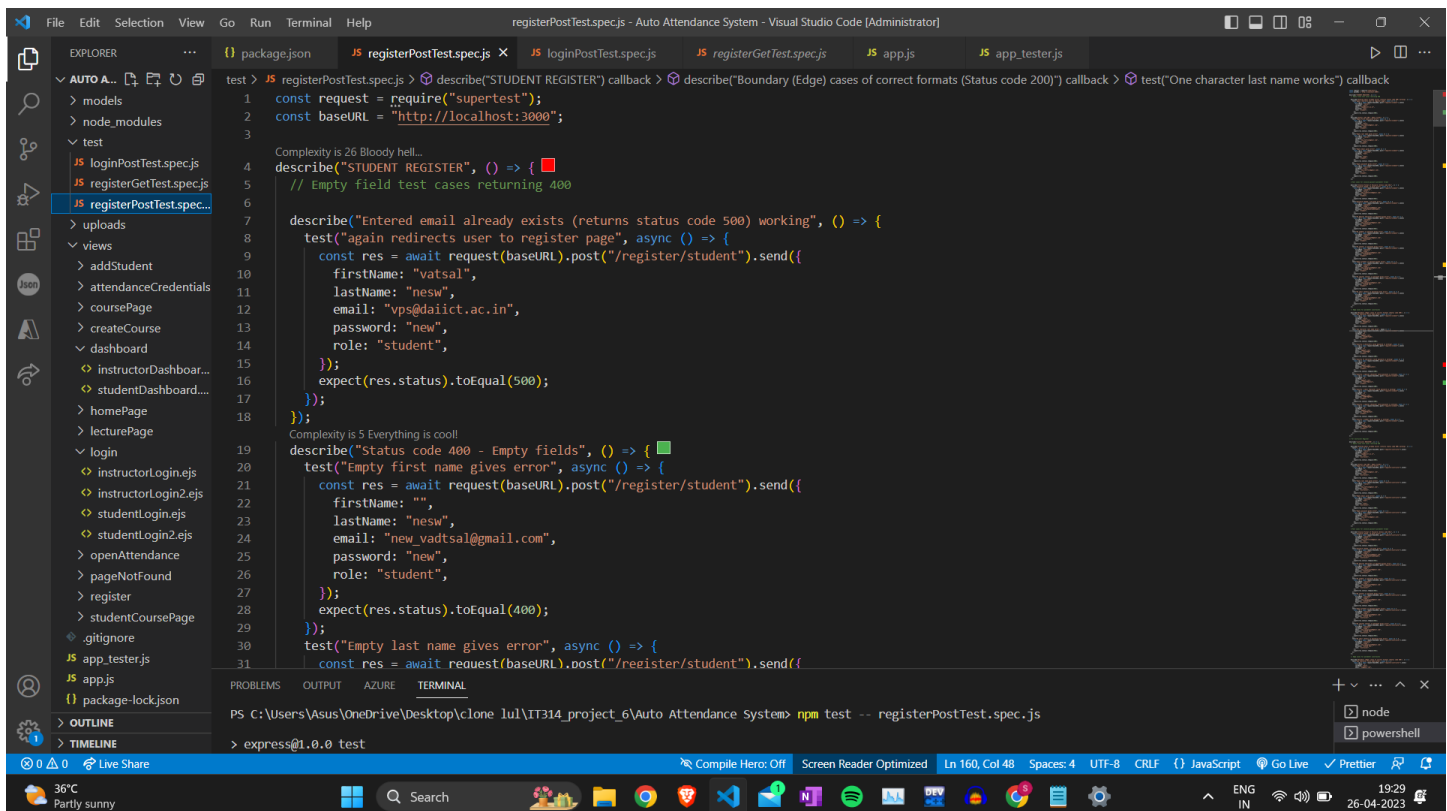
Register Unit Testing

The following testing module contains different scenarios for:

1. Instructor Register
2. Student Register

Code is saved as *registerPostTest.spec.js*

Code Snippet:



```
test > JS registerPostTest.spec.js > describe("STUDENT REGISTER") callback > describe("Boundary (Edge) cases of correct formats (Status code 200)") callback > test("One character last name works") callback

1  const request = require("supertest");
2  const baseUrl = "http://localhost:3000";
3
4  Complexity is 26 Bloody hell...
5  describe("STUDENT REGISTER", () => {
6    // Empty field test cases returning 400
7
8    describe("Entered email already exists (returns status code 500) working", () => {
9      test("again redirects user to register page", async () => {
10         const res = await request(baseUrl).post("/register/student").send({
11           firstName: "vatsal",
12           lastName: "nesw",
13           email: "vps@daict.ac.in",
14           password: "new",
15           role: "student",
16         });
17         expect(res.status).toEqual(500);
18       });
19     });
20
21     Complexity is 5 Everything is cool!
22     describe("Status code 400 - Empty fields", () => {
23       test("Empty first name gives error", async () => {
24         const res = await request(baseUrl).post("/register/student").send({
25           firstName: "",
26           lastName: "nesw",
27           email: "new_vadtsal@gmail.com",
28           password: "new",
29           role: "student",
30         });
31         expect(res.status).toEqual(400);
32       });
33       test("Empty last name gives error", async () => {
34         const res = await request(baseUrl).post("/register/student").send({
35           firstName: "vatsal",
36           lastName: "",
37           email: "new_vadtsal@gmail.com",
38           password: "new",
39           role: "student",
40         });
41         expect(res.status).toEqual(400);
42       });
43     });
44   });
```

PS C:\Users\Asus\OneDrive\Desktop\clone lul\IT314_project_6\Auto Attendance System> npm test -- registerPostTest.spec.js

> express@1.0.0 test

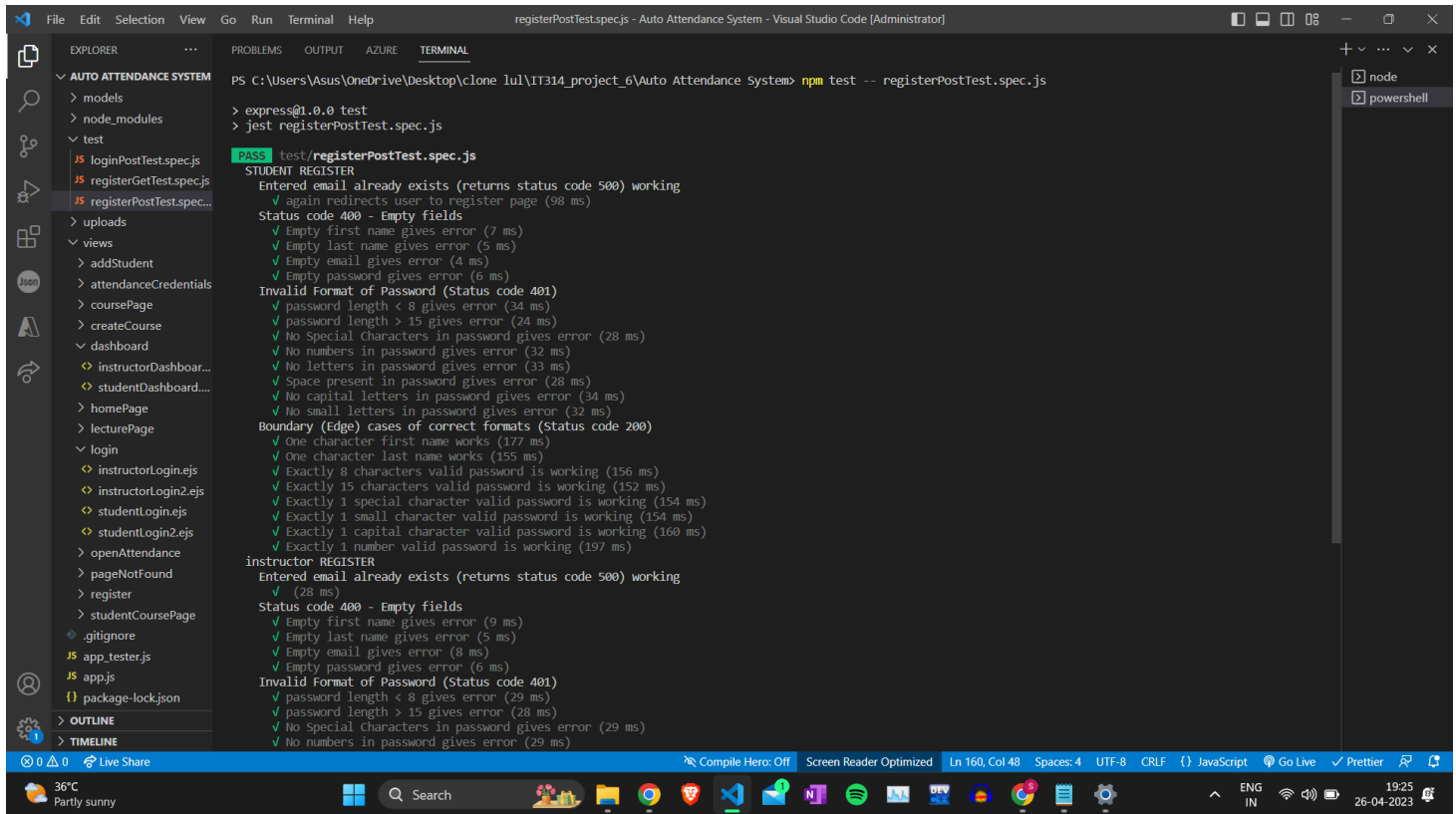
We took all kinds of scenarios possible and generated 42 test cases and hard coded them using *Jest and Supertest*.

Also entered the expected output for each test cases/scenarios and if the output matches the expected output then the test case will be shown as

✓passed

The results are shown below in screenshots.

Result after running 42 test cases



The screenshot displays the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal shows the command `npm test -- registerPostTest.spec.js` and its output. The output is organized into sections for 'STUDENT REGISTER' and 'instructor REGISTER'. Each section lists various test cases with their status (e.g., 'PASS', 'FAIL') and execution time. The 'STUDENT REGISTER' section includes tests for email existence, redirects, empty fields, and password formats. The 'instructor REGISTER' section includes tests for email existence, empty fields, and password formats. The terminal output shows that all 42 test cases passed.

```
PS C:\Users\Asus\OneDrive\Desktop\clone_lu1\IT314_project_6\Auto Attendance System> npm test -- registerPostTest.spec.js
> express@1.0.0 test
> jest registerPostTest.spec.js

PASS test/registerPostTest.spec.js
STUDENT REGISTER
  Entered email already exists (returns status code 500) working
    ✓ again redirects user to register page (98 ms)
  Status code 400 - Empty fields
    ✓ Empty first name gives error (7 ms)
    ✓ Empty last name gives error (5 ms)
    ✓ Empty email gives error (4 ms)
    ✓ Empty password gives error (6 ms)
  Invalid Format of Password (Status code 401)
    ✓ password length < 8 gives error (34 ms)
    ✓ password length > 15 gives error (24 ms)
    ✓ No Special Characters in password gives error (28 ms)
    ✓ No numbers in password gives error (32 ms)
    ✓ No letters in password gives error (33 ms)
    ✓ Space present in password gives error (28 ms)
    ✓ No capital letters in password gives error (34 ms)
    ✓ No small letters in password gives error (32 ms)
  Boundary (Edge) cases of correct formats (Status code 200)
    ✓ One character first name works (177 ms)
    ✓ One character last name works (155 ms)
    ✓ Exactly 8 characters valid password is working (156 ms)
    ✓ Exactly 15 characters valid password is working (152 ms)
    ✓ Exactly 1 special character valid password is working (154 ms)
    ✓ Exactly 1 small character valid password is working (154 ms)
    ✓ Exactly 1 capital character valid password is working (160 ms)
    ✓ Exactly 1 number valid password is working (197 ms)
instructor REGISTER
  Entered email already exists (returns status code 500) working
    ✓ (28 ms)
  Status code 400 - Empty fields
    ✓ Empty first name gives error (9 ms)
    ✓ Empty last name gives error (5 ms)
    ✓ Empty email gives error (8 ms)
    ✓ Empty password gives error (6 ms)
  Invalid Format of Password (Status code 401)
    ✓ password length < 8 gives error (29 ms)
    ✓ password length > 15 gives error (28 ms)
    ✓ No Special Characters in password gives error (29 ms)
    ✓ No numbers in password gives error (29 ms)
```

```
registerPostTest.spec.js - Auto Attendance System - Visual Studio Code [Administrator]

EXPLORER
├── AUTO ATTENDANCE SYSTEM
│   ├── models
│   ├── node_modules
│   ├── test
│   │   ├── loginPostTest.spec.js
│   │   ├── registerGetTest.spec.js
│   │   └── registerPostTest.spec.js
│   ├── uploads
│   ├── views
│   │   ├── addStudent
│   │   ├── attendanceCredentials
│   │   ├── coursePage
│   │   ├── createCourse
│   │   ├── dashboard
│   │   ├── instructorDashboard...
│   │   ├── studentDashboard...
│   │   ├── homePage
│   │   ├── lecturePage
│   │   ├── login
│   │   ├── instructorLogin.ejs
│   │   ├── instructorLogin2.ejs
│   │   ├── studentLogin.ejs
│   │   ├── studentLogin2.ejs
│   │   ├── openAttendance
│   │   ├── pageNotFound
│   │   ├── register
│   │   ├── studentCoursePage
│   │   ├── .gitignore
│   │   ├── app_tester.js
│   │   ├── app.js
│   │   └── package-lock.json
│   ├── OUTLINE
│   └── TIMELINE
├── .gitignore
├── app_tester.js
├── app.js
├── package-lock.json
├── OUTLINE
└── TIMELINE

TERMINAL
✓ Space present in password gives error (28 ms)
✓ No capital letters in password gives error (34 ms)
✓ No small letters in password gives error (32 ms)
Boundary (Edge) cases of correct formats (Status code 200)
✓ One character first name works (177 ms)
✓ One character last name works (155 ms)
✓ Exactly 8 characters valid password is working (156 ms)
✓ Exactly 15 characters valid password is working (152 ms)
✓ Exactly 1 special character valid password is working (154 ms)
✓ Exactly 1 small character valid password is working (154 ms)
✓ Exactly 1 capital character valid password is working (160 ms)
✓ Exactly 1 number valid password is working (197 ms)
instructor REGISTER
Entered email already exists (returns status code 500) working
(28 ms)
Status code 400 - Empty fields
✓ Empty first name gives error (9 ms)
✓ Empty last name gives error (5 ms)
✓ Empty email gives error (8 ms)
✓ Empty password gives error (6 ms)
Invalid Format of Password (Status code 401)
✓ password length < 8 gives error (29 ms)
✓ password length > 15 gives error (28 ms)
✓ No Special Characters in password gives error (29 ms)
✓ No numbers in password gives error (29 ms)
✓ No letters in password gives error (32 ms)
✓ Space present in password gives error (27 ms)
✓ No capital letters in password gives error (31 ms)
✓ No small letters in password gives error (24 ms)
Boundary (Edge) cases of correct formats (Status code 200)
✓ One character first name works (148 ms)
✓ One character last name works (185 ms)
✓ Exactly 8 characters valid password is working (168 ms)
✓ Exactly 15 characters valid password is working (176 ms)
✓ Exactly 1 special character valid password is working (139 ms)
✓ Exactly 1 small character valid password is working (153 ms)
✓ Exactly 1 capital character valid password is working (178 ms)
✓ Exactly 1 number valid password is working (139 ms)
Test Suites: 1 passed, 1 total
Tests: 42 passed, 42 total
Snapshots: 0 total
Time: 4.288 s, estimated 5 s
Ran all test suites matching /registerPostTest.spec.js/i.
PS C:\Users\Asus\OneDrive\Desktop\clone lul\IT314_project_6\Auto Attendance System>
```

```
✓ One character last name works (185 ms)
✓ Exactly 8 characters valid password is working (168 ms)
✓ Exactly 15 characters valid password is working (176 ms)
✓ Exactly 1 special character valid password is working (139 ms)
✓ Exactly 1 small character valid password is working (153 ms)
✓ Exactly 1 capital character valid password is working (178 ms)
✓ Exactly 1 number valid password is working (139 ms)

Test Suites: 1 passed, 1 total
Tests: 42 passed, 42 total
Snapshots: 0 total
Time: 4.288 s, estimated 5 s
Ran all test suites matching /registerPostTest.spec.js/i.
PS C:\Users\Asus\OneDrive\Desktop\clone lul\IT314_project_6\Auto Attendance System>
```

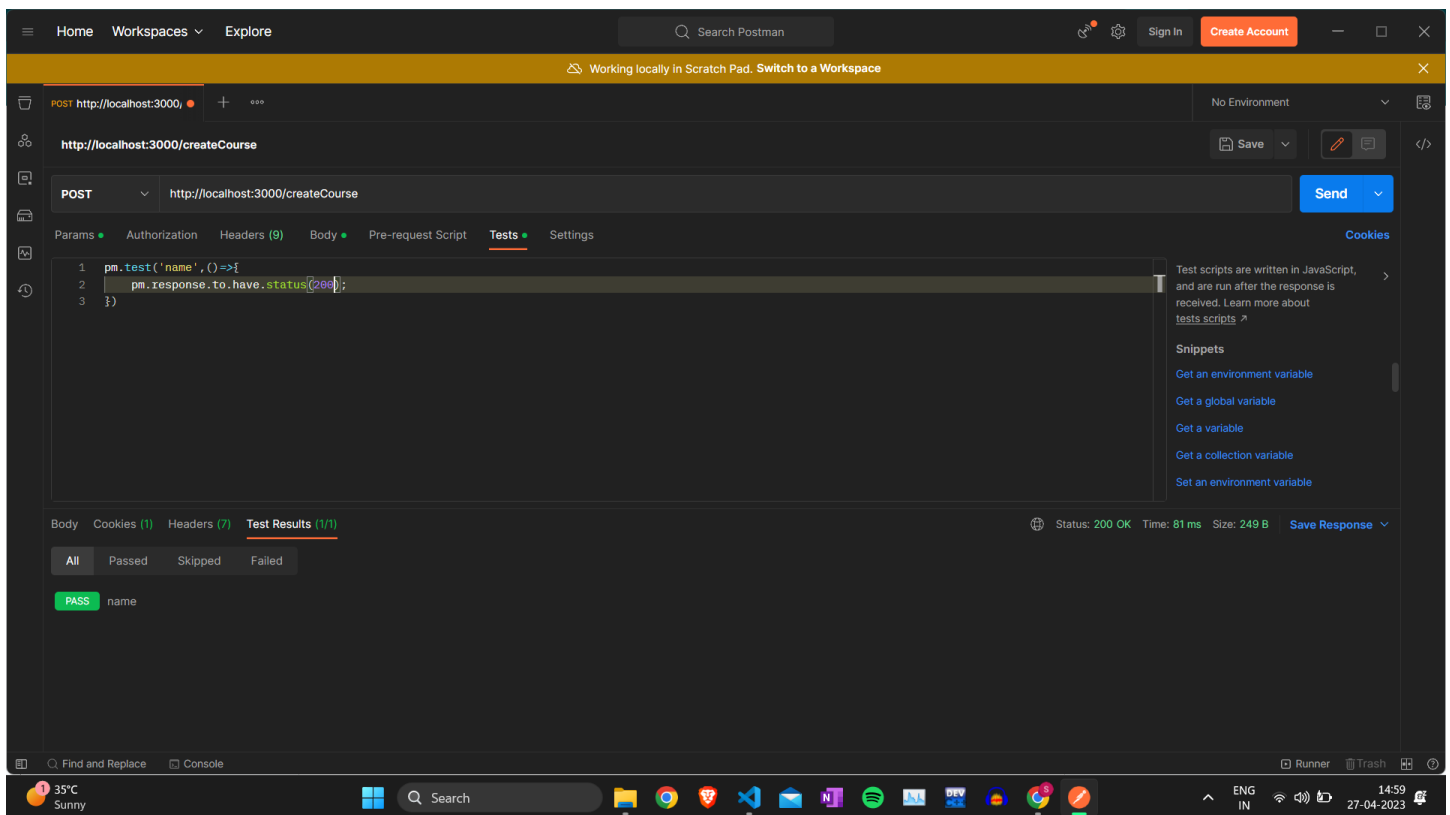
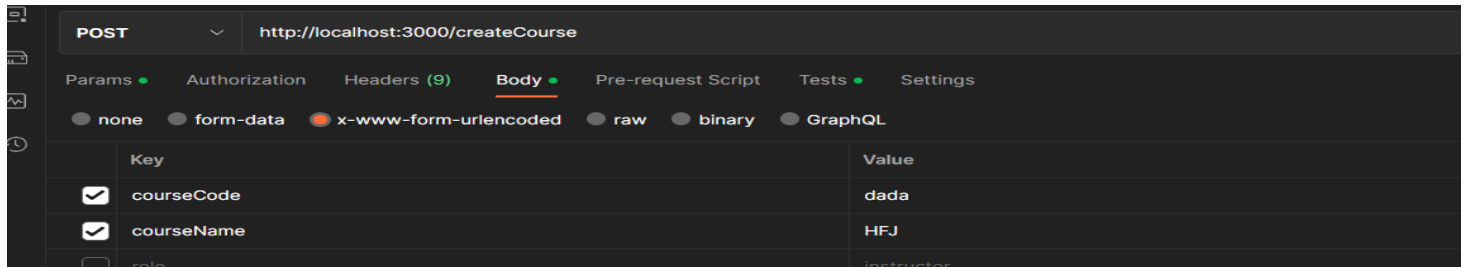
The next two modules are tested in *Postman* because the terminal will not store the data for login and it was a tedious code to implement the scenario in *Jest and Supertest*.

Testing for Create Course

Test Case 1:

courseCode: valid

courseName: valid



Test case2:

courseCode: empty

courseName: valid

Key	Value
<input checked="" type="checkbox"/> courseCode	
<input checked="" type="checkbox"/> courseName	Distributed Systems
<input type="checkbox"/>

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/createCourse`. The request body is empty. The response status is `400 Bad Request` with the message `input fields can't be null`. The test scripts section shows the following code:

```
1 pm.test('name', () => {
2   pm.response.to.have.status(400);
3 })
```

The interface also shows the 'Tests' tab selected, and the 'Body' tab showing the response message. The status bar at the bottom indicates the response is a 400 Bad Request, took 37 ms, and is 263 B in size.

Test case 3:

When courseName is empty and courseCode is valid.

courseCode: valid

courseName: empty

POST http://localhost:3000/createCourse	
Params • Authorization Headers (9) Body • Pre-request Script Tests • Settings	
none form-data x-www-form-urlencoded raw binary GraphQL	
Key	Value
<input checked="" type="checkbox"/> courseCode	IT783
<input checked="" type="checkbox"/> courseName	

Home Workspaces Explore Search Postman Sign In Create Account

Working locally in Scratch Pad. Switch to a Workspace

POST http://localhost:3000/createCourse No Environment Save Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

```
1 pm.test('name', () => {
2   pm.response.to.have.status(400);
3 })
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about [tests scripts](#).

Snippets

[Get an environment variable](#)

[Get a global variable](#)

[Get a variable](#)

[Get a collection variable](#)

[Set an environment variable](#)

Body Cookies (1) Headers (7) Test Results (1/1) Status: 400 Bad Request Time: 31 ms Size: 263 B Save Response

Pretty Raw Preview Visualize

input fields can't be null

Find and Replace Console Runner Trash 35°C Sunny Search ENG IN 15:06 27-04-2023


```
courseCode: empty
courseName: empty
```

courseName: empty

POST

▼

http://localhost:3000/createCourse

Params • Authorization Headers (9) Body • Pre-request Script Tests • Settings

☐ none

☐ form-data

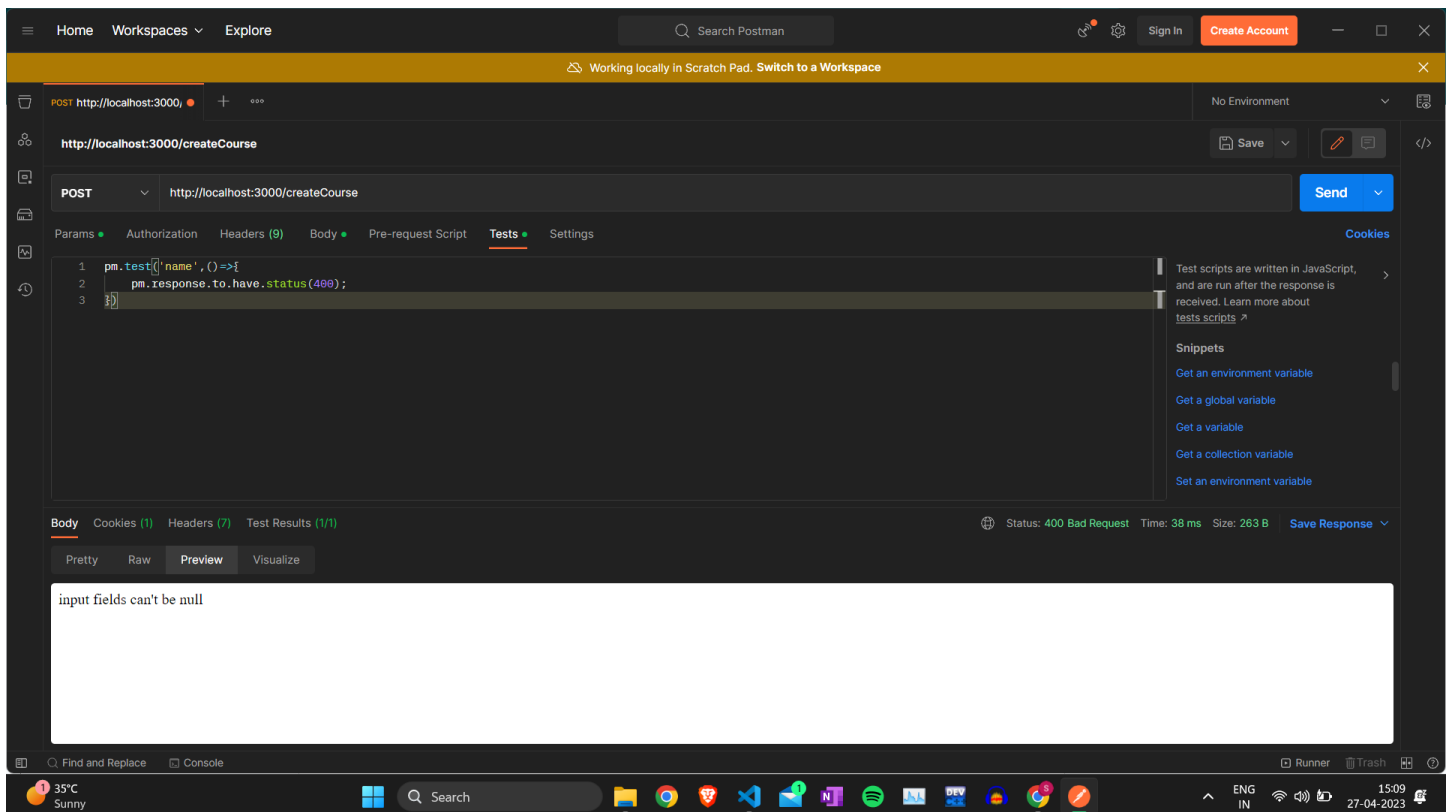
☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	courseCode	
<input checked="" type="checkbox"/>	courseName	



Testing for creating new Attendance

Test case 1:

lectureName: valid

Minutes: valid

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/openAttendance/644a435ab807a49c248606c3`. The request body is a JSON object with the following fields:

Key	Value	Description
<input type="checkbox"/> courseName		
<input type="checkbox"/> role	instructor	
<input type="checkbox"/> email	asdf@dailct.ac.in	
<input type="checkbox"/> password	123456	
<input checked="" type="checkbox"/> lectureName	Lecture 15	
<input checked="" type="checkbox"/> minutes	20	

The response status is 200 OK, with a time of 197 ms and a size of 264 B. The test results show a single test passing:

Test Results (1/1)
PASS name

The screenshot shows the 'Tests' tab in Postman with the following test script:

```
1 pm.test('name', () => {
2   pm.response.to.have.status(200);
3 })
```

Test case 2:
lectureName: empty
minutes: valid

POST http://localhost:3000/openAttendance/644a435ab807a49c248606c3

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description
courseName		
role	instructor	
email	asdf@daict.ac.in	
password	123456	
lectureName		
minutes	20	

Body Cookies (1) Headers (7) **Test Results (1/1)** Status: 400 Bad Request Time: 46 ms Size: 250 B Save Response

All Passed Skipped Failed

PASS name

POST http://localhost:3000/openAttendance/644a435ab807a49c248606c3

Params Authorization Headers (9) Body Pre-request Script **Tests** Settings

```
1 pm.test('name', () => {
2   pm.response.to.have.status(400);
3 })
```

Test case 3:
lectureName: valid
minutes : not valid

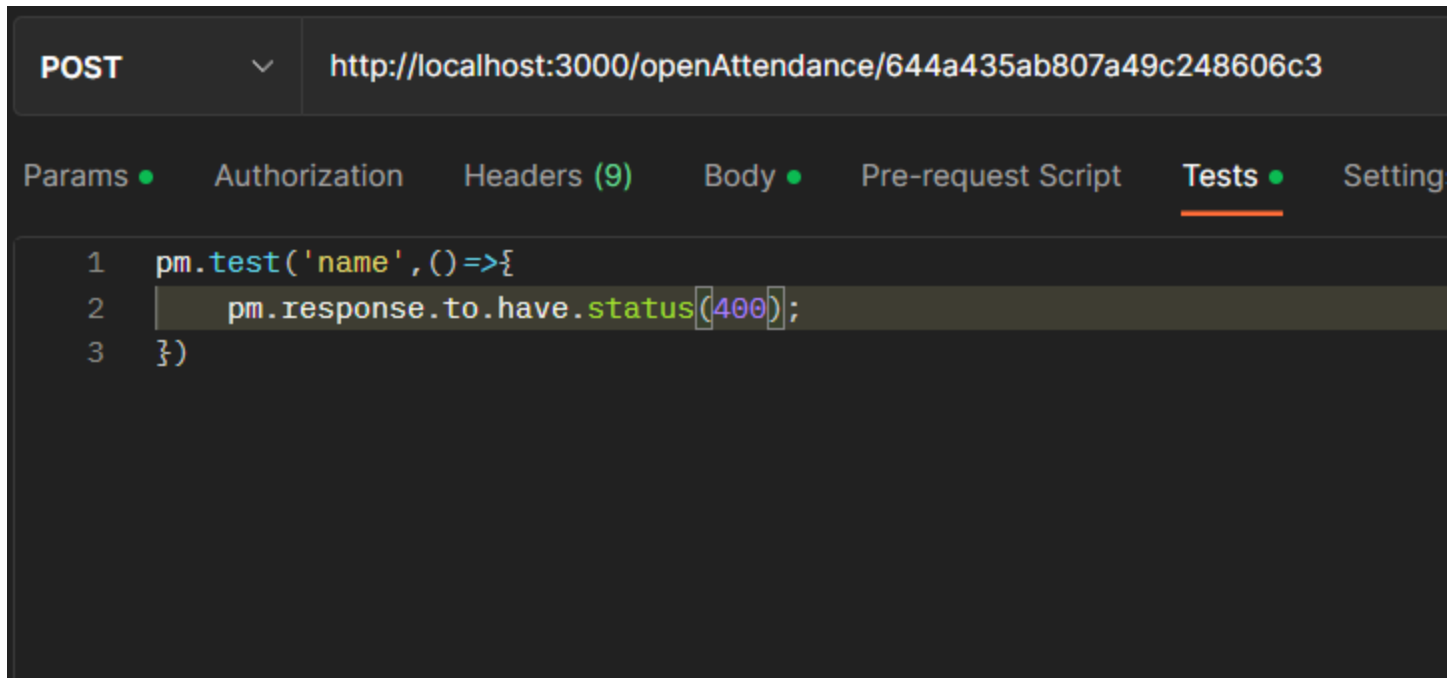
The screenshot shows the Postman interface with a POST request to `http://localhost:3000/openAttendance/644a435ab807a49c248606c3`. The request body is in the 'Body' tab, showing a JSON object with `lectureName` and `minutes` fields. The `minutes` field is highlighted in red, indicating it is the cause of the error. The response status is `400 Bad Request` with a time of `55 ms` and size of `250 B`. The test results show a `PASS` for the `name` test.

Key	Value	Description
<input type="checkbox"/> courseName		
<input type="checkbox"/> role	instructor	
<input type="checkbox"/> email	asdf@daict.ac.in	
<input type="checkbox"/> password	123456	
<input checked="" type="checkbox"/> lectureName	Lecture 1	
<input checked="" type="checkbox"/> minutes	2	

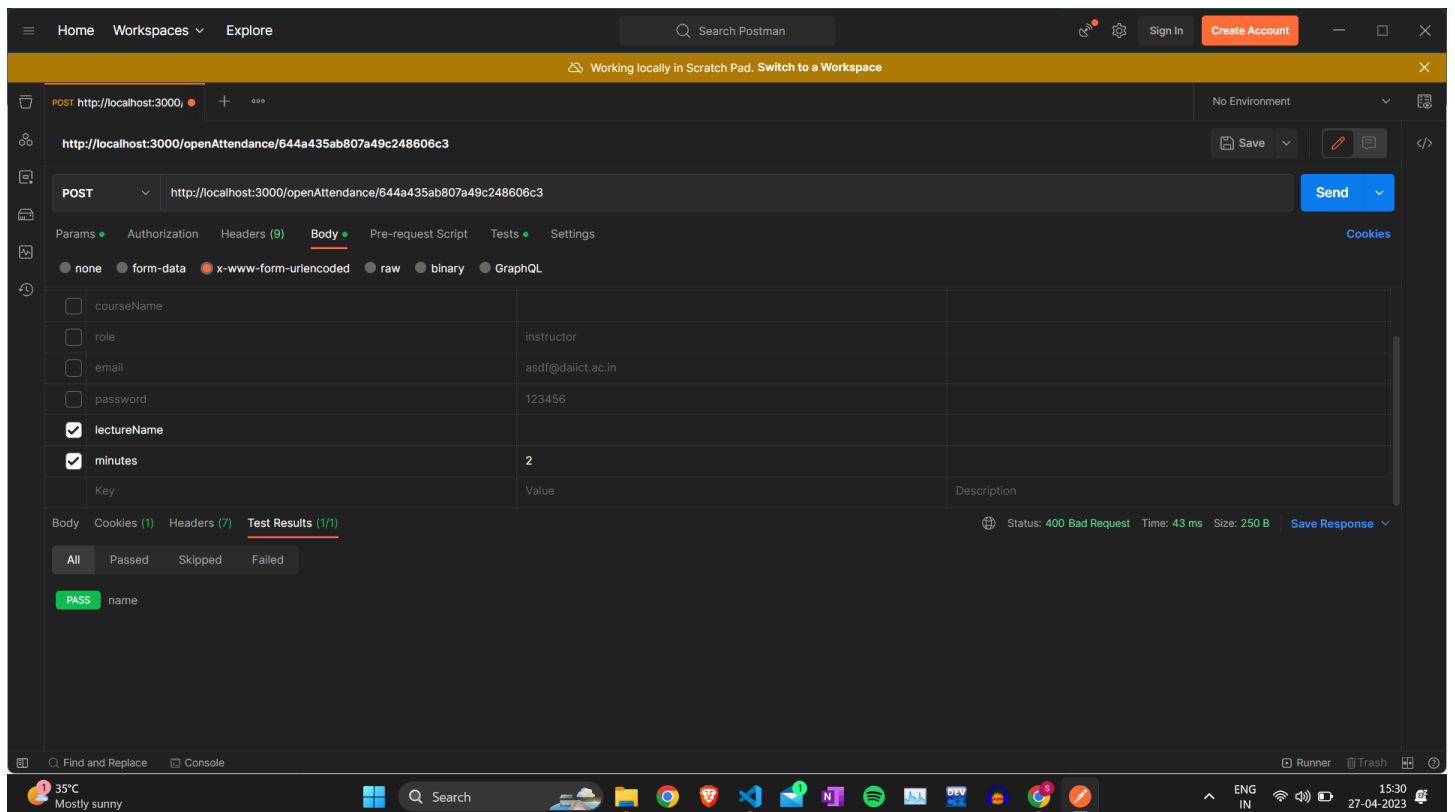
Body Cookies (1) Headers (7) **Test Results (1/1)** Status: 400 Bad Request Time: 55 ms Size: 250 B Save Response

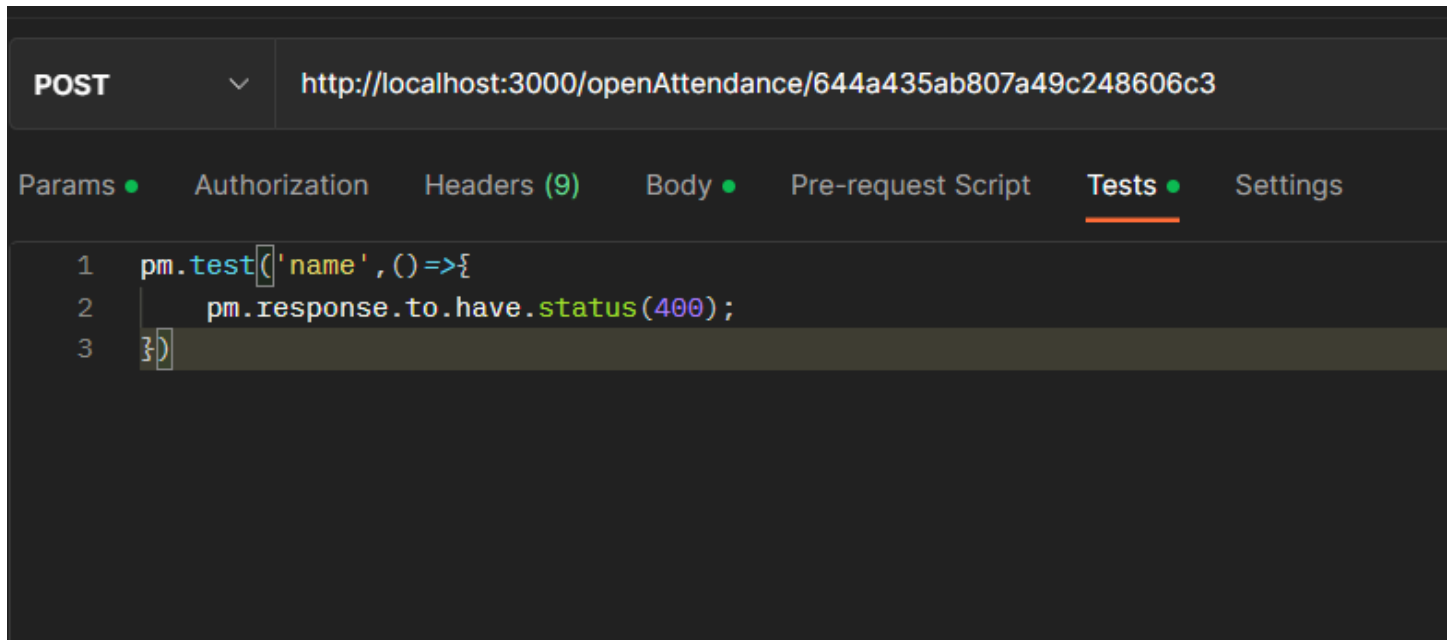
All Passed Skipped Failed

PASS name

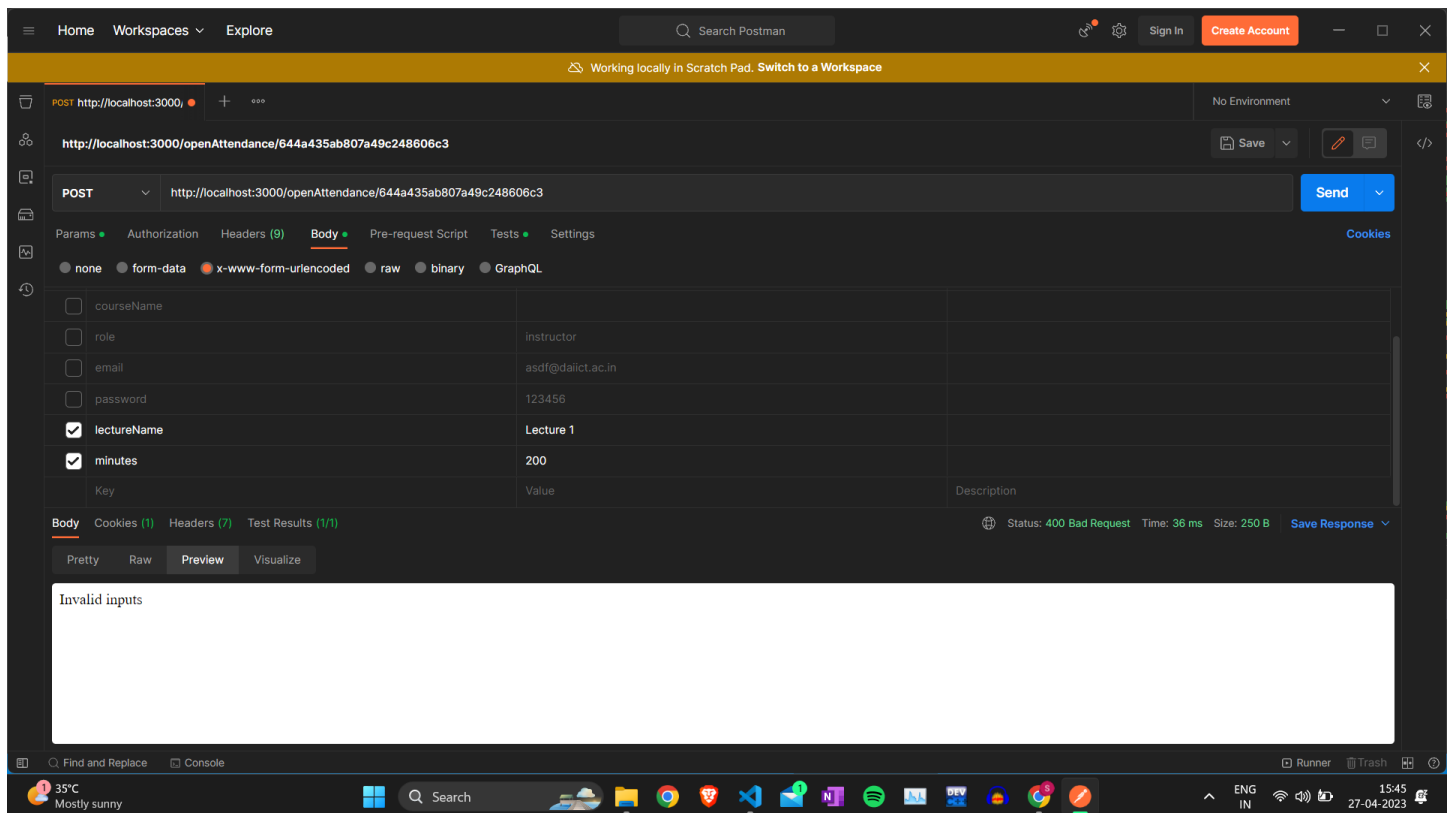


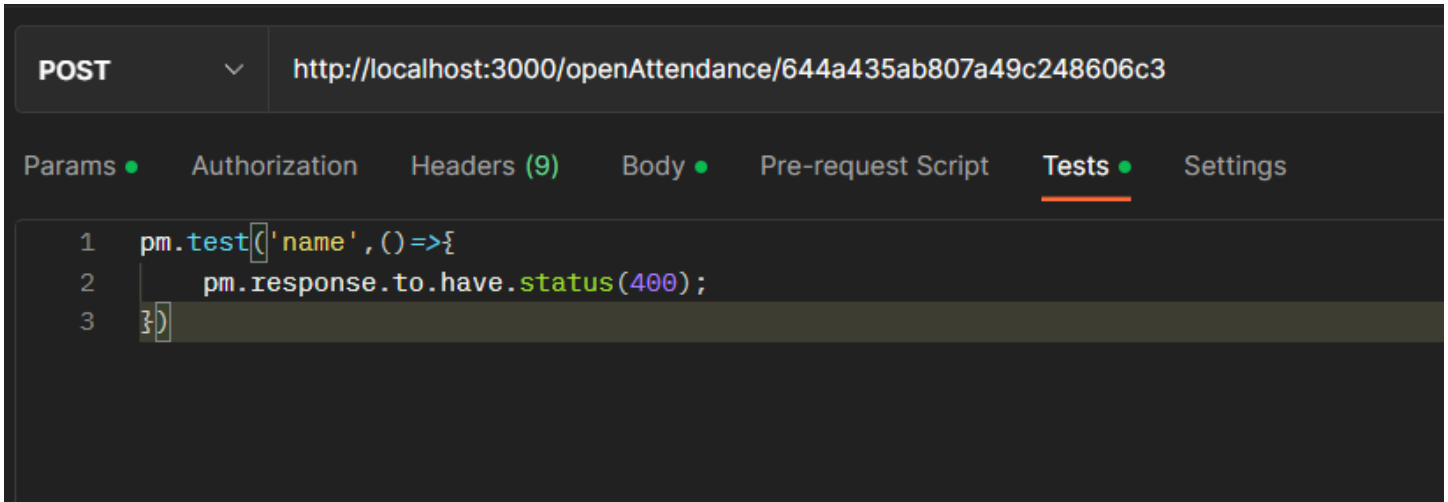
Test case 4:
lectureName: not valid
minutes: not valid





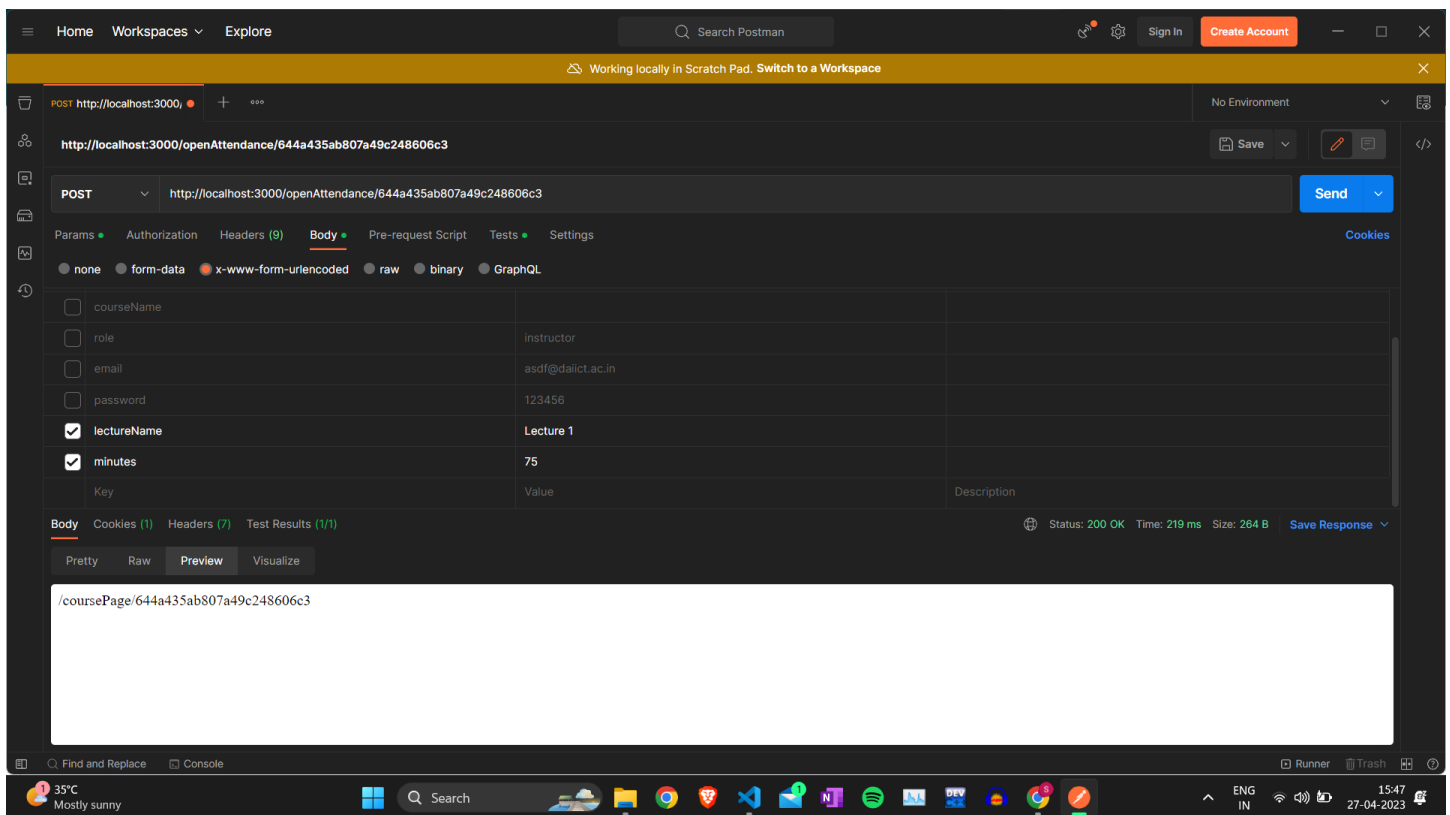
Test case 5:
lectureName: Valid
Minutes: >75

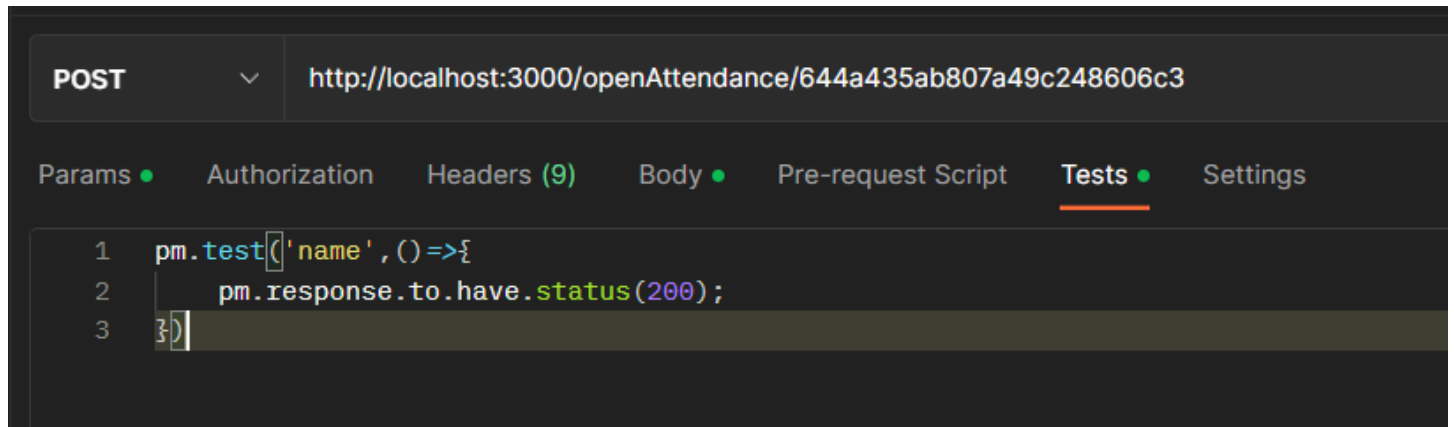




Test case 6:

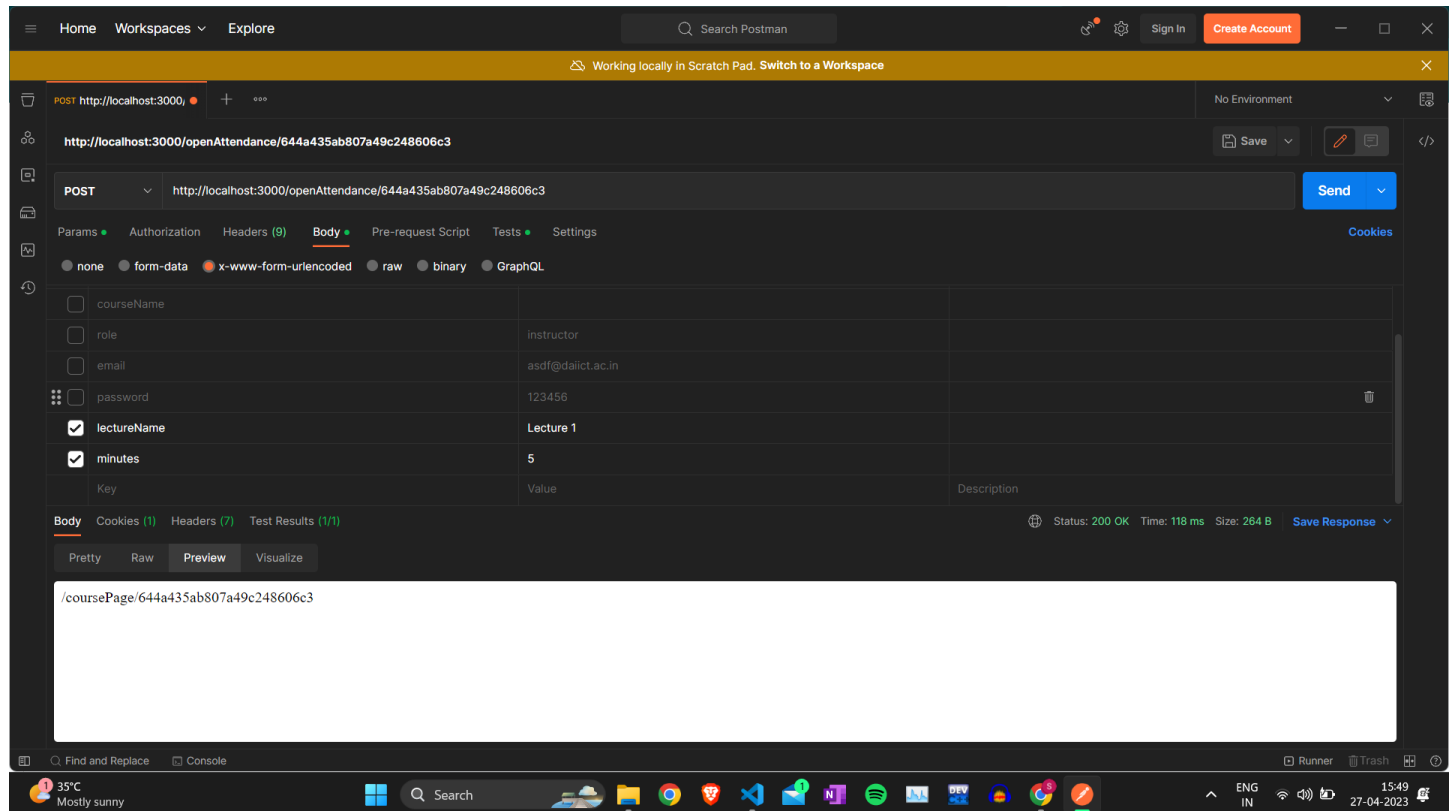
When lecture name is valid and minutes is exactly equal to 75.





Test 7:

When input is exactly equal to 5 and lecture name is valid.



POST



http://localhost:3000/openAttendance/644a435ab807a49c248606c3

Params ●

Authorization

Headers (9)

Body ●

Pre-request Script

Tests ●

Settings

```
1 pm.test('name', () => {  
2   pm.response.to.have.status(200);  
3 })
```