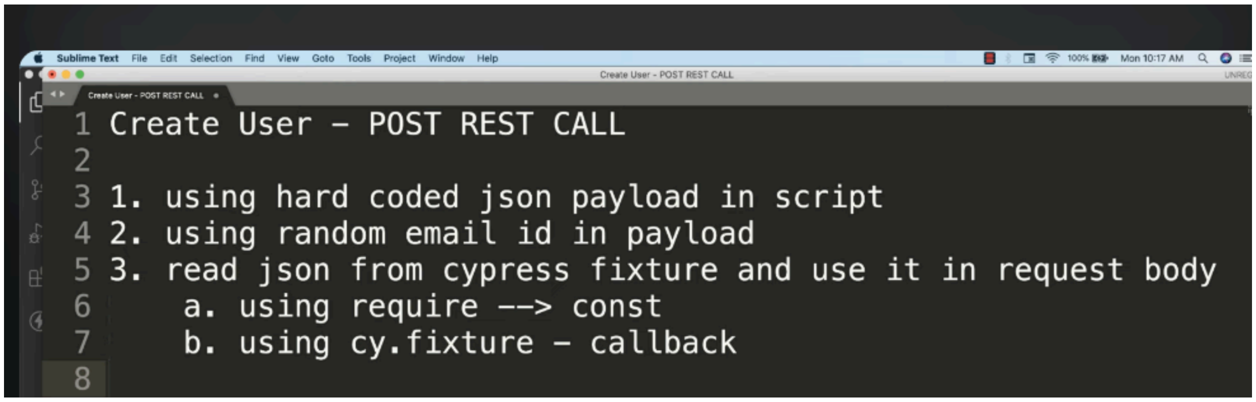
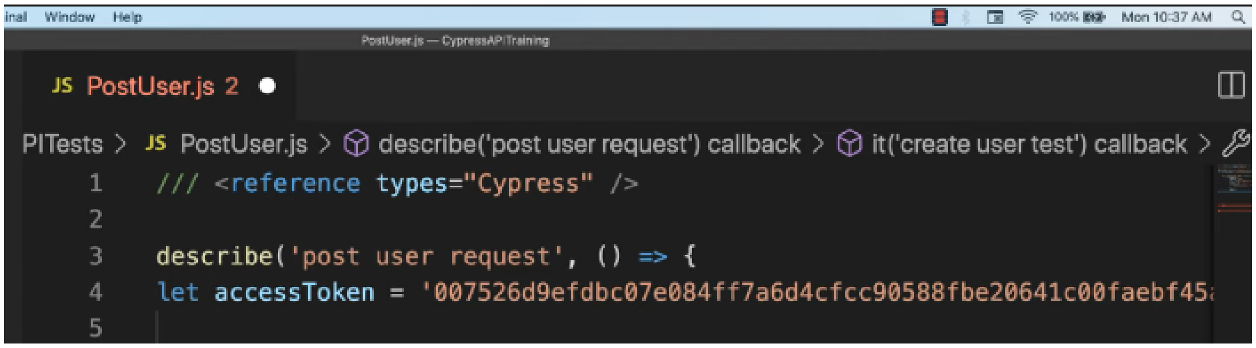
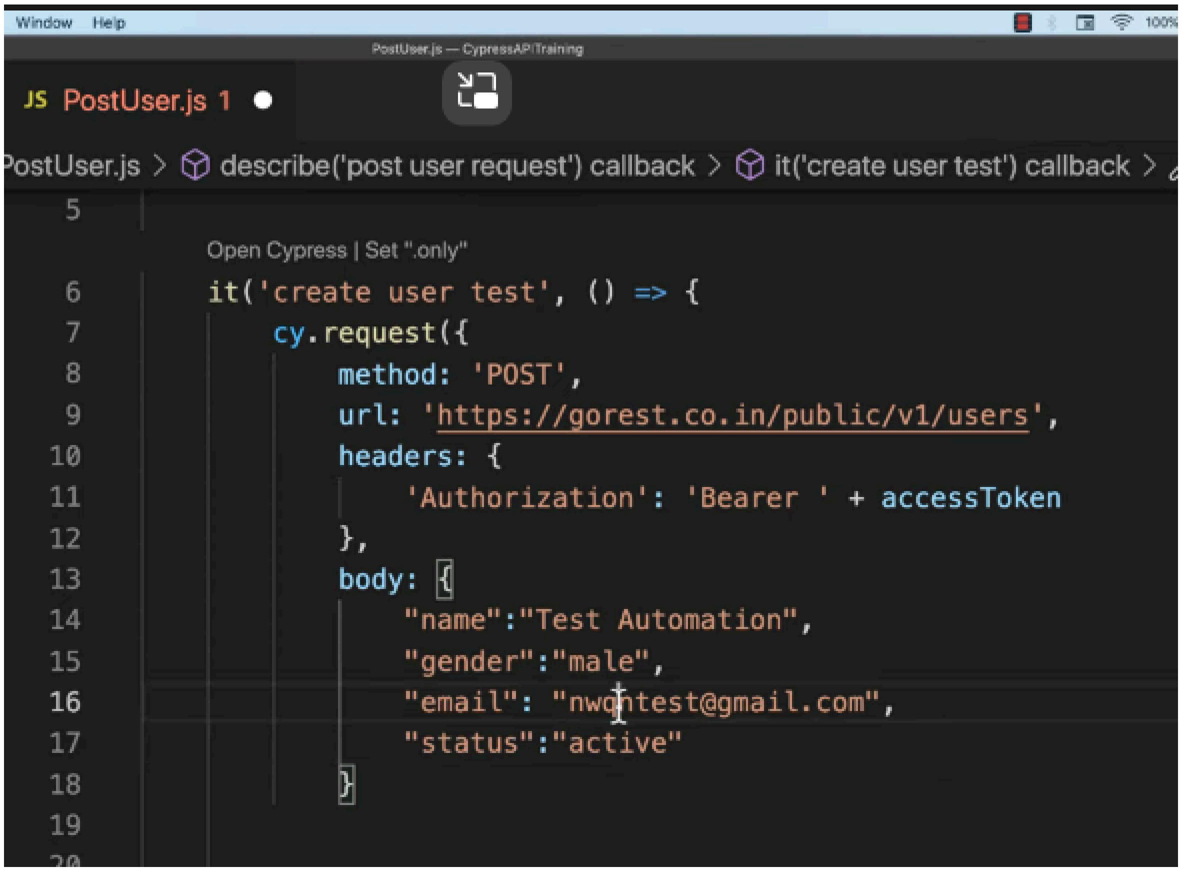
post



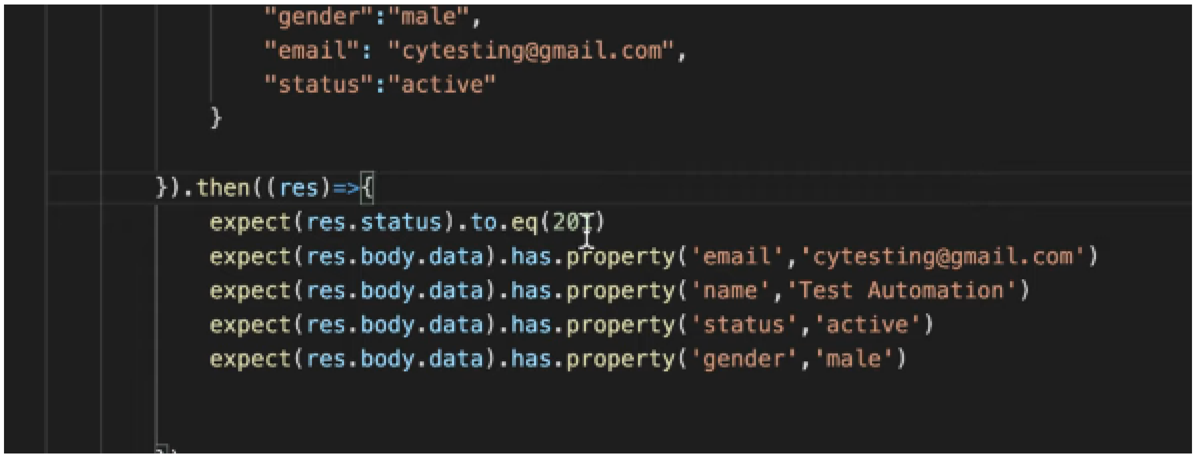
```
Create User - POST REST CALL

1. using hard coded json payload in script
2. using random email id in payload
3. read json from cypress fixture and use it in request body
    a. using require --> const
    b. using cy.fixture - callback
```



```
/// <reference types="Cypress" />

describe('post user request', () => {
let accessToken = '007526d9efdbc07e084ff7a6d4cfcc90588fbe20641c00faebf45
```

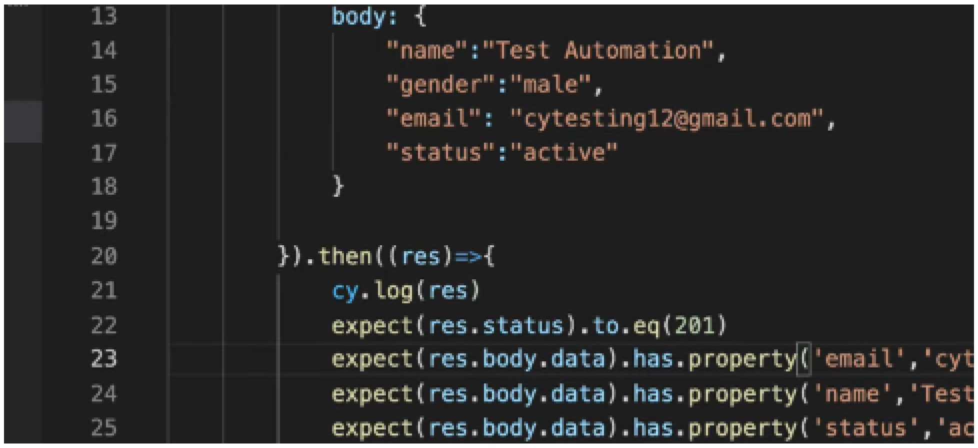


```
    it('create user test', () => {
        cy.request({
            method: 'POST',
            url: 'https://gorest.co.in/public/v1/users',
            headers: {
                'Authorization': 'Bearer ' + accessToken
            },
            body: {
                "name":"Test Automation",
                "gender":"male",
                "email": "nwqntest@gmail.com",
                "status":"active"
            }
```



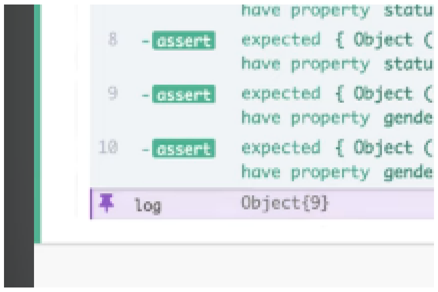
```
            "gender":"male",
            "email": "cytesting@gmail.com",
            "status":"active"
        }

    }).then((res)=>{
        expect(res.status).to.eq(201)
        expect(res.body.data).has.property('email','cytesting@gmail.com')
        expect(res.body.data).has.property('name','Test Automation')
        expect(res.body.data).has.property('status','active')
        expect(res.body.data).has.property('gender','male')
```

Added21 to log response



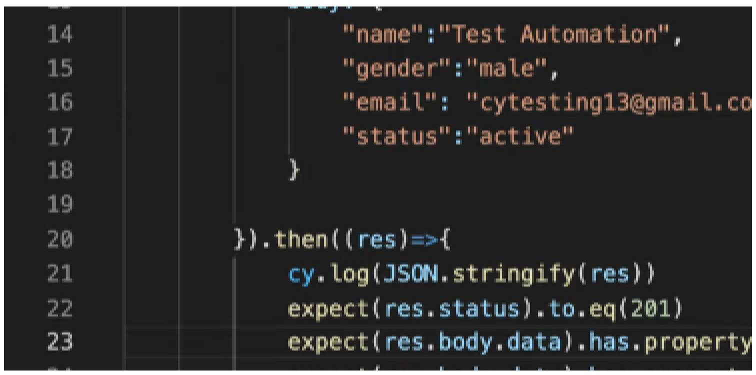
```
13                  body: {
14                      "name":"Test Automation",
15                      "gender":"male",
16                      "email": "cytesting12@gmail.com",
17                      "status":"active"
18                  }
19
20              }).then((res)=>{
21                  cy.log(res)
22                  expect(res.status).to.eq(201)
23                  expect(res.body.data).has.property('email','cyt
24                  expect(res.body.data).has.property('name','Test
25                  expect(res.body.data).has.property('status','ac
```

We get object in output.



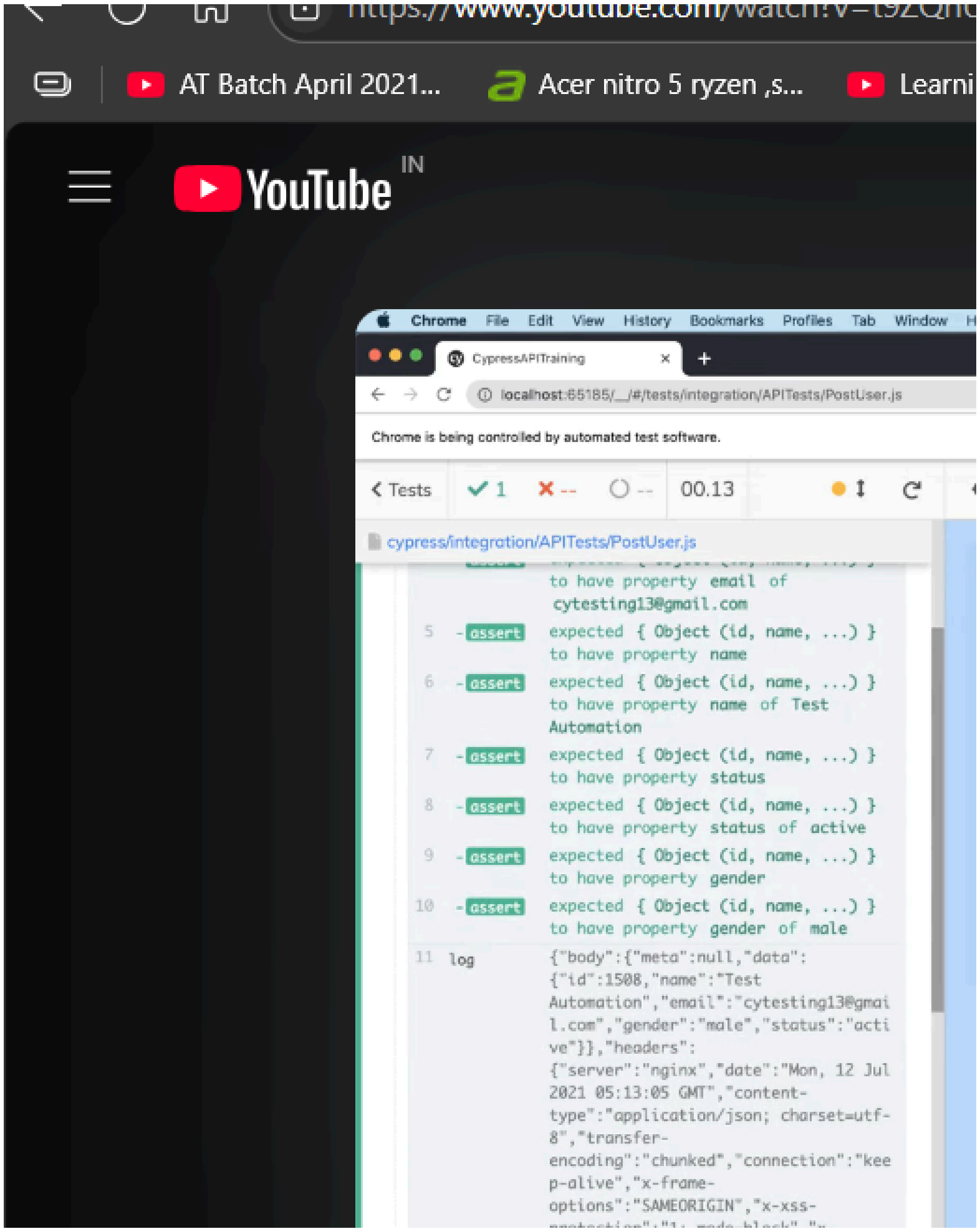
```
                have property statu
8  -assert      expected { Object (
                have property statu
9  -assert      expected { Object (
                have property gende
10 -assert      expected { Object (
                have property gende
   log          Object{9}
```

Updated21
Made to string form using stringify.



```
                "name":"Test Automation",
                "gender":"male",
                "email": "cytesting13@gmail.co
                "status":"active"
            }

        }).then((res)=>{
            cy.log(JSON.stringify(res))
            expect(res.status).to.eq(201)
            expect(res.body.data).has.property
```

Now nice response

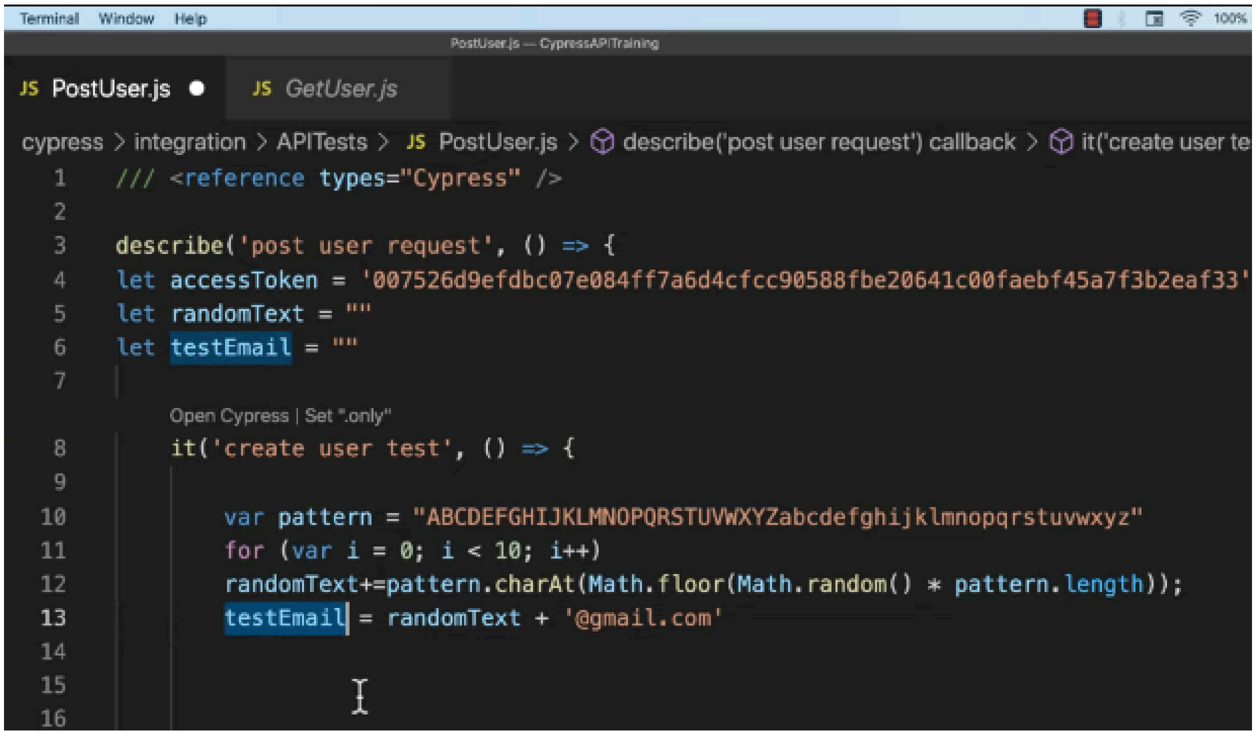

AT Batch April 2021...
Acer nitro 5 ryzen ,s...
YouTube
Chrome File Edit View History Bookmarks Profiles Tab Window
CypressAPITraining
localhost:65185/__/#/tests/integration/APITests/PostUser.js
Tests
00.13

Chrome is being controlled by automated test software.

cypress/integration/APITests/PostUser.js

```
                 to have property email of
                 cytesting13@gmail.com
5   -assert      expected { Object (id, name, ...) }
                 to have property name
6   -assert      expected { Object (id, name, ...) }
                 to have property name of Test
                 Automation
7   -assert      expected { Object (id, name, ...) }
                 to have property status
8   -assert      expected { Object (id, name, ...) }
                 to have property status of active
9   -assert      expected { Object (id, name, ...) }
                 to have property gender
10  -assert      expected { Object (id, name, ...) }
                 to have property gender of male
11  log          {"body":{"meta":null,"data":
                 {"id":1508,"name":"Test
                 Automation","email":"cytesting13@gmai
                 l.com","gender":"male","status":"acti
                 ve"}},"headers":
                 {"server":"nginx","date":"Mon, 12 Jul
                 2021 05:13:05 GMT","content-
                 type":"application/json; charset=utf-
                 8","transfer-
                 encoding":"chunked","connection":"kee
                 p-alive","x-frame-
                 options":"SAMEORIGIN","x-xss-
```

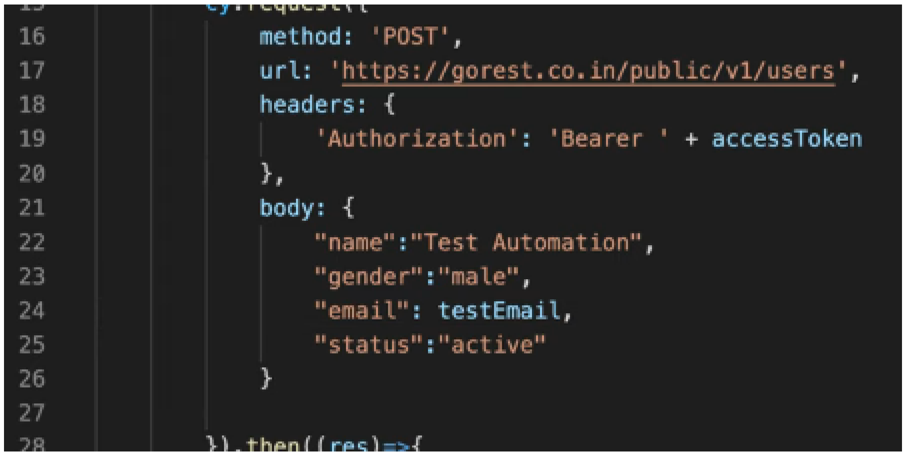
Random generator



cypress > integration > APITests > PostUser.js > describe('post user request') callback > it('create user te

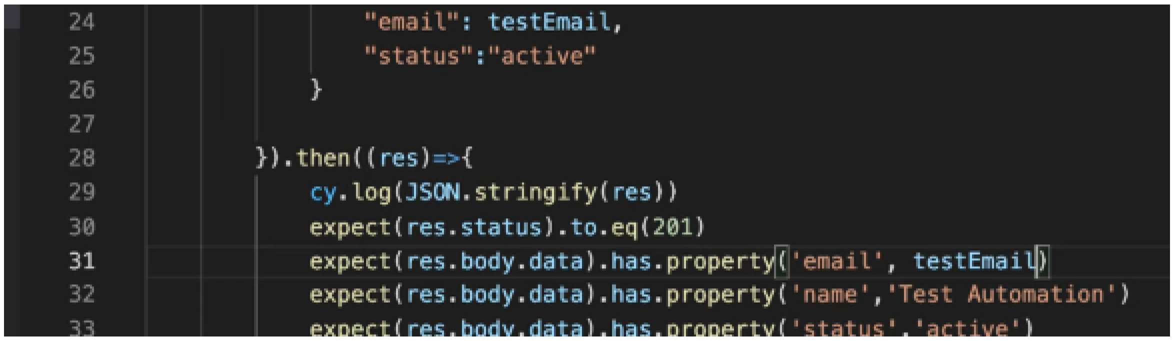
```
1   /// <reference types="Cypress" />
2
3   describe('post user request', () => {
4   let accessToken = '007526d9efdbc07e084ff7a6d4cfcc90588fbe20641c00faebf45a7f3b2eaf33'
5   let randomText = ""
6   let testEmail = ""
7
8       it('create user test', () => {
9
10          var pattern = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
11          for (var i = 0; i < 10; i++)
12          randomText+=pattern.charAt(Math.floor(Math.random() * pattern.length));
13          testEmail = randomText + '@gmail.com'
14
15
16
```

24 line
Changed to random generator variable.



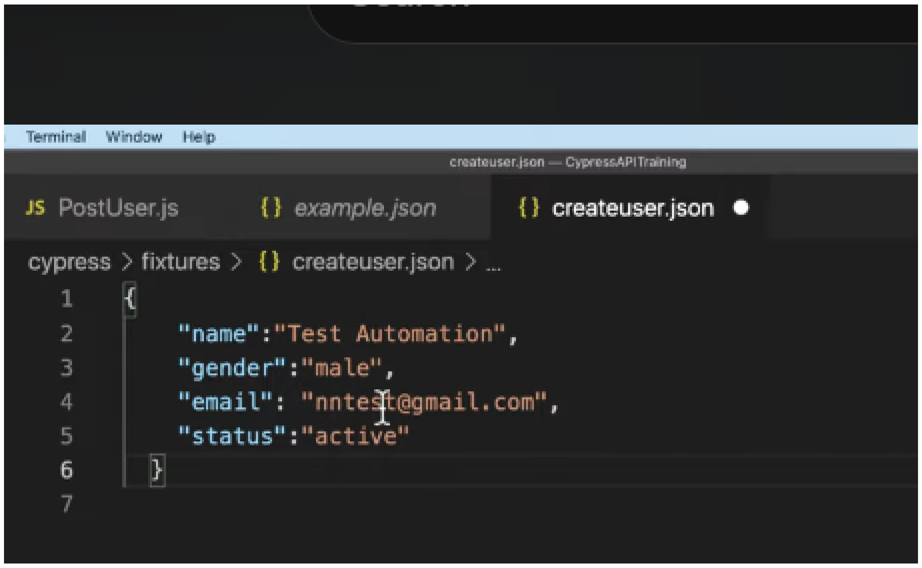
```
15          cy.request({
16              method: 'POST',
17              url: 'https://gorest.co.in/public/v1/users',
18              headers: {
19                  'Authorization': 'Bearer ' + accessToken
20              },
21              body: {
22                  "name":"Test Automation",
23                  "gender":"male",
24                  "email": testEmail,
25                  "status":"active"
26              }
27
28          }).then((res)=>{
```

31 line.
Changed to random generator variable.



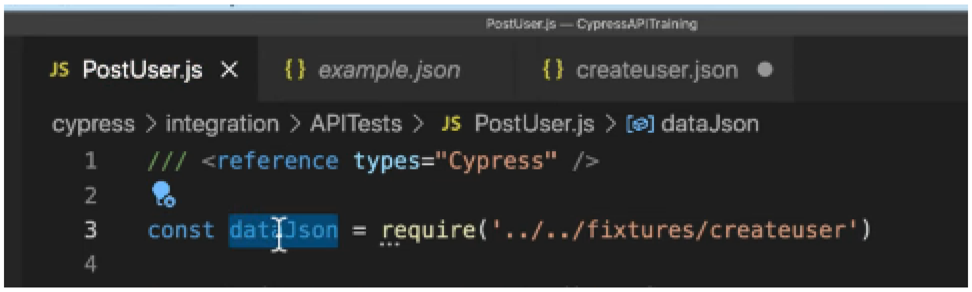
```
                "email": testEmail,
                "status":"active"
            }

        }).then((res)=>{
            cy.log(JSON.stringify(res))
            expect(res.status).to.eq(201)
            expect(res.body.data).has.property('email', testEmail)
            expect(res.body.data).has.property('name','Test Automation')
            expect(res.body.data).has.property('status','active')
```

Create user json
Data file.


Terminal
Window
Help
createuser.json — CypressAPITraining
PostUser.js
example.json
createuser.json
cypress > fixtures > createuser.json > ...

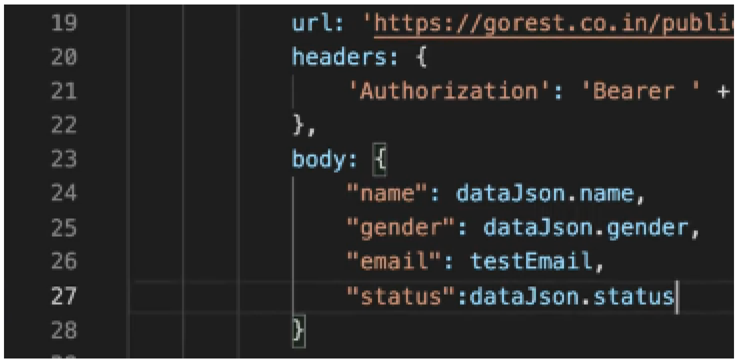
```
{
    "name":"Test Automation",
    "gender":"male",
    "email": "nntest@gmail.com",
    "status":"active"
}
```

Added3
No need of extension in import.


PostUser.js — CypressAPITraining
PostUser.js
example.json
createuser.json
cypress > integration > APITests > PostUser.js > dataJson

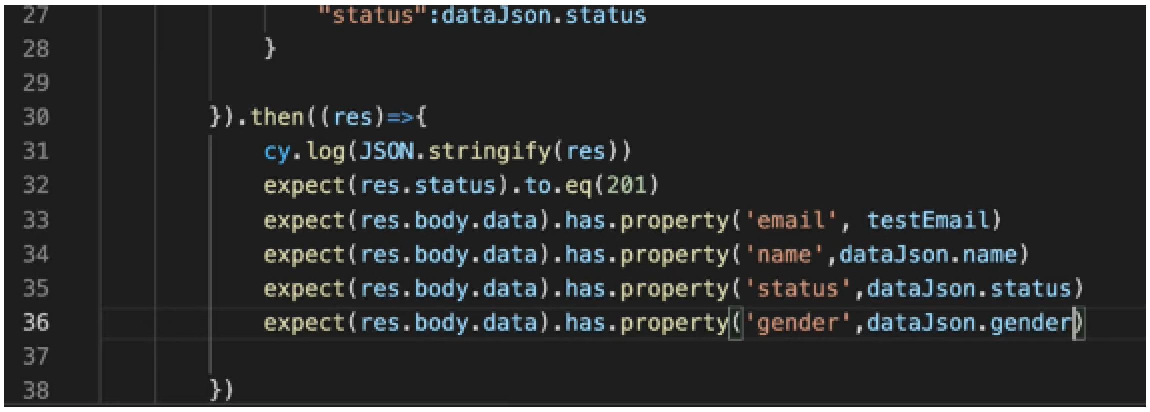
```
/// <reference types="Cypress" />

const dataJson = require('../../fixtures/createuser')
```

Changed body.
Using fixture variables.



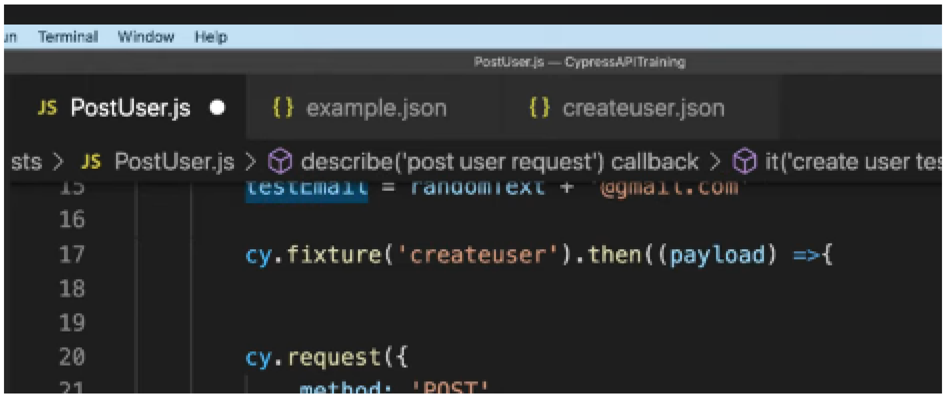
```
            url: 'https://gorest.co.in/publi
            headers: {
                'Authorization': 'Bearer ' +
            },
            body: {
                "name": dataJson.name,
                "gender": dataJson.gender,
                "email": testEmail,
                "status":dataJson.status
            }
```

Changed response.
Use fixture variables.



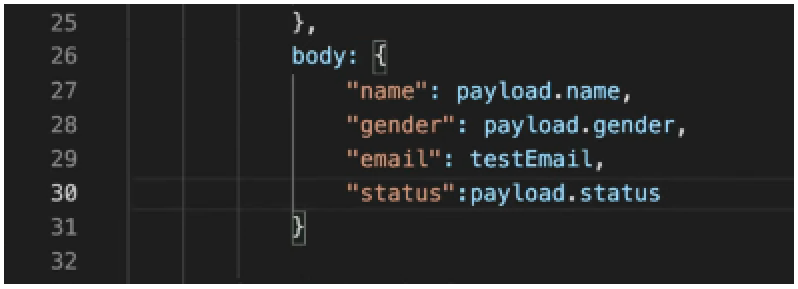
```
                "status":dataJson.status
            }

        }).then((res)=>{
            cy.log(JSON.stringify(res))
            expect(res.status).to.eq(201)
            expect(res.body.data).has.property('email', testEmail)
            expect(res.body.data).has.property('name',dataJson.name)
            expect(res.body.data).has.property('status',dataJson.status)
            expect(res.body.data).has.property('gender',dataJson.gender)

        })
```

This is another way to call fixtures directly.
Using payload as the variable name.
Store all fixture data inside payload.


Terminal Window Help
PostUser.js — CypressAPITraining
PostUser.js
example.json
createuser.json
sts > PostUser.js > describe('post user request') callback > it('create user tes

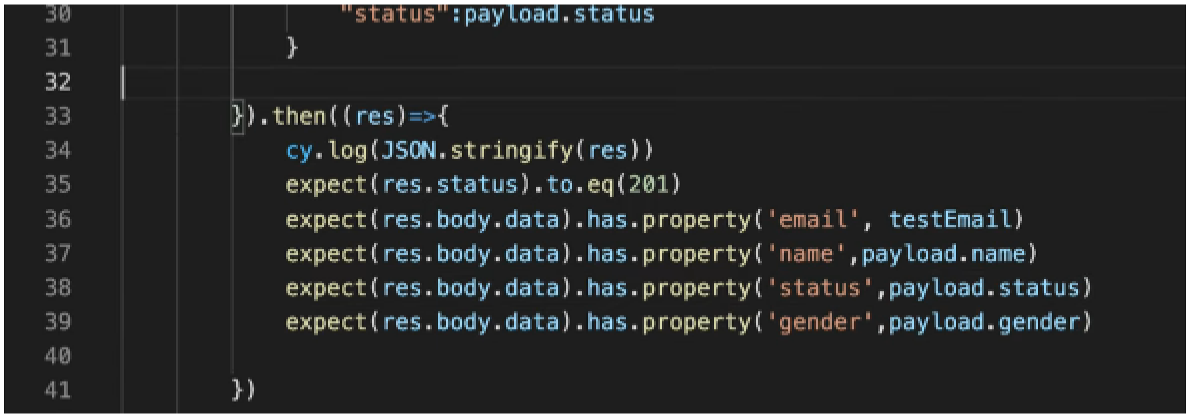
```
        testEmail = randomText + '@gmail.com'

        cy.fixture('createuser').then((payload) =>{


        cy.request({
            method: 'POST'
```

From url to test to assert should be inside fixture if we are using.
Pass in body using the help of payload variable.



```
                },
                body: {
                    "name": payload.name,
                    "gender": payload.gender,
                    "email": testEmail,
                    "status":payload.status
                }

```

Assert with fixture variable name - payload.



```
                    "status":payload.status
                }

            }).then((res)=>{
                cy.log(JSON.stringify(res))
                expect(res.status).to.eq(201)
                expect(res.body.data).has.property('email', testEmail)
                expect(res.body.data).has.property('name',payload.name)
                expect(res.body.data).has.property('status',payload.status)
                expect(res.body.data).has.property('gender',payload.gender)

            })
```