IT 314 - Software Engineering

Lab 8 - Group 14

Date - Apr 20, 2023

Team Members	Student ID
Pulkit Khandelwal	202001120
Shreyansh Khemesara	202001121
Prajapati Vrutik Navneetbhai(Team Leader)	202001124
Baraiya Dhruv Pravinbhai	202001140
Mandaviya Soham Hiteshbhai	202001142
Shah Nishadhkumar Nirajbhai	202001151
Buddhdev Harsh Nitesh	202001157
Chhagani Krunal Ajaybhai	202001158
Thakor Harshal Dipaksinh	202001169

● Take a module that you have implemented in any programming language. (Java, Python, C, C++, etc.)

    We will be considering the Sign-up and Sign-in modules for the submission. The language used is JavaScript.

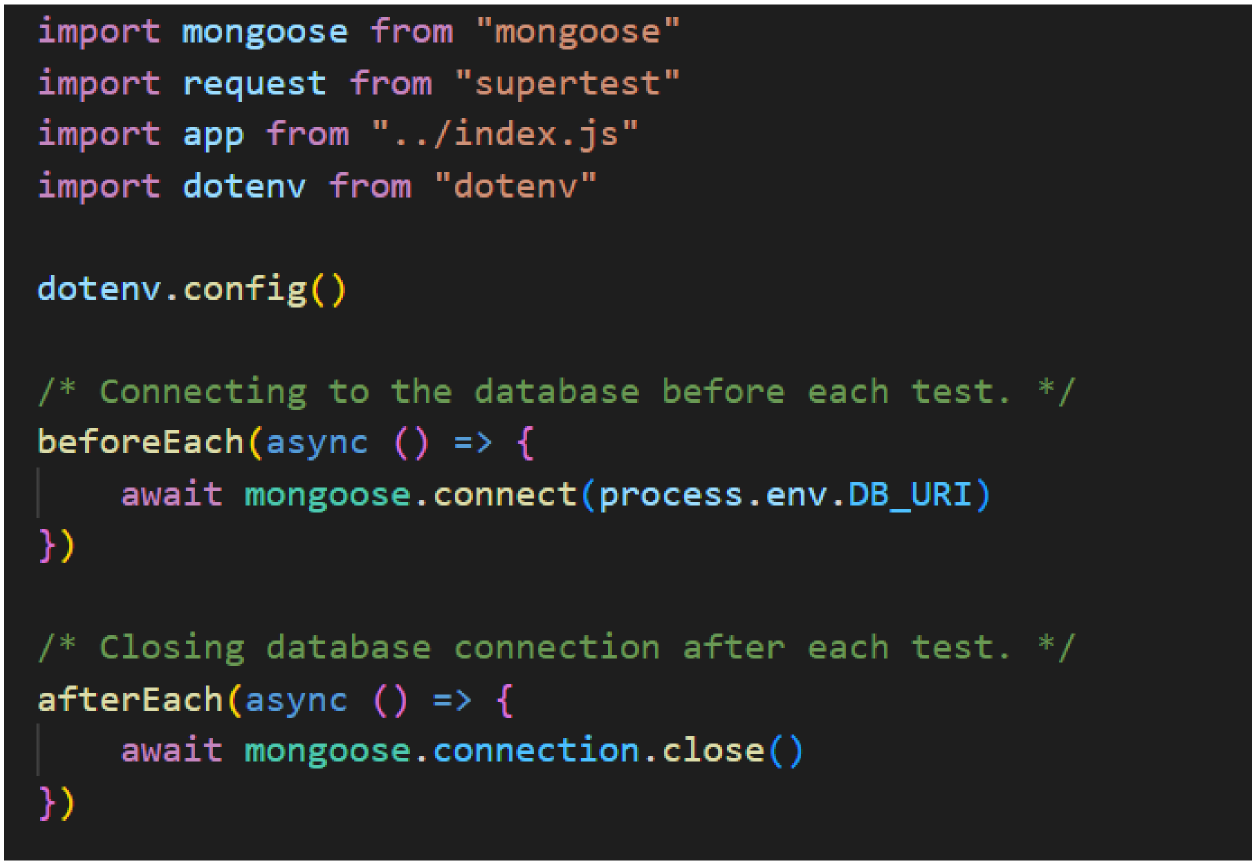
● Write the test cases.

Testing using Jest :

Here, we have implemented a log-in module and we have checked this module over various test cases.

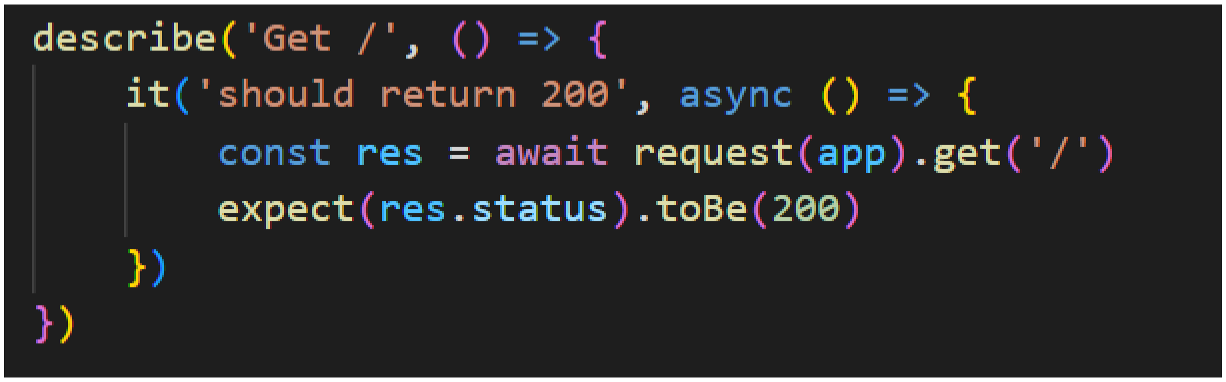
This particular case has many different test cases, so it is not necessary to run them all. If we miss even one field, the case should still be triggered.only four test cases should therefore cover almost all cases.

This is the configuration for the jest.



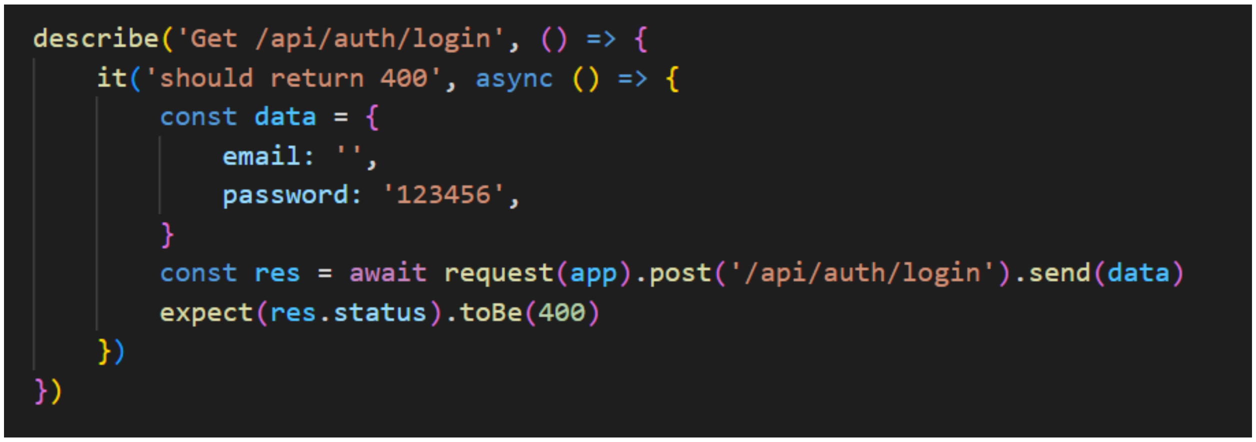
```
import mongoose from "mongoose"
import request from "supertest"
import app from "../index.js"
import dotenv from "dotenv"

dotenv.config()

/* Connecting to the database before each test. */
beforeEach(async () => {
    await mongoose.connect(process.env.DB_URI)
})

/* Closing database connection after each test. */
afterEach(async () => {
    await mongoose.connection.close()
})
```

1. This case is for the home route.



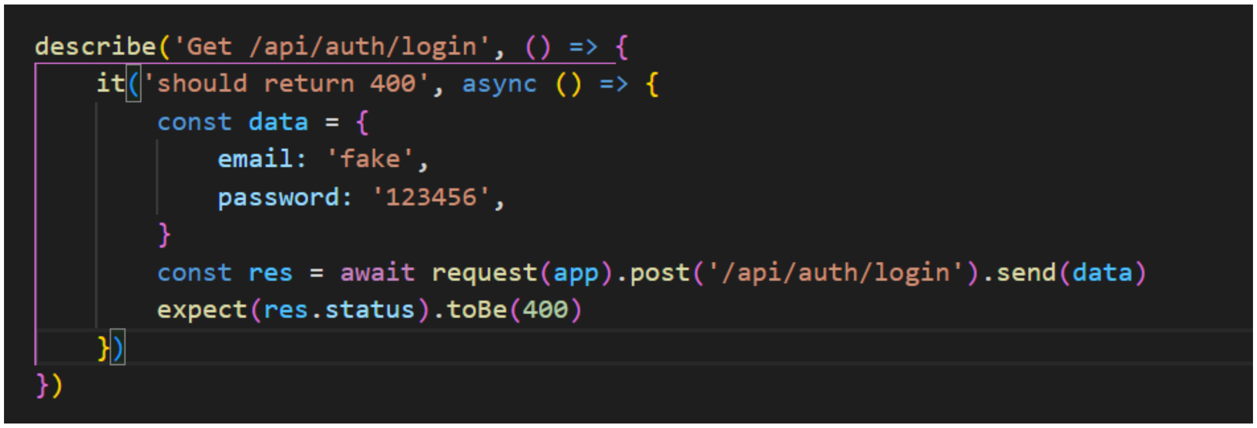
```
describe('Get /', () => {
    it('should return 200', async () => {
        const res = await request(app).get('/')
        expect(res.status).toBe(200)
    })
})
```

2. This case is triggered when the email is empty and it will return 400 as status code.



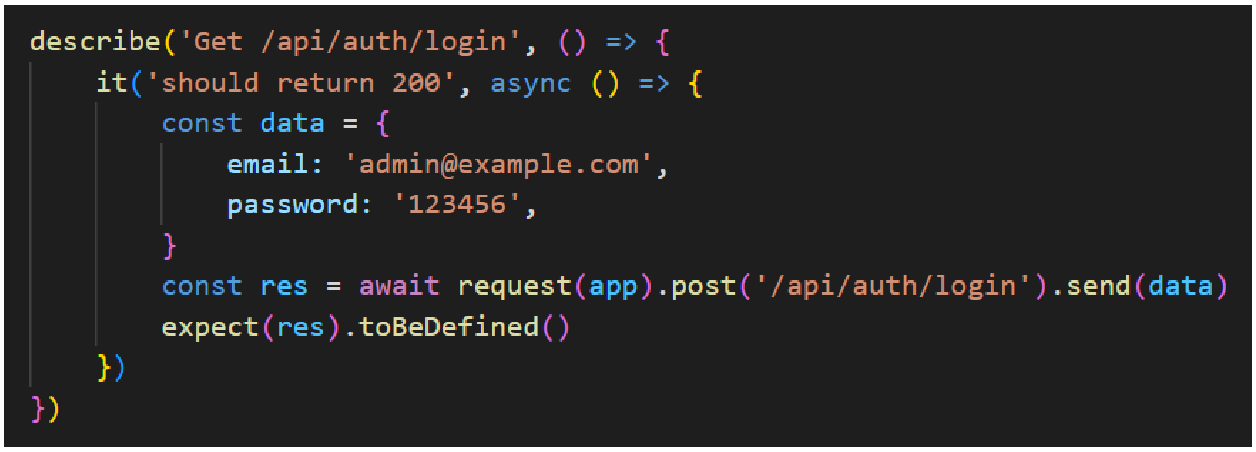
```
describe('Get /api/auth/login', () => {
    it('should return 400', async () => {
        const data = {
            email: '',
            password: '123456',
        }
        const res = await request(app).post('/api/auth/login').send(data)
        expect(res.status).toBe(400)
    })
})
```

3. This case is triggered when email is entered in the wrong format , this also will return a status code as 400.



```
describe('Get /api/auth/login', () => {
    it('should return 400', async () => {
        const data = {
            email: 'fake',
            password: '123456',
        }
        const res = await request(app).post('/api/auth/login').send(data)
        expect(res.status).toBe(400)
    })
})
```

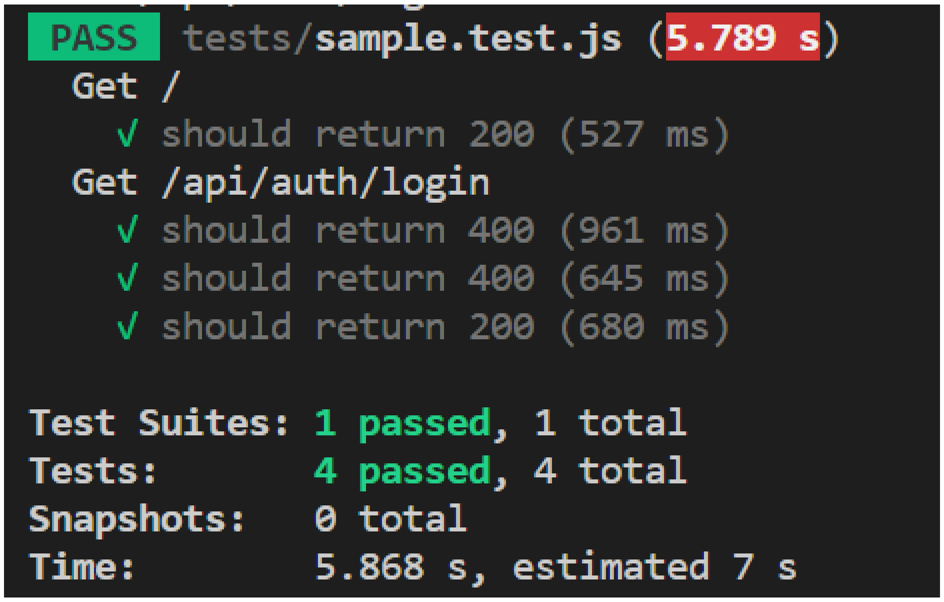
4. This case will trigger when email and password both are valid and they are correct , it will return a status code as 200.



```
describe('Get /api/auth/login', () => {
    it('should return 200', async () => {
        const data = {
            email: 'admin@example.com',
            password: '123456',
        }
        const res = await request(app).post('/api/auth/login').send(data)
        expect(res).toBeDefined()
    })
})
```

Output :

Here, we can see that output is correct for all the testcases, all the expected outputs are correct.



```
 PASS  tests/sample.test.js (5.789 s)
  Get /
    √ should return 200 (527 ms)
  Get /api/auth/login
    √ should return 400 (961 ms)
    √ should return 400 (645 ms)
    √ should return 200 (680 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        5.868 s, estimated 7 s
```

Once the test case failed, How did you fix that?

There is one test case that failed during module sign-up. According to terminology, the developer reports a defect, the bug is fixed, and the developer discovers a typo that the static analysis tool missed because the typographically incorrect variable name is already present in the file. After fixing the bug, the test case passed.