IT314 Software Engineering

Lab 8 - junit Testing

Guesthouse_booking_system_16

Group - 16

Unit Testing Jest:

We have used the jest framework for unit testing in node js. The testing is being performed in node js as the project is being implemented in node js.

These are the function which have been used to validate the user.

1. isValidEmail

```
function isValidEmail(email) {
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return emailRegex.test(email);
  }

  module.exports = isValidEmail;
```

Test case for the email validator

```
const isValidEmail = require('../utilsCheck/emailValidator');

test('valid email', () => {
  expect(isValidEmail('john@example.com')).toBe(true);
  expect(isValidEmail('joe123@example.net')).toBe(true);
  expect(isValidEmail('jane.doe@example.co.uk')).toBe(true);
  expect(isValidEmail('first.last@example.com')).toBe(true);
```

```
  expect(isValidEmail('user+label@example.com')).toBe(true);
  expect(isValidEmail('user123456789@example.com')).toBe(true);
  expect(isValidEmail('admin@example.org')).toBe(true);
  expect(isValidEmail('info@example.biz')).toBe(true);

});

test('invalid email', () => {
  expect(isValidEmail('invalidemail.com')).toBe(false);
  expect(isValidEmail('john@.com')).toBe(false);
  expect(isValidEmail('jane.doe@example')).toBe(false);
  expect(isValidEmail('joe123@example.')).toBe(false);
  expect(isValidEmail('info@example.com ')).toBe(false);
  expect(isValidEmail('@example.com')).toBe(false);
  expect(isValidEmail('admin@ example.com')).toBe(false);

});
```

Here we first import the our isvalid emai function.

Then we just write the two test case

First is valid email which return true when the email is valid syntax wise
Second is invalid email which return true when the email is invalid syntax wise

```
PS C:\Users\HARSH PATEL\Desktop\SW\IT314_Guesthouse_booking_system_16\api> jest emailValidator.test.js
 PASS  __test__/emailValidator.test.js
  √ valid email (4 ms)
  √ invalid email (1 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.678 s, estimated 1 s
Ran all test suites matching /emailValidator.test.js/i.
PS C:\Users\HARSH PATEL\Desktop\SW\IT314_Guesthouse_booking_system_16\api>
```

We can see the total two test cases are passed.

2. To check if mongo db is connected or not

```
app.get("/", async (req, res) => {
  mongoose.connect(process.env.MONGO_URL);
  res.json('test ok');
});
```

Here we call get api to check if mongo db is connected or not.

For that we just called mongoose.connect in app.get request.

Here MONGO_URL is in .env file so nobody can access it.

For testing it we wrote the jest test case

```
const request = require("supertest");
const app = require("../index");

describe("GET /", () => {
  test("should return 'test ok'", async () => {
    const response = await request(app).get("/");
    expect(response.status).toBe(200);
  });
});
```

Here describe is the Test suite in that test suite we wrote one unit test for checking if mongo db is connected or not.

Here is the result for that test cases

```
PS C:\Users\HARSH PATEL\Desktop\SW\IT314_Guesthouse_booking_system_16\api> jest auth.test.js --forceExit
  console.log
    Server is running on port 4000

      at Object.log (index.js:143:9)

 PASS  __test__/auth.test.js
  GET /
    √ should return 'test ok' (43 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.248 s, estimated 3 s
Ran all test suites matching /auth.test.js/i.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?
PS C:\Users\HARSH PATEL\Desktop\SW\IT314_Guesthouse_booking_system_16\api>
```

3. To get The profile page

From the main file we call the profile page see if it available or not

This is the test case for that

```
const request = require("supertest");
const app = require("../index");
const mongoose = require("mongoose");



describe("GET /profile", () => {
    test("should return null without a valid token", async () => {
        const response = await request(app).get("/profile");
        expect(response.status).toBe(200);
        expect(response.body).toBeNull();
      });
});
```

Here is the result for that test cases