Unit Testing the eCommerce Applications - 2

Assignment Solutions

Relevel
by Unacademy

Assignment Solutions

Write tests for getProductsUnderCategory method in product controller.

```
Run | Debug
describe('Product controller getProductsUnderCategory
method', () => {
    Run | Debug
    it('should return success message', async () => {
        const spy = jest.spyOn(ProductModel, 'findAll').
        mockImplementation(
            (data) => new Promise(function (resolve, reject) {
                resolve(testPayload);
        }));

        req.params.categoryId = 1;
        await ProductController.getProductsUnderCategory(req,
        res);

        expect(spy).toHaveBeenCalled();
        expect(res.status).toHaveBeenCalledWith(200);
        expect(res.send).toHaveBeenCalledWith(testPayload)
        ;
    });
})
```

```
Run | Debug
describe('Product controller getProductsUnderCategory
method', () => {
    Run | Debug
    it('should return error', async () => {
        const spy = jest.spyOn(ProductModel, 'findAll').
        mockImplementation(
            () => new Promise(function (resolve, reject) {
                reject(new Error('This is an error'));
        }));

        await ProductController.getProductsUnderCategory(req,
        res);

        expect(spy).toHaveBeenCalled();
    });
})
```

Please find the code link for reference.

Write unit tests for auth controller.

auth.controller.test.js

tests > controllers > auth.controller.test.js > testPayload

```
const db = require('../../models');
const UserModel = db.user;
const AuthController = require('../../controllers/auth.
controller');
const { mockRequest, mockResponse } = require('../
interceptor');
const bcrypt = require("bcryptjs");

let req, res;

beforeEach(() =>{
    req = mockRequest();
    res = mockResponse();
})

const testPayload =
    {
        username: "Test User",
        email: "test@email.com",
        password: "123456"
    };

Run | Debug
```

Please find the code

```
describe('Auth controller signup method', () => {
    Run | Debug
    it('should return user registered success message', async
    () => {
        const spy2 = jest.spyOn(UserModel, 'create').
        mockImplementation(
            (user) => new Promise(function (resolve, reject) {
                resolve(Promise.resolve("added roles"));
        }));

        req.body = testPayload;

        await AuthController.signup(req, res);

        expect(spy2).toHaveBeenCalled();
        // expect(res.status).toHaveBeenCalledWith(201);
        // expect(res.send).toHaveBeenCalledWith({ message:
        "User registered successfully!" });
    });
});
```

link for reference.

```
describe('Auth controller signup method', () => {
    Run | Debug
    it('should return user registered success message', async
    () => {
        const spy1 = jest.spyOn(bcrypt, 'compareSync').
        mockImplementation(() => true);
        const spy2 = jest.spyOn(UserModel, 'findOne').
        mockImplementation(
            (user) => new Promise(function (resolve, reject) {
                resolve("pass");
        }));

        req.body = {
            username: "Test User"
        };

        await AuthController.signin(req, res);

        expect(spy1).toHaveBeenCalled();
        expect(spy2).toHaveBeenCalled();
        // expect(res.status).toHaveBeenCalledWith(201);
        // expect(res.send).toHaveBeenCalledWith({ message:
        "User registered successfully!" });
    });
});
```

Please find the code link for reference.