ЛАБОРАТОРНА РОБОТА № 7

Тема: Розробка тестів з використанням фреймворка Mocha

Хід роботи:

Установка Mocha

- Створіть проект mocha (або наступний за рахунком)
- Встановіть фреймворк Mocha на рівні проекту
- В package.json пропишіть команду для запуску тестів

Лістининг:

```
{
  "name": "lab7",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "mocha --timeout 10000"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "mocha": "^10.4.0"
  },
  "dependencies": {
    "chai": "^5.1.1",
    "dotenv": "^16.4.5",
    "supertest": "^7.0.0"
  }
}
```

Завдання 1.

Створіть функцію пошуку факторіала числа factorial. Створіть специфікацію та тести перевірки роботи функції. Перевірте виконання тестів:

n	factorial(n)
5	120
6	720
0	1
-5	null
-6	null

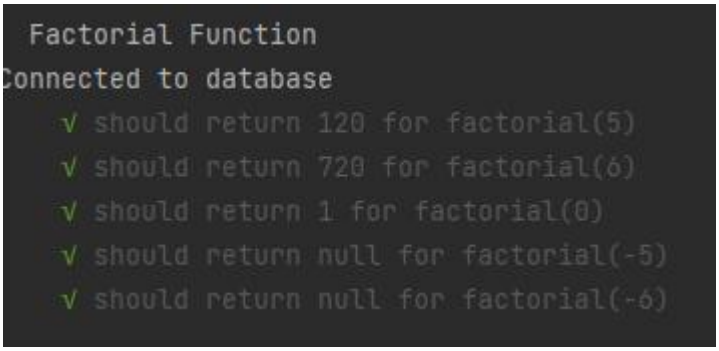
- Функція factorial повинна знаходитись в користувацькому модулі functions.js
- Файл з тестами повинен іменуватись functions.test.js

Лістининг functions.js:

```
function factorial(n) {
    if (n < 0) return null;
    if (n === 0) return 1;
    let result = 1;
    for (let i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}

module.exports = factorial;
```

Лістининг functions.test.js:

```
const assert = require('assert');
const factorial = require('../functions');

describe('Factorial Function', function() {
    it('should return 120 for factorial(5)', function() {
        assert.strictEqual(factorial(5), 120);
    });

    it('should return 720 for factorial(6)', function() {
        assert.strictEqual(factorial(6), 720);
    });

    it('should return 1 for factorial(0)', function() {
        assert.strictEqual(factorial(0), 1);
    });

    it('should return null for factorial(-5)', function() {
        assert.strictEqual(factorial(-5), null);
    });

    it('should return null for factorial(-6)', function() {
        assert.strictEqual(factorial(-6), null);
    });
});
```



Завдання 2. Розробка тестів для REST-застосунку (TaskApp)

Перед виконанням тестування відбувається очистка колекцій *users* і *tasks*;

#	***Тест***	***Результат проходження тесту***
1	Реєстрація користувача User1 з помилкою валідації	Статус 401 та назва помилки
2	Реєстрація користувача User1 без помилок	Статус 200. Отриманий об'єкт user з властивістю _id
3	Реєстрація користувача User2 без помилок	Статус 200 і отриманий об'єкт user з ідентифікатором

4	Вхід під User1 з вірними даними	Статус 200 і отримане повідомлення success
5	Додавання задачі Task1	Статус 200 і отриманий об'єкт task з ідентифікатором
6	Додавання задачі Task2	Статус 200 і отриманий об'єкт task з ідентифікатором
7	Отримання задач користувача User1	Статус 200. Довжина 2
8	Отримуємо задачу Task1 по ідентифікатору	Статус 200. Об'єкт task з властивостями title і completed.
9	Вихід	Повідомлення "logout success"
10	Вхід під User2 з вірними даними	Статус 200 і отримане повідомлення success
11	Додавання задачі Task3	Статус 200 і отриманий об'єкт task з ідентифікатором
12	Отримання задач користувача User2	Статус 200. Довжина 1.
13	Отримуємо задачу Task1 по ідентифікатору	Статус 404. Повідомлення "Not Found"
14	Вихід	Статус 200. Повідомлення "logout success"
15	Отримуємо задачу Task1 по її ідентифікатору	Статус 403. Повідомлення "Forbidden Access"

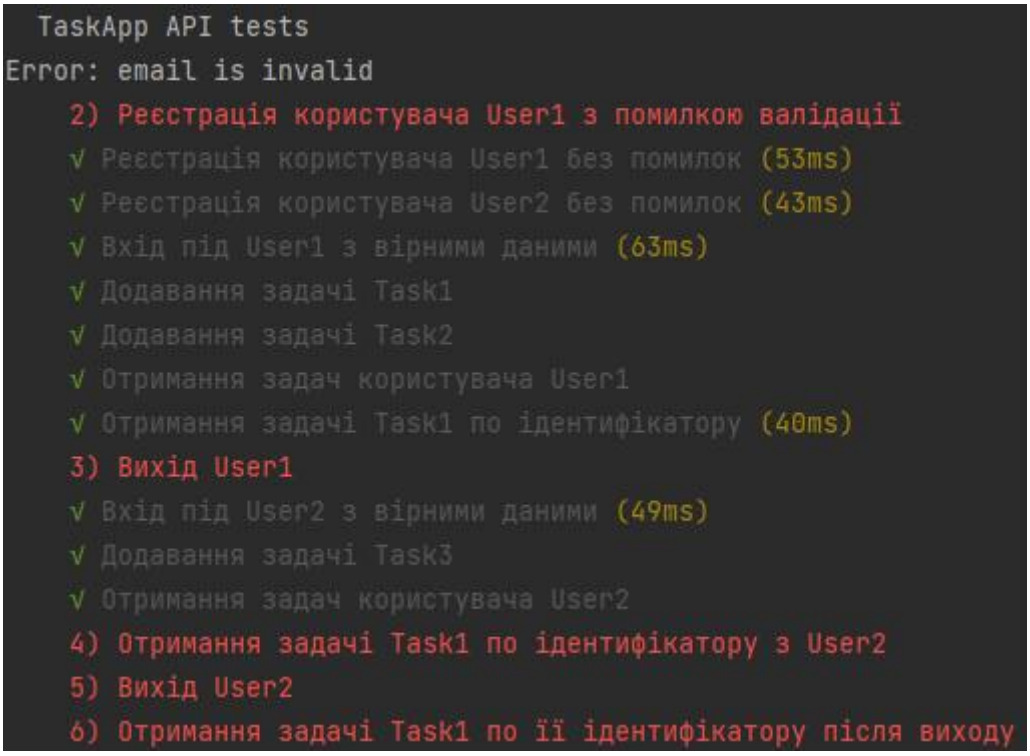
Лістининг taskapp.test.js:

```
const request = require('supertest');
const app = require('../../Lab3-4-5/TaskApp/src/app');
const User = require('../../Lab3-4-5/TaskApp/models/user');
const Task = require('../../Lab3-4-5/TaskApp/models/task');
require('../../Lab3-4-5/TaskApp/db/mongoose');
require('dotenv').config({ path: '../../Lab3-4-5/.env' });

(async () => {
    const { expect } = await import('chai');

    describe('TaskApp API tests', function() {
        let authToken;

        before(async function() {
            await User.deleteMany({});
            await Task.deleteMany({});
```

```
        });

        it('Реєстрація користувача User1 з помилкою валідації', async function() {
            const res = await request(app)
                .post('/users')
                .send({ name: 'User1', password: '12345', email: 'invalid-
email' });

            expect(res.status).to.equal(400);
            expect(res.body).to.have.property('error');
        });

        it('Реєстрація користувача User1 без помилок', async function() {
            const res = await request(app)
                .post('/users')
                .send({ name: 'User1', age: 20, password: '12345678', email:
'user1@gmail.com' });

            expect(res.status).to.equal(200);
            expect(res.body).to.have.property('_id');
            expect(res.body).to.have.property('name', 'User1');
            expect(res.body).to.have.property('email', 'user1@gmail.com');
        });

        it('Реєстрація користувача User2 без помилок', async function() {
            const res = await request(app)
                .post('/users')
                .send({ name: 'User2', age: 22, password: '12345678', email:
'user2@gmail.com' });

            expect(res.status).to.equal(200);
            expect(res.body).to.have.property('_id');
            expect(res.body).to.have.property('name', 'User2');
            expect(res.body).to.have.property('email', 'user2@gmail.com');
        });

        it('Вхід під User1 з вірними даними', async function() {
            const res = await request(app)
                .post('/users/login')
                .send({ email: 'user1@gmail.com', password: '12345678' });

            expect(res.status).to.equal(202);
            expect(res.body).to.have.property('token');
            authToken = res.body.token;
        });

        it('Додавання задачі Task1', async function() {
            const res = await request(app)
                .post('/tasks')
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task1', description: 'First task', completed:
false });

            expect(res.status).to.equal(201);
            expect(res.body).to.have.property('_id');
            expect(res.body).to.have.property('title', 'Task1');
            expect(res.body).to.have.property('description', 'First task');
            expect(res.body).to.have.property('completed', false);
        });

        it('Додавання задачі Task2', async function() {
            const res = await request(app)
                .post('/tasks')
```

```
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task2', description: 'Second task', completed:
false });

            expect(res.status).to.equal(201);
            expect(res.body).to.have.property('_id');
            expect(res.body).to.have.property('title', 'Task2');
            expect(res.body).to.have.property('description', 'Second task');
            expect(res.body).to.have.property('completed', false);
        });

        it('Отримання задач користувача User1', async function() {
            const res = await request(app)
                .get('/tasks')
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(200);
            expect(res.body).to.have.length(2);
            expect(res.body[0]).to.have.property('title');
            expect(res.body[1]).to.have.property('title');
        });

        it('Отримання задачі Task1 по ідентифікатору', async function() {
            const taskRes = await request(app)
                .post('/tasks')
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task1', description: 'First task', completed:
false });

            const res = await request(app)
                .get(`/tasks/${taskRes.body._id}`)
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(200);
            expect(res.body).to.have.property('title', 'Task1');
            expect(res.body).to.have.property('completed', false);
        });

        it('Вихід User1', async function() {
            const res = await request(app)
                .post('/users/logout')
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(202);
            expect(res.body).to.have.property('message', 'logout success');
        });

        it('Вхід під User2 з вірними даними', async function() {
            const res = await request(app)
                .post('/users/login')
                .send({ email: 'user2@gmail.com', password: '12345678' });

            expect(res.status).to.equal(202);
            expect(res.body).to.have.property('token');
            authToken = res.body.token;
        });

        it('Додавання задачі Task3', async function() {
            const res = await request(app)
                .post('/tasks')
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task3', description: 'Third task', completed:
false });
```

```
            expect(res.status).to.equal(201);
            expect(res.body).to.have.property('_id');
            expect(res.body).to.have.property('title', 'Task3');
            expect(res.body).to.have.property('description', 'Third task');
            expect(res.body).to.have.property('completed', false);
        });

        it('Отримання задач користувача User2', async function() {
            const res = await request(app)
                .get('/tasks')
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(200);
            expect(res.body).to.have.length(1);
            expect(res.body[0]).to.have.property('title', 'Task3');
        });

        it('Отримання задачі Task1 по ідентифікатору з User2', async function() {
            const taskRes = await request(app)
                .post('/tasks')
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task1', description: 'First task', completed:
false });

            const res = await request(app)
                .get(`/tasks/${taskRes.body._id}`)
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(404);
            expect(res.body).to.have.property('message', 'Not Found');
        });

        it('Вихід User2', async function() {
            const res = await request(app)
                .post('/users/logout')
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(202);
            expect(res.body).to.have.property('message', 'logout success');
        });

        it('Отримання задачі Task1 по її ідентифікатору після виходу', async
function() {
            const taskRes = await request(app)
                .post('/tasks')
                .set('Authorization', `Bearer ${authToken}`)
                .send({ title: 'Task1', description: 'First task', completed:
false });

            const res = await request(app)
                .get(`/tasks/${taskRes.body._id}`)
                .set('Authorization', `Bearer ${authToken}`);

            expect(res.status).to.equal(403);
            expect(res.body).to.have.property('message', 'Forbidden Access');
        });
    });
})();
```



Висновок: Під час виконання лабораторної роботи, ознайомився з розробкою тестів з використанням фреймворка Mocha