Український державний університет імені Михайла Драгоманова

Факультет математики, інформатики та фізики

Звіт про виконану роботу до лабораторного заняття № 2

*з курсу*

*Автоматизація тестування*

***Виконав/ла:*** студентка групи41ІПЗ

спеціальності «Інженерія програмного

забезпечення»

***Антосевич К.О.***

***Викладач:*** Кархут В.Я.

Лабораторна робота №2 1. Напишіть тест, щоб перевірити, що функція 1 яка викликає іншу функцію 2 з аргументами Name і Lastname викликається з значеннями name=John lastname=Doe. Використовуйте макет функції для створення імітації функції. 2. Створіть просту асинхронну функцію, яка повертає проміс, а потім протестуйте її за допомогою вбудованих функцій Jest expect та resolves. Щоб виконати це завдання, створіть асинхронну функцію, яка повертає проміс, що вирішується рядком "hello world". Потім використовуйте функцію expect з Jest, щоб перевірити, що проміс вирішується з очікуваним рядком. 3. Створіть асинхронну функцію, яка використовує синтаксис async/await та протестуйте її за допомогою Jest. Щоб виконати це завдання, створіть асинхронну функцію, яка викликає іншу асинхронну функцію, яка повертає обіцянку, що вирішується числом. Потім використовуйте синтаксис async/await, щоб отримати значення вирішеної обіцянки та використовуйте функцію expect з Jest, щоб перевірити, що значення є очікуваним числом. 4. Створіть асинхронну функцію, яка відхиляє з помилкою, і протестуйте її за допомогою Jest. Cтворіть асинхронну функцію, яка повертає проміс, яка відхиляється з помилкою. Потім скористайтеся функцією очікування Jest, щоб переконатися, що проміс відхилено з очікуваною помилкою. 5. Створіть асинхронну функцію, яка виконує симульований виклик API, і протестуйте її за допомогою Jest. Створіть асинхронну функцію, яка здійснює виклик API за допомогою fetch і повертає проміс, який вирішується за допомогою обєкта JSON. Потім скористайтеся функцією очікування Jest, щоб переконатися, що відповідь є очікуваним об’єктом JSON. 6. Створіть асинхронну функцію, яка викликає іншу функцію, яка здійснює виклик фіктивного API, і повертає проміс, який вирішується за допомогою відповіді JSON. Потім скористайтеся функцією очікування Jest, щоб переконатися, що відповідь є очікуваним об’єктом JSON

// Функції

function callFunctionWithArgs(name, lastName, callback) {

return callback(name, lastName);

}

async function resolveHelloWorld() {

return Promise.resolve("hello world");

}

async function fetchNumber() {

return Promise.resolve(42);

}

async function rejectWithError() {

return Promise.reject(new Error("Failed"));

}

async function fetchMockApi() {

return Promise.resolve({ status: "ok", data: { id: 1, name: "John Doe" } });

}

async function fetchAndProcessApi(callApi) {

const response = await callApi();

return response.data;

}

// Файл з тестами

const jest = require('jest');

describe('Function tests', () => {

// Тест 1: перевірка виклику функції з конкретними аргументами

test('callFunctionWithArgs should call function with specific arguments', () => {

const mockFn = jest.fn();

callFunctionWithArgs('John', 'Doe', mockFn);

expect(mockFn).toHaveBeenCalledWith('John', 'Doe');

});

// Тест 2: асинхронна функція, що повертає "hello world"

test('resolveHelloWorld resolves to "hello world"', () => {

return expect(resolveHelloWorld()).resolves.toBe("hello world");

});

// Тест 3: асинхронний тест з async/await

test('fetchNumber should fetch a number', async () => {

const number = await fetchNumber();

expect(number).toBe(42);

});

// Тест 4: перевірка відхилення проміса з помилкою

test('rejectWithError should reject with an error', () => {

return expect(rejectWithError()).rejects.toThrow("Failed");

});

// Тест 5: симульований API виклик, який повертає JSON

test('fetchMockApi returns expected JSON object', () => {

return expect(fetchMockApi()).resolves.toEqual({ status: "ok", data: { id: 1, name: "John Doe" } });

});

// Тест 6: асинхронна функція, яка використовує іншу функцію для виклику фіктивного API

test('fetchAndProcessApi returns processed data from API', async () => {

const mockApi = jest.fn().mockResolvedValue({ status: "ok", data: { id: 1, name: "John Doe" } });

const data = await fetchAndProcessApi(mockApi);

expect(data).toEqual({ id: 1, name: "John Doe" });

expect(mockApi).toHaveBeenCalled();

});

});