МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет
нафти і газу

Кафедра інженерії програмного забезпечення

ЛАБОРАТОРНА РОБОТА № 3

з дисципліни

Основи автоматизованого тестування

Виконав: ст. гр. ІП-23-1К
Гнатюк Д. М.

Івано-Франківськ

2025

Тема: Створення та запуск юніт-тестів з використанням Jest

Мета: Навчитися створювати модульні тести, розібратися в існуючих інструментах для створення модульних тестів та навчитися їх обирати.

Посилання на Github: https://github.com/DementialDima/Lab1-AutomatedTesting

Створено проєкт;

mkdir unit-testing

cd unit-testing

npm init -y

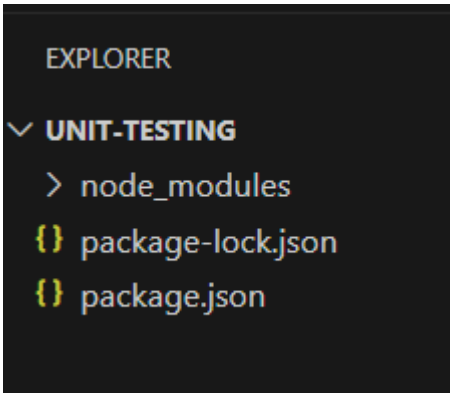
```
PS C:\Users\gnatu\unit-testing> npm init -y
>>
Wrote to C:\Users\gnatu\unit-testing\package.json:

{
  "name": "unit-testing",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Встановлено Jest:

npm install --save-dev jest

```
                        npm install --save-dev jest
>> C:\Users\gnatu\unit-testing>
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache
if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and power
ful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 266 packages, and audited 267 packages in 9s

32 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\gnatu\unit-testing>
```

Результат підготовки середовища:



Створено файл mathFunctions.js з кодом:

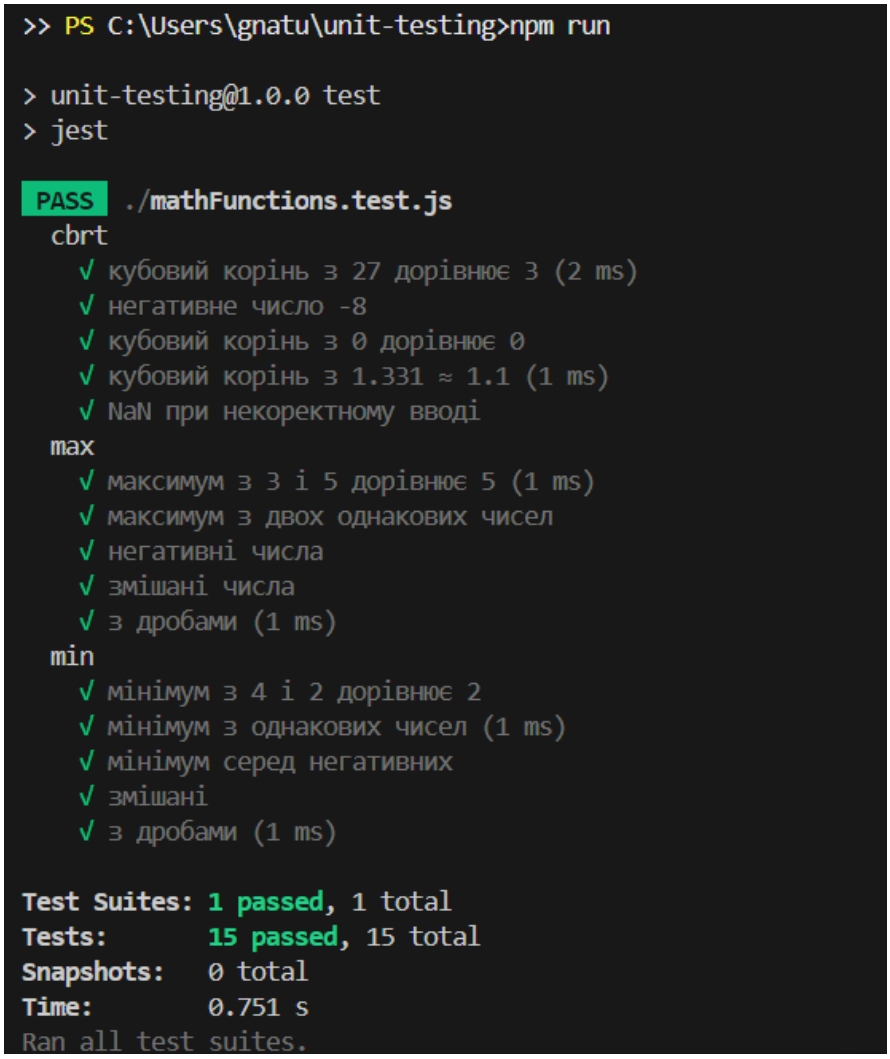
```
export const cbrt = a => Math.cbrt(a);
export const max = (a, b) => Math.max(a, b);
export const min = (a, b) => Math.min(a, b);
```

Створено файл mathFunctions.test.js з кодом тестів функцій:

```
import { cbrt, max, min } from "./mathFunctions";

describe("cbrt", () => {
  test("кубовий корінь з 27 дорівнює 3", () => {
    expect(cbrt(27)).toBe(3);
  });

  test("негативне число -8", () => {
    expect(cbrt(-8)).toBe(-2);
  });

  test("кубовий корінь з 0 дорівнює 0", () => {
    expect(cbrt(0)).toBe(0);
  });

  test("кубовий корінь з 1.331 ≈ 1.1", () => {
    expect(cbrt(1.331)).toBeCloseTo(1.1);
  });

  test("NaN при некоректному вводі", () => {
    expect(cbrt("abc")).toBeNaN();
  });
});

describe("max", () => {
  test("максимум з 3 і 5 дорівнює 5", () => {
    expect(max(3, 5)).toBe(5);
  });

  test("максимум з двох однакових чисел", () => {
    expect(max(7, 7)).toBe(7);
  });

  test("негативні числа", () => {
    expect(max(-3, -8)).toBe(-3);
  });

  test("змішані числа", () => {
```

```
    expect(max(-10, 0)).toBe(0);
  });

  test("з дробами", () => {
    expect(max(2.5, 2.499)).toBe(2.5);
  });
});

describe("min", () => {
  test("мінімум з 4 і 2 дорівнює 2", () => {
    expect(min(4, 2)).toBe(2);
  });

  test("мінімум з однакових чисел", () => {
    expect(min(9, 9)).toBe(9);
  });

  test("мінімум серед негативних", () => {
    expect(min(-4, -1)).toBe(-4);
  });

  test("змішані", () => {
    expect(min(0, -1)).toBe(-1);
  });

  test("з дробами", () => {
    expect(min(0.1, 0.01)).toBe(0.01);
  });
});
```

Скріншот з виконанням тестів:



Висновок: На цій лабораторній роботі було здобуто практичні навички опису програмного проєкту, формалізації вимог (бізнес-, користувацьких, функціональних і нефункціональних), створення словника термінів, побудови Use Case діаграми та структури класів. Також опановано роботу в середовищі Trello як інструменті організації та візуалізації проєктної документації. На прикладі реального вебзастосунку для стоматологічної клініки було закріплено принципи планування IT-проєктів у знання-орієнтованому підході.

Висновок: У ході виконання лабораторної роботи було налаштовано середовище для модульного тестування JavaScript-коду з використанням фреймворку Jest. Було створено три математичні функції: cbrt, max та min, і до кожної з них реалізовано по 5 юніт-тестів. Також було налаштовано конфігураційні файли package.json та .babelrc, а для підтримки синтаксису ES-модулів встановлено Babel. Після запуску тестів усі 15

перевірок пройдено успішно, що підтверджує коректність реалізації функцій і написаних до них тестів.

Отже, мету лабораторної роботи досягнуто — отримано практичні навички зі створення та запуску юніт-тестів, а також з використання Jest у сучасному JavaScript-проєкті.

Відповіді на запитання

1. Що таке модульний тест?

Модульний тест (unit test) — це тест, який перевіряє правильність роботи однієї конкретної функції або модуля програми ізольовано від інших частин системи.

2. Які переваги модульного тестування?

- Швидке виявлення помилок на ранніх етапах
- Полегшує рефакторинг коду
- Підвищує якість та надійність програмного забезпечення
- Спрощує розуміння логіки коду
- Дозволяє швидко перевірити змінений функціонал

3. До якого типу належить модульне тестування?

Модульне тестування належить до білих коробок (white-box testing), оскільки тестер має доступ до внутрішньої реалізації коду.

4. Які інструменти запуску тестів Вам відомі?

- Jest
- Mocha
- Jasmine
- AVA
- Vitest

5. Які бібліотеки перевірки Вам відомі?

- expect (вбудований у Jest)
- Chai (для Mocha)
- assert (вбудована в Node.js)
- should.js

6. Які вбудовані інструменти для створення тестів існують в Node.js?

- assert — базова бібліотека перевірок (assertions)
- node:test — вбудований модуль для створення тестів, доступний з Node.js 18+