**Лабораторна робота № 3  
Програмування засноване на тестуванні**

Виконав студент групи **КН-19-2 Ільченко Дмитро**

**Мета роботи:** навчитися розробляти й реалізовувати модульні тести з використанням систем автоматизованого тестування серії xUnit.

**Хід виконання роботи**

1. Вивчити теоретичні відомості.

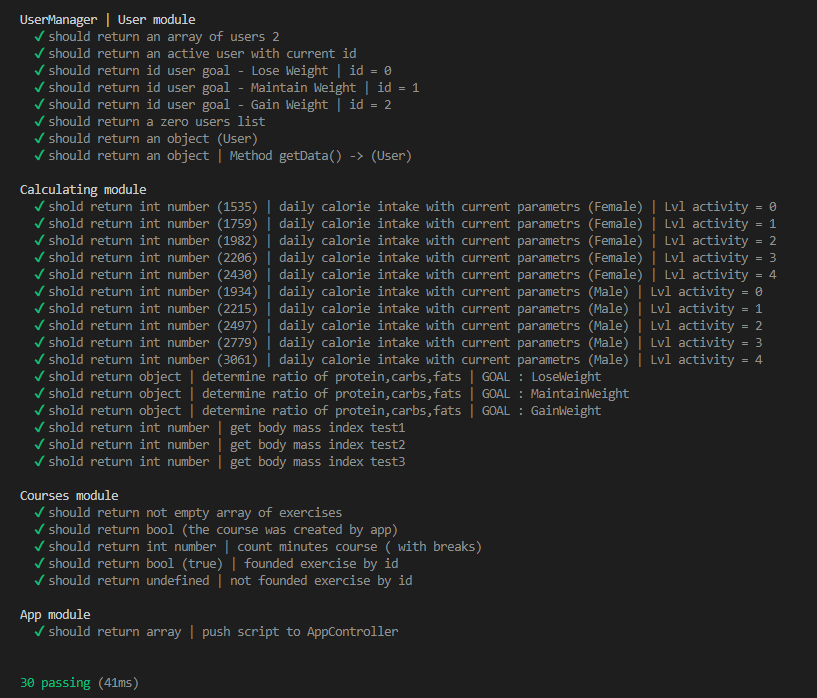
2. Ознайомитися із принципами роботи системи автоматизованого тестування JUnit.

3. Реалізувати модульні тести, що забезпечують перевірку основної функціональності кожного класу відповідно до розробленого проекту програмного продукту.

4. Домогтися успішного виконання всіх модульних тестів.

5. Зробити висновки з проекту програмного продукту

Модульні тести:



const { expect, assert } = require("chai");

import "mocha";

import {

    CalcDailyCalorieIntakeFEMALE\_TEST1,

    CalcDailyCalorieIntakeFEMALE\_TEST2,

    CalcDailyCalorieIntakeFEMALE\_TEST3,

    CalcDailyCalorieIntakeFEMALE\_TEST4,

    CalcDailyCalorieIntakeFEMALE\_TEST5,

    CalcDailyCalorieIntakeMALE\_TEST1,

    CalcDailyCalorieIntakeMALE\_TEST2,

    CalcDailyCalorieIntakeMALE\_TEST3,

    CalcDailyCalorieIntakeMALE\_TEST4,

    CalcDailyCalorieIntakeMALE\_TEST5,

    determineRatioOfPFC\_TEST1,

    determineRatioOfPFC\_TEST2,

    determineRatioOfPFC\_TEST3,

    getBodyMassIndex\_TEST1,

    getBodyMassIndex\_TEST2,

    getBodyMassIndex\_TEST3,

} from "./calcTest";

import { UserManagerTest, UserTest1, UserTest2 } from "./userTest";

import {  ABSCourseTEST1, ABSCourseTEST2 } from './calcCourse';

import AppController from "../scripts/Classes/AppController/AppController";

describe("UserManager | User module", () => {

    it("should return an array of users 2", () => {

        UserManagerTest.addNewUser(UserTest1);

        UserManagerTest.addNewUser(UserTest2);

        expect(UserManagerTest.users).to.have.lengthOf(2);

        UserManagerTest.clearData();

    });

    it("should return an active user with current id", () => {

        let \_testId = "1234567890";

        UserManagerTest.setActiveUser = UserTest1;

        expect(UserManagerTest.getctiveUser.about.id).to.equal(\_testId);

    });

    it("should return id user goal - Lose Weight | id = 0", () => {

        let \_testGoal = "Lose Weight";

        expect(UserManagerTest.getIdUserGoal(\_testGoal)).to.equal(0);

    });

    it("should return id user goal - Maintain Weight | id = 1", () => {

        let \_testGoal = "Maintain Weight";

        expect(UserManagerTest.getIdUserGoal(\_testGoal)).to.equal(1);

    });

    it("should return id user goal - Gain Weight | id = 2", () => {

        let \_testGoal = "Gain Weight";

        expect(UserManagerTest.getIdUserGoal(\_testGoal)).to.equal(2);

    });

    it("should return a zero users list ", () => {

        UserManagerTest.clearData();

        expect(UserManagerTest.users).to.have.lengthOf(0);

    });

    it("should return an object (User) ", () => {

        UserManagerTest.setActiveUser = UserTest1;

        expect(UserManagerTest.getctiveUser).to.be.an("object");

    });

    it("should return an object | Method getData() -> (User) ", () => {

        expect(UserTest1.about).to.be.an("object");

    });

});

describe("Calculating module", () => {

    it("shold return int number (1535) | daily calorie intake with current parametrs (Female) | Lvl activity = 0", () => {

        const \_testCalorie = 1535;

        expect(CalcDailyCalorieIntakeFEMALE\_TEST1).to.be.equal(\_testCalorie);

    });

    it("shold return int number (1759) | daily calorie intake with current parametrs (Female) | Lvl activity = 1", () => {

        const \_testCalorie = 1759;

        expect(CalcDailyCalorieIntakeFEMALE\_TEST2).to.be.equal(\_testCalorie);

    });

    it("shold return int number (1982) | daily calorie intake with current parametrs (Female) | Lvl activity = 2", () => {

        const \_testCalorie = 1982;

        expect(CalcDailyCalorieIntakeFEMALE\_TEST3).to.be.equal(\_testCalorie);

    });

    it("shold return int number (2206) | daily calorie intake with current parametrs (Female) | Lvl activity = 3", () => {

        const \_testCalorie = 2206;

        expect(CalcDailyCalorieIntakeFEMALE\_TEST4).to.be.equal(\_testCalorie);

    });

    it("shold return int number (2430) | daily calorie intake with current parametrs (Female) | Lvl activity = 4", () => {

        const \_testCalorie = 2430;

        expect(CalcDailyCalorieIntakeFEMALE\_TEST5).to.be.equal(\_testCalorie);

    });

    it("shold return int number (1934) | daily calorie intake with current parametrs (Male) | Lvl activity = 0", () => {

        const \_testCalorie = 1934;

        expect(CalcDailyCalorieIntakeMALE\_TEST1).to.be.equal(\_testCalorie);

    });

    it("shold return int number (2215) | daily calorie intake with current parametrs (Male) | Lvl activity = 1", () => {

        const \_testCalorie = 2215;

        expect(CalcDailyCalorieIntakeMALE\_TEST2).to.be.equal(\_testCalorie);

    });

    it("shold return int number (2497) | daily calorie intake with current parametrs (Male) | Lvl activity = 2", () => {

        const \_testCalorie = 2497;

        expect(CalcDailyCalorieIntakeMALE\_TEST3).to.be.equal(\_testCalorie);

    });

    it("shold return int number (2779) | daily calorie intake with current parametrs (Male) | Lvl activity = 3", () => {

        const \_testCalorie = 2779;

        expect(CalcDailyCalorieIntakeMALE\_TEST4).to.be.equal(\_testCalorie);

    });

    it("shold return int number (3061) | daily calorie intake with current parametrs (Male) | Lvl activity = 4", () => {

        const \_testCalorie = 3061;

        expect(CalcDailyCalorieIntakeMALE\_TEST5).to.be.equal(\_testCalorie);

    });

    it("shold return object | determine ratio of protein,carbs,fats | GOAL : LoseWeight ", () => {

        const \_testPFC = { dailyProtein: 188, dailyFat: 56, dailyCarbs: 313 };

        expect(determineRatioOfPFC\_TEST1).to.be.a("object");

        expect(JSON.stringify(determineRatioOfPFC\_TEST1)).to.be.equal(JSON.stringify(\_testPFC));

    });

    it("shold return object | determine ratio of protein,carbs,fats | GOAL : MaintainWeight ", () => {

        const \_testPFC = { dailyProtein: 188, dailyFat: 83, dailyCarbs: 250 };

        expect(determineRatioOfPFC\_TEST2).to.be.a("object");

        expect(JSON.stringify(determineRatioOfPFC\_TEST2)).to.be.equal(JSON.stringify(\_testPFC));

    });

    it("shold return object | determine ratio of protein,carbs,fats | GOAL : GainWeight ", () => {

        const \_testPFC = { dailyProtein: 219, dailyFat: 83, dailyCarbs: 344 };

        expect(determineRatioOfPFC\_TEST3).to.be.a("object");

        expect(JSON.stringify(determineRatioOfPFC\_TEST3)).to.be.equal(JSON.stringify(\_testPFC));

    });

    it("shold return int number | get body mass index test1", () => {

        const \_result = 23;

        expect(getBodyMassIndex\_TEST1).to.be.a("number").equal(\_result);

    });

    it("shold return int number | get body mass index test2", () => {

        const \_result = 25;

        expect(getBodyMassIndex\_TEST2).to.be.a("number").equal(\_result);

    });

    it("shold return int number | get body mass index test3", () => {

        const \_result = 24;

        expect(getBodyMassIndex\_TEST3).to.be.a("number").equal(\_result);

    });

});

describe("Courses module", () => {

    it("should return not empty array of exercises",()=>{

        expect(ABSCourseTEST1.data.exercises).to.be.a("array").not.be.empty;

    })

    it("should return bool (the course was created by app)",()=>{

        expect(ABSCourseTEST1.data.isCreateByUser).to.be.false;

    })

    it("should return int number | count minutes course ( with breaks)",()=>{

        const \_countCountMinutes = 22

        expect(ABSCourseTEST1.getAllTimeExercises()).to.be.equal(\_countCountMinutes);

    })

    it("should return bool (true) | founded exercise by id",()=>{

        expect(ABSCourseTEST1.currentExerciseById("abs-beginner-01")).to.be.an("object").that.is.not.empty;

    })

    it("should return undefined | not founded exercise by id",()=>{

        expect(ABSCourseTEST2.currentExerciseById("abs-beginner-01")).to.be.an("undefined");

    })

})

describe("App module",()=>{

    it("should return array | push script to AppController",()=>{

        const res = () => "Hello world";

        AppController.pushScript(res)

        expect(AppController.\_scripts).to.have.lengthOf(1)

    })

})

**Висновок:**

Я ознайомився із принципами роботи системи автоматизованого тестування JUnit (Mocha & Chai). Реалізував модульні тести, що забезпечують перевірку основної функціональності кожного класу відповідно до розробленого проекту програмного продукту. Домігся успішного виконання всіх модульних тестів.