[ASSIGNMENT 3]
[STP304X_01 KIỂM THỬ TỰ ĐỘNG]

Họ và tên : Phạm Hải Thanh
MSV : FX20171

TP HCM.2023

MỤC LỤC

KIỂM TRA PHẢN HỒI API - CÁCH 3

Giải thích: Ở cách làm này, chúng ta sẽ sử dụng file excel "TokenSave.xlsx" trong folder **ExceFilelWriten** để tiến hành lưu thông tin token.

Trình tự làm như sau:

- Đối với yêu cầu đăng nhập vào tài khoản:

B1. Lấy các thông tin cần thiết như baseUrl loginPath, accountLogin, passwordLogin từ file config.properties

B2. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra

B3. Lấy dữ liệu token (nếu có) ghi vào file excel "TokenSave.xlsx"

- Đối với tạo thông tin công việc mới cho tài khoản:

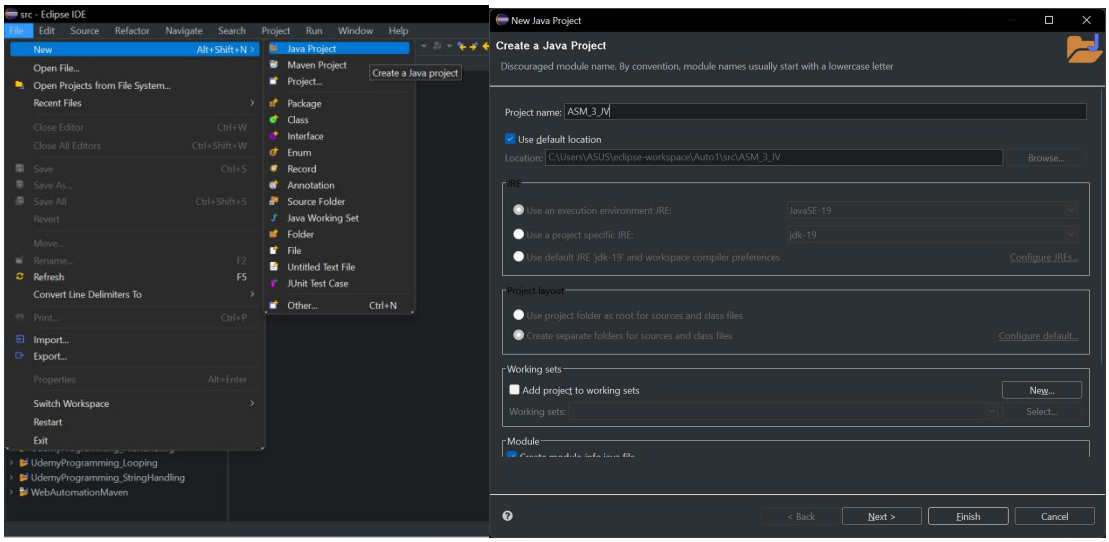
B1. Lấy dữ liệu token (nếu có) từ file "TokenSave.xlsx" ghi được ở bên trên để sử dụng làm header cho phần này

B2. Tạo thông tin công việc mới

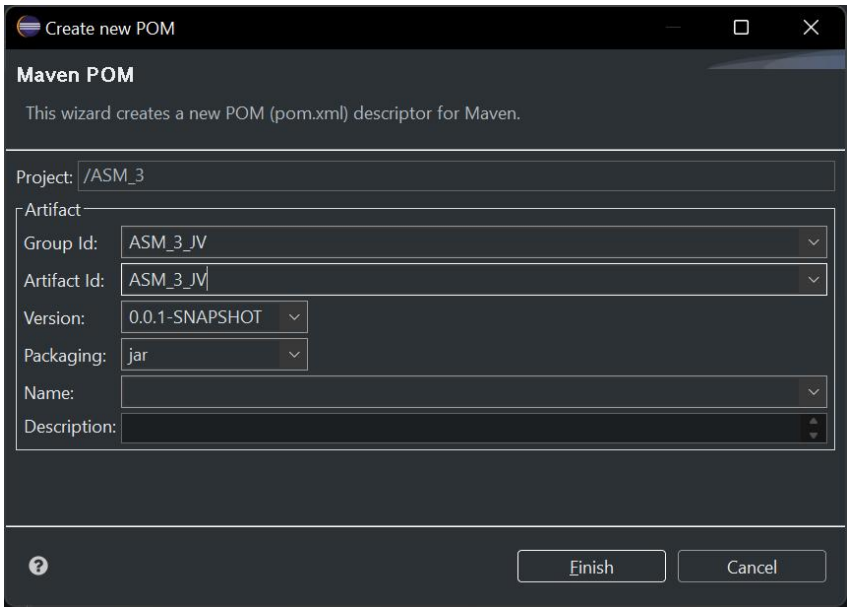
B3. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra

PHẦN 1: TẠO PROJECT, PACKAGE CHO ASM3

- B1: Chọn dạng dự án: Java Project



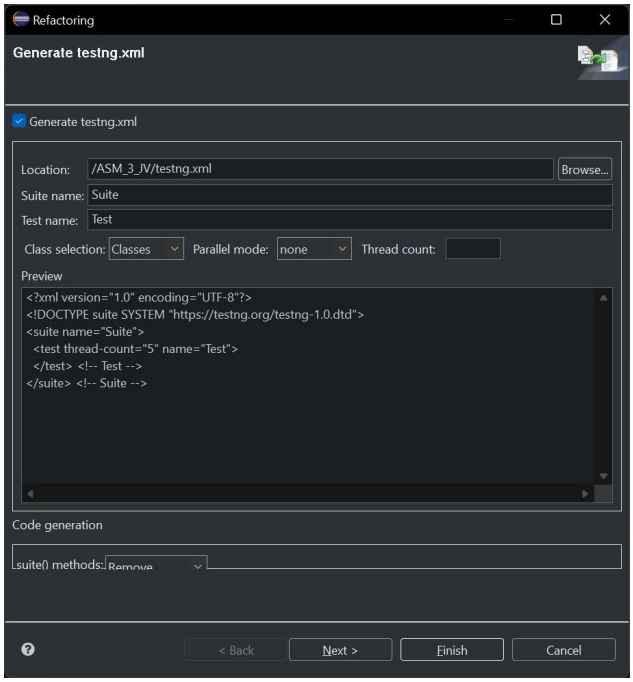
- B2: Convert sang Maven



- B3: Truy cập https://mvnrepository.com/ để cài đặt các Dependencies vào pom.xml của project để sử dụng

1. Cài poi và poi-ooxml
<dependency> <groupId>org.apache.poi</groupId> <artifactId>poi</artifactId> <version>5.2.3</version> </dependency>
<dependency> <groupId>org.apache.poi</groupId> <artifactId>poi-ooxml</artifactId> <version>5.2.3</version> </dependency>
2. Cài testng
<dependency> <groupId>org.testng</groupId> <artifactId>testng</artifactId> <version>7.4.0</version> <scope>test</scope> </dependency>
3. Cài rest-assured và dependencies hỗ trợ đọc json
<dependency> <groupId>io.rest-assured</groupId> <artifactId>rest-assured</artifactId> <version>4.4.0</version> <scope>test</scope> </dependency>
<dependency> <groupId>com.google.inject</groupId> <artifactId>guice</artifactId> <version>3.0</version> </dependency>
<dependency> <groupId>org.hamcrest</groupId> <artifactId>hamcrest</artifactId> <version>2.2</version> </dependency>
<dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> <version>2.13.4.2</version> </dependency>
<dependency> <groupId>org.apache.logging.log4j</groupId> <artifactId>log4j-core</artifactId> <version>2.20.0</version> </dependency>

- B4: Convert sang TestNG



PHẦN 2: TẠO CÁC THƯ MỤC VÀ VIẾT CÁC CÀI ĐẶT BỔ TRỢ

Tạo các thư mục chứa các file ,drive bổ trợ

Tên folder	File con	Giải thích
Properties	Config.properties	Chứa các thông tin như đường link web cần test, tên user, password để đăng nhập
ExceFilelWriten	TokenSave.xlsx	Là file token sẽ được lưu lại sau khi login thành công và sử dụng cho phần header ở CreateWork

Xác định và thêm thông tin vào Config.properties
- Config.properties chứa các thông tin URL page , username, password

```
#Maintaining Application Configuration Data

baseUrl=http://13.228.39.1:5000
loginPath=/api/users/login
createWorkPath=/api/work-user/createWork-user
accountLogin=testerFunix
passwordLogin=Abc13579
```

PHẦN 3: TẠO CÁC PACKAGES VÀ CLASSES TRONG PROJECT

Tạo 2 folder packages trong file ASM_3_JV: là src.test và src.main

Tên packages	Mục đích	Các class con
src.test		
com.automation.testcase	chứa các class test sử dụng TestNG	TC_API_Login_Excel_Writen.java
		TC_API_CreateWork_Excel_Read.java
src.main		
com.automation.untils	chứa class hỗ trợ đọc properties file ...	PropertiesFileUtils.java

3.1 Tạo package com.automation.utils

Mục đích: chứa các class hỗ trợ đọc properties file …

3.1.1 Class PropertiesFileUtils.java

- Class PropertiesFileUtils.java giúp đọc các giá trị từ file Configs.proterties trong folder Properties.

```
package com.api.auto.utils;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Properties;

public class PropertiesFileUtils {

//Tạo đường dẫn đến properties files trong folder Properties
   public static String CONFIG_PATH = "./Properties/Config.properties";

//Tạo đường dẫn đến token properties files trong folder Properties
   public static String TOKEN_PATH = "./Properties/Token.properties";

// Đọc giá trị đã lưu từ file Config.properties
   public static String getProperty(String key) {
            Properties properties= new Properties();
            String value = null;
            FileInputStream fileInput = null;

//Tạo exception nếu giá trị cần đọc rỗng
      try {
            fileInput = new FileInputStream(CONFIG_PATH);
           properties.load(fileInput);
           value = properties.getProperty(key);

          if (value != null) {              // nếu không có key, trả về một chuỗi rỗng
           value = value.trim();            // loại bỏ khoảng trắng đầu, cuối chuỗi
          }
         return value;
         }catch (Exception ex) {
         System.out.println("Xảy ra lỗi khi đọc giá trị của  " + key);
         ex.printStackTrace();
         } finally {    //luôn nhảy vào đây dù có xảy ra exception hay không.
                        if (fileInput!= null) {
```

```
                    try {
                        fileInput.close(); //thực hiện đóng luồng đọc
                    } catch (IOException e) {
                         e.printStackTrace();
                              }
                    }
        }
        return value;
    }
```

3.2 Tạo package com.automation.testcase

3.2.1 Chứa class Token.java

- Chứa class Token.java là nơi để định nghĩa Token.

```
package com.api.auto.testcase.Excel;

public class Token {
//Tạo đối tượng Token bao gồm các thông tin cơ bản: token
   private String tokenValue;
//Tạo phương thức get,set cho tokenValue
   public String gettokenValue() {
      return tokenValue;
   }
   public void settokenValue(String tokenValue) {
      this.tokenValue = tokenValue;
   }
}
```

3.2.2 Chứa class TC_API_Login_Excel_Writen.java

- Chứa class TC_API_Login_Excel_Writen.java là nơi để thực hiện việc test API login. Chúng ta sẽ sử dụng dữ liệu đã cho là URL, user, password từ **"Config.properties"** để đăng nhập và kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi. Sau đó sẽ lưu lại giá trị token vào file excel **"TokenSave.xlsx"**. Token này sẽ được sử dụng để thêm vào header ở phần CreateWork.

```
package com.api.auto.testcase.Excel;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.io.File;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import com.api.auto.utils.PropertiesFileUtils;

import io.restassured.RestAssured;
```

```
import io.restassured.http.ContentType;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;

public class TC_API_Login_Excel_Writen {
//TẠO LỆNH GHI GIÁ TRỊ TOKEN VÀO EXCEL
	public static void writeToExcelFile(ArrayList<Token> TokenSave) {

		XSSFWorkbook wk = new XSSFWorkbook();
		XSSFSheet s1 = wk.createSheet("TokenWriten");
		XSSFRow r1 = s1.createRow(0);

// Tạo và gán giá trị cho cột 0 với tên gọi là Token
		XSSFCell c1 = r1.createCell(0);
		c1.setCellValue("TokenValue");

// Tạo vòng lặp for để gán các giá trị sẽ nhập sẽ được ghi vào cột và dòng tương ứng trong excel file
		for (int i = 0; i < TokenSave.size(); i++) {
			Token std = TokenSave.get(i);
			XSSFRow row1 = s1.createRow(i + 1);
			XSSFCell cell1 = row1.createCell(0);
			cell1.setCellValue(String.valueOf(std.gettokenValue()));
		}

// Tạo lệnh báo nếu thông tin nhập đã được ghi lại trong file excel có tên là "TokenSave.xlsx"
// File excel này có đường path tại folder:"./ExceFilelWriten/TokenSave.xlsx"
		try {
			FileOutputStream out = new FileOutputStream(new
File("./ExceFilelWriten/TokenSave.xlsx"));
			wk.write(out);
			out.close();
			System.out.println("TokenSave.xlsx written successfully on disk");
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}
// ĐẶT CÁC BIẾN SỬ DỤNG TRONG TEST

   String postEndPoint = "http://13.228.39.1:5000";
   String basePath = "/api/users/login";

   private String account;
	private String password;
	private Response postresponse;
	private ResponseBody responseBody;
	private JsonPath bodyJson;
	private String token ;

	@BeforeClass
	public void init() {
// Đọc đường dẫn URL và account/password đã lưu từ file Config.properties
		String baseUrl = PropertiesFileUtils.getProperty("baseUrl");
		String loginPath = PropertiesFileUtils.getProperty("loginPath");;
		account = PropertiesFileUtils.getProperty("accountLogin");
```

```
                password = PropertiesFileUtils.getProperty("passwordLogin");

                RestAssured.baseURI = baseUrl;

// Tạo và đọc body dưới dạng JSON
                Map<String, Object> user = new HashMap<String, Object>();
                user.put("account", account);
                user.put("password", password);

    RequestSpecification request = RestAssured.given()
                 .contentType(ContentType.JSON)
                 .body(user);

    postresponse = request.post(loginPath);
                responseBody = postresponse.body();
                bodyJson = responseBody.jsonPath();

                System.out.println("RESPONSE:"+"\n" + postresponse.asPrettyString());
        }

//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
  @Test(priority = 1)
  public void T01_StatusCode200() {
// Kiểm chứng status code có là 200 hay không
    assertEquals(200, postresponse.getStatusCode(), "Status Check Failed!");
  }

  @Test(priority = 2)
  public void T02_ContainMessage() {
// Kiểm chứng response body có chứa trường message hay không
    assertTrue(postresponse.asString().contains("message"), "message field check Failed!");
  }

  @Test(priority = 3)
  public void T03_VerifyOnMatchMessage() {
// Kiểm chứng trường message có hiển thị nội dung "Đăng nhập thành công" hay không
   String resMessage = bodyJson.get("message");
    assertEquals(resMessage, "Đăng nhập thành công", "message field check Failed!");
  }

  @Test(priority = 4)
  public void TC04_ContainToken() throws Exception {
// Kiểm chứng response body có chứa trường token hay không
    token = bodyJson.getString("token");
    assertTrue(postresponse.asString().contains("token"), "Token field check Failed!");
  }

  @Test(priority = 5)
        public void TC05_SaveTokenToExcel() {

                ArrayList<Token> TokenSave = new ArrayList<Token>();
                Token token = new Token();
                token.settokenValue(bodyJson.getString("token"));
//Thêm token vào danh sách TokenSave
                TokenSave.add(token);
//Viết danh sách TokenSave vào file Excel
                writeToExcelFile(TokenSave);
```

```
                    System.out.println("Token save successfully");
            }

    @Test(priority = 6)
    public void TC06_ContainUser() {
// Kiểm chứng response chứa thông tin user hay không
        assertTrue(postresponse.asString().contains("user"), "User field check Failed!");
            }

    @Test(priority = 7)
    public void TC07_ContainType() {
// Kiểm chứng response body có chứa thông tin trường type hay không
        assertTrue(postresponse.asString().contains("type"), "Type field check Failed!");
            }

    @Test(priority = 8)
    public void TC08_VerifyOnMatchType() {

// Kiểm chứng trường type nội dung có phải là “UNGVIEN” hay không
                    assertEquals(bodyJson.getString("user.type"),"UNGVIEN", "Type check Failed!");
            }

    @Test(priority = 9)
    public void TC09_ContainAccount() {
// Kiểm chứng response body có chứa thông tin trường account hay không
        assertTrue(postresponse.asString().contains("account"), "account field check Failed!");
            }

    @Test(priority = 10)
    public void TC10_VerifyOnMatchAccount() {
// Kiểm chứng trường account có khớp với account đăng nhập đã cho hay không
      assertEquals(bodyJson.getString("user.account"), "testerFunix","Account check Failed!");
            }

    @Test(priority = 11)
    public void TC11_ValidatePassword()  {
 // Kiểm chứng response chứa thông tin trường password hay không
        assertTrue(postresponse.asString().contains("password"), "Type field check Failed!");
            }

    @Test(priority = 12)
    public void TC12_VerifyOnMatchPassword() {
// Kiểm chứng trường password có khớp với password đăng nhập đã cho hay không
      assertEquals(bodyJson.getString("user.password"), "Abc13579","Password check Failed!");
            }
}
```

3.2.3 Chứa class TC_API_CreateWork_Excel_Read.java

- Class TC_API_CreateWork_Excel_Read.java là nơi để thực hiện việc tạo công việc. Chúng ta sẽ sử dụng dữ liệu token ở file excel đã lấy được từ lúc login thành công để làm header và tạo dữ liệu. Sau khi tạo dữ liệu, nhận được phản hồi thì sẽ kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi

```
package com.api.auto.testcase.Excel;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import io.restassured.RestAssured;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;

public class TC_API_CreateWork_Excel_Read {

// Tạo các biến sử dụng trong test

   String postEndPoint = "http://13.228.39.1:5000";
   String createWorkPath = "/api/work-user/createWork-user";

   RequestSpecification request;
   Response postresponse;
   ResponseBody responseBody;
   JsonPath bodyJson;
   String jsonContent;
   private String token ;
   File jsonBody;

 // Đọc giá trị từ file excel TokenSave.xlsx
   private String readExcelValue(String filePath, String sheetName, int rowIndex, int cellIndex) throws
IOException {
      FileInputStream fis = new FileInputStream("./ExceFilelWriten/TokenSave.xlsx");
      XSSFWorkbook wk = new XSSFWorkbook(fis);
      XSSFSheet s1 = wk.getSheet("TokenWriten");
      XSSFRow r1 = s1.getRow(rowIndex);
      XSSFCell c1 = r1.getCell(cellIndex);
      String token = c1.getStringCellValue();
      wk.close();
      fis.close();
      return token;
      }

   @BeforeMethod
   public void setConfiguration() throws IOException {

      String tokenFilePath = "./ExceFilelWriten/TokenSave.xlsx";
```

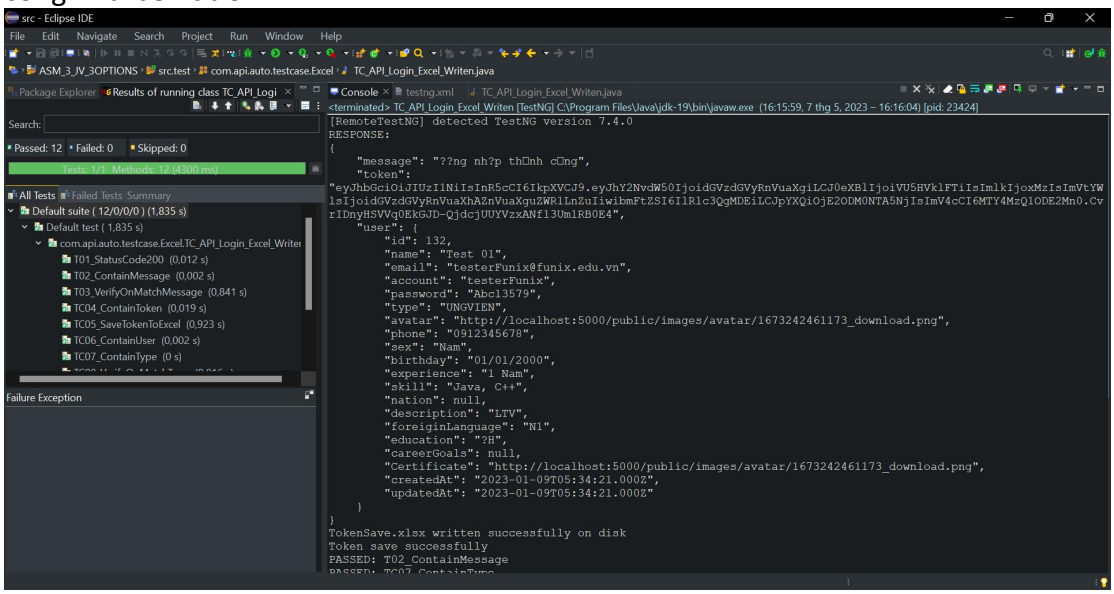
```
        String sheetName = "TokenWriten";
        int rowIndex = 1; // Assuming the token value is stored in the second row
        int cellIndex = 0; // Assuming the token value is stored in the first cell of the row
        this.token = readExcelValue(tokenFilePath, sheetName, rowIndex, cellIndex);

        String inputData = "";

        File jsonBody = new File("./RequestPayload/CreateWork.json");
        FileReader fr = new FileReader(jsonBody);
        BufferedReader br = new BufferedReader(fr);
        jsonContent = br.readLine();

        while (jsonContent != null) {
           inputData = inputData + jsonContent;
           jsonContent = br.readLine();
        }

        request = RestAssured.given();
        request.baseUri(postEndPoint);
        request.basePath(createWorkPath);

// Lưu giá trị token đã đọc vào header
        request.header("token", token);
        request.header("content-type", "application/json");

        request.body(inputData);

        postresponse = request.post();
        responseBody = postresponse.getBody();
        bodyJson = responseBody.jsonPath();
             }

//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
   @Test(priority = 1)
   public void TC01_Validate201Created() {
// Kiểm chứng status code có là 201 hay không
        assertEquals(postresponse.getStatusCode(), 201);
        System.out.println("\nStatus code is: " + postresponse.getStatusCode()+"\n");
        System.out.println(" " + postresponse.asPrettyString());
   }

   @Test(priority = 2)
   public void TC02_ContainWorkId() {
 // Kiểm chứng response body có chứa trường id hay không
        assertTrue(postresponse.asString().contains("id"), "Id field check Failed!");
   }

   @Test(priority = 3)
   public void TC03_ContainNameOfWork() {
// Kiểm chứng response body có chứa trường nameWork hay không
        assertTrue(postresponse.asString().contains("nameWork"), "nameWork field check Failed!");
   }

   @Test(priority = 4)
   public void TC04_ValidateNameOfWorkMatched() {
// Kiểm chứng trường nameWork có khớp với thông tin đã tạo hay không
        assertEquals(bodyJson.getString("nameWork"), "Giang vien","nameWork check Failed!");
```

```
    }

    @Test(priority = 5)
    public void TC05_ContainExperienceOfWork() {
// Kiểm chứng response body có chứa trường experience hay không
        assertTrue(postresponse.asString().contains("experience"), "experience field check Failed!");
    }

    @Test(priority = 6)
    public void TC06_ValidateExperienceMatched() {
// Kiểm chứng trường experience có khớp với thông tin myExperience đã tạo hay không
        assertEquals(bodyJson.getString("experience"), "3","experience check Failed!");
    }

    @Test(priority = 7)
    public void TC07_ContainEducationOfUser() {
// Kiểm chứng response body có chứa trường education hay không
        assertTrue(postresponse.asString().contains("education"), "education field check Failed!");
    }

    @Test(priority = 8)
    public void TC08_ValidateEducationMatched() {
// Kiểm chứng trường education có khớp với thông tin myEducation đã tạo hay không
        assertEquals(bodyJson.getString("education"), "Thac si","education check Failed!");
    }
}
```

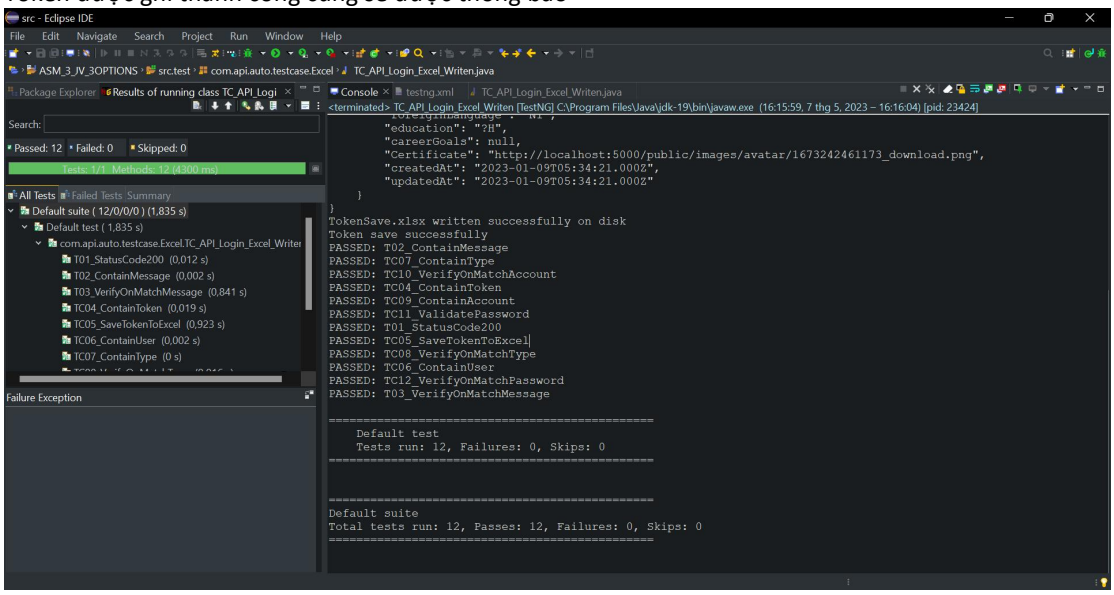
PHẦN 4: KẾT QUẢ CHẠY CHƯƠNG TRÌNH

4.1 Trường hợp TC_API_Login_Excel_Writen.java

Khi chạy chương trình sẽ in ra các thông tin hiển thị của phản hồi sau khi login thành công như bên dưới.



Token được ghi thành công cũng sẽ được thông báo



4.2 Trường hợp TC_API_CreateWork_Excel_Read.java