

BỘ LAO ĐỘNG THƯƠNG BINH & XÃ HỘI
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THÔNG TIN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN - ĐIỆN TỬ



BÁO CÁO ĐỒ ÁN

MÔN CÔNG NGHỆ KIỂM THỦ ỨNG DỤNG

KIỂM THỦ WEBSITE

GIẤY BỘM

Giảng viên hướng dẫn: Lê Thị Quỳnh Chi

Sinh viên thực hiện: Pham Hồng Huy MSSV: 501210037

Huỳnh Phúc Công Anh **MSSV: 501210009**

Đăng Thi Khánh Hiền MSSV: 501210004

Tên nhóm học phần: 015 CNKTUD HK1.22-23 CD21CT1

Hoc ky: 1 Năm hoc: 2022-2023

Tp.Hồ Chí Minh, tháng 11 năm 2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Tp. HCM, ngày....tháng.....năm.....

(chữ ký và ghi rõ họ tên)

MỤC LỤC

1.	GIỚI THIỆU.....	1
1.1.	Mô tả yêu cầu.	1
2.	NỘI DUNG.....	3
2.1.	Các yêu cầu.	3
2.1.1.	Yêu cầu B:.....	3
2.1.1.1.	Dựa vào trang web lựa chọn, thiết kế database cho phù hợp với dữ liệu trên web.	3
2.1.1.2.	Làm đầy đủ các quy trình có thể có (giống như thao tác trên web).	4
2.1.2.	Yêu cầu C: Selenium.	14
2.1.2.1.	Cài đặt môi trường JDK, ChromeDriver và Selenium Webdriver:	14
2.1.2.2.	Kiểm thử tự động Unit Test:	18
2.1.2.3.	Kiểm thử tự động Integration Test:	21
2.1.2.3.1.	Test case: Chức năng đăng nhập.....	21
2.1.2.3.2.	Test case: Chức năng đăng xuất.....	22
2.1.2.3.3.	Test case: Chức năng làm trống giỏ hàng khi người dùng đăng xuất	22
2.1.2.4.	Kiểm thử tự động System:	23
2.1.2.4.1.	Test case: Chức năng tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong giỏ	23
2.1.3.	Yêu cầu D: Kiểm thử hiệu năng đối với 3 thông số sau cho trang web mà nhóm lựa chọn.	24
PHẦN 3: TỔNG KẾT.....		32
TÀI LIỆU THAM KHẢO		33

DANH MỤC HÌNH ẢNH , BIỂU ĐỒ

hình 1: mẫu data 1 sản phẩm	3
hình 2: mẫu data 1 người dùng	3
hình 3: mẫu data 1 đơn hàng.....	4
hình 4: đăng ký thành công	4
hình 5: lỗi đăng ký với thông tin email và pass đã tồn tại với test status code 401.....	5
hình 6: đăng nhập thành công với code 200	6
hình 7: đăng nhập thất bại.....	7
hình 8: lấy tất cả sản phẩm và kiểm tra status code 200	8
hình 9: lấy ra 1 sản phẩm bằng id và kiểm tra status code 200	8
hình 10: lấy toàn bộ đơn hàng thành công lưu mã đơn hàng đầu tiên vào biến đơn hàng	9
hình 11: lấy đơn hàng theo mã đơn hàng đã lưu. check status code 200.....	10
hình 12: thêm đơn hàng thành công, trả về số lượng đơn hàng hiện có	11
hình 13: xoá đơn hàng bằng mã đơn hàng. code thành công 200.....	12
hình 14: cập nhật đơn hàng thành công	13
hình 15: kết quả chạy newman	14
hình 16: thiết lập môi trường jdk.....	15
hình 17: cài đặt driver chromium	16
hình 18: giải nén và đặt file chromedriver.exe vào resources của project.....	16
hình 19: , chọn nút add để thêm selenium vừa tải	17
hình 20: 2.1.2.2.1.test case: kiểm tra trường tên không được phép trống 1	18
hình 21: 2.1.2.2.1.test case: kiểm tra trường tên không được phép trống 2	18
hình 22: 2.1.2.2.2.test case: kiểm tra trường tên không được chứa số (1)	19
hình 23: 2.1.2.2.2.test case: kiểm tra trường tên không được chứa số (2)	19
hình 24: test case: kiểm tra trường tên không được chứa kí tự đặc biệt	20
hình 25: test case: kiểm tra trường tên không được có khoảng trắng ở đầu	20
hình 26: 2.1.2.2.5.test case: kiểm tra trường họ không được phép trống	20
hình 27: test case: chức năng đăng nhập.....	21
hình 28: test case: chức năng đăng xuất.....	22
hình 29: test case: chức năng làm trống giỏ hàng khi người dùng đăng xuất.....	23
hình 30: kiểm tra lại số lượng món hàng.....	23
hình 31: 2.1.2.4.1.test case: chức năng tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong giỏ (1)	23
hình 32: 2.1.2.4.1.test case: chức năng tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong giỏ (2)	24
hình 33: khởi động và tạo test plan đặt tên là giaybom trong jmeter.....	24
hình 34: add jmeter elements	25
hình 35:điền thông tin	25
hình 36: add listener	26
hình 37: kết quả (1)	27
hình 38: kết quả (2)	27
hình 39: thay đổi số lượng 100 user	28
hình 40: kết quả 100 user (1).....	28
hình 41: kết quả 100 user (2).....	29
hình 42: tăng lên 1000 user.....	29
hình 43: kết quả 1000 user (1).....	30
hình 44: kết quả 1000 user (2).....	30

1. GIỚI THIỆU.

1.1. Mô tả yêu cầu.

Đò án kết thúc môn Công nghệ kiểm thử ứng dụng cần đạt được các yêu cầu do giáo viên đặt ra như sau:

A. Manual Testing: (Đã làm KTGK)

SV làm tất cả các testcase có thể có cho các chức năng trong trang web đã lựa chọn cho 3 level: unit test, intergration test và system test. Testcase nào bug thì tô màu đỏ.(Lưu ý: tạo 3 sheet riêng cho từng level.)

Do số lượng testcase khá nhiều nên đối với phần Manual testing các thành viên phải phân chia nhau để viết testcase. Trong sheet "phân công" ghi rõ thành viên nào đảm nhận test chức năng nào, v.vv Sử dụng template đính kèm để hoàn thành phần A.

B. API: (4đ)

- Dựa vào trang web lựa chọn, thiết kế database cho phù hợp với dữ liệu trên web.
- Làm đầy đủ các quy trình có thể có (giống như thao tác trên web)
- Viết testcase như check status code, check value trả về, kiểm tra giá trị input, v.vv
- Chạy runner + newman

C. Selenium: (4đ)

- Những task nào làm auto selenium trong file testcase thì note vào "Yes"
- Quay video chạy các testcase auto -> tải lên google drive và nộp link vào assignment.
- Nén file source code đồ án, tạo folder Selenium và tải toàn bộ lên link github repository của nhóm.

D. Kiểm thử hiệu năng đối với 3 thông số sau cho trang web mà nhóm lựa chọn: (1đ)

- Thread group (users): 10/100/1000
- Ram up: 10
- Loop: 1

Sau khi chạy jmeter, dựa vào Listener là View Results Tree/Graph Results/ Aggregate Report, đưa ra các kết luận, nhận xét về kết quả đạt được cũng như giải pháp đề xuất để nâng cấp hệ thống (nếu có).

2. NỘI DUNG.

2.1. Các yêu cầu.

Đường link website: <https://giaybom.com/>

Tiến hành các bước kiểm thử dựa trên trang Web đã chọn

2.1.1. Yêu cầu B:

2.1.1.1. *Dựa vào trang web lựa chọn, thiết kế database cho phù hợp với dữ liệu trên web.*

Database thiết kế theo website gồm có:

- Sản phẩm chứa các thông tin như: ID, Loại, Tên, Giá.

```
{  
    "id": 1,  
    "type": "GIÀY BOOTS",  
    "title": "[MUA 1 TẶNG 1] [LIMITED] GB Boots B1825",  
    "price": 47000000.0  
},
```

Hình 1: Mẫu data 1 sản phẩm

- Người dùng chứa các thông tin: ID, Tên người dùng, Email, Mật khẩu.

```
{  
    "id": 4,  
    "name": "HongHuy",  
    "email": "honghuy@gmail.com",  
    "password": "123"  
},
```

Hình 2: Mẫu data 1 người dùng

- Đơn hàng gồm có các thông tin: Mã đơn hàng, mã người dùng, tên người dùng, email, địa chỉ, điện thoại, mã hàng, giá, số lượng, tổng tiền.

```
{
  "codeOrder": "order04",
  "userId": 4,
  "username": "user4",
  "email": "user@example.com",
  "address": "Ha Noi",
  "phone": "033245678",
  "id": "2",
  "name": "GB CLASSIC B1817",
  "price": 370,
  "quantity": 1,
  "total": 370
},
```

Hình 3: Mẫu data 1 đơn hàng

2.1.1.2. Làm đầy đủ các quy trình có thẻ có (giống như thao tác trên web).

Đối với người dùng, cho phép đăng ký, đăng nhập. Yêu cầu đăng ký không trùng với tài khoản hiện có và email đã được dùng cho tài khoản khác. Kiểm tra đăng ký thành công với Status code 200.

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** {{url}}/register
- Body (JSON):**

```

1 {
2   "name": "HongHuy",
3   "email": "honghuy@gmail.com",
4   "password": "1234567"
5 }
```
- Response Body (Pretty JSON):**

```

1 {
2   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
3 }
```

Hình 4: Đăng ký thành công

KTCK / POST Register with existed email

POST

Params Authorization Headers (9) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "name": "HongHuy",  
3   "email": "honghuy@gmail.com",  
4   "password": "1234567"  
5 }
```

Body Cookies Headers (12) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "status": 401,  
3   "message": "User already exist"  
4 }
```

Hình 5: Lỗi đăng ký với thông tin email và pass đã tồn tại với test Status code 401.

Đăng nhập bằng email và pass thành công kiểm tra Status code 200 sau đó lưu token vào Collection Variable.

KTCK / POST Login

The screenshot shows a POST request to `http://{{url}}/auth/login`. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "email": "honghuy@gmail.com",  
3   "password": "1234567"  
4 }
```

The Response tab shows a JSON object with one key:

```
1 {  
2   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWl  
3 }
```

Hình 6: Đăng nhập thành công với code 200
Thất bại với Status code 401.

KTCK / POST Login with Wrong Email

POST {{url}}/auth/login

Params Authorization Headers (9) Body Pre-request Script Tests

Query Params

	KEY	VALUE
	Key	Value

Body Cookies Headers (12) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "status": 401,  
3   "message": "Incorrect email or password"  
4 }
```

Hình 7: Đăng nhập thất bại

Đối với sản phẩm. Thiết kế tính năng yêu cầu sản phẩm theo mã hàng hoặc lấy tất cả và Status code trả về 200. Sau đó lấy id của sản phẩm đầu tiên lưu vào Collection Variable.

The screenshot shows a Postman request for 'GET All Product'. The URL is {{url}}/products/. The 'Body' tab is selected, showing a JSON response with two items:

```

1 [
2   {
3     "id": 1,
4     "type": "GIÀY BOOTS",
5     "title": "[MUA 1 TẶNG 1] [LIMITED] GB Boots B1825",
6     "price": 47000000
7   },
8   {
9     "id": 2,
10    "type": "GIÀY BOOTS",
11    "title": "[MUA 1 TẶNG 1] [LIMITED] GB Boots B1826",
12    "price": 48000000
13  }

```

Hình 8: Lấy tất cả sản phẩm và kiểm tra Status code 200

The screenshot shows a Postman request for 'GET One Product'. The URL is {{url}}/products/{{id}}. The 'Body' tab is selected, showing a JSON response with one item:

```

1 {
2   "id": 1,
3   "type": "GIÀY BOOTS",
4   "title": "[MUA 1 TẶNG 1] [LIMITED] GB Boots B1825",
5   "price": 47000000
6 }

```

Hình 9: Lấy ra 1 sản phẩm bằng id và kiểm tra Status code 200

Đối với đơn hàng, yêu cầu phải xác thực bằng Bearer Token với giá trị token đã lưu sau khi đăng nhập thành công. Cho phép xem tất cả đơn hàng, thêm, xoá, sửa đơn hàng.

KTCK / GET All Order

GET [Body Cookies Headers \(12\) Test Results \(1/1\)

Pretty Raw Preview Visualize JSON ↻

```

1 \[
2 {
3     "codeOrder": "order04",
4     "userId": 4,
5     "username": "user4",
6     "email": "user@example.com",
7     "address": "Ha Noi",
8     "phone": "033245678",
9     "id": "2",
10    "name": "GB CLASSIC B1817",
11    "price": 370,
12    "quantity": 1,
13    "total": 370
14 },
15 {
16     "codeOrder": "order03",
17     "userId": 4,
18     "username": "user4",
19     "email": "user@example.com",
20     "address": "HCM",
21     "phone": "03324567899",
22     "id": "2",
23     "name": "GB CLASSIC B1817",
24     "price": 370,
25     "quantity": 1,
26     "total": 370

```]({{url}}/auth/orders</a></p>
<p>Params Authorization ● Headers (8) Body Pre-request Script Tests ●</p>
<pre>
1 pm.collectionVariables.set()

Hình 10: Lấy toàn bộ đơn hàng thành công lưu mã đơn hàng đầu tiên vào biến đơn hàng

KTCK / GET One Order

GET {{url}}/auth/orders/{{codeOrder}}

Params Authorization (8) Headers (8) Body Pre-request Script Tests (1)

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(200);  
3 });
```

Body Cookies Headers (12) Test Results (1/1)

Pretty Raw Preview Visualize JSON

```
1 {  
2     "codeOrder": "order04",  
3     "userId": 4,  
4     "username": "user4",  
5     "email": "user@example.com",  
6     "address": "Ha Noi",  
7     "phone": "033245678",  
8     "id": "2",  
9     "name": "GB CLASSIC B1817",  
10    "price": 370,  
11    "quantity": 1,  
12    "total": 370  
13 }
```

Hình 11: Lấy đơn hàng theo mã đơn hàng đã lưu. Check Status code 200

KTCK / POST Add Order

POST {{url}}/auth/orders/

Params Authorization Headers (10) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "codeOrder": "order01",  
3   "userId": 4,  
4   "username": "user4",  
5   "email": "user@example.com",  
6   "address": "HCM",  
7   "phone": "03324567899",  
8   "id": "2",  
9   "name": "GB CLASSIC B1817",  
10  "price": 370,  
11  "quantity": 1,  
12  "total": 370  
13 }
```

Body Cookies Headers (12) Test Results (1/1)

Pretty Raw Preview Visualize JSON

1 3

Hình 12: Thêm đơn hàng thành công, trả về số lượng đơn hàng hiện có

The screenshot shows the Postman interface for a DELETE request to delete an order. The URL is `http://{{url}}/auth/orders/{{codeOrder}}`. The 'Tests' tab contains a script to check if the status code is 200. The 'Body' tab shows a JSON response with fields like codeOrder, userId, username, email, address, phone, id, name, price, quantity, and total.

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(200);  
3});
```

```
1 {  
2     "codeOrder": "order04",  
3     "userId": 4,  
4     "username": "user4",  
5     "email": "user@example.com",  
6     "address": "Ha Noi",  
7     "phone": "033245678",  
8     "id": "2",  
9     "name": "GB CLASSIC B1817",  
10    "price": 370,  
11    "quantity": 1,  
12    "total": 370  
13}
```

Hình 13: Xoá đơn hàng bằng mã đơn hàng. Code thành công 200

KTCK / PUT Update Order

PUT `{url}/auth/orders/{codeOrder}`

Params Authorization Headers (10) Body Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1
2   "codeOrder": "order03",
3   "userId": 3,
4   "username": "user3",
5   "email": "user@example.com",
6   "address": "HCM",
7   "phone": "03324567899",
8   "id": "2",
9   "name": "GB CLASSIC B1817",
10  "price": 370,
11  "quantity": 1,
12  "total": 370
13

```

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1
2   "codeOrder": "order03",
3   "userId": 3,
4   "username": "user3",
5   "email": "user@example.com",
6   "address": "HCM",
7   "phone": "03324567899",
8   "id": "2",
9   "name": "GB CLASSIC B1817",
10  "price": 370,
11  "quantity": 1,
12  "total": 370
13

```

Hình 14: Cập nhật đơn hàng thành công
Chạy runner + newman

```

KTCK

→ POST Register
POST localhost:3000/register [200 OK, 580B, 77ms]
✓ Status code is 200

→ POST Register with existed email
POST localhost:3000/register [401 Unauthorized, 426B, 5ms]
✓ Status code is 401

→ POST Login
POST localhost:3000/auth/login [200 OK, 580B, 5ms]
✓ Status code is 200

→ POST Login with Wrong Email
POST localhost:3000/auth/login [401 Unauthorized, 435B, 4ms]

→ GET All Product
GET localhost:3000/products/ [200 OK, 915B, 8ms]
✓ Status code is 200

→ GET One Product
GET localhost:3000/products/1 [200 OK, 511B, 7ms]
✓ Status code is 200

→ GET All Order
GET localhost:3000/auth/orders [200 OK, 893B, 3ms]
✓ Status code is 200

→ GET One Order
GET localhost:3000/auth/orders/order04 [200 OK, 599B, 3ms]
✓ Status code is 200

→ POST Add Order
POST localhost:3000/auth/orders/ [200 OK, 361B, 3ms]
✓ Status code is 200

→ PUT Update Order
PUT localhost:3000/auth/orders/order04 [200 OK, 598B, 5ms]
✓ Status code is 200

→ DELETE Delete Order
DELETE localhost:3000/auth/orders/order04 [200 OK, 598B, 3ms]
✓ Status code is 200


```

| | executed | failed |
|----------------------------|--------------|--------|
| iterations | 1 | 0 |
| requests | 11 | 0 |
| test-scripts | 22 | 0 |
| prerequest-scripts | 14 | 0 |
| assertions | 18 | 0 |
| total run duration: | 300ms | |

Hình 15: Kết quả chạy Newman

2.1.2. Yêu cầu C: Selenium.

2.1.2.1. Cài đặt môi trường JDK, ChromeDriver và Selenium Webdriver:

2.1.2.1.1. Cài đặt môi trường JDK:

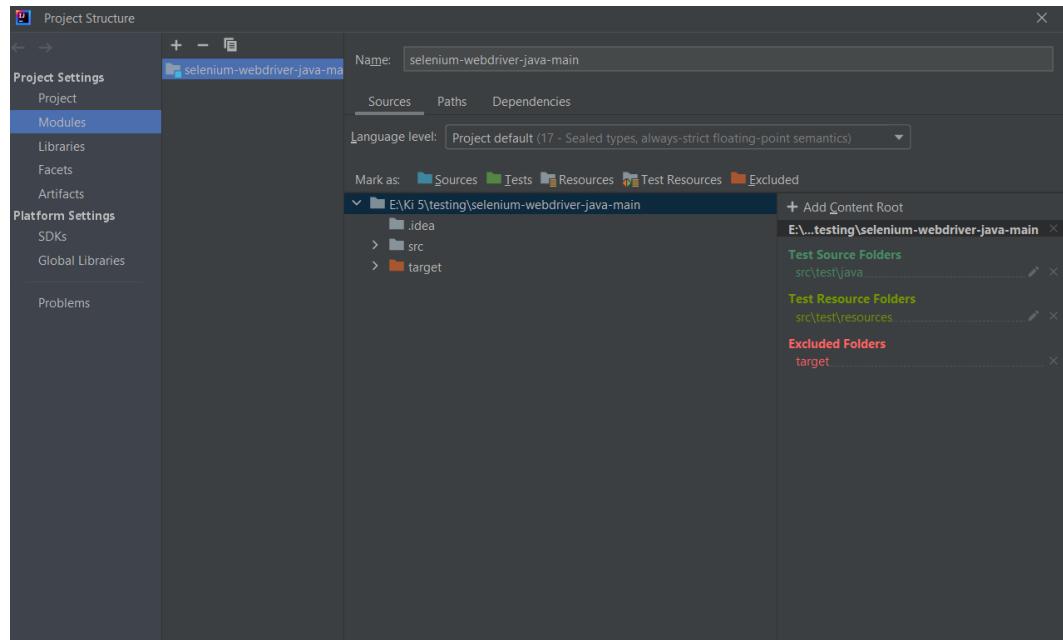
Truy cập vào [Java Downloads | Oracle](#) để download môi trường jdk.

Sau đó giải nén và chạy chương trình để Install

2.1.2.1.2. Cài đặt IDE:

Ở đây nhóm em sử dụng IntelliJ để viết code nên các bước cài đặt như sau:

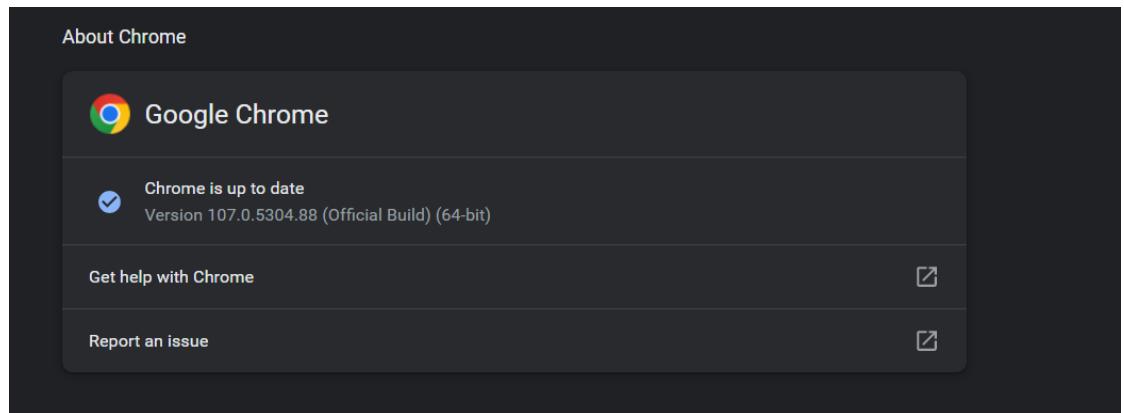
- Truy cập vào [Download IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains](#) để download phần mềm và cài đặt chúng
- Mở IntelliJ, mở project mới và thiết lập môi trường tương ứng với jdk đã tải về:



Hình 16: Thiết lập môi trường JDK

2.1.2.1.3. Cài đặt Chrome Driver:

Mở Google Chrome và vào phần Setting kiểm tra version của trình duyệt web đang sử dụng để download driver tương ứng. Đường dẫn: Google Chrome > Dấu ba chấm ở góc phải trên của trình duyệt > Setting > About

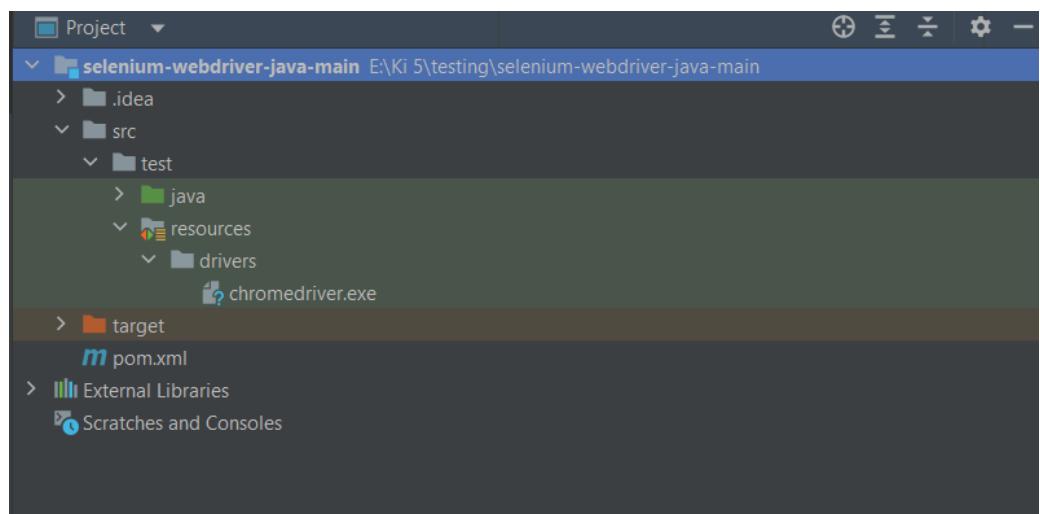


Hình 17: Cài đặt Driver Chromium

Ở đây sử dụng phiên bản 107...

Tiếp theo vào <https://chromedriver.chromium.org/downloads> để tải xuống chrome driver.

Giải nén và đặt file chromedriver.exe vào resources của project:



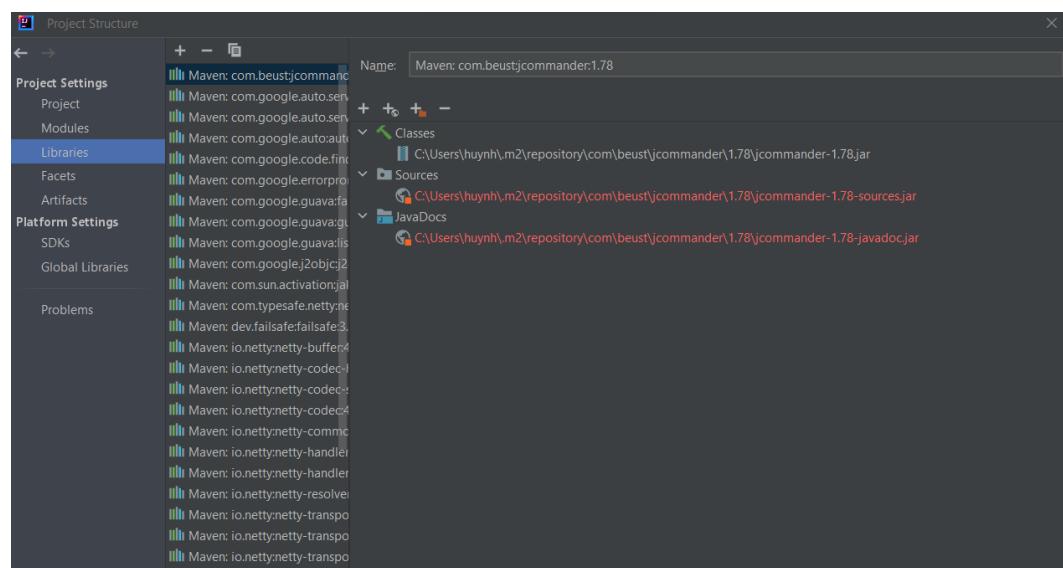
Hình 18: Giải nén và đặt file chromedriver.exe vào resources của project

2.1.2.1.4. Cài đặt Selenium Webdriver:

Trước hết tải Selenium Webdriver về tại đường dẫn: [\[Selenium Java\] TÀI NGUYÊN CÀI ĐẶT MÔI TRƯỜNG | Anh Tester](#). Đây là file có đuôi là *.jar và hãy nhớ đường dẫn của nó sau khi tải về.

Trong IntelliJ, click chuột phải vào project đang làm việc và chọn Open Module Settings.

Tại tab Libraries, chọn nút add để thêm selenium vừa tải về vào bằng đường dẫn của nó:



Hình 19: , chọn nút add để thêm selenium vừa tải

Vậy là đã hoàn tất cho việc chuẩn bị môi trường và công cụ để xây dựng các testcases chạy auto rồi nhé! Tiếp theo ta hãy điểm sơ qua một vài ví dụ minh họa của từng hạng mục Manual Testing như Unit test, Integration test, System test.

Lưu ý: đây chỉ là một vài minh họa, toàn bộ source nằm tại link github:
[phamhonghuy/DACK-1 \(github.com\)](https://github.com/phamhonghuy/DACK-1)

2.1.2.2. Kiểm thử tự động Unit Test:

2.1.2.2.1. Test case: Kiểm tra trường Tên không được phép trống

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng kí
- Tại trường Tên, không nhập bất cứ gì và điền tất cả trường còn lại
- Nhấn Submit để Đăng kí

Code đã xây dựng:

```
@Test
public class TC9 {
    public static void tc9(){
        WebDriver driver = driverFactory.getChromeDriver();
        try {
            //Step 1: Goto https://giaybom.com/
            driver.get("https://giaybom.com/");
            WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));
            //Step 2: Click Register button
            WebElement registerBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("body > header:nth-child(8) > div:nth-child(1) > div > div > div > div > form > div > div > div > div > div > input"))));
            registerBtn.click();
            //Step 3: In Name field, not enter and observe
            //3.1 fill first name:
            WebElement fieldFirstName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("FirstName"))));
            fieldFirstName.sendKeys("keysToSend");
            //3.2 fill last name:
            WebElement fieldLastName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("LastName"))));
            fieldLastName.sendKeys("keysToSend");
            //3.3 fill email:
            WebElement fieldEmail = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("Email"))));
            fieldEmail.sendKeys("keysToSend");
            //3.4 fill phone:
            WebElement fieldPhone = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("Phone"))));
            fieldPhone.sendKeys("keysToSend");
        }
        //3.5 fill password:
        WebElement fieldPassword = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("CreatePassword"))));
        fieldPassword.sendKeys("keysToSend");
        Thread.sleep( millis: 3000);
        //Step 4: Click Submit
        WebElement submitBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("input[value='Đăng ký']"))));
        submitBtn.click();
        //Verify Register status
        String checkTitle = driver.getTitle();
        System.out.println(checkTitle);
        if(checkTitle.equals("Tạo tài khoản - giayBOM")){
            System.out.println("Đăng kí chưa thành công!!!");
        }else{
            System.out.println("Đã đăng kí!!!");
            System.out.println("Kết quả form không đạt yêu cầu");
        }
        Thread.sleep( millis: 3000);
    }catch(Exception e){
        e.printStackTrace();
        System.out.println("Fail!!!");
    }
    driver.close();
    driver.quit();
}
}
```

Hình 20: 2.1.2.2.1. Test case: Kiểm tra trường Tên không được phép trống 1

```
//3.5 fill password:
WebElement fieldPassword = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("CreatePassword"))));
fieldPassword.sendKeys("keysToSend");
Thread.sleep( millis: 3000);
//Step 4: Click Submit
WebElement submitBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("input[value='Đăng ký']"))));
submitBtn.click();
//Verify Register status
String checkTitle = driver.getTitle();
System.out.println(checkTitle);
if(checkTitle.equals("Tạo tài khoản - giayBOM")){
    System.out.println("Đăng kí chưa thành công!!!");
} else{
    System.out.println("Đã đăng kí!!!");
    System.out.println("Kết quả form không đạt yêu cầu");
}
Thread.sleep( millis: 3000);
} catch(Exception e){
    e.printStackTrace();
    System.out.println("Fail!!!");
}
driver.close();
driver.quit();
}
```

Hình 21: 2.1.2.2.1. Test case: Kiểm tra trường Tên không được phép trống 2

2.1.2.2.2. Test case: Kiểm tra trường Tên không được chứa số

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng kí
- Tại trường Tên, nhập ‘anh123’ và điền tất cả trường còn lại
- Nhấn Submit để Đăng kí

Code đã xây dựng:

```
@Test
public class TC10 {
    public static void tc10(){
        WebDriver driver = driverFactory.getChromeDriver();
        try {
            //Step 1: Goto https://giaybom.com/
            driver.get("https://giaybom.com/");
            WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));
            //Step 2: Click Register button
            WebElement registerBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("body > header:nth-child(8) > div:nth-child(1) > div:nth-child(2) > a"))));
            registerBtn.click();
            //Step 3: In Name field, not enter and observe
            //3.1 fill first name:
            WebElement fieldFirstName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("FirstName"))));
            fieldFirstName.sendKeys("anh123");

            //3.2 fill last name:
            WebElement fieldLastName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("LastName"))));
            fieldLastName.sendKeys("Huyễn Phúc");

            //3.3 fill email:
            WebElement fieldEmail = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("Email"))));
            fieldEmail.sendKeys("jIAshf167673asd2@gmail.com");

            //3.4 fill phone:
            WebElement fieldPhone = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("Phone"))));
            fieldPhone.sendKeys("123123123");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Hình 22: 2.1.2.2.2. Test case: Kiểm tra trường Tên không được chứa số (1)

```
//3.5 fill password:
WebElement fieldPassword = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("CreatePassword"))));
fieldPassword.sendKeys("1234567");
Thread.sleep( 3000 );
//Step 4: Click Submit
WebElement SubmitBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("input[value='Bằng kí']"))));
SubmitBtn.click();
//Verify Register status
String checkTitle = driver.getTitle();
System.out.println(checkTitle);
if(checkTitle.equals("Tạo tài khoản - giayBOM")){
    System.out.println("Đăng kí chưa thành công!!!");
} else{
    System.out.println("Oke đăng kí!!!");
    System.out.println("Kiểm lỗi form không đạt yêu cầu!");
}
Thread.sleep( 3000 );
} catch(Exception e){
    e.printStackTrace();
    System.out.println("Fail!!!");}
}
driver.close();
driver.quit();
}
}
```

Hình 23: 2.1.2.2.2. Test case: Kiểm tra trường Tên không được chứa số (2)

2.1.2.2.3. Test case: Kiểm tra trường Tên không được chứa kí tự đặc biệt

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng kí
- Tại trường Tên, nhập ‘anh!@#’ và điền tất cả trường còn lại
- Nhấn Submit để Đăng kí

Code đã xây dựng:

Các bước 1, 2 và nhấn Submit giống như trên sẽ không chụp lại. Ta chỉ thay đổi giá trị ở bước nhập dữ liệu cho trường Tên.

```
//3.1 fill first name:  
WebElement fieldFirstName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("FirstName"))));  
fieldFirstName.sendKeys(...keysToSend: "anh!@#");
```

Hình 24: Test case: Kiểm tra trường Tên không được chứa kí tự đặc biệt

2.1.2.2.4. Test case: Kiểm tra trường Tên không được có khoảng trắng ở đầu

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng kí
- Tại trường Tên, ‘ anh’ và điền tất cả trường còn lại
- Nhấn Submit để Đăng kí

Code đã xây dựng:

Các bước 1, 2 và nhấn Submit giống như trên sẽ không chụp lại. Ta chỉ thay đổi giá trị ở bước nhập dữ liệu cho trường Tên.

```
//3.1 fill first name:  
WebElement fieldFirstName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("FirstName"))));  
fieldFirstName.sendKeys(...keysToSend: " Anh");
```

Hình 25: Test case: Kiểm tra trường Tên không được có khoảng trắng ở đầu

2.1.2.2.5. Test case: Kiểm tra trường Họ không được phép trống

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng kí
- Tại trường Họ, không nhập bất cứ gì và điền tất cả trường còn lại
- Nhấn Submit để Đăng kí

Code đã xây dựng:

Các bước 1, 2 và nhấn Submit giống như trên sẽ không chụp lại. Ta chỉ thay đổi giá trị ở bước nhập dữ liệu cho trường Họ.

```
//3.2 fill last name:  
WebElement fieldLastName = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("LastName"))));  
fieldLastName.sendKeys(...keysToSend: "");
```

Hình 26: 2.1.2.2.5. Test case: Kiểm tra trường Họ không được phép trống

2.1.2.3. Kiểm thử tự động Integration Test:

2.1.2.3.1. Test case: Chức năng đăng nhập

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
- Nhấn vào nút Đăng nhập
- Tại trường Email nhập: ‘test6@gmail.com’
- Tại trường password nhập: ‘123456’
- Nhấn Submit để Đăng nhập

Code đã xây dựng:

```
@Test  
public class TC3 {  
    public static void tc3(){  
        WebDriver driver = driverFactory.getChromeDriver();  
        try{  
            //Step 1: Goto https://giaybom.com/  
            driver.get("https://giaybom.com/");  
            WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));  
            System.out.println("Opened Website");  
  
            //Step 2: Click Log in button  
            WebElement loginBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("body > header:nth-child(8) > div:nth-child(1) > a"))));  
            loginBtn.click();  
  
            //Step 3: Fill all fields  
  
            //3.1 fill email:  
            WebElement fieldEmail = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("CustomerEmail"))));  
            fieldEmail.sendKeys("huynh@gmail.com");  
  
            //3.2 fill password:  
            WebElement fieldPassword = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("CustomerPassword"))));  
            fieldPassword.sendKeys("123456");  
  
            //Step 4: Click Submit  
            WebElement SubmitBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("input[value='Đăng nhập']"))));  
            SubmitBtn.click();  
  
            //Verify Register status  
            try{  
                WebElement checkStatus = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("div[class='errors'] ul li"))));  
                String checkValue = checkStatus.getText();  
                System.out.println(checkValue);  
                if(checkValue.equals("Thông tin đăng nhập không hợp lệ.")){  
                    System.out.println("Đăng nhập thất bại !!!");  
                }  
            }catch(Exception e){  
                System.out.println("Đăng nhập thành công!!!");  
            }  
  
            Thread.sleep( millis: 3000);  
        }catch(Exception e){  
            e.printStackTrace();  
            System.out.println("Fail!!!");  
        }  
        driver.close();  
        driver.quit();  
    }  
}
```

Hình 27: Test case: Chức năng đăng nhập

2.1.2.3.2. Test case: Chúc năng đăng xuất

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
 - Nhấn vào nút Đăng nhập
 - Tại trường Email nhập: ‘test6@gmail.com’
 - Tại trường password nhập: ‘123456’
 - Nhấn Submit để Đăng nhập
 - Nhấn vào nút Đăng xuất

Code đã xây dựng:

Sau khi đăng nhập theo các bước ở testcase trên thì ta code đoạn đăng xuất như sau:

```
    WebElement logoutBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/header[1]/div[1]/div[1]/div[1]/div[1]/div[1]")));
    logoutBtn.click();
    System.out.println("Đang xuất thành công!!!");
} catch(Exception e){
    System.out.println("Đang xuất thất bại !!!");
}
```

Hình 28: Test case: *Chức năng đăng xuất*

2.1.2.3.3. Test case: *Chức năng làm trống giỏ hàng khi người dùng*

dǎng xuá̄t

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
 - Đăng nhập một tài khoản đã đăng ký
 - Thêm một vài món hàng vào giỏ
 - Đăng xuất và kiểm tra số lượng món hàng trong giỏ sau đó

Code đã xây dựng:

Sau khi đăng nhập theo các bước ở testcase trên thì ta code đoạn đăng thêm hàng và đăng xuất như sau:

Hình 29: 2.1.2.3.3. Test case: Chức năng làm trống giỏ hàng khi người dùng đăng xuất
Sau đó kiểm tra lại số lượng món hàng:

```
//Step 8: Verify
if(amountNumber.equals(amountNumber2)){
    System.out.println("Not return empty cart when user log out! That is BUG");
} else{
    System.out.println("Returned empty cart when user log out");
}
Thread.sleep(millis: 3000);
```

Hình 30: kiểm tra lại số lượng món hàng

2.1.2.4. Kiểm thử tự động System:

2.1.2.4.1. Test case: Chức năng tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong gio

Các bước kiểm tra:

- Đi đến trang <https://giaybom.com/>
 - Đăng nhập một tài khoản đã đăng ký
 - Thêm một vài món hàng vào giỏ
 - Thay đổi số lượng của một món hàng trong giỏ
 - Nhấn ‘Cập nhật giỏ hàng’ để tính lại giá tiền
 - Xác nhận lại đã được thay đổi chưa

Code đã xây dựng cho phần chỉnh sửa số lượng:

```

//Step 5: Add a product to cart:
WebElement homeBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("a[href='Quay trở về trang chủ']"))));
homeBtn.click();
WebElement viewProductBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.cssSelector("div[id='collection1'] div[class='grid-uniform md-mg-left-15 viewProductBtn.click()")));
WebElement addProductBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/div[5]/main[1]/section[1]/div[1]/div[1]/addProductBtn.click())));
Thread.sleep( 3000 );
WebElement oc = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/div[6]/div[1]/div[1]/form[1]/div[1]/table String oldcost = oc.getText();
System.out.println("Old cost: "+oldcost);
System.out.println("Old cost: "+oldcost);
WebElement inputQuantity = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/div[6]/div[1]/div[1]/form[1]/div[1]/inputQuantity.sendKeys("1"));
inputQuantity.sendKeys("1");
WebElement updateCostBtn = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/div[6]/div[1]/div[1]/form[1]/div[1]/updateCostBtn.click());
Thread.sleep( 3000 );
WebElement nc = wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.xpath( xpathExpression: "/html[1]/body[1]/div[6]/div[1]/div[1]/form[1]/div[1]/table String newcost = nc.getText();
System.out.println("New cost: "+newcost);

```

Hình 31: 2.1.2.4.1. Test case: *Chức năng tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong giỏ (1)*

Xác nhận giá tiền sau khi cập nhật:

```

    //Verify
    if(oldcost.equals(newcost)){
        System.out.println("Change quantity success!!!!");
    }
    Thread.sleep( millis: 6000);

} catch(Exception e){
}

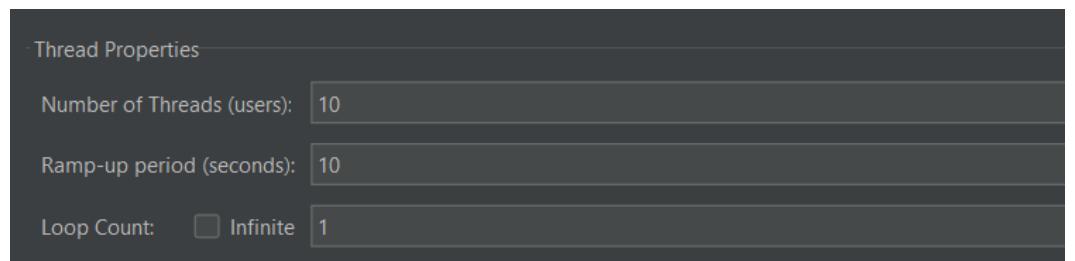
```

Hình 32: 2.1.2.4.1. Test case: Chắc chắn tính lại tổng tiền khi người dùng thay đổi số lượng món hàng trong giỏ (2)

- 2.1.3. Yêu cầu D: Kiểm thử hiệu năng đối với 3 thông số sau cho trang web mà nhóm lựa chọn.

Kiểm thử hiệu năng với JMeter

- B1. Khởi động và tạo Test Plan đặt tên là giayBom trong Jmeter.



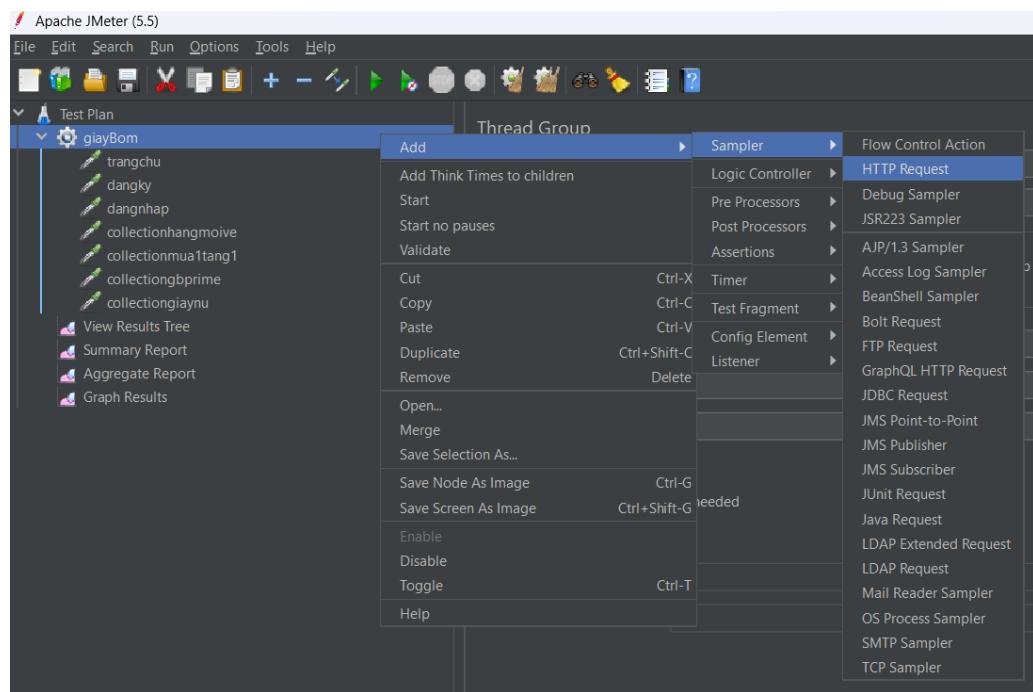
Hình 33: Khởi động và tạo Test Plan đặt tên là giayBom trong Jmeter.

- Number of Threads: 10 (Số lượng người dùng truy cập trang web: 100)
- Loop Count: 1 (Thời gian để thực hiện test)
- Ramp-Up Period: 10s (Trong vòng 10s tạo ra 10 user, tức là trung bình mỗi giây tạo 1 user)

B2: Add JMeter elements

- HTTP request: Click chuột phải Thread Group and chọn: Add ->

Sampler -> HTTP Request.



Hình 34: Add JMeter elements

Chúng ta đang thực hiện test trang giayBom.com (<https://giaybom.com/>)
nên chúng ta sẽ điền thông tin như sau

Thread Group

Name: giayBom

Comments:

Action to be taken after a Sampler error

Continue Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 10

Ramp-up period (seconds): 10

Loop Count: Infinite 1

Same user on each iteration

Delay Thread creation until needed

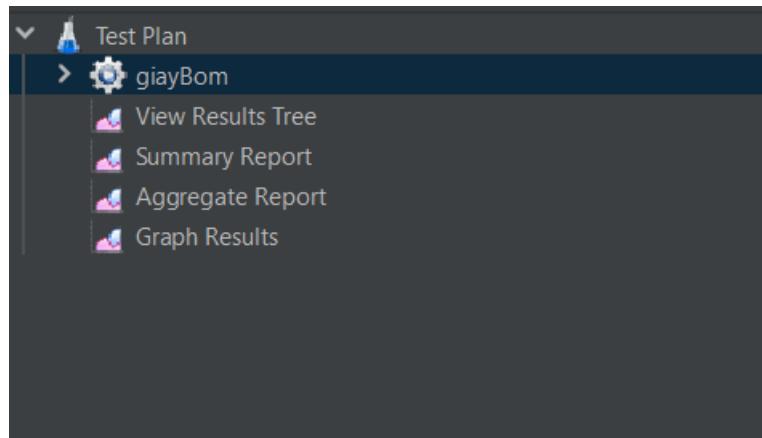
Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Hình 35:điền thông tin

B3: Add Listener Click chuột phải vào Test Plan: Add -> Listener -> Graph Results/View Results Tree/Aggregate Report. Ta sẽ có một loại các thư mục tương ứng

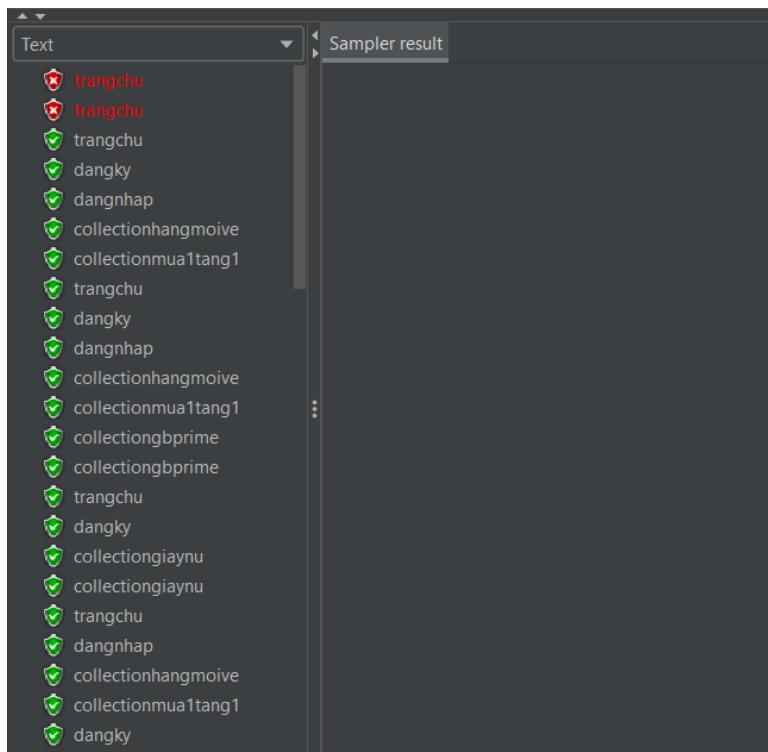


Hình 36: Add Listener

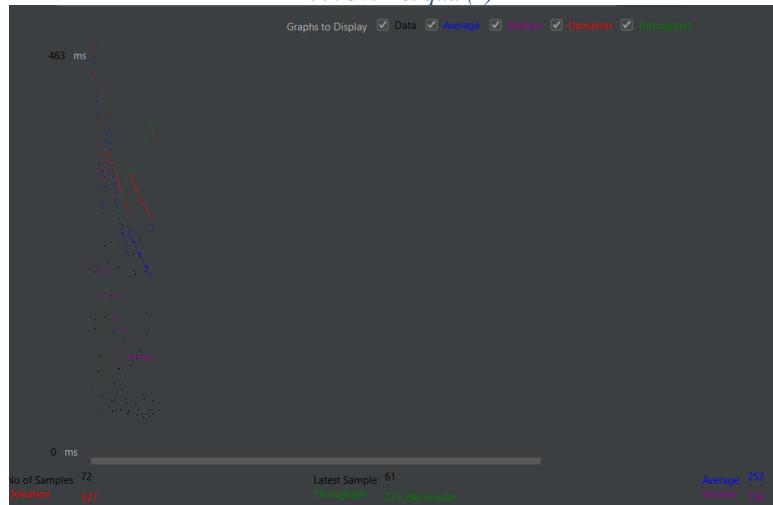
Bước 4: Run Test và xem kết quả Án nút Run (Ctrl + R) trên Toolbar để bắt đầu chạy.

Bạn sẽ nhìn thấy kết quả test hiển thị trên Graph. Ở dưới cùng của hình ảnh, bạn sẽ thấy có các số liệu thống kê và được thể hiện bằng màu sắc:

- Đen: Tổng số Sample hiện tại được gửi.
- Màu xanh dương: Trung bình hiện tại của tất cả các Sample được gửi.
- Màu đỏ: Độ lệch chuẩn hiện tại.
- Màu xanh lá cây: Tỷ lệ throughput mà đại diện cho số lượng yêu cầu trên mỗi phút mà máy chủ xử lý



Hình 37: Kết quả (1)



Hình 38: Kết quả (2)

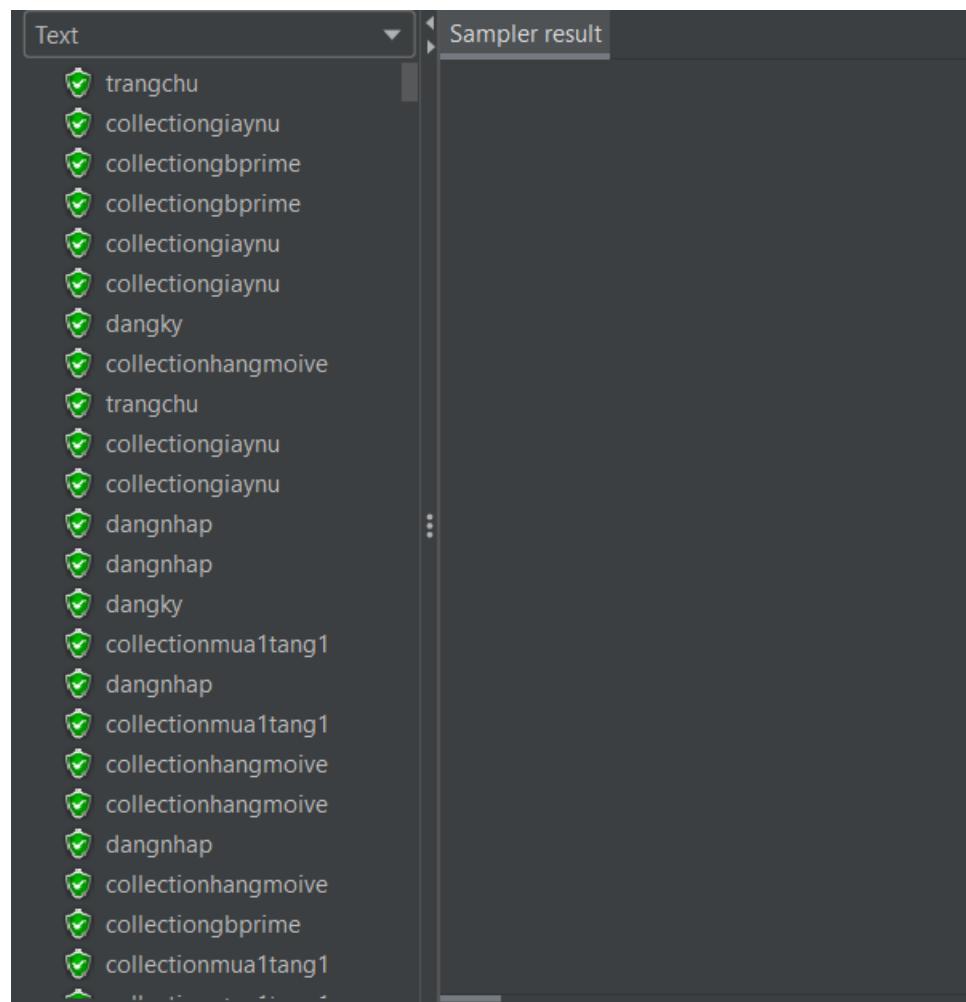
Hình ảnh trên chạy tương trưng cho trang chủ, trang đăng ký, đăng nhập, các trang collection sản phẩm tương ứng.

Sau đó ta tiến hành đổi thông số Thread group (users) thành 100 rồi 1000, để kiểm thử hiệu năng trang web với trường hợp số lượng người dùng truy cập lớn hơn trước.

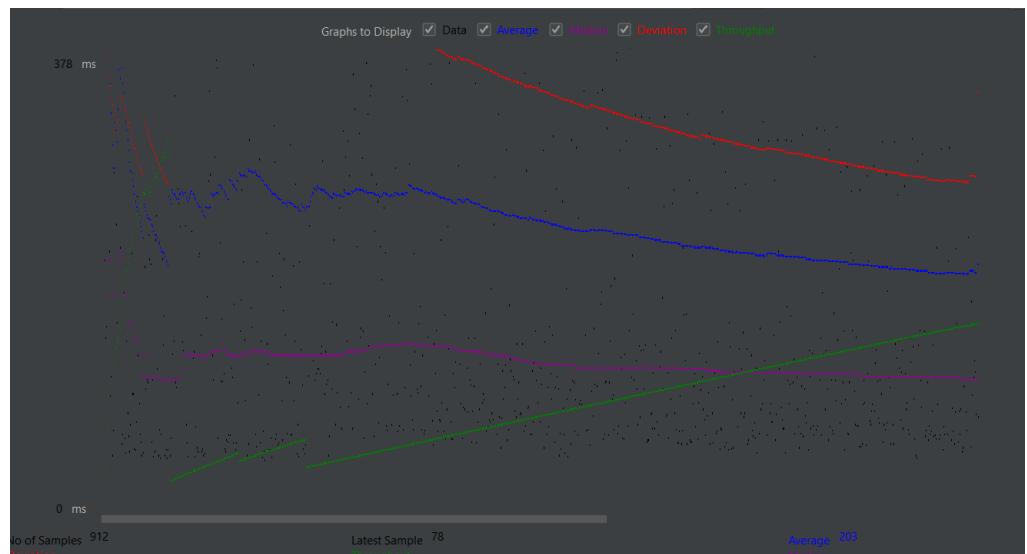


Hình 39: Thay đổi số lượng 100 user

Kết quả cho thấy trang web chạy khá mượt mà và chưa phát sinh lỗi đối với mức 100 user



Hình 40: Kết quả 100 User (1)



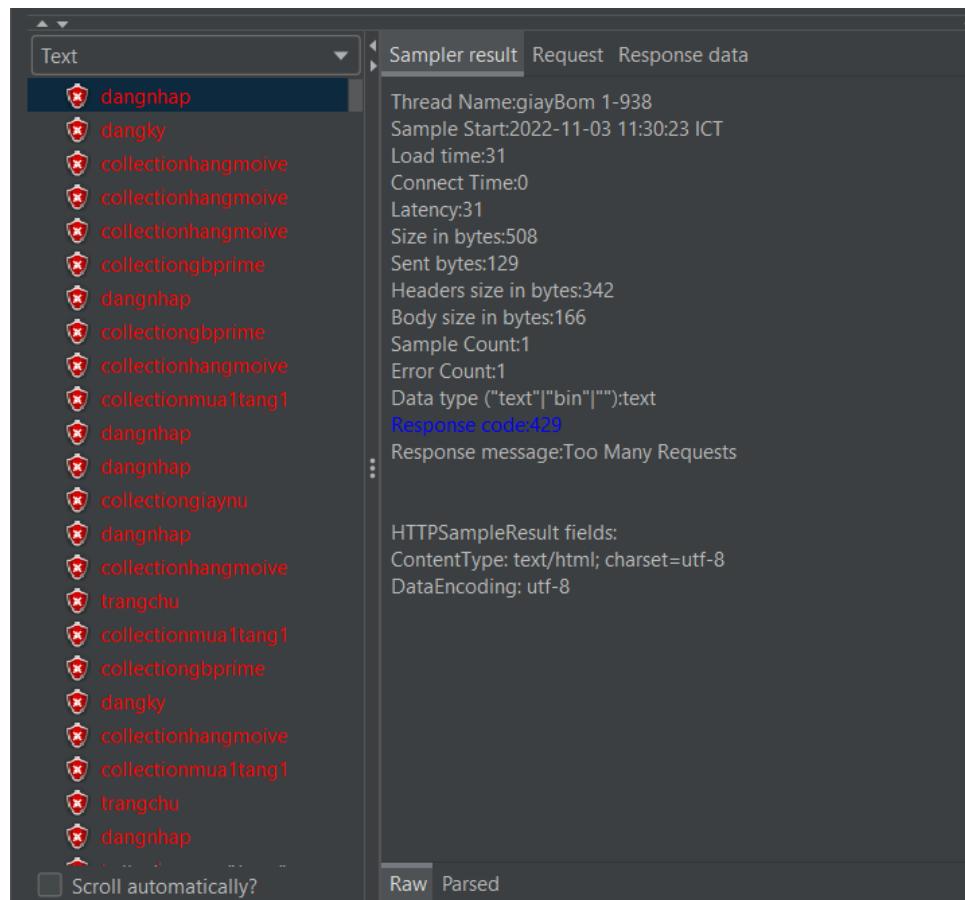
Hình 41: Kết quả 100 User (2)

Tiếp tục đổi sang 1000 user như dữ liệu phía dưới

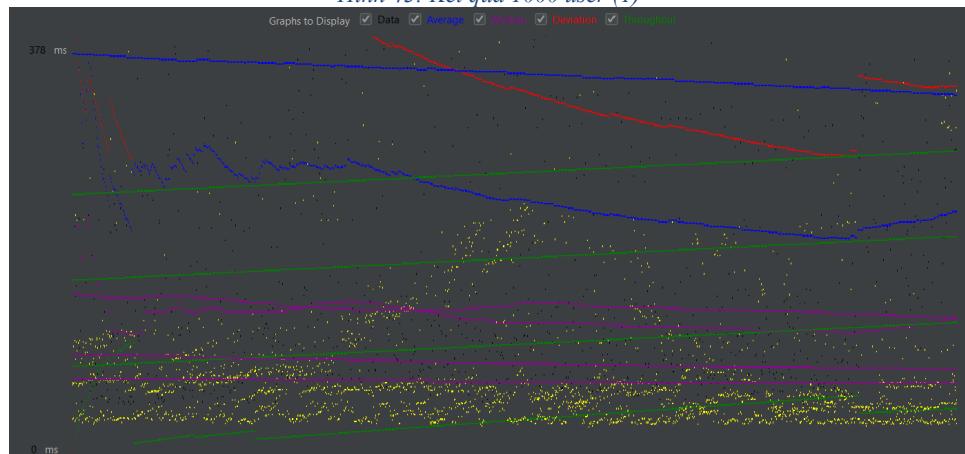
The screenshot shows the 'Thread Properties' configuration window. It includes fields for 'Number of Threads (users)' set to 1000, 'Ramp-up period (seconds)' set to 10, and 'Loop Count' set to 1. There is also an 'Infinite' checkbox which is unchecked.

Hình 42: Tăng lên 1000 User

Kết quả khi thử nghiệm 1000 user đã phát sinh lỗi rất nhiều trên hầu hết các trang được test.



Hình 43: Kết quả 1000 user (1)



Hình 44: Kết quả 1000 user (2)

Nhận xét kết quả đặt được:

Trang website chỉ có thể chạy tốt với mức truy cập 100 user, 1 giây 10 user truy cập nhưng khi con số lên đến 1000 thì website gặp tình trạng lỗi quá nhiều, cần khắc phục tình trạng trên nếu không muốn gặp phải tình huống sập website khi lưu lượng người truy cập nhiều đột biến.

Giải pháp để xuất xử lý vấn đề:

- Mua thêm băng thông, thiết lập nhiều server hơn.
- Tối ưu webserver để có thể tiếp nhận nhiều người truy cập hơn.
- Giám sát lưu lượng truy cập để điều chỉnh cân bằng tải.
- Giới hạn lượt truy cập.
- Sử dụng Firewall.

Phần 3: TỔNG KẾT.

Qua môn học Công nghệ kiểm thử ứng dụng, nhóm em đã có được kỹ năng cần thiết để tiến hành kiểm thử một sản phẩm từ đó cho ra các nhận định, đánh giá về tính ổn định, khả năng thực thi chức năng. Với kiến thức đã học đáp ứng được yêu cầu đã đặt ra, và ứng dụng được ngay vào thực tế sử dụng.

Xin chân thành cảm ơn Cô Lê Thị Quỳnh Chi, giáo viên bộ môn Công nghệ kiểm thử ứng dụng. Đã truyền đạt lại rất nhiều kiến thức bổ ích, thực tế về ngành Công nghệ thông tin nói chung và môn Công nghệ kiểm thử ứng dụng nói riêng, đã giúp nhóm em có thêm nhiều kỹ năng cũng như kiến thức bổ ích.

TÀI LIỆU THAM KHẢO

1. Tài liệu giảng dạy bộ môn Công nghệ kiểm thử ứng dụng.
2. Các nguồn tư liệu từ Internet.