University of Windsor

Assignment 1

Aleena Ali Azeem

110190830

Computing Concept

CERTIFICATION

Introduction

This report details the process of using Selenium with Java to scrape pricing data from AWS. The selected website was AWS S3 Pricing (https://aws.amazon.com/s3/pricing/). Steps taken in creating an environment to run selenium scripts.

1. Download IntelliJ
2. Download browser driver (chrome in this case)
3. Download selenium
4. Add chrome driver to PATH
5. Use project structure in IntelliJ to add jar files from selenium library that we download earlier
6. NOW RUN THE SCRIPTS

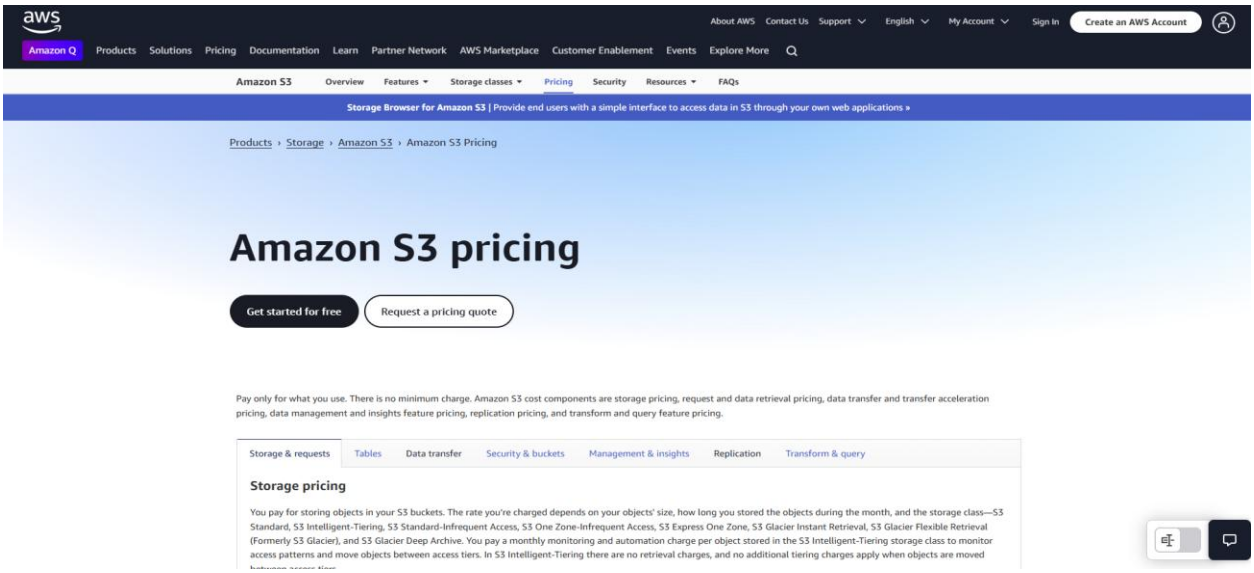
Objectives

Task 1: Extracting Data from AWS S3 Pricing

✓ Open the website in a web browser using Selenium.

```
//here we are initializing driver for the browser and our browser is chrome
here
WebDriver driver = new ChromeDriver();
//We then give the driver path to the website we have chose
driver.get("https://aws.amazon.com/s3/pricing/"); // Open AWS S3 Pricing page

//here we have a wait time to give enough time for selenium to find the
elements
```

Output:



Explanation:

1. We are initializing the driver for the browser and our browser is chrome here
2. //We then give the driver path to the website we have chosen
3. //here we have a wait time to give enough time for selenium to find the elements

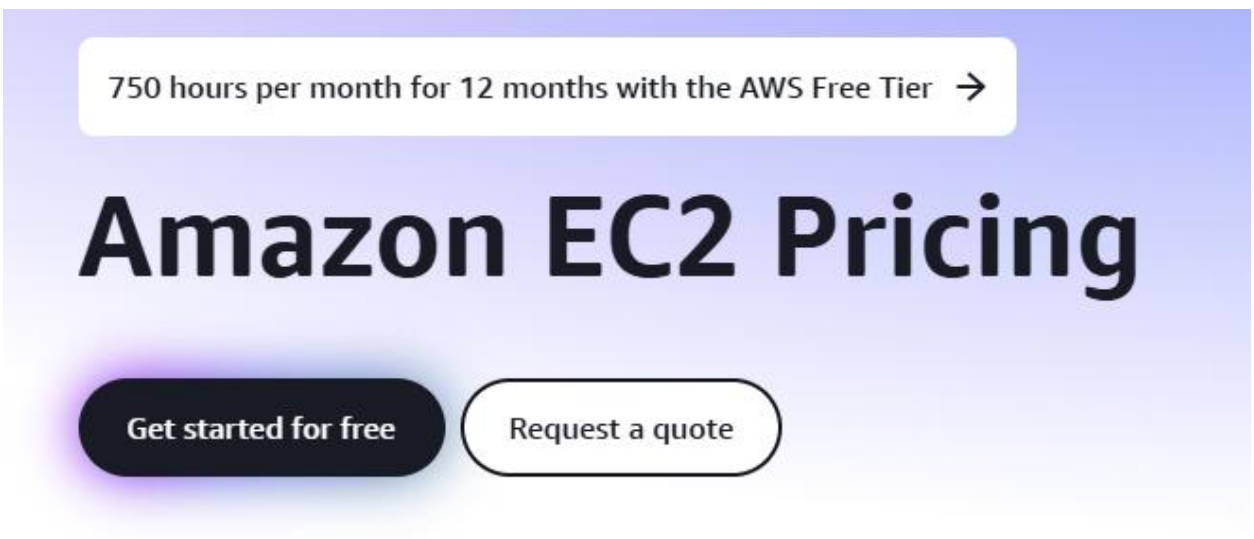
✓ Find and interact with various elements on the page (e.g., links, buttons, text boxes) using Selenium commands.

Code:

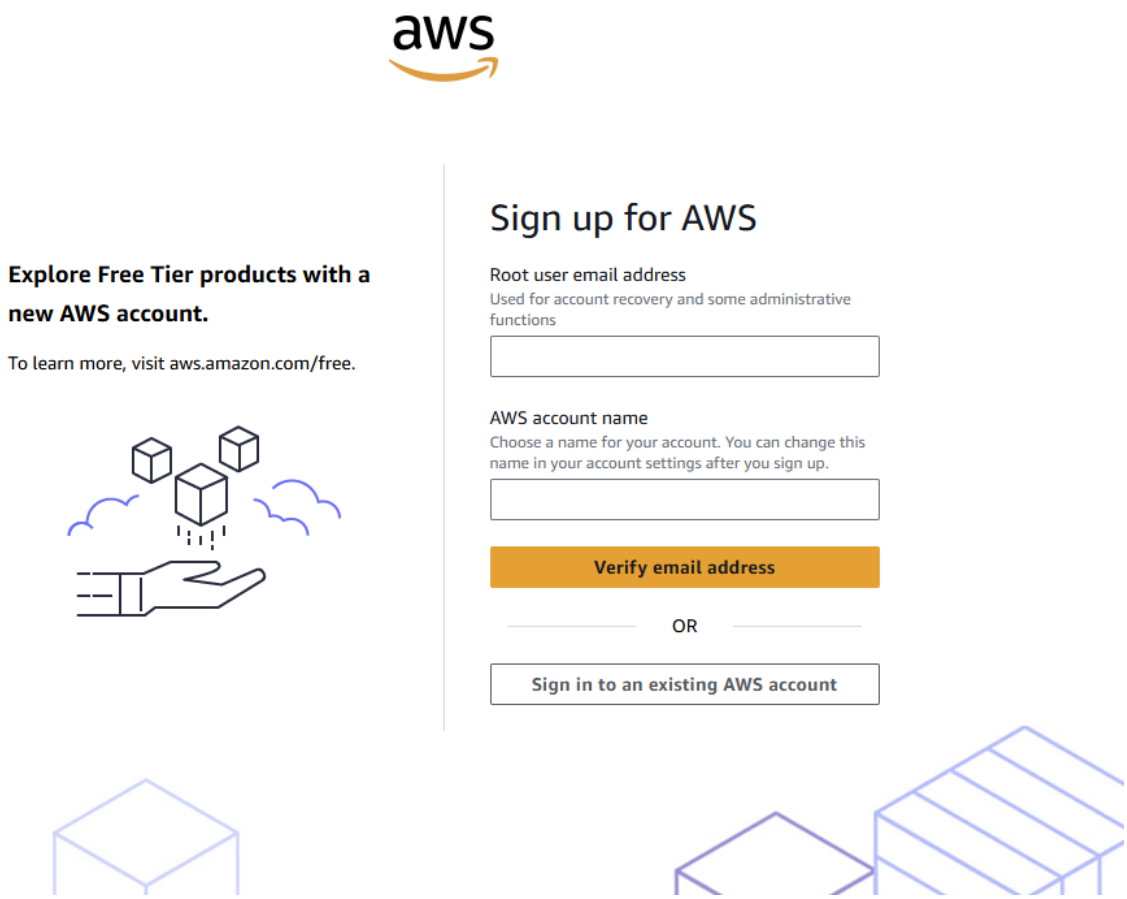
```
// TASK 1, here we are clicking a button
try {
    //again using a css selector we will be spotting it
    WebElement b2 =
wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("a[href*='r
egistration/index.html']")));
```

```
    // Scroll to the button before clicking
    ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);", b2);

    // Click the button
    b2.click();

    // Wait for redirection
    wait.until(ExpectedConditions.urlContains("portal.aws.amazon.com"));
    System.out.println("Clicked 'Get Started for Free' button.");

    //here we are redirecting to another page
    //task 2
    WebElement b3 =
wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("#m-nav >
div.m-nav-header.lb-clearfix.m-nav-double-row > nav > a:nth-child(1)")));
    b3.click();

    // Clear any existing text before sending keys
    b3.sendKeys(Keys.CONTROL + "a");
    b3.sendKeys(Keys.BACK_SPACE);
    b3.sendKeys("Aleena Ali Azeem");
```

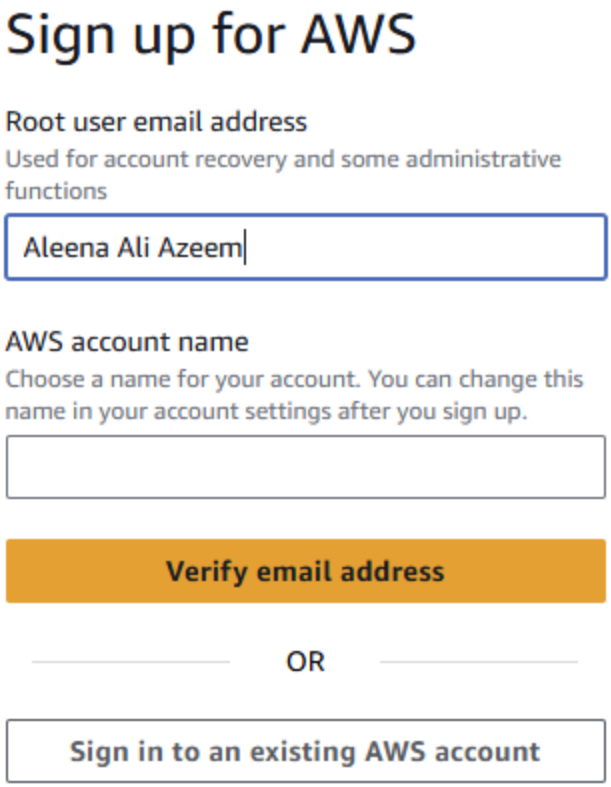
Output:



This will go to the next page



Then we add “Aleena Ali Azeem” into the root user email address box



Explanation:

1. Detected and closed pop-ups before interacting.
2. Located the "Get Started for Free" button using CSS Selector (a.aws-button[href*='registration/index.html']).
3. Scroll to the button before clicking (since it may be hidden).
4. Clicked the button and waited for the page redirection

Objective

✓ Extract data from the page using Selenium commands, such as finding and storing text, images, or other content.

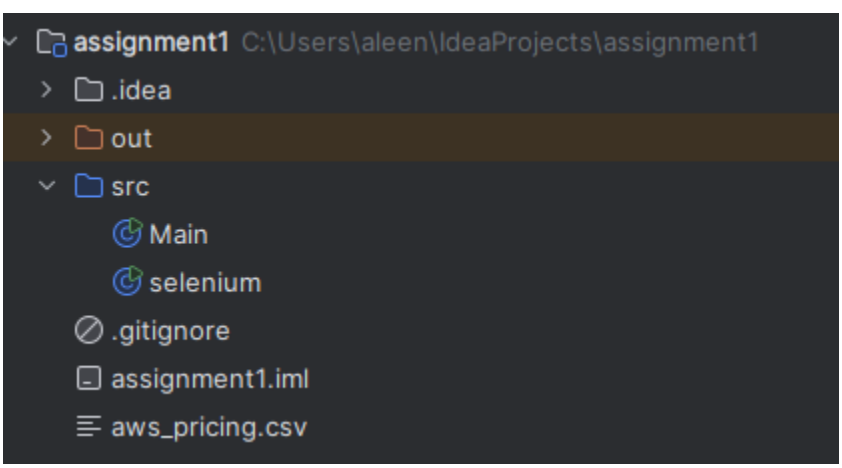
✓ Save the scraped data in a CSV file or other format of your choice

Code:

```
//here we are trying to get data from the website its a table and then we
```

```
will store it in a csv file
try { //task1
    //here we have gone to the website-inspected the table elements and then
copied css selector path
    WebElement table =
wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("#aws-
element-460808b2-3d0a-407a-9d2a-97009814657c table")));

    //Here we are getting all of the rows by using tr
    List<WebElement> rows = table.findElements(By.tagName("tr"));

    //here we are using the built-in java function in order to create a csv
file
    FileWriter writer = new FileWriter("aws_pricing.csv");

    //here we will have a loop over the table in order to get elements of the
table
    for (WebElement row : rows) {
        List<WebElement> cells = row.findElements(By.tagName("td"));

        //skip if the rows are empty
        if (cells.isEmpty()) continue;

        //here we are storing the data in to a variable called data
        StringBuilder data = new StringBuilder();
        for (WebElement cell : cells) {
            data.append(cell.getText().trim()).append(",");
        }

        //After getting the data we will write into the table
        writer.append(data.substring(0, data.length() - 1)).append("\n");
    }

    //Close CSV file
    writer.flush();
    writer.close();
    System.out.println("AWS S3 Pricing data saved to aws_pricing.csv");
```

Output:



Explanation

1. Located the pricing table using its CSS Selector (#aws-element-460808b2-3d0a-407a-9d2a-97009814657c table).
2. Extracted rows (<tr> elements) and columns (<td> elements) using Selenium.
3. Stored extracted data in a CSV file named aws_pricing.csv.

Task 2

Objective:

- Students need to scrape multiple pages from the same website and combine the results.

Code:

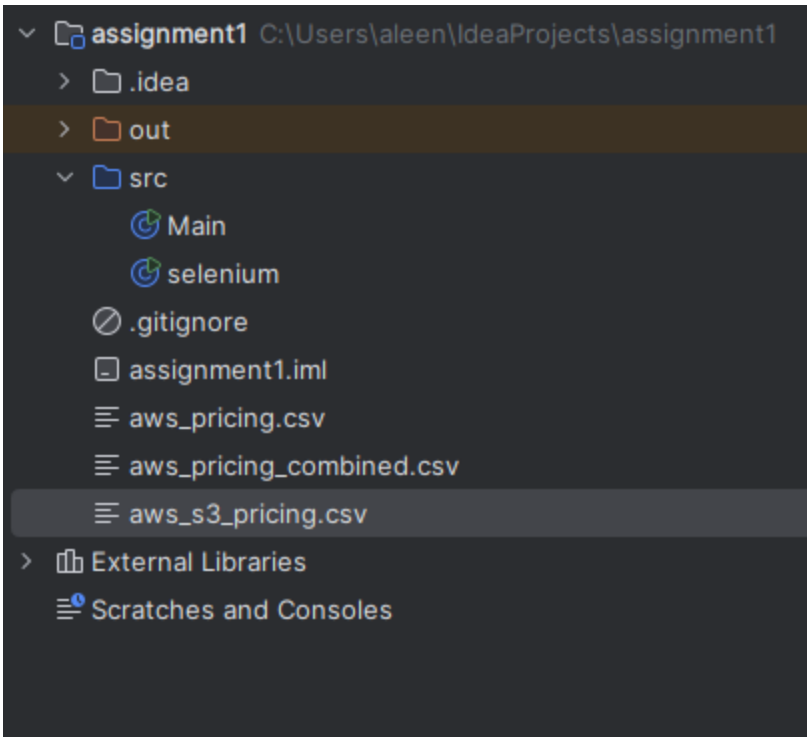
```
try {
    // Open CSV file
    FileWriter writer = new FileWriter("aws_pricing_combined.csv");
    writer.append("Service,Storage Type,Price per GB\n"); // CSV Header

    // TASK 1: Scrape AWS S3 Pricing
    driver.get("https://aws.amazon.com/s3/pricing/");
    System.out.println("Scraping AWS S3 Pricing...");

    try {
        WebElement s3Table =
wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("#aws-
element-460808b2-3d0a-407a-9d2a-97009814657c table")));
        List<WebElement> s3Rows = s3Table.findElements(By.tagName("tr"));

        for (WebElement row : s3Rows) {
            List<WebElement> cells = row.findElements(By.tagName("td"));
            if (cells.isEmpty()) continue;

            StringBuilder rowData = new StringBuilder();
            rowData.append("AWS S3,"); // Label the data as S3 pricing
            for (WebElement cell : cells) {
                rowData.append(cell.getText().trim()).append(",");
            }
            writer.append(rowData.substring(0, rowData.length() -
1)).append("\n");
        }
    } catch (Exception e) {
        System.out.println("Error extracting AWS S3 pricing.");
    }

    // TASK 2: Scrape AWS EC2 Pricing
    driver.get("https://aws.amazon.com/ec2/pricing/");
    System.out.println("Scraping AWS EC2 Pricing...");
```

```
    try {
        WebElement ec2Table =
wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("table"
))); // Adjust selector if needed
        List<WebElement> ec2Rows = ec2Table.findElements(By.tagName("tr"));

        for (WebElement row : ec2Rows) {
            List<WebElement> cells = row.findElements(By.tagName("td"));
            if (cells.isEmpty()) continue;

            StringBuilder rowData = new StringBuilder();
            rowData.append("AWS EC2,"); // Label the data as EC2 pricing
            for (WebElement cell : cells) {
                rowData.append(cell.getText().trim()).append(",");
            }
            writer.append(rowData.substring(0, rowData.length() -
1)).append("\n");
        }
    } catch (Exception e) {
        System.out.println("Error extracting AWS EC2 pricing.");
    }

    // Close CSV file
    writer.flush();
    writer.close();
    System.out.println("AWS S3 & EC2 Pricing data saved to
aws_pricing_combined.csv");

} catch (IOException e) {
    System.out.println("Error writing to file.");
    e.printStackTrace();
}
```

Explanation:

- Navigated to AWS EC2 Pricing Page (https://aws.amazon.com/ec2/pricing/).
- Extracted table data using the table tag.
- Saved the extracted data into aws_pricing_combined.csv alongside AWS S3 pricing.

Output: CSV file aws_pricing_combined.csv containing both AWS S3 & EC2 pricing



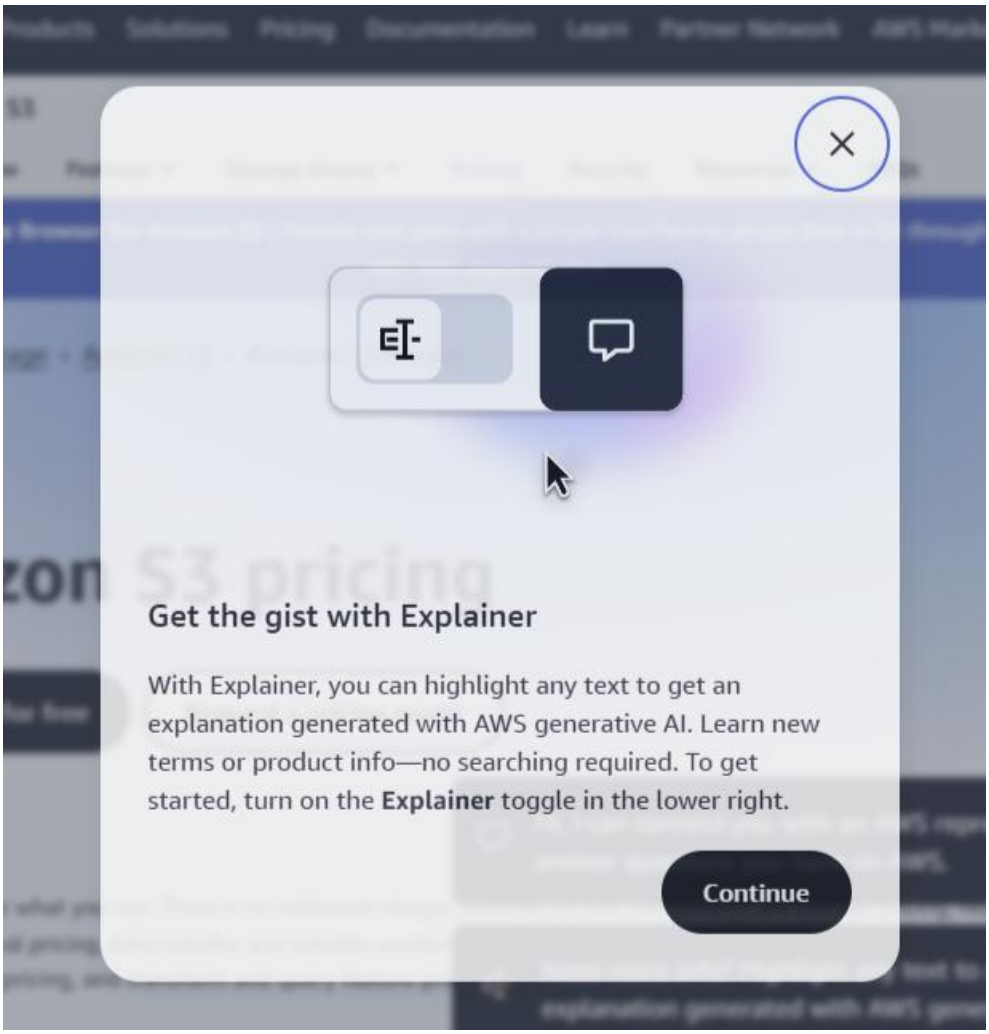
Task 3

Objectives

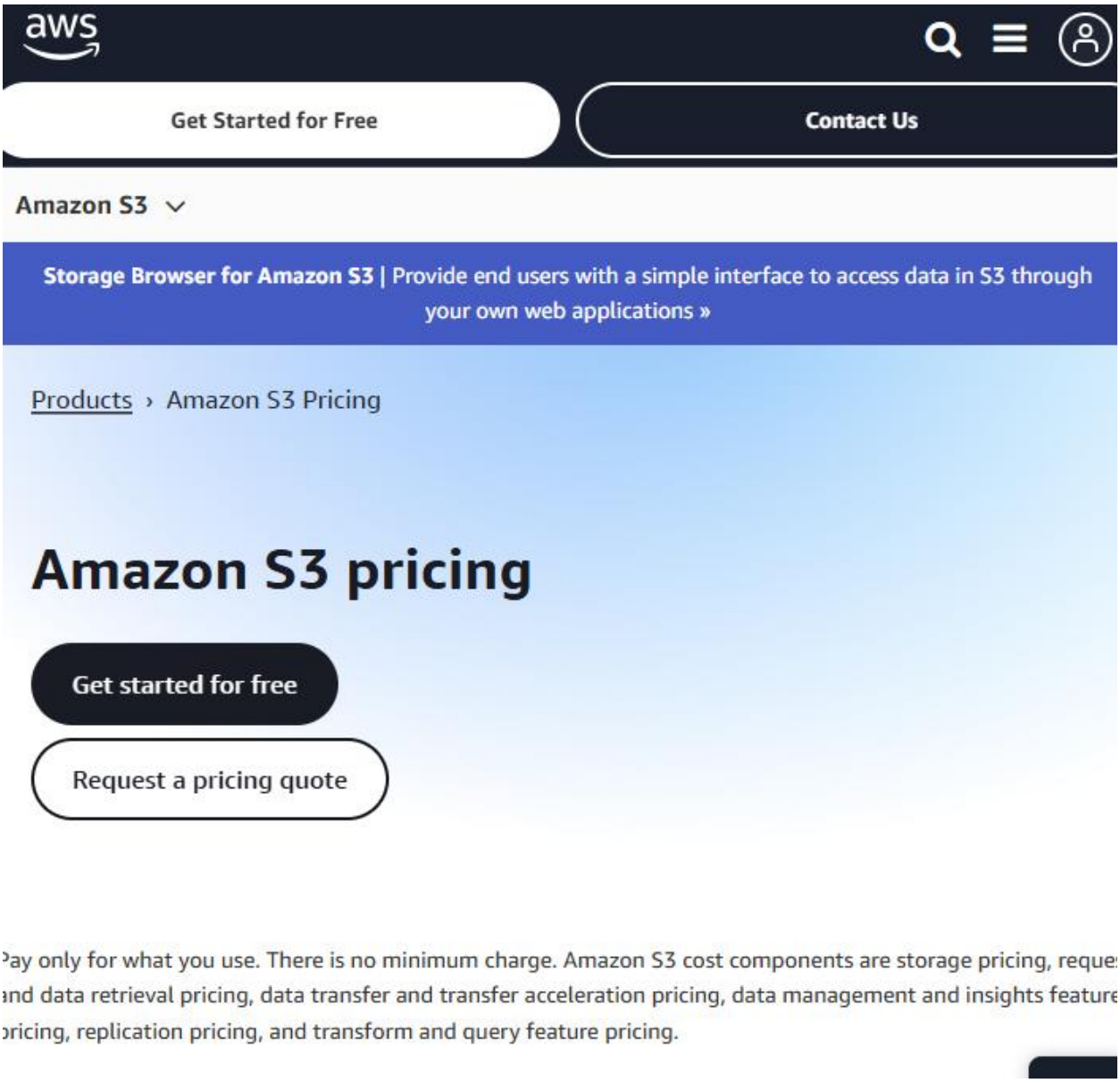
Students need to use advanced Selenium commands, such as waiting for elements to load or handling pop-up windows.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(30));

try {
    //TASK 3 ATTEMPTED closed up the pop as the selenium cant go further to
retrieve anything
    try {
        // here we are using css selector in order to click on the pop up
button
        WebElement b1 =
wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[ari
a-label='Close']")));
        b1.click();
        //this will be printed when the button is pressed, this is done in
order to debug
        System.out.println("Closed the pop-up.");
    } catch (Exception e) {
        System.out.println("No pop-up found, continuing."); // throw an
exception if there is no pop up
    }
```

Output:


Get the gist with Explainer
With Explainer, you can highlight any text to get an explanation generated with AWS generative AI. Learn new terms or product info—no searching required. To get started, turn on the Explainer toggle in the lower right.
Continue

Redirects to



Explanation:

We have implemented a wait at the beginning so that selenium can search for the elements to interact with. Secondly we clicked on the button on the pop up in order to close it and scrap the website.

Challenges & Solutions

Issue	Solution Implemented
Pop-ups blocking interactions	Closed pop-ups using **CSS Selector &** `click()`
Data extraction failing	Used **CSS Selectors & XPath for stability**

References

- AWS Official Site: https://aws.amazon.com/
- Selenium Documentation: https://www.selenium.dev/