Automation framework using Cucumber for the JWT website on link: https://jwt.io

Steps:

Example 1:

1. Go to the link.
2. Paste the token in the Encoded?Embedded text area.
3. Assert if the value of "c" is 3 in the payload.
4. Assert if you're getting 'Invalid Signature' under the Embedded text area.

Example 2:

2. Go to the link.
3. Paste the token in the Encoded/Embedded text area.
4. Assert if you're getting 'Invalid Signature' under the Embedded text area.
5. Paste the secret in the input under verify signature.
6. Assert if you're getting 'Signature Verified' under the Embedded text area.
7. Assert if the token is the same as the one you entered.
8. Assert if the value of "c" is still 3.
9. Assert if changing the secret changes the token but the payload remains the same.

`JwtTests.feature` (Cucumber Feature File)

```
Feature: JWT Token Verification

Scenario: Validate invalid signature and payload

  Given I open JWT website

  When I paste the token into the Encoded text area

  Then The payload field "c" should be 3

  And I should see "Invalid Signature" message
```

```
  Scenario: Validate correct secret verification
    Given I open JWT website
    When I paste the token into the Encoded text area
    And I enter the correct secret
    Then I should see "Signature Verified"
    And The payload field "c" should still be 3
    And I change the secret to something invalid
    Then I should see "Invalid Signature" again
```

TestRunner.java

```
package runners;

import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;

@CucumberOptions(
    features = "src/test/resources/features",
    glue = {"steps"},
    plugin = {"pretty", "html:target/cucumber-reports.html"},
    monochrome = true
)
public class TestRunner extends AbstractTestNGCucumberTests {
}
```

JwtPage.java (Page Object)

```
package pages;

import org.openqa.selenium.*;

public class JwtPage {
    WebDriver driver;

    public JwtPage(WebDriver driver) {
        this.driver = driver;
    }

    By encodedBox = By.cssSelector("div#editor > div > textarea");
    By payloadSection = By.xpath("//section[contains(@class,
'jwt__section--second')]//div[contains(@class, 'CodeMirror')]");
    By signatureMessage = By.xpath("//section[contains(@class,
'jwt__section--third')]//div[contains(text(),'Signature')]");
    By secretInput = By.cssSelector("input#jwt-verify-secret");

    public void navigate() {
        driver.get("https://jwt.io");
    }

    public void enterToken(String token) {
        WebElement encoded = driver.findElement(encodedBox);
        encoded.clear();
        encoded.sendKeys(token);
    }

    public String getPayloadText() {
```

```
        WebElement payload = driver.findElement(payloadSection);
        return payload.getText();
    }

    public void enterSecret(String secret) {
        WebElement secretBox = driver.findElement(secretInput);
        secretBox.clear();
        secretBox.sendKeys(secret);
    }

    public String getSignatureMessage() {
        return driver.findElement(signatureMessage).getText();
    }
}
```

JwtSteps.java

```
package steps;

import io.cucumber.java.en.*;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import pages.JwtPage;

public class JwtSteps {
    WebDriver driver;
    JwtPage page;

    String token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhIjoxLCJiIjoyLCJjIjozLCJpYXQiOjE2OTYzOTc5Mj
d9.6S9s1qxsu454fCUtOOME3Y_LLw7jq1owBleccDmPwvo";

    @Given("I open JWT website")
    public void openWebsite() {
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        page = new JwtPage(driver);
        page.navigate();
    }
```

```
    @When("I paste the token into the Encoded text area")
    public void pasteToken() {
        page.enterToken(token);
    }

    @Then("The payload field {string} should be {int}")
    public void verifyPayload(String field, int expectedValue) {
        String payload = page.getPayloadText();
        Assert.assertTrue(payload.contains("\"" + field + "\":" + expectedValue), "Payload does
not contain expected value");
    }

    @Then("I should see {string} message")
    public void verifyMessage(String message) {
        Assert.assertTrue(page.getSignatureMessage().contains(message), "Expected message
not found");
    }

    @And("I enter the correct secret")
    public void enterCorrectSecret() {
        page.enterSecret("helloworld");
    }

    @And("I change the secret to something invalid")
    public void changeSecret() {
        page.enterSecret("wrongsecret");
    }

    @Then("I should see {string} again")
    public void verifyInvalidMessageAgain(String message) {
        Assert.assertTrue(page.getSignatureMessage().contains(message), "Invalid Signature
message not seen again");
        driver.quit();
    }
}
```