Cucumber Expression

Great! Here's a **mini-project template** that helps you practice writing **clean and readable Cucumber step definitions** using **Cucumber Expressions**, the **Page Object Model (POM)**, and **parameterization**.

✅ Mini Project: Login to Demo Site

🎯 Goal:

Automate the login functionality of a demo site using:

- Clean Gherkin scenarios
- Cucumber Expressions
- Page Object Model
- Parameterized steps

🧱 1. Project Structure

```
pgsql

login-test/
├── src/
│   ├── test/
│   │   ├── java/
│   │   │   ├── stepdefs/
│   │   │   │   └── LoginSteps.java
│   │   │   ├── pages/
│   │   │   │   └── LoginPage.java
│   │   │   └── runner/
│   │   │       └── TestRunner.java
│   ├── resources/
│   │   └── features/
│   │       └── login.feature
```

📄 2. `login.feature`

```
gherkin

Feature: Login Functionality

  Scenario: Successful login with valid credentials
    Given the user is on the login page
    When the user logs in with username "admin" and password "password123"
    Then the user should see the welcome message
```

💬 3. `LoginSteps.java`

java

```java
package stepdefs;

import io.cucumber.java.en.*;
import org.junit.Assert;
import pages.LoginPage;

public class LoginSteps {
    LoginPage loginPage = new LoginPage();

    @Given("the user is on the login page")
    public void the_user_is_on_login_page() {
        loginPage.openLoginPage();
    }

    @When("the user logs in with username {string} and password {string}")
    public void the_user_logs_in(String username, String password) {
        loginPage.login(username, password);
    }

    @Then("the user should see the welcome message")
    public void the_user_should_see_welcome() {
        Assert.assertTrue(loginPage.isWelcomeMessageDisplayed());
    }
}
```

🧾 4. LoginPage.java

java

```java
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginPage {
    WebDriver driver;

    // Locators
    By usernameField = By.id("username");
    By passwordField = By.id("password");
    By loginButton = By.id("login");
    By welcomeMessage = By.id("welcome");

    public LoginPage() {
        driver = new ChromeDriver(); // Or use WebDriverManager
    }

    public void openLoginPage() {
        driver.get("https://example.com/login"); // Replace with a real login demo site
    }

    public void login(String username, String password) {
        driver.findElement(usernameField).sendKeys(username);
        driver.findElement(passwordField).sendKeys(password);
        driver.findElement(loginButton).click();
    }

    public boolean isWelcomeMessageDisplayed() {
        return driver.findElement(welcomeMessage).isDisplayed();
    }
}
```

🚀 5. `TestRunner.java`

```java
package runner;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepdefs",
    plugin = {"pretty", "html:target/cucumber-report.html"},
    monochrome = true
)
public class TestRunner {}
```

🛠️ 6. Required Dependencies (Maven `pom.xml`)

```xml
<dependencies>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>7.14.0</version>
    </dependency>
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>7.14.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.20.0</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

🎯 Assignment:

1. Replace the login URL with a working test site like:
    - https://the-internet.herokuapp.com/login
    - or a local test app
2. Add another scenario:

```gherkin
Scenario: Login fails with invalid credentials
  Given the user is on the login page
  When the user logs in with username "admin" and password "wrongpass"
  Then an error message should be displayed
```

3. Add a step definition + method in `LoginPage` to validate the error.

Would you like this as a downloadable `.zip` project or want the next mini-project for registration or dashboard validation?