National University
of computer and emerging sciences

Assignment #2

Course:

Software Quality Engineering

Topic:

LAYP
(UI Testing Automation Framwork)

Submitted by:

Roll No.	Name
22F-3714	Faizan Tariq
22F-3722	Muhammad Abdullah

Submitted to:

Mr. Jawad Khalid

Contents

Project Overview

This project is designed to automate the testing of Daraz.com using Java, Maven, and Cucumber for Behavior-Driven Development (BDD). The framework follows a **Page Object Model (POM)** design and integrates **Cucumber for BDD**, **MariaDB for data storage**, and **Cucumber HTML Reporting** for test execution results.

Features:

- **Scalable Web UI Automation Framework**
- **Supports Gherkin for writing BDD tests**
- **Data-driven testing using MariaDB**
- **HTML Reporting for test results**

Prerequisites

1. Install Java

Ensure that you have **Java 1.8** or higher installed. You can check the version by running:

If you don't have it installed, download and install it from the Oracle website.

2. Install Maven

Maven is required for managing project dependencies and running the tests. You can check if Maven is installed by running:

➢ mvn -version

If Maven is not installed, download it from Maven's official website.

3. Install Eclipse IDE

Download and install Eclipse IDE if not already installed.

Project Setup

1. Clone the Repository

Open Eclipse and clone the GitHub repository where your project is hosted. Follow these steps:

1. Go to **File > Import > Git > Projects from Git**.
2. Choose **Clone URI**, and paste the repository URL.
3. Click **Next** and follow the instructions to clone the project into your workspace.

2. Import as Maven Project

After cloning, ensure that the project is imported as a Maven project:

1. Right-click on the project in the **Project Explorer** and select **Configure > Convert to Maven Project** (if it's not already).
2. Ensure that the pom.xml is properly loaded, as it defines all dependencies.

3. Install Dependencies

Once the project is imported, Maven should automatically download all necessary dependencies. You can force this process by running:

- **In Eclipse**: Right-click the project > **Maven > Update Project**.
- **Using Terminal**: Navigate to the project root and run:

➢ mvn clean install

4. Configure MariaDB (Optional for Data-Driven Testing)

If your test cases depend on **MariaDB**, ensure that MariaDB is installed and running. To set up MariaDB:

1. Download and install MariaDB from here.
2. Create a database for your tests and update the DatabaseConnection.java file with your connection details (username, password, and database URL).

Running Tests in Eclipse

1. Running All Tests

To run all test cases at once:

1. Navigate to the src/test/TestRunner directory where your TestRunner. files are stored
2. Right-click on the feature file (or the entire features folder) and choose **Run As > JUNIT Test**.

2. Running Specific Tests

To run a specific test, follow the same steps as above, but select the specific feature file (e.g., Login.feature).

3. Running Tests via Maven

If you prefer to run the tests through Maven, open a terminal in the project directory and run the following command:

- mvn test

This command will execute all the tests and generate a Cucumber report.

4. Viewing Reports

After running the tests, the Cucumber HTML report will be generated in the target/cucumber-reports directory.

- Navigate to target/cucumber-reports.
- Open the index.html file to view the detailed HTML report.

This report will show which tests passed, which failed, and will include logs and screenshots (if applicable).

Troubleshooting

1. Maven Build Issues

If Maven is not able to resolve dependencies, try the following:

- Ensure that you have an active internet connection for Maven to download dependencies.
- Use **Maven > Update Project** from Eclipse to force Maven to re-download dependencies.

2. Database Connection Issues

If you encounter issues connecting to **MariaDB**, check the following:

- Ensure MariaDB is running.
- Verify the connection details (username, password, and database URL) in your db.properties file.
- Test the connection manually to ensure that MariaDB accepts connections.

3. Tests Failing

If specific tests are failing:

- Review the logs in the target/cucumber-reports folder for any specific errors.
- Ensure the environment (browser, MariaDB, etc.) is properly configured.

Project Structure

Here is a brief explanation of the project structure:

Files and Folders

- **pom.xml**: Manages dependencies, plugins, and project build lifecycle.
- **src/test/resources/features**: Contains the .feature files written in Gherkin.
- **src/test/java**: Contains the step definition files that bind the Gherkin scenarios to Java code.
- **target/cucumber-reports**: The generated HTML reports after test execution.

Best Practices

- **Modular Design**: The project follows the **Page Object Model (POM)** pattern, where page-specific methods are abstracted into separate classes to ensure reusability.
- **BDD with Cucumber**: All tests are written in Gherkin for clear communication and collaboration between stakeholders.
- **Data-Driven Testing**: Use the Scenario Outline feature in Cucumber and integrate **MariaDB** for dynamic test data.

Github Link :

https://github.com/whis-19/LAYP.git