src/main/java/com.qa.factory

1.Factory Layer

**DriverFactory.java**- Property File Loading  
- Browser Decision

**OptionsManager.java**   
- Setting Browser Options

**Hybrid Framework Folder Structure**  
*POM design pattern +  
Data Driven Framework +  
TestNG*

**Page Object Model with page chaining design**

|  |  |
| --- | --- |
| *Language - Java Automation - Selenium WebDriver Execution Configuration - TestNG Build Tool - Maven DataDriver - Apache POI Reports - Extent Reports/TestNG*  *Loggers - Log4j* | *CICD - Jenkins Repo – Git/GitHub Parallel Testing - GRID Browsers –Chrome/FF/Edge/Safari*  *OS - Windows/Max/Linux RemoteExecution - Docker/LambdaTest* |

test-output.reports  
**TestReports   
(contains pass/fail status, excution time, environment, browser, OS etc)**

6.Reports

**Extent Reports**

**TestNG Reports**

**HTML Reports**

src/main/java/com.qa.util

5.Utility Layer Configurations

**UtilityClasses**-TestUtils: common methods  
-ExcelUtils: Read/write for datadriven approach  
-ListenerUtils: For log triggering

src/test/resources/

4.Resources

**config.properties**-Execution Environment  
-URL  
-Others

**TestRunners**-TestNG\_Master  
-TestNG\_Smoke  
-TestNG\_Regression

**TestData  
Excel/CSV Sheets**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

3.Test Layer

src/main/java/com.qa.base

src/test/java/com.qa.testcases  
**TestCase Library  
(contains all testcases)**

**AddToCartPageTest.java**

**CheckoutPageTest.java**

**LoginPageTest.java**

**BaseTest.java (Parent Class of all page classes)**  
-BeforeTestSetUp  
DriverFactoryCall  
Page chaining initiation -RootPage Call  
-AfterTestTearDown

src/main/java/com.qa.base

src/main/java/com.qa.pages  
**Page Library  
(contains webelement locators and actions)**

2.Page Layer

**LoginPage.java**

**AddToCartPage.java**

**CheckoutPage.java**

**BasePage.java (Parent Class of all page classes)**  
-Pagefactory init

**✅ End-to-End Test Case: Place an Order & Validate in DB**

🔍 Test Case Name: PlaceOrderAndValidateInDB

💡 Objective: Verify that a user can place an order successfully and confirm that order details are correctly saved in the database.

🧪 Test Steps:

🔹 UI Automation Steps (Selenium):

1. Launch OpenCart
   * Example URL: http://localhost/opencart/ (or your hosted version)
2. Login to User Account
   * Go to My Account → Login
   * Enter valid credentials (e.g., testuser@example.com / test123)
3. Add Product to Cart
   * Navigate to a category (e.g., Desktops)
   * Select a product (e.g., MacBook)
   * Click Add to Cart
4. Checkout
   * Go to Cart → Click Checkout
   * Fill billing/shipping details (if not already saved)
   * Select Payment Method → Confirm Order
5. Get the Order ID
   * On the success page, capture the Order ID shown in confirmation.

🔹 DB Validation Steps (via JDBC):

1. Connect to OpenCart DB (MySQL)
   * Use JDBC to connect
2. Query the oc\_order table:

sql

Copy code

SELECT \* FROM oc\_order WHERE order\_id = '<captured\_order\_id>';

1. Validate Order Details:
   * Assert customer name/email match
   * Assert total price is correct
   * Assert order status is "Pending" or "Processing"
2. Optionally, Validate oc\_order\_product:

sql

Copy code

SELECT \* FROM oc\_order\_product WHERE order\_id = '<captured\_order\_id>';

* + Confirm product name, quantity, and price

✅ Expected Result:

* Order is placed successfully via UI
* Order record exists in the DB with correct customer and product info
* Order totals match between UI and DB
* Product entries exist in oc\_order\_product

🧰 Tech Stack Tips:

* Frontend: Selenium + TestNG or JUnit
* DB: JDBC + MySQL Driver
* Maven Dependencies:

xml

Copy code

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

</dependency>

🔧 JDBC DB Validation Snippet:

java

Copy code

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/opencart", "root", "password");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM oc\_order WHERE order\_id = '" + orderId + "'");

if (rs.next()) {

String dbEmail = rs.getString("email");

Double dbTotal = rs.getDouble("total");

Assert.assertEquals(dbEmail, "testuser@example.com");

Assert.assertEquals(dbTotal, expectedTotal);

}

conn.close();