### EXECUTION WORKFLOW

### (21K3198,21K3178,21K3401)

### 1. ****Test Class Initialization (****TestExecution****)****

* The test class TestExecution is annotated with [TestClass], which signals to MSTest that it contains test methods.
* The class uses Selenium WebDriver for interacting with the web application and defines various page objects (LoginPage, ProductPage, YourCartPage, ItemPage, YourInformationPage, OverviewPage).

### 2. ****Test Lifecycle Hooks****

* **[AssemblyInitialize()]**: This method runs once before all tests in the assembly. It creates a report file for logging the test execution using ExtentReports. The report file is named dynamically with the current timestamp.
* **[AssemblyCleanup()]**: This method runs once after all tests in the assembly. It finalizes the ExtentReports log by flushing the reports.
* **[ClassInitialize()]**: This method runs before any tests in the class. It's empty in this case but can be used to set up class-level resources.
* **[ClassCleanup()]**: This method runs after all tests in the class. It is empty in this case as well.
* **[TestInitialize()]**: This method runs before each test method. It initializes the Selenium WebDriver for the specified browser (Chrome) by reading from the ConfigurationManager settings and creates a new test in ExtentReports.
* **[TestCleanup()]**: This method runs after each test method. It closes the Selenium WebDriver session after each test completes.

### 3. ****Test Case Execution****

Each test case follows a standard pattern, which includes the following key steps:

* **Test Data Fetching**: The test case uses the [DataSource] attribute to pull test data from an XML file. For each test case, it fetches the necessary values such as user-name, password, url, and other relevant data to execute the tests.
* **Test Logic**: The actual test logic is performed by interacting with various page objects that represent different parts of the web application. For example:
  + loginPage.Login(): Logs in using the provided credentials.
  + productPage.Add\_Or\_Remove\_ToCart(): Adds or removes items from the shopping cart.
  + yourcart.YourCart\_CheckOut(): Proceeds to the checkout page.
  + yourInformationPage.YourInfo\_Continue(): Fills out the user information form during checkout.
  + overviewPage.CheckPrice(): Verifies the prices in the overview page before completing the order.
* **Assertions**: The test asserts conditions on the UI elements to verify that the application behaves as expected. For example, verifying if the correct message appears when the order is completed or if the cart contains the expected items.
* **Screenshots**: After each assertion, the test case takes a screenshot of the current page to capture the test result. If the test passes, it logs a pass screenshot; if it fails, it logs a failure screenshot along with the error message.

### 4. ****Test Methods****

The individual test methods (annotated with [TestMethod]) define specific test scenarios. Below are the key test methods:

* **Complete\_Order\_TC001**: Verifies the complete order flow, from login to checkout and order completion.
* **Filter\_Check\_TC002**: Tests the filtering functionality on the product page.
* **Locked\_Out\_User\_TC003**: Checks if the correct error message appears for locked-out users.
* **User\_Add\_And\_Remove\_From\_Cart\_ProductPage\_TC004**: Tests adding/removing items from the cart on the product page.
* **YourCartPage\_ContinueShopping\_TC005**: Tests the "Continue Shopping" functionality on the cart page.
* **YourCartPage\_Remove\_TC006**: Verifies the functionality of removing items from the cart.
* **YourInfo\_CancelBtn\_TC007**: Tests if the cancel button works correctly during the user info step of checkout.
* **CheckOut\_OverView\_CancelBtn\_TC008**: Verifies the cancel functionality on the overview page during checkout.
* **UserLogout\_TC009**: Tests user logout functionality.
* **AllItems\_TC010**: Verifies if all items are shown correctly on the product page.
* **YourInfo\_Empty\_TC011**: Tests for validation errors when the user information form is submitted empty.
* **Add\_Item\_From\_Item\_Page\_Status\_Not\_Change\_TC012**: Tests if the item status remains unchanged when navigating from the item page.
* **Add\_Item\_From\_Item\_Page\_Status\_Change\_TC013**: Verifies if the item status changes as expected when navigating from the item page.

### 5. ****Test Result Reporting****

Each test uses **ExtentReports** to log the test status (Pass/Fail) and capture screenshots. The TakeScreenshot() method is used to capture the screen at specific checkpoints. These screenshots are logged along with the test status in the generated report.

### 6. ****Final Cleanup****

* After all tests have been executed, the AssemblyCleanup() method is invoked to finalize the ExtentReports and generate the final HTML report.