Project Overview

This project automates the extraction and validation of upcoming bike details and used car models from the Zigwheels website. The automation script is developed using Selenium, TestNG, and Cucumber, with results captured in structured Excel reports.

Problem Statement

Identify New Bikes

Objective: Display details of upcoming bikes including bike name, price, and expected launch date in India with the following criteria:

1. Manufacturer should be Honda.

2. Bike price should be less than 4 Lakhs.

Additional Objectives:

1. Extract and display all popular used car models in Chennai.

2. Attempt to 'Login' with Google using invalid account details and capture the error message.

Detailed Description

Requirements:

1. Display the "Upcoming" bikes details like bike name, price and the expected launch date for “Honda” manufacturer with price less than 4 Lakhs.

2. For used cars in Chennai, extract and display a list of all popular models.

3. Try to log in with Google using invalid details and capture the error message.

Key Automation Scope

1. **Handling windows & frames:** Manage multiple windows and frames during the automation process.

2. **Filling simple forms and capturing warning messages:** Automate form filling and capture any warning messages encountered.

3. **Extracting menu items from frames and storing them in collections:** Extract data from various frames and store them for further processing.

4. **Navigating back to the home page:** Maintain smooth navigation across the website, including returning to the website's home page when necessary.

Steps to Run the Project

1. **Configure the Browser:**

   Properties file is used to set the driver.

   - `1` for Google Chrome

   - `2` for Microsoft Edge

2. **Run TestNG.xml File:**

   When this testng.xml file is executed, TestNG will perform the following actions in this order

   - **Suite Initialization:** TestNG begins by starting the test suite named "Suite."
   - **Listener Setup:** Custom listeners (CustomListener for reporting/logging and RetryTransformer for test retries) are activated.
   - **Test Block Start:** The "Test" block within the suite begins execution, configured to run tests using up to 5 parallel threads.
   - **Class & Test Discovery:** TestNG loads the testRunner.TestNGTestRunner class and identifies all test methods (e.g., @Test methods, or Cucumber scenarios if it's a Cucumber Test Runner) and setup/teardown methods (@BeforeX, @AfterX).
   - **Test Method Execution:**
     - Any initial setup methods (@BeforeSuite, @BeforeTest, etc.) are run.
     - Discovered test methods/Cucumber scenarios are executed.
     - Tests run in parallel across up to 5 threads.
     - If a test fails, the RetryTransformer might re-execute it.
     - Cleanup methods (@AfterMethod, @AfterClass, etc.) are run after test completion.
   - **Reporting & Logging:** Throughout the process, the CustomListener generates reports, logs events, and performs other defined actions (e.g., taking screenshots on failure).

- **Test Block Completion:** All tests within the "Test" block finish execution.
- **Suite Completion:** The entire "Suite" completes, and TestNG generates its final reports.

Prerequisites

- Java JDK
- Eclipse IDE
- Selenium WebDriver
- TestNG
- Cucumber
- Apache POI (for Excel handling)

File Structure

- **src/main/java**: Contains the POM and Utilities.
- **src/test/java**: Contains the base class, hooks file, listener utility, step definition and test runner.

-**src/test/resources-** contains feature files, excel file, json file, xml file, config properties

- **TestNG.xml**: The TestNG configuration file.

Conclusion

This project demonstrates the automation of web data extraction and validation using Selenium, TestNG, and Cucumber, effectively handling browser interactions, data extraction, and error capturing with detailed reporting.