Automate an E-Commerce Web Application Writeup

Project Objective:

To automate a real-world web application

Flipkart is an e-commerce platform, and they have launched a new feature to search for a product in a particular category. Once the product is searched, Flipkart displays it as a list of product items. To enhance the performance of the application, Flipkart has implemented lazy loading. It displays only a few products that can come on the screen.

To display or load more products, the user must scroll down.

As a Test Engineer, you are expected to test this feature end-to-end.

GitHub Link: https://github.com/Prem-Ravi/Automate-an-E-Commerce-Web-Application.git

STEP1: (Setup & Configuration)

- Launch Eclipse IDE and create a maven project in your IDE.
- Install TestNG and other required files.
- Install Selenium IDE Extension in Google Chrome and Firefox Browsers.
- Download and Configure Web Driver for Chrome and Firefox.

STEP2: (Coding)

- Create the maven project and configure the build path.
- Create the TestNG class in ecommerce website.
- Import the required files and also add necessary dependencies in the .xml file.
- Create the class name Automation Flipkart.
- Two test groups are added for chrome and Firefox and necessary web drivers such as Chrome Driver and gecko driver are included.
- For Testing the www.flipkart.com page with search function of iPhone 13 and scroll effect are included with screen shot feature.

STEP3: (Execution)

- The code is compiled by the TestNG.
- The Execution is done in both google chrome and Firefox browsers with Selenium IDE and web drivers.
- Selenium controls the browser and outputs are generated.
- Screenshots are automatically updated in the workspace.
- Finally compile the code and execute the code in browser.

STEP4: (Save)

- The Result will be obtained in the console.
- Push the code to the GitHub Repository.

Outcome:

- Navigate to the Flipkart homepage (https://www.flipkart.com/)
- Determine a page load time with a performance test
- Search for a product, say, “iPhone 13” under the “Mobile” category
- Check if the images are loaded and visible till the screen height only
- Check if the page has a scroll feature
- Check the frequency at which the content will be refreshed while scrolling
- Verify that the image is downloaded just before the user scrolls to its position and gets displayed in time
- Verify that it navigates to the bottom of the page
- Check whether different browsers and screen resolutions render it the same way