**Write UP**

**Automating the Star Health Application Using Selenium**

1. Create a quickstart maven project in eclipse

2. Install Cucumber plugin in Eclipse

3. Select all trusted sites and click on Restart eclipse.

Go to Your cucumber project → POM.xml file and remove the existing depenedecies section.Compelte section remove it.

4. For this create 3 packages in the project under src/test/java

5. Create a feature file.

6. Create a file with extension .feature

7. Right click on the file and select the option pretty format.

Now we will change Maven project to Cucumber project

Right click on your project →go to Configure → click on convert to Cucumber project

8. Save the feature file → you steps will be highlighted

That means -> for these steps we have not written selenium code

9. Create Step definition file

10. Create the BaseTest.java file for Hooks

11. Create a Cucumber runner file in the folder runner

12. CUCUmber integration with jenkins

=============================

Download chromedriver.exe from selenium webpage

<https://googlechromelabs.github.io/chrome-for-testing/>

And add the below line in the step definition file

13. Above give the path of chromedriver exe in your downloads folder.

14. Now push the project to github and run using jenkins job.

In jenkins → manage jenkins→ Plugins→ available plugin

Search for HTML → Install it

15. Also download cucumber plugin

16. Go to your job -> click on configure

17. Save the job and run it. You will find reports in the path given in console as below

**Components:**

Maven Project Setup: Utilized Maven as the build tool for dependency management. The primary dependencies include Selenium WebDriver, Cucumber, and JUnit.

Project Structure: Organized the project into a structured format with distinct packages for pages, utilities, step definitions, and hooks to maintain modularity and scalability.

WebDriver Manager: Implemented a DriverManager class to manage the WebDriver instance. This class initializes the ChromeDriver for browser interactions.

Page Object Model (POM): Created a HomePage class within the pages package. This class encapsulates the methods related to the Amazon homepage, such as navigating to the homepage and searching for products.

Step Definitions: Defined step definitions in the StepDefinitions class under the tests package. The step definitions are written using Cucumber's Gherkin syntax to represent the Given-When-Then scenarios.

Cucumber Hooks: Integrated Cucumber hooks by implementing @Before and @After annotations in the CucumberHooks class. These hooks ensure that the WebDriver instance is initialized before the test execution and closed after the test completion.

Integration with Jenkins: Established integration with Jenkins for continuous integration. Configured a Maven project in Jenkins and set up a build job to trigger the test execution using the mvn clean test command.

**Execution Flow:**

Jenkins triggers the build job upon a specific event or schedule.

Maven builds the project, resolving dependencies and compiling the source code.

During the execution phase, Cucumber reads the feature files and maps them to the corresponding step definitions.

The WebDriver interacts with the Amazon application, executing the defined test scenarios.

Upon test completion, the results are reported, and Jenkins provides insights into the test execution status.

Conclusion:

The implemented Cucumber framework facilitates the automation of basic functionalities for the Amazon application, ensuring reliability, maintainability, and scalability. By leveraging Selenium WebDriver and Cucumber, the framework streamlines the testing process, enabling continuous integration and delivery through Jenkins. Future enhancements can include integrating additional functionalities, implementing data-driven testing, and enhancing reporting mechanisms for comprehensive test coverage.