GREENKART

TEST PLAN AND CASE STUDIES

About the Application:

- Green-kart is a Custom Dummy ECommerce application designed purely for the Testing purpose.
- Through this website, you can play around with the UIs to create an end to end automation framework, like searching a product, adding product(s) to the cart, applying promo code and finally placing the order.

Test Plan:

- Application Type: Web based ecommerce application
- Links (if any)- Greenkart
- Automation Tool: Selenium

- Framework approach: TestNG, Test driven, POM, Base page
- Tools to be used: Eclipse (IDE), Github for sharing the code to remote repository,
- Programming language: Java
- Features to be Tested and points to be covered:

☐ Search a product from the global search
☐ Click on a product
☐ select the total quantity of the product
☐ add the product to the cart
☐ remove the product from the cart
☐ Add billing details
☐ Payment method
☐ Confirm order
☐ Assertions

- Test env: Windows 10

Test Scenario:

1. Go to google.com
2. In the search bar search for green-kart and inspect all the elements of the autocomplete box.
3. Select green-kart and then select the link of rahul shetty academy
4. Search for a product from the global search, perform assertions to check whether correct product is displayed
5. Add the product(s) to the cart (Single and bulk)
6. Click on proceed to checkout

7. **Perform the assertions to check whether correct product, quantity and price is displayed**
8. **Perform assertions after adding promo codes (if any)**
9. **Click on place order**
10. **Select the country**
11. **Agree to the T/C**
12. **Perform assertion to see if proceed can be clicked with/ without agreeing the T/C**
13. **Click on Proceed**
14. **Perform assertion to check if you are redirected to the home page after placing the order**

Framework Approach:

1. **Tool- Selenium**
2. **Framework tool(if any)- TestNG**
3. **Programming language- Java**
4. **Reporting library- Extent reports library**
5. **Libraries used for Project Setup- Maven dependencies of Selenium, TestNG, WebDriver manager**
6. **Additional(If any)- Created config.properties file to store all the required sensitive informations and locator.properties file to store the web-elements**
7. **Page Object Model- Created different classes for different projects**
8. **Base Page- Created a separate class to store all the reusable methods**

9. **Utils- Created a Utility package to store all the basic components such as test listeners, retry methods**

What went better:

- **Got to know about in depth concepts regarding creating an end to end framework for any ecommerce project**
- **Learnt about how to add multiple products, single products with multiple quantity to the items cart and place their order**
- **Got to know about the OOPS concept in depth practically**

What are the challenges faced:

- **Inconsistent web-element locators which resulted in delay of executions and unexpected errors in the run time as well as in the compile time errors**
- **Sometimes encountered with unexpected runtime errors despite having valid web-element locators which delayed the completion of the project**

GITHub Repository(If any): GITHub-Repo