Problem Statement:

**1. Implement below case study using Selenium with Java concepts**

1. Launch a below URL and verify the Title using Assertions https://www.saucedemo.com/

2. Login to the saucedemo portal using below valid credentials and verify that able to login to the portal successfully

User Name: standard\_user

Password: secret\_sauce

3. Login to the saucedemo portal using below Invalid credentials and verify that not able to login to the portal

User Name: standard\_Invaliduser

Password: secret\_Invalidpwd

4. Read the username and password from TestNG Parameters (TestNG.xml)

5. Verify that User is navigated to right page and verify the Title using TestNG Assertions for Positive Case

6. Verify that User is not navigated to right page and verify the Title using TestNG Assertions for Negative Case

Note:

Please implement below concepts as mandatory while designing this Case Study

1. Create a Maven Project and update POM.XML accordingly to implement this Case Study.

2. Create a branch name – CaseStudy\_Intermediate and implement your code in that branch. After coding is completed commit and push your code into that branch.

3. URL should be initialized at a Suite level (Using @BeforeSuite Annotation)

4. Create Generic Functions to read the data

5. Create Generic Functions to capture screen shots in the report

6. Create TestNG.xml and run the test cases from TestNG.xml

**2.Implement below Case Study using Selenium with Python Concepts**

1. Launch a below URL https://www.saucedemo.com/

2. Verify SWAG LABS is present on the Web Page

3. Add any one of the Item to cart (Click on ADD TO CART button)

4. Click on right corner of the button and verify item is added to the cart

5. Click on left corner of the button and click on LOGOUT button

Note: Implement Python Case Study using pytest framework and write Generic Functions while calling the methods.

**3.Implement below Case Study using POSTMAN API Automation**

1. Create a collection in Postman and create positive and negative cases for below API. And use get method to get valid responses.

Verify the HTTP Status Codes, while running Positive and Negative Test Cases.

To validate the HTTP Status Codes and responses use Assertions concept in PostMan

Run all requests using Collection in Postman

User can parameterize the delayId to read data from CSV Files with Positive and Negative combinations.

https://reqres.in/api/users?delay={delayId}

https://reqres.in/api/users?delay=3

4.Introduction:

In this case study, creation of a Test Automation Framework using Android and Appium to ensure its quality and reliability.

TOOL CHAIN

Competency Skill Skill Detail

Automation Testing

Framework Appium, Maven, UIAutomator Viewer, Appium Inspector Medium Complexity

BUSINESS-REQUIREMENT

User Story # User Story Name User Story

US\_01 Developing a Test Automation Framework using Android and Appium: Framework Architecture and Design Analyze the app's architecture and identify key elements for automation. Design a modular automation framework architecture, considering the Page Object Model (POM) pattern. Choose appropriate design patterns and create a project structure for the framework.

US\_02 Set Up Environment and Dependencies Install Appium and necessary dependencies for test automation. Create a new project for the automation framework using a suitable programming language (e.g., Java).

US\_03 Page Object Model Implementation Create page classes representing different app screens and their elements. Implement methods to interact with UI elements using Appium's APIs. Utilize the Page Object instances to perform actions and validations in test scripts.

US\_04 Test Case Design and Execution Design test cases covering various app functionalities and user scenarios. Implement test scripts using Appium's API to interact with the app. Incorporate assertions and validations to verify expected outcomes. Implement dynamic data generation or data-driven testing where required.

US\_05 Configuration Management and Parallel Execution Implement configuration management for different Android devices and versions. Utilize Appium's capabilities to run tests in parallel on multiple devices. Integrate with a testing framework (e.g., TestNG) to manage test execution and reporting.

US\_6 Reporting and Continuous Integration Implement reporting mechanisms to capture test results and logs. Integrate the framework with CI/CD tools (e.g., Jenkins) for automated test execution. Ensure that reports are generated and accessible within the CI environment.

EXPECTED DELIVERABLES

Provide the architecture of the automation framework, sample code snippets, test cases, and evidence of successful test executions.

SUBMISSION

Submit the project report, framework source code, configuration files, and any additional resources.