UNIT 5:

1. In your test automation project, you encounter a web page with dynamic elements that change their properties or IDs every time the page loads. How would you handle these dynamic elements in your test automation scripts?

Solution:

1. Use Stable Locators:

- **Relative Locators: Use relative locators (e.g., finding elements based on their position relative to other elements) instead of relying on IDs or classes that change dynamically.**
- **XPath/CSS Selectors: Use XPath or CSS selectors that are resilient to changes in the element's properties. For example, use attributes that are less likely to change, like text or hierarchical structure.**

2. Implement Explicit Waits:

- **WebDriverWait: Use explicit waits to wait for elements to become present, visible, or clickable before interacting with them. This approach helps handle elements that load dynamically.**

2. You want to implement data-driven testing for a login functionality where the input data (username and password) varies. How would you design and implement a data-driven test?

Solution:

1. Prepare Test Data:

- **Store test data in an external file, such as an Excel sheet, CSV file, or JSON file.**

2. Read Data in Test Scripts:

- **Use libraries to read data from the file and iterate through it to run tests with different data sets.**

3. Implement Data-Driven Test:

- **Example Using Python + Selenium + CSV File:**
- **CSV File (data.csv)**
- **username,password**
- **testuser1,securepass1**
- **testuser2,securepass2**
- **testuser3,securepass3**

3.Your test automation suite is growing, and maintaining the scripts has become challenging. What strategies would you use to manage and maintain your test scripts effectively?

Solution:

1. Modularize Test Scripts:

- **Reusable Components: Break down test scripts into reusable functions or methods. For example, create separate functions for login, registration, and other common actions.**

2. Use Page Object Model (POM):

- **Encapsulation: Implement Page Object Model to separate test logic from page-specific locators and actions. This makes it easier to update locators and interactions when the UI changes.**

3. Implement Version Control:

- **Source Control: Use version control systems like Git to manage changes to test scripts and collaborate with team members.**

4. Regularly Review and Refactor:

- **Code Reviews: Periodically review and refactor test scripts to remove redundancy and improve maintainability.**

5. Use Test Automation Frameworks:

- **Frameworks: Implement test automation frameworks (e.g., TestNG for Java, Pytest for Python) that provide structure and best practices for organizing and executing tests.**