WireApps - Intern Quality Assurance Engineer

Technical Assessment

3. Test Automation Basics

1. Requirement Analysis

- Understand the Application: Gather detailed requirements for functionalities such as user login, registration, product search, add to cart, checkout, and order history.

- Identify Test Scenarios: Define what needs to be tested for each functionality, including edge cases and error handling scenarios.

2. Choose the Right Tools

- Test Automation Tools: Select appropriate tools based on the technology stack of the application. Common choices include:

  - Selenium: For web application testing.

- Test Management Tools: Tools like JIRA for managing test cases and tracking results.

- Continuous Integration (CI) Tools: Jenkins, GitLab CI, or GitHub Actions for automating test execution.

3. Framework Design

- Architecture: Decide on a framework architecture such as Page Object Model (POM), Model-View-Controller (MVC), or Behavior-Driven Development (BDD).

- Modularity: Structure the framework to promote reusability and separation of concerns. For example:

  - Page Objects: Create classes to represent different pages of the application.

  - Test Data Management: Separate test data from test scripts, possibly using external data sources like Excel or databases.

- Utilities: Implement utility classes for common tasks like reading configuration files, handling screenshots, or managing test reports.

- Assertions: Define a set of reusable assertions to verify expected outcomes.

4. Test Case Design

- Define Test Cases: Write detailed test cases for each functionality. Ensure they cover positive, negative, and boundary scenarios.
- Parameterization: Use data-driven testing to handle multiple data sets and test different scenarios with varying inputs.
- Test Environment: Set up test environments that mirror production as closely as possible.

5. Implementation

- Setup Framework: Configure the framework, including integration with CI/CD tools.
- Write Test Scripts: Develop test scripts based on the designed test cases and framework structure.
- Implement Error Handling: Ensure proper logging and error handling in test scripts to facilitate debugging.

6. Execution and Reporting

- Run Tests: Execute tests regularly, especially after new builds or deployments.
- Generate Reports: Implement reporting mechanisms to capture test results, logs, and screenshots. Tools like Allure or Extent Reports can be useful for this.

7. Maintenance

- Update Test Cases: Regularly update test cases and scripts to accommodate changes in the application.
- Monitor and Optimize

8. Best Practices

- Version Control
- Code Reviews
- Documentation