Name: Ashwini S Borse
Su-id: 4205533
Course: 5210

Assignment 5: Web Testing

HW5 – Web Testing

- **3 Learnings from paper "Black-Box and White-Box Test Case Generation for RESTful APIs: Enemies or Allies?"**
  1. **Hybrid Testing Superiority:** The empirical study suggests that combining black-box and white-box testing in a hybrid approach tends to be more effective than using either approach in isolation. The hybrid technique achieved the highest code coverage in the majority of scenarios and was the only technique capable of uncovering bugs in all tested systems. This highlights the potential benefits of integrating both testing strategies for comprehensive and effective testing of RESTful APIs.
  2. **System Complexity Impact on Testing Approach:** The study reveals that the choice between black-box and white-box testing depends on the complexity of the system under test (SUT). While black-box testing is faster and preferred for large and complex systems, white-box testing may be more effective for simpler or smaller systems. This understanding emphasizes the importance of tailoring testing strategies based on the characteristics of the specific API being tested.
  3. **Manual Setup Trade-Off and Deterministic Test Suites:** The lessons learned and challenges highlighted the trade-off between automation and test thoroughness. Despite the advantages of automation, all tested approaches (black-box, white-box, and hybrid) required some level of manual setup. Additionally, the deterministic nature of white-box test suites, which allows for resetting the system state between test cases, was contrasted with the non-deterministic nature of black-box test suites, especially in stateful APIs. This trade-off and the considerations around manual effort underscore the practical challenges in achieving fully automated testing and the need for a balanced approach.

- **3 Learnings from postman:**
  1. **Versatile API Testing Capabilities:** Postman supports various types of API testing, including REST, SOAP, and plain HTTP. This versatility allows testers to work with different types of APIs and protocols within a single tool. Whether you are dealing with RESTful services or SOAP-based web services, Postman provides a unified environment for creating and executing test requests.
  2. **Ease of Use and Quick Adoption:** Postman is known for its user-friendly interface, making it easy for both beginners and experienced testers to quickly adopt and start using the tool. Testers can create and send HTTP requests within minutes, and the intuitive design contributes to a smooth testing experience. The tool's simplicity is a key factor in its popularity, with over 8 million users choosing Postman as their preferred API testing tool.
  3. **Comprehensive Feature Set for Testing and Collaboration:** Postman offers a rich set of features that cater to various aspects of API testing and collaboration:
     - Testing Capabilities: Postman serves as an easy-to-use REST client, allowing testers to create and run manual and automated API tests. It supports the creation of test scripts, assertions, and validations.
     - Cross-Platform Support: Postman can be run on multiple platforms, including Mac, Windows, Linux, and Chrome Apps. This cross-platform compatibility ensures that testing teams can use the tool regardless of their operating system.

- Integration with CI/CD Tools: Postman integrates seamlessly with CI/CD tools like Jenkins and TeamCity, facilitating the incorporation of API testing into continuous integration and deployment pipelines.
- Knowledge Sharing: Postman enables users to package requests and expected responses, making it easy to share test scenarios and knowledge within a team. This collaborative feature promotes efficient communication and consistency in testing approaches.

♦ 3 Learning from test:

1. **Cross-Browser Testing is Essential:** When automating UI interaction with a Page Model (Test 1) or using a JavaScript framework (Test 2), it's crucial to conduct cross-browser testing. Browsers interpret and render web pages differently, which can lead to inconsistent test results. By running tests across popular browsers like Chrome, Firefox, Safari, and Edge, you ensure that your application functions correctly for a diverse user base. Tools like Selenium WebDriver can be employed to execute tests on multiple browsers, providing confidence in the application's compatibility.
2. **Multi-Platform Testing Enhances Reliability:** In addition to cross-browser testing, extending your tests to different operating systems is crucial (desktop and mobile). Test 1 and Test 2 should be executed on various operating systems, such as Windows, macOS, and Linux. This ensures that your application is robust and performs consistently across different environments. Additionally, including mobile devices in your testing scope is essential, given the growing usage of smartphones and tablets. Tools like Appium can facilitate mobile testing, allowing you to validate the responsiveness and functionality of your application on diverse platforms.
3. **Comprehensive Test Coverage Requires Both UI and API Testing:** While Test 1 and Test 2 focus on UI automation, it's important to incorporate API testing into your overall testing strategy (Test 3). UI tests validate the user interface and user experience, but API tests examine the functionality and reliability of the underlying services. By using tools like Postman for API testing, you gain insights into how different components of your application interact. This dual approach ensures comprehensive test coverage, addressing both front-end and back-end aspects of your application. It also helps identify issues early in the development process, promoting a more robust and resilient application.

♦ Test Summary:

1. Test 1: Use a Page Model to automate UI interaction
    - Test-Steps: Open URL, maximize window, Go to Search Bar, Enter Query, Select Query from dropdown menu, click, navigate to new page and validate data.
    - Challenges faced: Tried to execute test-cases in different OS enviornement, such as Window, Mac and OS. Faced challenges while inspecting elements. Also, these elements are different for different platforms. So difficult to use same set of test-cases across different browsers and platforms.

- Able execute tests in 3 Web Browsers and Windows and MAC OS. Unable to run on Mobile.

2. Test 2: Use a JavaScript framework to automate the UI framework
   - Test-Steps: Used Cypress framework
     - Test 1: Open URL, Navigate to Search Query and validate results.
     - Test 2: Open URL, click hyper_link and navigate to new page
   - Challenges: I faced challenges for setting up and learning Cypress. Refering its page helps.
   - Able to execute and validate Query.

3. Test 3: Use Postman or an alternate harness to automate API testing, for example https://datausa.io/api/data?drilldowns=Nation&measures=Population
   - Test-Steps: Used Postman Framework
     - Test-case 1: Get data from above API request using parameter Year=2020
     - Test-case 2: Get data from above API for the entries whose population is more than 3.25B.
   - Challenges Faced: Challenges faced in learning and setting up
   - Able to execute test-cases.

- GitHub: https://github.com/AshwiniBorsee/Software_Testing_And_Debugging/tree/main/HW5

Word Count 1123