**Module -5 Automation and Selenium**

**Q.1 What is automation Testing?**

**Automation Testing** is a software testing technique in which automated scripts and tools are used to perform tests on software applications. The goal of automation testing is to increase the efficiency, effectiveness, and coverage of the testing process while reducing the manual effort involved. Automation testing involves running predefined test cases using software tools rather than performing the tests manually.

**Key Points of Automation Testing:**

1. **Automated Test Scripts**: Instead of testers manually executing test cases, they write automated scripts that simulate user interactions with the software. These scripts are typically written in programming languages such as Python, Java, C#, or scripting languages like JavaScript or Ruby, depending on the testing tool being used.
2. **Test Automation Tools**: Automation testing relies on various tools that help in creating, running, and managing automated tests. These tools can be specific to the type of testing being done (e.g., functional, performance, regression). Some popular test automation tools include:
   * **Selenium**: For web application testing.
   * **QTP/UFT (Unified Functional Testing)**: For functional testing of desktop and web applications.
   * **JUnit/TestNG**: For unit testing in Java applications.
   * **Appium**: For mobile application testing.
   * **LoadRunner**: For performance and load testing.
   * **Cucumber**: For behavior-driven development (BDD) testing.
3. **Test Execution**: Once the test scripts are written, they are executed by the automation tools. These tools interact with the application under test (AUT) and perform various operations such as data entry, clicking buttons, navigating through pages, etc.
4. **Test Validation and Reporting**: After the test execution, the tool compares the actual results with the expected results. If there are any discrepancies, the test is marked as failed. Most automation tools provide detailed reports on the test execution, including logs, screenshots, and performance metrics.
5. **Continuous Integration/Continuous Delivery (CI/CD)**: Automated tests are often integrated into the CI/CD pipeline, where they are executed every time new code is pushed to the repository. This allows early detection of issues, providing rapid feedback to developers.

**Advantages of Automation Testing:**

1. **Speed and Efficiency**: Automated tests can be executed much faster than manual tests, especially for repetitive tasks. This makes it possible to run tests more frequently and in parallel (on different environments or browsers).
2. **Reusability**: Once test scripts are written, they can be reused across different versions of the application, reducing the need for re-writing tests.
3. **Regression Testing**: Automation is particularly effective for regression testing. It can quickly verify that previously working functionality has not been broken by new code changes.
4. **Increased Test Coverage**: Automated tests can cover a large number of test cases, including edge cases that might be impractical to test manually.
5. **Cost-Effectiveness (Long-Term)**: Although automation requires an initial investment in tools, scripts, and setup, it can save money in the long run by reducing the amount of manual effort needed for repetitive testing tasks.
6. **Early Bug Detection**: Automated tests can be run frequently, catching issues early in the development cycle and providing quick feedback to developers.
7. **Consistency**: Automated tests eliminate human errors associated with manual testing. Tests are executed exactly the same way every time.

**Disadvantages of Automation Testing:**

1. **Initial Investment**: Setting up automated tests requires time, effort, and resources for creating test scripts, setting up tools, and training team members. It might not be cost-effective for small projects or one-off tests.
2. **Maintenance**: Automated scripts require maintenance, especially when the application changes. If the user interface or underlying code changes, test scripts may need to be updated.
3. **Not Suitable for All Test Cases**: Automation testing is most beneficial for repetitive tasks and large test suites, but it may not be effective for tests that require human judgment (e.g., user experience testing, visual testing, etc.).
4. **Tool Limitations**: Not all tools are capable of testing every type of application (e.g., some tools may not be suitable for mobile applications or certain technologies).
5. **False Sense of Security**: Automation can give a false sense of security. Just because automated tests pass, it doesn’t mean that the software is entirely bug-free. Manual testing, including exploratory testing, is still necessary.

**Types of Automation Testing:**

1. **Unit Testing**: Testing individual components or functions in isolation. Common tools: JUnit, TestNG.
2. **Functional Testing**: Ensures that the application functions as expected. Common tools: Selenium, QTP/UFT.
3. **Regression Testing**: Ensures that new code changes don't negatively impact existing functionality. Automation is ideal for regression testing.
4. **Performance Testing**: Tests the application's behavior under load. Tools like **LoadRunner**, **JMeter**, and **Gatling** are used for performance testing automation.
5. **Acceptance Testing**: Ensures that the application meets the requirements and specifications. BDD tools like **Cucumber** are often used for automated acceptance testing.
6. **Smoke Testing**: Basic tests that check the core functionality of the application. Often automated to quickly check if the build is stable enough for further testing.

**When to Use Automation Testing:**

* **Repetitive Test Cases**: If you need to run the same tests multiple times across different environments or versions of the application.
* **Regression Testing**: When new code is integrated, it’s important to ensure that previously working features are not broken.
* **Large Test Suites**: For projects with large numbers of test cases that would be time-consuming to execute manually.
* **Performance and Load Testing**: When you need to simulate a large number of users interacting with the application, such as load testing or stress testing.
* **Continuous Integration (CI)**: When automated tests need to run frequently as part of a CI pipeline to catch issues early.

**In Summary:**

Automation testing uses specialized tools and scripts to automatically execute test cases on software applications. It is especially valuable for repetitive, large-scale testing tasks like regression, performance, and load testing. While it offers speed, efficiency, and cost-effectiveness in the long term, it requires initial investment in terms of time, tools, and maintenance. It's not a replacement for manual testing but is best used to complement it by covering repetitive tasks and enhancing test coverage.

**Q.2 which are the browsers supported By Selenium Ide?**

**Selenium IDE** (Integrated Development Environment) supports the following browsers for testing:

**1. Mozilla Firefox:**

* Selenium IDE was originally developed as a Firefox extension, and it continues to be one of the main supported browsers for Selenium IDE.
* It supports a wide range of features, including recording, running, and debugging tests within Firefox.

**2. Google Chrome:**

* Selenium IDE is also available as a Chrome extension.
* Users can record and play tests in Chrome, making it one of the most popular browsers for automation testing with Selenium IDE.

**3. Microsoft Edge:**

* As of recent updates, **Selenium IDE** also supports **Microsoft Edge** via its extension, similar to the way it works with Chrome and Firefox.

**4. Internet Explorer (Limited support):**

* Selenium IDE had support for **Internet Explorer (IE)** earlier, but this is now limited and largely obsolete due to the discontinuation of support for older versions of Internet Explorer and the focus on Microsoft Edge.

**5. Safari (Mac OS only):**

* While **Safari** does not natively support Selenium IDE, there have been workarounds and tools available for using it in certain versions of macOS, though this is less common.

**Summary:**

* **Firefox** and **Chrome** are the primary browsers supported by **Selenium IDE** for recording and running automated tests.
* **Microsoft Edge** also has support for Selenium IDE.
* **Internet Explorer** and **Safari** have limited or less active support, with IE being phased out in favor of Edge.

For broader Selenium WebDriver testing (including beyond IDE), Selenium can support many browsers, including Firefox, Chrome, Edge, Safari, and Internet Explorer, but for Selenium IDE, the focus is primarily on Firefox and Chrome.

**Q.3 What are the benefits of Automation Testing?**

**Automation Testing** offers a variety of benefits, especially in large, complex, or repetitive software projects. Here are the key advantages:

**1. Increased Speed and Efficiency**

* Automated tests can be executed **much faster** than manual tests. Once the test scripts are created, they can run in parallel or across multiple machines, significantly reducing the time taken to run tests, especially for large test suites.
* Tests that would take hours to complete manually can be done in minutes or seconds, speeding up the overall testing process.

**2. Reusability of Test Scripts**

* Once test scripts are written, they can be **reused** across different versions of the application without significant changes. This is particularly useful for regression testing, where the same tests need to be run across multiple builds or versions of the software.
* Reusable test scripts save time in the long run and reduce the effort involved in rewriting tests for every release.

**3. Better Coverage and Depth**

* Automation allows you to run a **larger number of test cases** in a shorter amount of time, ensuring more extensive test coverage.
* Automation can cover a wide range of test scenarios, including edge cases, that might be impractical or time-consuming to test manually.

**4. Cost-Effectiveness in the Long Run**

* While setting up automation requires an initial investment (time, tools, and resources), it **saves costs** over time by reducing the need for extensive manual testing.
* With automated tests, you can continuously execute tests throughout the development lifecycle, minimizing the risk of defects going undetected and reducing post-release maintenance costs.

**5. Faster Feedback and Early Bug Detection**

* Automated tests can be integrated into **Continuous Integration (CI)**/Continuous Delivery (CD) pipelines, providing developers with **rapid feedback** as soon as code changes are made.
* This leads to **early detection of bugs**, as tests are executed frequently (even multiple times a day), enabling quicker fixes and reducing the cost of fixing defects later in the development cycle.

**6. Consistency and Reliability**

* Automated tests execute exactly the same way every time they are run. This **eliminates human errors** associated with manual testing, such as forgetting steps, incorrect data entry, or missing edge cases.
* Automation ensures **consistent test execution**, which is especially valuable when testing across different environments or platforms.

**7. Parallel Execution**

* Automated tests can be executed across **multiple machines, browsers, or devices** simultaneously, which helps in reducing the time taken to validate the application across various environments.
* This is particularly beneficial for cross-browser testing, where you need to validate that the application works on different browsers or devices (e.g., Chrome, Firefox, Safari, mobile browsers).

**8. Improved Accuracy and Precision**

* Automated tests are precise because they follow predefined steps and compare actual outcomes to expected results without human involvement.
* This ensures that there is no **variation in test execution** that could occur due to human error, such as misclicking or skipping steps.

**9. Better Resource Utilization**

* By automating repetitive tests, manual testers can focus on more complex tasks, like **exploratory testing**, usability testing, or test case design.
* Automation helps make better use of resources, as testers can focus on areas that require human judgment while the automation tools handle repetitive and time-consuming tests.

**10. Support for Complex Scenarios**

* Automation is effective for testing **complex scenarios** that require a high degree of precision or are too repetitive and time-consuming for manual testers. Examples include:
  + **Performance testing**: Automated tools can simulate large numbers of virtual users to test the application under load.
  + **Stress testing**: Automated tests can push the system to its limits to ensure it can handle extreme conditions.

**11. Continuous Testing in Agile and DevOps**

* Automation is crucial in Agile and DevOps environments, where code changes frequently, and testing needs to be **continuous**.
* Automated tests can be run frequently as part of the CI/CD pipeline, ensuring the software is always tested and validated after each change or update.

**12. Regression Testing**

* Automated tests are particularly useful for **regression testing**, which ensures that new changes or features do not introduce defects into the existing system.
* Running automated regression tests on every new build helps catch issues early and ensures stability over time.

**13. No Need for Human Intervention**

* Once automated test scripts are created, they can run **without human involvement**. This allows testing to occur outside of regular working hours, such as during the night or over weekends, providing more flexibility and efficiency.

**14. Improved Test Reporting**

* Most automation tools offer comprehensive **test reports** that include detailed logs, screenshots, and performance metrics, helping testers and developers to quickly analyze issues and determine the root cause.
* These reports are consistent and can be easily shared with stakeholders for better communication.

**15. Supports Complex Applications and Large-Scale Systems**

* Automation testing can handle **large-scale systems** and complex applications with many interdependencies and scenarios that might be difficult to cover manually.
* It ensures that the entire application is tested across different modules, providing a more thorough check than manual testing.

**Summary:**

Automation testing significantly enhances the efficiency, reliability, and coverage of software testing. It offers benefits such as **speed**, **reusability**, **early bug detection**, **cost savings**, and **better test coverage**. Automation is especially valuable for repetitive, time-consuming tasks like regression testing, load testing, and performance testing. However, it requires initial effort to create the automation scripts and may not be suitable for all test cases, particularly those that require human judgment (e.g., visual testing). It’s most effective when used in combination with manual testing to ensure comprehensive test coverage.

**Q.4 what are the Advantage of Selenium?**

**Selenium** is one of the most popular open-source tools for automating web applications. It provides numerous benefits for web application testing, making it a preferred choice for many organizations and testing teams. Here are the **key advantages of Selenium**:

**1. Open-Source and Free**

* **No Cost**: Selenium is completely open-source and free to use. This makes it an attractive choice for organizations with limited budgets or those looking to avoid licensing fees associated with proprietary testing tools.

**2. Supports Multiple Browsers**

* **Cross-Browser Testing**: Selenium supports all major web browsers, including **Google Chrome**, **Mozilla Firefox**, **Safari**, **Internet Explorer**, and **Microsoft Edge**. This allows you to run automated tests on multiple browsers to ensure cross-browser compatibility.
* **Browser Independence**: Selenium's ability to test on multiple browsers without changing test scripts makes it highly flexible and useful for testing in various environments.

**3. Cross-Platform Compatibility**

* Selenium can be run on multiple operating systems, including **Windows**, **macOS**, and **Linux**. This means that you can test web applications in different environments to ensure compatibility across platforms.
* It also supports **cloud-based testing** and can be integrated with tools like **Sauce Labs** or **BrowserStack** for testing on a wide range of operating systems and browsers.

**4. Multiple Language Support**

* **Programming Language Flexibility**: Selenium supports a variety of programming languages, including **Java**, **Python**, **C#**, **Ruby**, **JavaScript**, and **Kotlin**. This means testers and developers can use their preferred language to write test scripts.
* **Ease of Integration**: Selenium integrates well with various test frameworks like **JUnit**, **TestNG**, **NUnit**, and **Cucumber**, allowing teams to leverage the power of these frameworks for creating robust test suites.

**5. Support for WebDriver**

* **WebDriver** is an interface provided by Selenium that allows you to interact directly with web browsers. WebDriver executes tests by interacting with the browser’s native API, providing a more reliable and stable way to automate web applications compared to Selenium’s older **Selenium RC** (Remote Control).
* WebDriver simulates real user interactions, such as clicking buttons, typing in text fields, selecting options, and verifying element visibility.

**6. Supports Parallel Test Execution**

* Selenium can execute tests in parallel, which allows for faster test execution. This is achieved by using tools like **TestNG**, **JUnit**, or **Selenium Grid** to run tests simultaneously on multiple machines, reducing the overall test cycle time.
* **Selenium Grid** allows you to distribute tests across different environments and hardware, making it possible to run tests on multiple devices, browsers, and operating systems at once.

**7. Large Community and Documentation**

* **Community Support**: Being one of the most widely used tools in the testing community, Selenium has a vast and active community. This ensures that help is readily available, and you can find solutions to common issues easily.
* **Comprehensive Documentation**: Selenium has excellent documentation that includes tutorials, guides, and reference materials for users of all experience levels, making it easier to get started and troubleshoot issues.

**8. Integration with Other Tools**

* **CI/CD Integration**: Selenium integrates well with popular **Continuous Integration (CI)** tools like **Jenkins**, **CircleCI**, and **Travis CI**, enabling automated testing to be part of the CI/CD pipeline.
* **Test Reporting Tools**: Selenium can also be integrated with **reporting tools** like **Allure**, **ExtentReports**, and **TestNG Reports** to generate detailed test reports, logs, and visualizations of test results.
* **Bug Tracking and Project Management**: Selenium can be integrated with bug tracking and project management tools like **JIRA** and **Bugzilla** to link test cases, failures, and issues directly to the testing workflow.

**9. Supports Mobile Testing (Appium)**

* **Mobile Testing**: While Selenium is designed primarily for web applications, it can be extended for mobile testing by using **Appium**, a tool that allows Selenium WebDriver to interact with mobile browsers and apps (both Android and iOS).
* This makes it possible to use Selenium for testing both mobile web applications and native mobile applications.

**10. Easy to Use**

* Selenium provides an easy-to-use interface and allows testers to write tests in simple, understandable code, especially for those familiar with object-oriented programming languages.
* Many testers and developers are already familiar with languages like Java and Python, making it easier to adopt Selenium for their testing needs.

**11. Extensive Support for Web Elements**

* Selenium provides support for various web elements like **forms**, **checkboxes**, **radio buttons**, **dropdowns**, and **dynamic content**. It allows testers to easily interact with and verify these elements through the **WebDriver** API.
* It also supports actions like **mouse hovering**, **drag-and-drop**, and **keyboard actions**, which are often required in modern web applications.

**12. Rich Set of Features**

* **Handling Popups**: Selenium can handle popups, alert boxes, and modal dialogs that are common in web applications.
* **Handling Dynamic Web Elements**: Selenium provides mechanisms for waiting for elements to appear or become interactive before performing actions, such as **Implicit Wait**, **Explicit Wait**, and **Fluent Wait**.

**13. Scalability**

* Selenium tests can scale easily as your application grows. With tools like **Selenium Grid**, you can add more machines or environments to your test infrastructure to run more tests in parallel.
* This scalability is especially useful for large test suites and complex applications.

**14. Flexibility and Control**

* **Customizability**: Selenium offers great flexibility in test creation, allowing you to customize tests as per your needs. You can write complex test scenarios using the full power of programming languages.
* **Direct Control**: Selenium allows direct control over web browsers, making it possible to test a wide range of use cases and scenarios that cannot be easily tested with other tools.

**15. Support for Headless Browsers**

* **Headless Browsers**: Selenium supports headless browsers like **ChromeHeadless** and **PhantomJS**, which can run tests without launching a visible browser window. This can be useful for running tests on a server or in CI pipelines where a GUI is not required.

**Summary:**

Selenium offers a range of advantages, including **open-source availability**, **cross-browser and cross-platform compatibility**, **language flexibility**, **integration with CI/CD pipelines**, and **parallel test execution**. It is highly flexible, scalable, and supports mobile testing through Appium. Its large community, rich documentation, and support for various testing tools make Selenium an excellent choice for automated web application testing.

**Q.5why tester should opt for Selenium and not QTP?**

When choosing between **Selenium** and **QTP (QuickTest Professional)** — now known as **UFT (Unified Functional Testing)** — there are several factors to consider based on the project requirements, team skills, and overall budget. Here are the key reasons why a **tester might opt for Selenium** instead of QTP/UFT:

**1. Cost-Effectiveness**

* **Selenium** is **open-source and free** to use, which makes it an attractive option for organizations with limited budgets. In contrast, **QTP/UFT** is a **commercial tool** that requires a significant **licensing fee** for each user, making it costly for small and medium-sized enterprises (SMEs).
* Over time, Selenium can result in substantial savings in tool costs, especially when scaling the automation effort.

**2. Cross-Browser and Cross-Platform Support**

* **Selenium** supports all major browsers, including **Google Chrome**, **Mozilla Firefox**, **Safari**, **Internet Explorer**, and **Microsoft Edge**. This makes it ideal for testing across different browser versions and platforms.
* Selenium also supports multiple operating systems, such as **Windows**, **macOS**, and **Linux**, offering **cross-platform compatibility**.
* **QTP/UFT** has good support for web applications but has limited browser support, especially with modern browsers. It also has more stringent licensing for cross-platform testing.

**3. Programming Language Support**

* **Selenium** supports a wide range of programming languages such as **Java**, **Python**, **C#**, **Ruby**, **JavaScript**, and **Kotlin**. This provides flexibility for testers to work in the language they are most comfortable with or the one already used by the development team.
* **QTP/UFT**, on the other hand, primarily uses **VBScript** for writing tests, which may require testers to learn a new scripting language if they are not already familiar with it. Many testers find VBScript restrictive compared to more modern programming languages.

**4. Community Support and Open-Source Ecosystem**

* **Selenium** has a vast and active **community**, as it is one of the most widely used open-source testing tools. The community provides **free resources**, such as forums, blogs, tutorials, and documentation, which makes it easier to troubleshoot issues and learn.
* **QTP/UFT** has commercial support through **Micro Focus**, but it doesn't have the same level of active community-driven innovation and contributions as Selenium. Also, the **knowledge base** for QTP is not as extensive or freely available as Selenium’s open-source resources.

**5. Flexibility and Extensibility**

* **Selenium** is highly **extensible** and can be integrated with a variety of tools for test reporting, CI/CD, performance testing, and more. It integrates seamlessly with tools like **Jenkins**, **TestNG**, **JUnit**, **Maven**, and **Cucumber**.
* It also supports **headless browsers** (e.g., **ChromeHeadless**, **PhantomJS**) for running tests without UI, which is beneficial for running tests in CI pipelines or on servers.
* **QTP/UFT** is more of a **closed ecosystem** and may not offer the same level of integration with open-source tools. Its extensibility is limited compared to Selenium.

**6. Parallel Test Execution**

* **Selenium** supports parallel test execution using **Selenium Grid**, which allows running tests on multiple machines, browsers, and environments simultaneously, thus speeding up the test execution process.
* **QTP/UFT** also supports parallel execution, but **Selenium** provides more flexibility in terms of the number of machines and browsers, and it integrates more easily with modern CI/CD pipelines.

**7. Mobile Testing (Appium)**

* Selenium can be used for **mobile testing** via **Appium**, which allows automation for both **Android** and **iOS** applications. This makes Selenium a better choice for projects that require both web and mobile testing.
* **QTP/UFT** supports mobile testing as well but requires a separate license for **mobile testing**, adding to the cost and complexity of usage. Appium, being open-source, provides a more affordable and flexible alternative for mobile automation.

**8. Faster Test Execution and Scalability**

* **Selenium** is faster compared to **QTP/UFT** for certain types of tests, especially when tests are executed in parallel. It scales easily by adding more machines or environments via **Selenium Grid**.
* **QTP/UFT** might require more setup and resources for similar scalability, and its licensing model can complicate this process, especially when scaling to multiple machines.

**9. Easy to Learn and Use (for Developers and Testers)**

* **Selenium** is favored by developers due to its support for standard programming languages like **Java**, **Python**, and **C#**, which are commonly used in development teams. Testers who are familiar with these languages can easily learn Selenium without needing to switch to a proprietary language.
* **QTP/UFT**, on the other hand, uses **VBScript**, which might require testers to learn a new scripting language. This could be a barrier for those who are not familiar with VBScript, especially for developers who are more accustomed to programming languages like Java or C#.

**10. No Vendor Lock-In**

* **Selenium** is **vendor-independent**, meaning that there is no lock-in to a specific vendor or proprietary tool. You are free to use it with any development stack and can easily switch tools or vendors without being restricted by licensing terms.
* **QTP/UFT** is a proprietary tool from **Micro Focus** and is tied to their licensing terms and conditions. Switching to another tool or vendor might require a more complicated transition process.

**11. Test Reporting and Debugging**

* **Selenium** integrates easily with test reporting tools such as **TestNG**, **JUnit**, and **ExtentReports**, which provide detailed logs and reports.
* **QTP/UFT** has built-in reporting features, but Selenium’s integration with open-source reporting tools gives more flexibility and customization options.

**12. Support for Cloud-Based Testing**

* **Selenium** integrates easily with **cloud-based testing platforms** like **Sauce Labs**, **BrowserStack**, and **CrossBrowserTesting**, allowing tests to be run on multiple browsers and devices in the cloud, eliminating the need for on-premises hardware.
* While **QTP/UFT** has some cloud testing support, it is not as extensive or flexible as Selenium’s open-source integrations with these platforms.

**Summary of Key Reasons to Choose Selenium Over QTP/UFT:**

* **Cost**: Selenium is free, while QTP/UFT requires costly licenses.
* **Flexibility**: Selenium supports a wide range of programming languages, browsers, and platforms, and integrates seamlessly with CI/CD pipelines.
* **Cross-Browser and Cross-Platform Support**: Selenium supports all major browsers and operating systems, offering better flexibility for testing across environments.
* **Community Support**: Selenium has a vast and active open-source community, making it easy to find resources and get help.
* **Mobile Testing**: Selenium, through Appium, supports mobile testing for both Android and iOS.
* **Integration with Other Tools**: Selenium integrates with a variety of tools for reporting, CI/CD, and test management.
* **No Vendor Lock-In**: Selenium is vendor-independent, while QTP/UFT is a proprietary tool.

In conclusion, while **QTP/UFT** may still be a good choice for certain enterprise environments with heavy reliance on desktop applications or specific integration requirements, **Selenium** is the better choice for most modern, cost-conscious, and flexible web and mobile automation testing needs.

**Q.6 To validate the tops technologies website Contact us page and enter your friend detail at last “Login and sidemenu” https://www.saucedemo.com/**

**Part 1: TOPS Technologies – Contact Us Page Validation**

URL: https://www.tops-int.com/contact-us

**Test Scenario 1: Check all form fields are visible**

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_01 | Verify all form fields are displayed | Open contact page | All fields: Name, Email, Phone, Course, City, Message are visible |

**Test Scenario 2: Field validation with blank input**

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_02 | Submit form with empty fields | Click submit without filling anything | Error messages should be displayed |
| TC\_03 | Enter invalid email | Enter abc in email and submit | Show email format error |
| TC\_04 | Enter invalid phone number | Enter 123 in phone and submit | Show phone number format error |

**Test Scenario 3: Valid data submission (Friend details)**

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_05 | Submit form with valid friend details | Enter: Priya Shah, priyashah@email.com, 9876543210, Software Testing, Ahmedabad, and a message | Show success or thank you message |

**Part 2: Saucedemo.com – Login and Side Menu Validation**

URL: [https://www.saucedemo.com](https://www.saucedemo.com/)  
 Username: standard user  
 Password: secret\_sauce

**Test Scenario 1: Valid Login**

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_06 | Login with correct credentials | Enter username and password, click Login | Redirect to product inventory page |

**Test Scenario 2: Side menu items verification**

| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| TC\_07 | Open and verify side menu | Click on side menu (≡) | Menu items: All Items, About, Logout, Reset App State are visible |
| TC\_08 | Click on About | Click “About” from side menu | Redirect to company info page |
| TC\_09 | Click on Logout | Click “Logout” from side menu | Redirect back to login page |
| TC\_10 | Click on Reset App State | Click “Reset App State” from side menu | Cart and filters should reset |

**Defect Reporting Example**

| **Defect ID** | **Module** | **Description** | **Steps to Reproduce** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| DEF\_01 | Contact Form | No error on invalid phone number | Enter 123 as phone and submit form | Should show error | No error message shown | Open |

**Bottom of Form**