105 – Automation Testing Framework

Short Questions

Unit-1: Concepts of Software Testing

1. What is software testing, and why is it important?
2. Explain the difference between functional and non-functional testing.
3. What are the key differences between manual and automated testing?
4. What are the phases of the software testing life cycle (STLC)?
5. Define black-box testing and white-box testing.
6. What is regression testing, and when is it performed?
7. What is unit testing, and who is responsible for performing it?
8. How does smoke testing differ from sanity testing?
9. What are the main goals of acceptance testing?
10. Explain the concept of test coverage in software testing.

Unit-2: Selenium Python

1. What is Selenium, and how is it used with Python?
2. What are the key advantages of using Python with Selenium for automation?
3. How do you set up a Selenium environment in Python?
4. What is the purpose of the `webdriver` module in Selenium?
5. How do you locate web elements in Selenium using Python?
6. How can you interact with input fields, buttons, and links using Selenium Python?
7. What is the role of `WebDriverWait` in Selenium Python?
8. How can you handle pop-ups and alerts using Selenium in Python?
9. How do you capture screenshots during test execution in Selenium Python?
10. What is the use of the `implicitly_wait()` method in Selenium Python?

Unit-3: Selenium WebDriver Basics

1. What is Selenium WebDriver, and how does it differ from Selenium RC?
2. How do you start a WebDriver session in Selenium?
3. What are the different types of locators used in Selenium WebDriver?
4. Explain how to navigate between pages using Selenium WebDriver.
5. How can you handle drop-down menus using Selenium WebDriver?
6. What is the difference between `find_element()` and `find_elements()` in Selenium WebDriver?
7. How do you simulate mouse actions like clicking and double-clicking using WebDriver?
8. How do you handle browser windows or tabs using Selenium WebDriver?
9. What is the role of `execute_script()` in Selenium WebDriver?
10. How can you close a browser using Selenium WebDriver?

Unit-4: Advanced Selenium Topics

1. What is the Page Object Model (POM) in Selenium, and why is it used?
2. How do you implement data-driven testing using Selenium WebDriver?

3. What are some best practices for writing maintainable Selenium test scripts?
4. How do you handle dynamic elements in Selenium WebDriver?
5. How can you deal with CAPTCHA and other complex validations in Selenium tests?
6. Explain the concept of parallel test execution in Selenium.
7. How do you manage timeouts and retries in Selenium WebDriver?
8. What is the significance of the `Actions` class in Selenium?
9. How can you execute JavaScript code within a Selenium WebDriver test?
10. How do you integrate Selenium WebDriver with test frameworks like PyTest or UnitTest?

Unit-5: Advanced Selenium

1. What is Selenium Grid, and how does it facilitate distributed testing?
2. How do you configure Selenium Grid to run tests on multiple machines?
3. What are the advantages of using Selenium with Docker for test automation?
4. How can you use Selenium WebDriver for mobile testing?
5. Explain how to test web applications in different browsers using Selenium.
6. What is headless browser testing, and how do you set it up with Selenium?
7. How do you integrate Selenium WebDriver with CI/CD tools like Jenkins?
8. How can you handle file uploads and downloads using Selenium?
9. What is browser profiling, and how do you use it with Selenium WebDriver?
10. How do you generate detailed test reports after running Selenium tests?

Long Questions

Unit-1: Concepts of Software Testing

1. What is the main difference between manual and automation testing, and what are the pros and cons of each approach?
2. Name some tests that can be effectively performed using automated testing tools.
3. Explain the components of Selenium, including Selenium IDE, RC, WebDriver, and Grid.
4. How do you install Selenium IDE, Firebug, and Firepath in your browser for Selenium testing?
5. Describe the architecture and installation process of Selenium.
6. What is the Selenium Client Library, and how does it relate to the JSON Wire Protocol over HTTP?
7. Explain the concept of Browser Drivers in Selenium and why they are important.
8. Give an example of when you might choose manual testing over automated testing and vice versa.

Unit-2: Selenium Python

1. What are the advantages of using Selenium with Python for web automation?
2. How do you navigate web links using the get() method in Selenium with Python?
3. Describe the process of interacting with a webpage using Selenium in Python.
4. Explain the various methods for locating single elements in a web page, such as find_element_by_id and find_element_by_name.

5. What is the significance of the Action Chains object in Selenium, and what are some common actions you can perform using it?
6. Provide examples of Action Chain methods, such as click, double click, and drag and drop, and explain their usage.
7. How do you use the Key down and Key up actions in Action Chains with Selenium in Python?
8. Discuss the importance of the perform(), pause(), and release() methods in Action Chains.

Unit-3: Selenium WebDriver Basics

1. Explain how to import Selenium WebDriver packages and initialize the browser for web automation.
2. Describe the process of navigating to a website, accessing its login page, and fetching user_id and password fields using Selenium.
3. What are some commonly used methods in Selenium WebDriver, such as maximize_window(), get(), find_element_by_name(), and send_keys()?
4. Why is the Keys class from Selenium.webdriver.common.keys important, and how is it used in Selenium?
5. Provide a use case of automating the Facebook login using any web browser with Selenium WebDriver.
6. Describe the steps to automate the login process for Gmail using Selenium WebDriver in any browser.
7. How do you differentiate between find_element() and find_elements() in Selenium WebDriver?
8. Explain the concept of locators in Selenium and discuss dynamic Xpath and dynamic CSS in detail.

Unit-4: Advanced Selenium Topics

1. Describe the differences between find_element() and find_elements() methods in Selenium and when to use each.
2. What are locators in Selenium, and why are they essential for identifying elements on a web page?
3. Explain dynamic Xpath and dynamic CSS in the context of Selenium test automation.
4. How do you handle drop-downs in Selenium, and what methods are available for interacting with them?
5. Discuss the techniques for handling file uploads in Selenium.
6. Explain how to handle alerts, popups, and multi-windows in Selenium test automation.
7. What are mouse events in Selenium, and how do you perform actions like mouse hover, right-click, double-click, and drag and drop?
8. Describe the methods for capturing screenshots in Selenium, including capturing full-page screenshots.

Unit-5: Advanced Selenium

1. Differentiate between Implicit, Explicit, and Fluent Wait in Selenium. When would you use each of these wait strategies in your automation scripts?

2. Explain the purpose of the Apache POI library in Selenium testing. How can you use it to read and write data from an Excel file in your test automation?
3. Describe the process of conducting database testing using Selenium. Include details on testing with MySQL and DB2 databases.
4. How can you handle Ajax calls in Selenium test automation? Provide an example scenario.
5. What are listeners in Selenium, and how do they help in test automation? Give an example of a situation where you would use listeners.
6. Discuss how to handle JavaScript interactions in Selenium. Provide examples of using JavaScript to perform actions in your automation scripts.
7. How can you implement a robust exception handling mechanism in Selenium scripts, and why is it important?
8. Explain the process of integrating Selenium with TestNG for test management and reporting. What are the benefits of using TestNG with Selenium?

104 – Web Development Operations

Short Questions

Unit-1: Concepts of DevOps

1. What is the primary goal of DevOps?
2. Define DevOps and its key principles.
3. How does DevOps differ from traditional software development methods?
4. What are the major benefits of adopting DevOps practices?
5. What is the role of collaboration in DevOps?
6. What does the term “continuous delivery” mean in DevOps?
7. How does automation contribute to DevOps practices?
8. What are some common challenges faced when implementing DevOps?
9. What is the importance of version control in DevOps?
10. How do DevOps practices improve software quality?

Unit-2: DevOps Lifecycle and Automation Tools

1. What are the key phases of the DevOps lifecycle?
2. How does continuous integration fit into the DevOps lifecycle?
3. What role does monitoring play in the DevOps lifecycle?
4. Name three automation tools commonly used in DevOps.
5. What is the significance of configuration management in DevOps?
6. How does continuous deployment differ from continuous delivery?
7. Why is automated testing essential in the DevOps lifecycle?
8. What is the purpose of infrastructure as code (IaC) in DevOps?
9. Which tools are commonly used for infrastructure automation in DevOps?
10. How does containerization aid the DevOps lifecycle?

Unit-3: Ansible and Playbooks

1. What is Ansible, and why is it used in DevOps?
2. Explain the purpose of Ansible playbooks.
3. How does Ansible differ from other configuration management tools like Puppet or Chef?
4. What are the main components of an Ansible playbook?
5. What is the role of YAML in Ansible playbooks?
6. How do you define tasks in an Ansible playbook?
7. What is an inventory file in Ansible?
8. How does Ansible achieve agentless automation?
9. What is a module in Ansible, and how is it used?
10. How can you run an Ansible playbook on multiple hosts?

Unit-4: Advanced Ansible Concepts

1. What are Ansible roles, and how are they used in complex environments?
2. Explain the concept of Ansible Galaxy.
3. How does Ansible handle variables and facts?

4. What are handlers in Ansible, and how are they different from tasks?
5. How does Ansible Vault ensure secure handling of sensitive data?
6. What are conditionals in Ansible, and when would you use them?
7. How can you implement error handling in Ansible playbooks?
8. Explain the use of loops in Ansible playbooks.
9. What is the significance of Jinja2 templates in Ansible?
10. How do you manage large-scale deployments using Ansible Tower?

Unit-5: Jenkins and CI/CD

1. What is Jenkins, and how does it support CI/CD?
2. How does Jenkins automate the build and deployment process?
3. What are the key components of a Jenkins pipeline?
4. What is a Jenkinsfile, and how is it used?
5. How do you configure Jenkins to trigger a build on a code commit?
6. What role does Jenkins play in continuous integration (CI)?
7. Explain the difference between freestyle projects and pipelines in Jenkins.
8. What plugins are commonly used in Jenkins for CI/CD?
9. How does Jenkins integrate with other DevOps tools like Docker or Git?
10. What are the benefits of using Jenkins for continuous delivery (CD)?

Long Questions

Unit-1: Concepts of DevOps

1. What are the key components of DevOps, and how do they contribute to the integration of Developers and Operations?
2. Explain the purpose of DevOps Architecture and its significance in modern software development.
3. Compare and contrast the workflow of the waterfall method and agile software development. How does DevOps fit into each of these workflows?
4. Define DevOps and Agile and highlight the key differences between these two approaches to software development.
5. Describe the role of Developers and Operations in DevOps and how their integration benefits the development process.
6. How does DevOps promote collaboration between development and operations teams? Provide examples of how this collaboration enhances software development.
7. Explain the concept of Continuous Integration (CI) and Continuous Delivery (CD) in the context of DevOps. Why are they essential in DevOps practices?
8. Discuss the importance of automation in DevOps and how it contributes to improving the software development lifecycle.

Unit-2: DevOps Lifecycle and Automation Tools

1. Describe the key phases of the DevOps lifecycle and the typical workflow involved in DevOps practices.
2. What are the primary goals of DevOps automation tools? Provide examples of popular automation tools and their specific purposes.

3. Explain the significance of Maven in the context of DevOps. How does it contribute to the software development process?
4. Introduce Ansible and its role in automation. How does Ansible simplify the management of IT infrastructure?
5. Compare and contrast the features and purposes of Jira and Splunk in the DevOps toolchain.
6. How can automation tools like Ansible help in the deployment and configuration of software and systems? Provide practical examples.
7. What are the main advantages of using automation tools in DevOps practices, and how do they contribute to efficiency and reliability?
8. Discuss the role of version control systems (e.g., Git) in the DevOps lifecycle and explain how they aid collaboration and code management.

Unit-3: Ansible and Playbooks

1. Provide an overview of Ansible, its working principles, and how it simplifies system administration and automation.
2. Describe the installation process of Ansible and the prerequisites for setting up an Ansible environment.
3. Explain the basics of YAML and its role in Ansible playbooks. Discuss key YAML elements, such as keys, values, lists, and dictionaries.
4. Explain the process of transferring files to remote servers using Ansible and provide examples of quick commands for file transfer.
5. How does Ansible manage packages on remote servers, and what is the role of Ansible modules in package management?
6. Define Ansible playbooks and their significance in automation. Explain the concepts of playbooks, tasks, and hosts in Ansible.
7. Create a simple Ansible playbook to accomplish a specific task (e.g., installing software) and explain the structure of the playbook.
8. Discuss the various tags used in Ansible playbooks (e.g., name, hosts, vars, tasks) and their respective roles in defining playbook behavior.

Unit-4: Advanced Ansible Concepts

1. Explain the process of creating an Ansible role, including the creation of a role directory and its utilization in playbooks.
2. Explain the concept of Ansible variables and how they can be used to customize playbook behavior.
3. How can you handle exceptions and errors in Ansible playbooks, and what are best practices for ensuring robust automation?
4. Describe the different control structures in Ansible playbooks, including blocks, loops, and conditionals. Provide examples of how each is used.
5. Discuss the benefits of breaking down a complex playbook into smaller, reusable roles and how this enhances playbook maintainability.
6. Explain the role of variables and dynamic inventories in customizing Ansible playbooks for various environments and scenarios.
7. Explore the concept of role-based access control (RBAC) in Ansible and how it enhances security and access control.

8. Describe the role of Ansible roles and how it facilitates the sharing and reuse of Ansible roles in the DevOps community.

Unit-5: Jenkins and CI/CD

1. Provide an overview of Jenkins, including its core concepts and architecture. How does Jenkins facilitate continuous integration and continuous delivery?
2. What are the key applications of Jenkins in the software development process, and how does it support various stages of the DevOps lifecycle?
3. Enumerate the essential features of Jenkins and explain how they contribute to building a robust and efficient CI/CD pipeline.
4. Discuss the advantages of using Jenkins as a CI/CD tool and how it helps in automating software delivery processes.
5. Explain the installation process of Jenkins and highlight any prerequisites or considerations for setting up a Jenkins server.
6. Explain the concepts of Continuous Integration (CI) and Continuous Delivery (CD) and their role in automating the software development and deployment process.
7. Outline the key components of a CI/CD pipeline and describe how it streamlines the development, testing, and delivery of software.
8. Provide a step-by-step guide on how to build a basic CI/CD pipeline using Jenkins, including the stages involved and the tools and plugins commonly used.