SECTION A

Very Short Answer Questions

Attempt all seven (7) questions [2 × 7 = 14]

1. How do you define the Career role as Software Quality Engineer?

A Software Quality Engineer is responsible for ensuring that software products meet quality standards by developing and executing test plans, automating tests, identifying bugs, and working closely with development teams to improve product quality.

1. Write down the code to run the postman collection through Command Prompt with Newman along with its environment file and data file.

**newman run collection.json -e environment.json -d data.json**

1. Illustrate the use of ‘Assertions’ in JMeter. List out different assertion elements in JMeter?

**Assertions** in JMeter are used to verify that the server's response to a request meets the expected criteria, such as content, response time, or status.  
Common Assertion Elements:

* **Response Assertion**: Validates the response text.
* **Duration Assertion**: Ensures response time is within limits.
* **Size Assertion**: Checks the size of the response.
* **XML/JSON Assertion**: Validates the structure of XML/JSON responses.

1. What is a wireframe Document? What do we use Wireframes for?

A wireframe document is a visual guide that represents the skeletal framework of a website or application. Wireframes are used to outline the layout, structure, and flow of user interfaces without focusing on design elements like color or graphics.

1. What are some benefits of Software Testing?

**Bug Detection**: Helps identify and fix defects early.

**Quality Assurance**: Ensures the software meets user requirements and expectations, leading to a more reliable product.

1. What do you mean by End-to-End Testing in Quality Assurance?

End-to-End (E2E) testing validates the entire software system's flow, ensuring that all integrated components work together as expected, from start to finish, mimicking real-world scenarios.

1. Distinguish Between Severity and Priority.

 **Severity**: Indicates the impact of a defect on the system's functionality (e.g., critical, major).

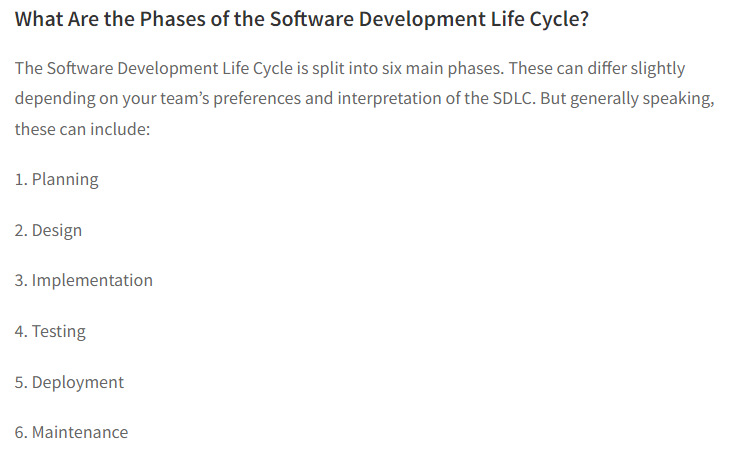
 **Priority**: Refers to the urgency of fixing a defect based on business needs (e.g., high, medium, low).

SECTION B

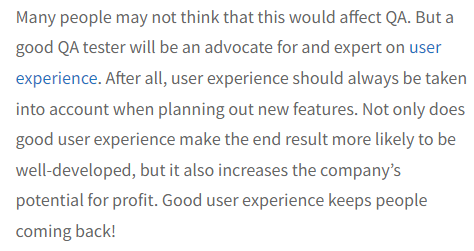
Short Answer Questions

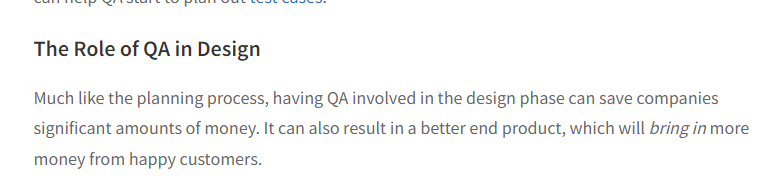
Attempt only seven (7) questions out of nine (9) questions [8 × 7 = 56]

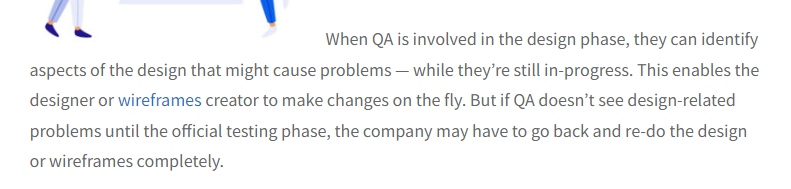
**Q.1 Illustrate with a Diagram and Explain each of the roles of QA in each of the phases of the Software Development Life Cycle (SDLC). [Chapter 1: Introduction to Software Quality Assurance]**













QA could weigh in on the potential impacts on user experience, or suggest using a loading animation to make up for any lag time.

If issues are brought up during daily standups, QA can jump in and help out as needed with the implementation phase.

**The Role of QA in Testing**

The QA team executes the test cases, reports defects, and ensures that the software meets the quality standards. This phase involves rigorous testing including functional, non-functional, and regression testing.

### The Role of QA in Deployment

QA typically needs to be on-call during a deployment. Often these happen late at night, when users aren’t as likely to be affected. During a deployment, QA is on standby. As soon as the release goes live, QA again jumps into gear, [smoke testing](https://mindfulqa.com/what-is-smoke-testing/) to make sure the deploy didn’t cause any issues.

### The Role of QA in Maintenance

Rinse and repeat for above! There are sometimes bugs that slip through — even the best auttomation/manual testing can’t find 100% of bugs on 100% of browsers/devices 100% of the time. QA can be on hand to test the [bug fixes](https://mindfulqa.com/prioritize-bugs/), or upgrades to features.

**Q.2 What are the Types of Testing? List and explain at least 4 types for functional and non-functional types of testing each. [Chapter 3: Types of Testing in Quality Assurance]**

**Types of Testing:**

**Functional Testing:**

1. **Unit Testing:**
   * **Description:** Involves testing individual components or modules of the software to ensure that they work correctly. It is usually done by developers during the coding phase.
   * **Purpose:** To verify that each unit of the software performs as expected.
2. **Integration Testing:**
   * **Description:** Focuses on testing the interfaces between modules or components to ensure they work together as intended. It can be performed incrementally or as a "big bang."
   * **Purpose:** To ensure that the integrated components work as expected when combined.
3. **System Testing:**
   * **Description:** Tests the complete and integrated software application to validate that it meets the specified requirements. It includes both functional and non-functional testing.
   * **Purpose:** To verify the overall functionality and behavior of the system as a whole.
4. **User Acceptance Testing (UAT):**
   * **Description:** Conducted by the end-users to ensure that the software meets their needs and requirements. It is the final testing phase before the software is released to production.
   * **Purpose:** To validate the software against business requirements and user expectations.

**Non-Functional Testing:**

1. **Performance Testing:**
   * **Description:** Evaluates the speed, responsiveness, and stability of the software under different loads. It includes load testing, stress testing, and endurance testing.
   * **Purpose:** To ensure that the software performs well under expected and peak load conditions.
2. **Security Testing:**
   * **Description:** Focuses on identifying vulnerabilities, threats, and risks in the software. It includes penetration testing, vulnerability scanning, and risk assessment.
   * **Purpose:** To ensure that the software is secure from external and internal threats.
3. **Usability Testing:**
   * **Description:** Assesses the software’s ease of use and user-friendliness. It involves testing the UI, user flows, and overall user experience.
   * **Purpose:** To ensure that the software is intuitive and easy for end-users to navigate and use.
4. **Compatibility Testing:**
   * **Description:** Verifies that the software works correctly across different environments, including various browsers, operating systems, and devices.
   * **Purpose:** To ensure that the software is compatible with different environments and user configurations.

**Q.3 What is a Gantt chart Document? What are some Pros and Cons of it? Illustrate Gantt Chart Document with a suitable Diagram. [Chapter 2: QA Documentation Practices and Types]**

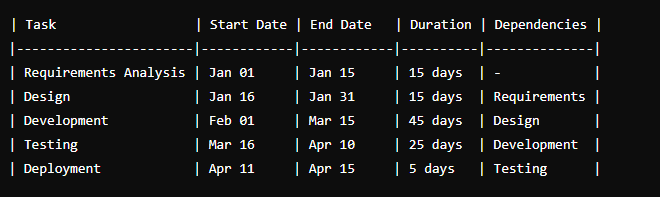
A **Gantt chart** is a project management tool that visually represents the schedule of a project. It displays tasks, their durations, start and end dates, and dependencies between tasks. Each task is represented by a horizontal bar, with the length of the bar corresponding to the task duration.

**Pros of Gantt Charts:**

* **Visual Clarity:** Provides a clear visual representation of the project schedule and timeline.
* **Task Management:** Helps in tracking task progress and managing task dependencies.
* **Resource Allocation:** Facilitates the allocation and monitoring of resources across tasks.
* **Progress Tracking:** Allows easy monitoring of the project's progress against the planned schedule.

**Cons of Gantt Charts:**

* **Complexity:** Can become complicated and difficult to manage for large projects with many tasks.
* **Static Representation:** Gantt charts are static and may require frequent updates to reflect changes in the project schedule.
* **Time-Consuming:** Creating and maintaining a Gantt chart can be time-consuming, especially for complex projects.



|--- Requirements Analysis ---|

|--- Design ---|

|--- Development ---|

|--- Testing ---|

|--- Deployment ---|

**Q.4 When do we use global variables, collection variables, and local variables? [Chapter 7: API Automation using Postman]**

**Global Variables:**

* **Usage:** Global variables are used when you need to use the same data across multiple collections, requests, or environments within Postman. They are accessible throughout the entire workspace and can be used across different collections and environments.
* **Example:** Base URLs, API keys, or environment-specific configurations that are consistent across multiple collections.

**Collection Variables:**

* **Usage:** Collection variables are scoped to a specific collection and are used within that collection’s requests and tests. These variables are particularly useful when the same value is needed across multiple requests within a single collection.
* **Example:** Authentication tokens, specific endpoints, or any data shared within requests of the same collection.

**Local Variables:**

* **Usage:** Local variables are used within a specific request or test script. They are limited to the scope of that request or script and are not accessible outside of it. These variables are useful for temporary data storage or when passing data between requests.
* **Example:** Temporary values like dynamic request payloads or response data that is only needed within a single request.

**Q.5 Describe the Different Forms of Software Quality Assurance Documentation. Explain**

**why each of them are essential. [Chapter 2: QA Documentation Practices and Type]**

 **Test Plan Document:**

A comprehensive document outlining the strategy, approach, resources, and schedule for testing activities. It includes details on the scope of testing, objectives, deliverables, timelines, and responsibilities.

Provides a roadmap for the testing process, ensuring that all aspects of the software are tested systematically and within the project timeline.

 **Test Cases Document:**

A set of conditions or variables used to determine whether a software application is working correctly. Test cases include input data, execution steps, and expected results.

Ensures that each function of the software is tested with a clear set of steps, helping to identify defects and verify that the software meets requirements.

 **Defect Report:**

A document that records any issues or bugs found during testing. It includes information on the defect, such as steps to reproduce, severity, priority, and status.

Facilitates communication between testers and developers, ensuring that issues are documented, tracked, and resolved.

**Traceability Matrix:**

A document that maps and traces user requirements with test cases. It ensures that all requirements are covered by test cases and helps in tracking the testing progress.

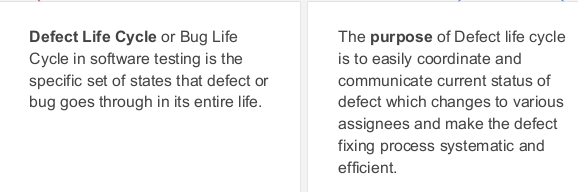
Ensures comprehensive testing by confirming that all requirements are tested, helping to avoid missed test scenarios.

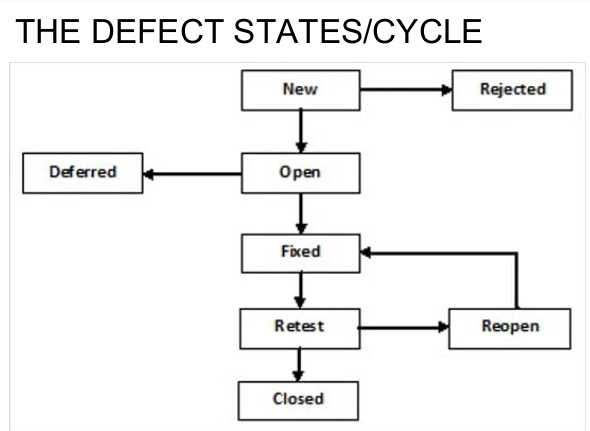
 **Test Summary Report:**

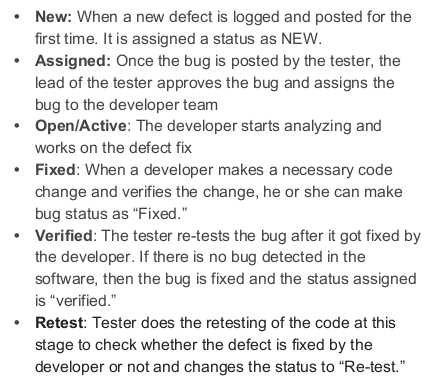
A document that provides an overview of the testing activities, including what was tested, the number of defects found, the number of tests passed/failed, and overall test coverage.

Offers a high-level view of the testing process and outcomes, helping stakeholders make informed decisions about the software’s release readiness.

**Q.6 What is a Defective Life Cycle? Illustrate and explain some of the Defect states of Defect Life Cycle. [Chapter 4: Defect Reporting and Defect Tracking]**







**Q.7 What Tracing Document do we use to make sure all the requirements have been covered? Explain some Types of Traceability Matrix form. What are some advantages of the document? [Chapter 5: Test plan Document and Test Cases]**

* **Document Used:** The **Traceability Matrix** is used to ensure that all requirements have been covered by corresponding test cases. It maps requirements to test cases, ensuring that every requirement is validated during testing.

**Types of Traceability Matrix Forms:**

1. **Forward Traceability Matrix:**
   * **Description:** Maps requirements to test cases. Ensures that all requirements are tested by at least one test case.
   * **Usage:** Verifies that each requirement has been implemented and validated.
2. **Backward Traceability Matrix:**
   * **Description:** Maps test cases back to the requirements. Ensures that each test case is linked to a specific requirement.
   * **Usage:** Helps identify any test cases that do not trace back to a requirement, reducing unnecessary tests.
3. **Bi-Directional Traceability Matrix:**
   * **Description:** Combines both forward and backward traceability, mapping requirements to test cases and vice versa. Provides a comprehensive view of traceability.
   * **Usage:** Ensures complete coverage and identifies missing requirements or test cases.

**Advantages of Traceability Matrix:**

* **Ensures Full Coverage:** Guarantees that all requirements are tested and nothing is missed.
* **Improves Test Case Organization:** Helps organize test cases systematically by linking them to requirements.
* **Facilitates Change Management:** Easier to manage changes in requirements by identifying the impacted test cases.
* **Enhances Communication:** Provides clear documentation that can be used to communicate testing status to stakeholders.

**Q.8 What is an API? List and explain some of the types of API testing. [Chapter 6: API Testing]**

* An API is a set of rules and protocols that allow different software applications to communicate with each other. It defines the methods and data formats that applications can use to interact with each other.

**Types of API Testing:**

1. **Functional Testing:**
   * **Description:** Validates that the API functions as expected and produces the correct responses for various inputs. It checks the correctness of the API's endpoints.
   * **Purpose:** Ensures that the API meets the specified functional requirements.
2. **Load Testing:**
   * **Description:** Evaluates the API’s performance under various load conditions. It tests the API’s ability to handle multiple requests simultaneously.
   * **Purpose:** Ensures the API can handle high traffic and performs well under load.
3. **Security Testing:**
   * **Description:** Examines the API for vulnerabilities, data security, and authentication mechanisms. It tests how the API handles unauthorized access and data breaches.
   * **Purpose:** Ensures the API is secure from attacks and unauthorized access.
4. **Compatibility Testing:**
   * **Description:** Verifies that the API works correctly across different environments, such as various browsers, operating systems, and devices.
   * **Purpose:** Ensures that the API is compatible with different client environments and configurations.
5. **Regression Testing:**
   * **Description:** Re-executes previously conducted tests to ensure that the API’s functionality remains intact after changes or updates to the code.
   * **Purpose:** Confirms that new changes have not introduced any new bugs.

**Q.9 What is Postman? What are some of the common HTTP methods that we used in Postman to send requests. Explain each of them briefly [Chapter 7: API Automation using Postman]**

Postman is a popular API testing tool that allows developers to design, test, and document APIs. It provides a user-friendly interface to create and execute HTTP requests, validate responses, and automate API testing.

**Common HTTP Methods in Postman:**

1. **GET:**
   * **Description:** Retrieves data from the server. It is used to request data from a specific resource.
   * **Usage:** Fetches data such as user details, product lists, or any other information without modifying the resource.
2. **POST:**
   * **Description:** Sends data to the server to create a new resource. It is often used to submit forms or upload files.
   * **Usage:** Creates a new entry in a database, such as creating a new user or posting a new blog article.
3. **PUT:**
   * **Description:** Updates an existing resource on the server. It is used to modify an existing resource with new data.
   * **Usage:** Updates user information, such as changing a user’s email address or updating a product’s price.
4. **DELETE:**
   * **Description:** Deletes a specified resource from the server. It is used to remove data.
   * **Usage:** Deletes a user account, removes a product from a catalog, or clears a specific entry from a database.
5. **PATCH:**
   * **Description:** Partially updates an existing resource. It is similar to PUT but only modifies specific fields of the resource.
   * **Usage:** Updates a single field, like changing just the email address of a user without altering other fields.

**Group C**

**Long Analytical or Case Question**

**Attempt any two (2) questions out of three (3) questions [15 × 2 = 30]**

**Q.1 What are Software Development methodologies? List any Five models of SDLC. Explain Agile Methodology and Waterfall Methodology in detail with a Diagram. [Chapter 2: QA Documentation Practices and Types] (5 + 10)**

Software development methodologies are structured approaches to software development that guide the process from conception through deployment and maintenance. These methodologies define the processes, tasks, roles, and workflows involved in developing software systems.

**Five Models of SDLC:**

1. **Waterfall Model:**
   * A linear and sequential approach where each phase must be completed before the next begins. It’s often used for projects with well-defined requirements.
2. **Agile Model:**
   * An iterative and incremental approach that emphasizes flexibility, customer feedback, and rapid delivery of small, functional segments of the software.
3. **Spiral Model:**
   * Combines elements of both iterative and waterfall models, with a focus on risk assessment. The development process passes through four phases repeatedly: Planning, Risk Analysis, Engineering, and Evaluation.
4. **V-Model (Validation and Verification):**
   * An extension of the waterfall model, where testing activities are planned parallel to corresponding development activities. Each development phase is directly associated with a testing phase.
5. **DevOps Model:**
   * Integrates development and operations teams to improve collaboration and automate deployment processes. It emphasizes continuous integration, continuous delivery, and continuous deployment.

**Agile Methodology:**

**Overview:**

* Agile methodology is an iterative and incremental approach to software development. It focuses on delivering small, functional parts of the software (called "increments") frequently, usually in cycles called "sprints" that last 1-4 weeks. Agile encourages active collaboration between the development team and stakeholders, allowing for continuous feedback and adaptation.

**Key Principles:**

* **Customer Collaboration:** Continuous interaction with customers to gather feedback and make necessary adjustments.
* **Responding to Change:** Flexibility to change requirements even in late development stages.
* **Frequent Delivery:** Regular delivery of functional software to ensure progress and quality.
* **Individuals and Interactions:** Emphasis on team collaboration and communication over processes and tools.

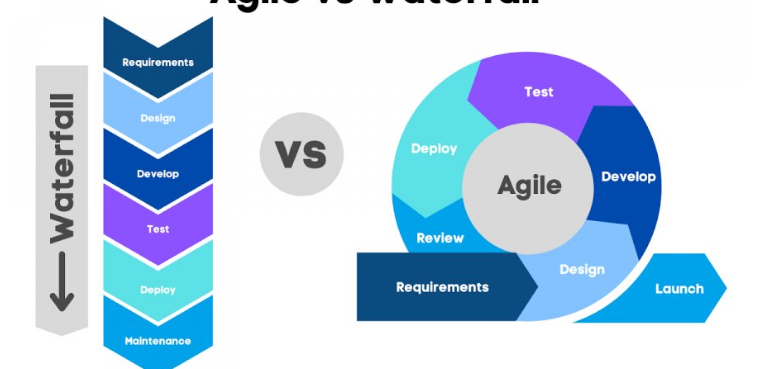
**Waterfall Methodology:**

**Overview:**

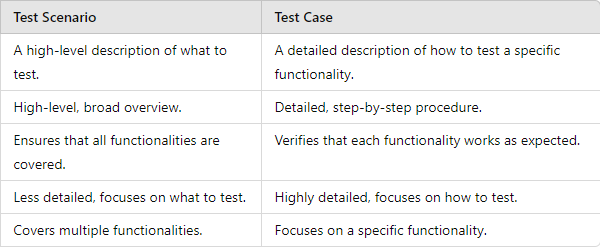
* The Waterfall methodology is a traditional and linear approach to software development. It divides the software development process into distinct phases, each of which must be completed before the next phase begins. This model works best when requirements are well-understood and unlikely to change.

**Phases:**

1. **Requirements Gathering:** Detailed documentation of all the software requirements.
2. **System Design:** Planning the software architecture based on requirements.
3. **Implementation (Coding):** Writing the actual code based on the design.
4. **Integration and Testing:** Integrating the different modules and testing the entire system.
5. **Deployment:** Releasing the software to the production environment.
6. **Maintenance:** Ongoing support and fixing issues as they arise.

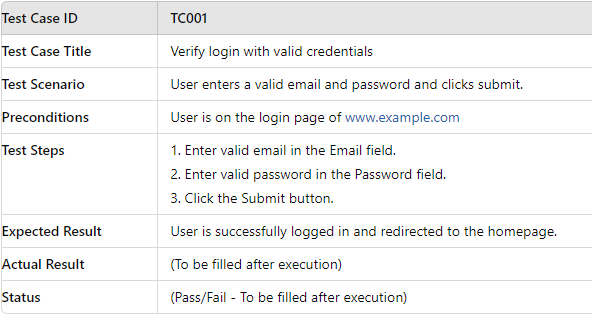


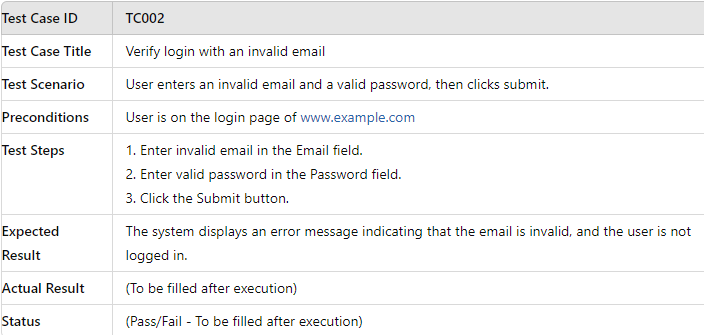
**Q.2 List out key differences between Test Scenario and Test Case? List out any 5 Test cases for a Login Scenario of the system that has Email, Password Field and Submit Button on its UI. And present it on a proper Test Case Template with proper for any two of those Test Cases. You can assume the site name as &quot;www.example.com&quot; to mention on your test case document. [Chapter 5: Test plan Document and Test Cases] (4+5+6)**



**Five Test Cases for a Login Scenario:**

1. **Test Case 1:** Verify that the user can log in with a valid email and password.
2. **Test Case 2:** Verify that the user cannot log in with an invalid email.
3. **Test Case 3:** Verify that the user cannot log in with an invalid password.
4. **Test Case 4:** Verify that the login button is disabled when the email or password fields are empty.
5. **Test Case 5:** Verify that the system displays an error message when incorrect login credentials are entered.





**Q.3 Explain the architecture of Apache JMeter. How does it facilitate performance testing? Discuss the components such as Samplers, listeners, Config Elements and Processors. [Chapter 8: Performance Testing with JMeter] (2+2+11)**

Apache JMeter is a popular open-source tool used for performance testing and load testing of web applications. It is designed to simulate a heavy load on a server, group of servers, network, or object to test its strength and analyze overall performance.

**Architecture Components:**

1. **Test Plan:**
   * **Description:** The Test Plan is the container for the various elements of the JMeter test. It defines what to test, how to test, and the criteria for success.
   * **Purpose:** It organizes and configures all the testing components, setting the foundation for executing the performance test.
2. **Thread Group:**
   * **Description:** Thread Groups define the number of users (threads) to simulate, the ramp-up period (time to start all threads), and the loop count (number of times the test will be executed).
   * **Purpose:** Simulates multiple users performing the same actions on the system under test.
3. **Samplers:**
   * **Description:** Samplers define the type of request to be sent to the server, such as HTTP requests, FTP requests, or JDBC requests.
   * **Purpose:** Generates the actual requests to the server and measures the response times. Examples include HTTP Sampler, FTP Sampler, JDBC Request.
4. **Listeners:**
   * **Description:** Listeners are used to collect and display the results of the test execution. They can output the results to various formats like tables, graphs, or logs.
   * **Purpose:** Provide visualization of the performance test results and help in analyzing the performance metrics. Examples include View Results Tree, Graph Results, and Summary Report.
5. **Config Elements:**
   * **Description:** Config Elements are used to set up defaults and variables for Samplers. They provide configuration information that Samplers and other test elements use.
   * **Purpose:** Facilitate the configuration of testing conditions such as HTTP request defaults, CSV Data Set Config, and User Defined Variables.
6. **Processors:**
   * **Description:** Processors are used to modify the Samplers' requests or responses. There are two types: Pre-Processors (executed before the sampler) and Post-Processors (executed after the sampler).
   * **Purpose:** Allow manipulation of the test elements before or after the request is sent, such as altering parameters dynamically or extracting data from responses. Examples include Regular Expression Extractor and BeanShell Preprocessor.
7. **Timers:**
   * **Description:** Timers introduce delays between requests to simulate realistic user behavior.
   * **Purpose:** Helps in spacing out requests to better simulate real-world traffic patterns, such as Constant Timer or Gaussian Random Timer.
8. **Assertions:**
   * **Description:** Assertions are used to validate that the response received from the server matches the expected result.
   * **Purpose:** Ensures that the server response is correct and meets specified criteria, such as Response Assertion and Duration Assertion.

**How JMeter Facilitates Performance Testing:**

* **Load Simulation:** JMeter can simulate multiple users with configurable ramp-up times and loops, creating realistic load conditions on the system under test.
* **Distributed Testing:** JMeter supports distributed testing, allowing the load to be spread across multiple machines to test large-scale systems.
* **Extensive Protocol Support:** JMeter supports various protocols like HTTP, HTTPS, FTP, JDBC, SOAP, and more, making it versatile for different types of performance tests.
* **Real-Time Monitoring:** With listeners and graphical reports, JMeter provides real-time feedback on the performance metrics, helping to identify bottlenecks quickly.
* **Customizability:** Through the use of processors, config elements, and scripting (e.g., BeanShell, Groovy), JMeter offers a high degree of customization, allowing complex scenarios to be tested.