1. What are the types of Testing?

There are various types of testing in software development, each serving a specific purpose to ensure the quality and reliability of software. Here are some common types of testing with examples.

Unit Testing

To test individual components or functions in isolation.

Example: Testing a function that calculates the square of a number, ensuring it returns the correct result.

Integration Testing

To test the interactions between different components or modules.

Example: Testing how different software modules communicate when processing a user login request.

Functional Testing

To verify that the software meets its functional requirements.

Example: Testing an e-commerce website to ensure that users can add items to their shopping cart and proceed to checkout.

Regression Testing:

To ensure that recent code changes do not negatively impact existing functionality.

Example: After fixing a bug, running a set of tests to confirm that the bug is resolved without introducing new issues.

Performance Testing:

To evaluate how a system performs under various conditions, such as load, stress, or scalability.

Example: Testing a web application to determine how it handles concurrent user requests or simulating high traffic.

Security Testing:

To identify vulnerabilities and security issues in the software.

Example: Conducting penetration testing to discover and address potential weaknesses in a web application's security.

Usability Testing:

To assess the user-friendliness of the software and its overall user experience.

Example: Observing and collecting feedback from users as they perform specific tasks on a mobile app to improve its usability.

Compatibility Testing:

To ensure that the software works correctly on different devices, browsers, and operating systems.

Example: Testing a website on various browsers (e.g., Chrome, Firefox, Safari) to verify that it displays and functions consistently.

Load Testing:

To evaluate how a system performs under expected load conditions.

Example: Simulating a large number of concurrent users on a web server to assess its response time and resource utilization.

Acceptance Testing:

To determine whether the software meets the acceptance criteria set by stakeholders.

Example: End-users testing a mobile app to ensure it meets their business requirements before deployment.

Alpha Testing:

To test the software in a controlled environment before releasing it to a broader audience.

Example: In-house testing of a new video game by the development team before it is sent to beta testers.

Beta Testing:

To gather feedback from a selected group of external users before a full release.

Example: Providing a pre-release version of a social media app to a group of volunteers for testing and feedback.

**2) What are the phases of STLC?**

Requirement Analysis, Test Planning, Test Case Development, Test Environment setup, Test execution, Test cycle Closure.

1. **What are the qualities you possess as a Manual tester?**

As a manual tester, there are several qualities that I possess which contribute to my effectiveness and success in my role. Here are a few examples:

1. Attention to Detail

2. Communication Skills

3. Adaptability

4. Problem-Solving Skills

**4.) What is the difference between Agile model and Waterfall model?**

Agile and Waterfall are two distinct project management methodologies used in software development and other industries. They have fundamental differences in their approach to project planning, execution, and adaptability. Here are the key differences between Agile and Waterfall methodologies:

Project Approach:

Agile: Agile is an iterative and incremental approach. Projects are divided into small, manageable increments called iterations or sprints. Each iteration typically lasts 2-4 weeks and results in a potentially shippable product increment. Agile allows for flexibility and frequent reassessment and adaptation of project goals based on feedback.

Waterfall: Waterfall is a linear and sequential approach. It follows a predefined sequence of phases, including requirements, design, implementation, testing, deployment, and maintenance. Each phase must be completed before moving on to the next. Waterfall is less flexible, and changes are often costly and time-consuming.

Requirements and Scope:

Agile: Agile projects often begin with a high-level vision or set of requirements but allow for flexibility in detailing requirements as the project progresses. Requirements can evolve and change based on user feedback and evolving priorities.

Waterfall: Waterfall requires detailed upfront documentation of all project requirements. Changes to requirements after the project has started are discouraged and can be expensive to implement.

Customer Involvement:

Agile: Agile encourages close collaboration with customers and stakeholders throughout the project. Frequent feedback from customers is used to guide the project's direction.

Waterfall: Customer involvement tends to be more limited in Waterfall, often occurring primarily at the beginning and end of the project.