Assignment

**1.What is Exploratory Testing?**

Ans: Exploratory testing is a **hands-on, unscripted approach to software testing** where testers actively explore an application to identify defects, understand its functionality, and evaluate its behavior. Unlike scripted testing, which relies on pre-written test cases, exploratory testing emphasizes flexibility, creativity, and tester intuition.

**2.What is a traceability matrix?**

Ans: It provides visibility into where something came from, where it has been, and what has been done to it. Traceability is critical in various industries, such as manufacturing, supply chain management, food safety, software development, and pharmaceuticals, to ensure quality, compliance, and accountability.

### **Types of Traceability**

**Upstream (Backward) Traceability**:

**Downstream (Forward) Traceability**:

**End-to-End Traceability**

**3.What is boundary value testing?**

Ans: **Boundary Value Testing (BVT)** is a software testing technique used to identify defects at the boundaries of input domains rather than within the center of the input range. It is based on the idea that errors are more likely to occur at the edges of these boundaries, making them critical areas to test.

Concepts of boundary value testing:

Boundary values

Focus areas

Test cases

**4. What is Equivalence partitioning testing?**

Equivalence Partitioning (EP) is a **software testing technique** used to divide input data of a software application into distinct **equivalence classes**. The goal is to reduce the number of test cases while still achieving good test coverage by identifying test cases that are expected to produce similar results.

**5. What is integration testing?**

**Integration testing** is a level of software testing where individual modules or components of an application are combined and tested as a group to verify their interaction and data flow. The primary goal is to ensure that these integrated components work together correctly and that there are no issues in their interfaces.

**6. What determines the level of risk?**

The level of risk in any context, such as software development, business operations, or safety-critical systems, is determined by two primary factors:

The level of risk in any context, such as software development, business operations, or safety-critical systems, is determined by two primary factors:

### **1. Likelihood (Probability)**

* **Definition**: The probability that a specific event, issue, or defect will occur.

### **2. Impact (Severity)**

* **Definition**: The extent of damage or loss that would result if the risk materializes.

By systematically assessing **likelihood** and **impact**, organizations can prioritize and manage risks effectively.

**7. What is Alpha testing?**

**Alpha Testing** is a type of software testing performed during the development phase to identify bugs, ensure functionality, and evaluate the usability of a software application before it is released to external users. It is usually conducted by **internal testers**, such as the development or quality assurance (QA) team, and sometimes includes a small group of potential end-users in a controlled environment.

**8. What is Beta testing?**

**Beta Testing** is the second phase of user validation testing in the software development lifecycle. It involves releasing a nearly complete version of the software to a **limited number of real users** outside the development team to test the application in a **real-world environment**. The goal is to identify issues that may not have been discovered during earlier testing phases and gather user feedback for final improvements.

**9. What is component testing?**

**Component Testing** is a type of software testing that focuses on verifying the functionality, reliability, and performance of individual components or modules of a system in isolation. This ensures that each component works as expected before it is integrated with other components.

### **Advantages of Component Testing:**

1. **Early Bug Detection**: Identifies issues at the module level before integration.
2. **Improves Code Quality**: Ensures each component meets requirements independently.
3. **Simplifies Debugging**: Since components are tested in isolation, issues are easier to locate and fix.

**10. What is functional system testing?**

**Functional System Testing** is a type of software testing that focuses on validating the functionality of a complete system or application against its functional requirements. It ensures that the system performs as intended and delivers the expected outcomes for various user inputs, scenarios, and interactions.

### **Example of Functional System Testing:**

Consider an **online shopping application**:

* **Functional Requirements**:
  + Users can search for products.
  + Users can add items to a cart.
  + Users can make payments using credit cards or digital wallets.

**11. What is non functional testing?**

**on-Functional Testing** refers to testing the aspects of a software application that do not directly relate to specific behaviors or functions but instead focus on how the system performs under various conditions. It evaluates the non-functional attributes of the system, such as **performance**, **usability**, **security**, and **scalability**, among others. Non-functional testing ensures that the system meets the broader quality attributes required for successful deployment and user satisfaction.

**12. What is GUI testing?**

**GUI Testing** (Graphical User Interface Testing) is the process of evaluating and verifying the functionality, usability, and appearance of a software application’s graphical user interface. The goal is to ensure that the interface behaves as expected and provides a smooth, intuitive user experience, with proper alignment, color schemes, fonts, buttons, and interactive elements.

**13. What is Adhoc testing?**

**Adhoc Testing** is an informal and unstructured type of software testing where the tester explores the application without any predefined test cases or plans. The goal of adhoc testing is to identify defects or issues in the system by randomly testing different parts of the application. It relies heavily on the tester's experience, intuition, and understanding of the application, rather than following a detailed testing process.

**14. What is load testing?**

The goal of load testing is to determine how the application behaves when it is subjected to a normal or expected number of concurrent users or requests, typically under standard or typical usage conditions. It helps assess the system's capacity to handle the expected load and identify potential performance issues before the application goes live.

**15. What is white box testing and list the types of white box testing?**

**White Box Testing** is a type of software testing where the internal workings or structure of the application are tested. In this approach, the tester has full visibility into the internal code, architecture, and logic of the system.

Types of White box testing:

Unit testing

Integration testing

**16. What is black box testing? What are the different black box testing techniques?**

**Black Box Testing** is a software testing technique in which the tester focuses on testing the functionality of the application without knowledge of its internal code or structure.

The different black box testing techniques:

**Equivalence Partitioning**

**Boundary Value Analysis**

**Decision Table Testing**

**State Transition Testing**

**Use Case Testing**

**Exploratory Testing**

**Random Testing**

**17. Mension: what are the categories of defect?**

**Defects in a product, system, or process are typically categorized to help identify, analyze, and address issues effectively. The categories of defects can vary depending on the context (e.g., software development, manufacturing, construction, etc.). Here are some general categories across various industries:**

### **1. Critical Defects**

* **These defects pose a serious safety risk or cause the system to fail entirely.**
* **Examples: Hardware failure, data loss, or a system crash.**

### **\*\*2 Conclusion-specific**

**Test environment ​**

**18. Mention what bigbang testing is?**

**Big Bang Testing is a software testing approach where all components or modules of a system are integrated simultaneously after completing individual development. Once integrated, the entire system is tested as a whole to evaluate its functionality.**

### **When to Use:**

* **Big Bang Testing is most suitable for small systems with fewer components or when development teams are confident that individual modules are defect-free.**

### **19. What is the purpose of exit criteria?**

### **Purposes of Exit Criteria:**

1. **Ensure Quality Standards Are Met:**
   * Ensures the deliverables (e.g., software, product, or documentation) meet predefined quality requirements.
   * Helps maintain consistency and reliability across processes.
2. **Avoid Incomplete Work:**
   * Prevents moving to the next phase or delivering a product that is incomplete or defective.
3. **Facilitate Risk Management:**
   * Reduces risks by verifying that all required testing or validation steps have been completed before deployment or release.
4. **Provide Accountability:**
   * Establishes clear goals and accountability for teams to ensure their work aligns with project objectives.
5. **Support Decision-Making:**
   * Helps stakeholders determine if it is appropriate to proceed with the next phase, release, or delivery.
6. **Compliance with Standards:**
   * Ensures compliance with organizational, regulatory, or contractual standards.

**20. When should “Regression Testing” be performed?**

### **Situations When Regression Testing Should Be Performed:**

1. **After Bug Fixes:**
   * To ensure that the resolved issue has not introduced new bugs or broken other features in the system.
2. **After New Features Are Added:**
   * To verify that integrating new features has not affected the existing functionality.
3. **After Enhancements or Updates:**
   * When enhancements or performance improvements are made, regression testing ensures that these changes do not have unintended side effects.
4. **During Code Refactoring:**
   * To validate that restructuring the code for readability, maintainability, or optimization has not altered its behavior.
5. **After Integration of Modules:**
   * When individual modules are integrated into the overall system, regression testing ensures the system works cohesively.
6. **During Environment Changes:**
   * If there are changes in the environment, such as an OS upgrade, database migration, or web server update, regression testing ensures the application remains stable.
7. **Before Major Releases or Updates:**
   * Comprehensive regression testing is essential before releasing a product to production to ensure reliability.
8. **Periodic Maintenance Cycles:**
   * In ongoing projects, regression testing can be part of regular maintenance to ensure system stability over time.

**21. What is 7 key performance? Explain in detail?**

These are the following 7 key performence:

### **1. Customer Satisfaction (CSAT):**

* **Description:** Measures how satisfied customers are with a product, service, or interaction.
* **How It's Measured:**
  + Customer surveys.
  + Net Promoter Score (NPS): Evaluates how likely a customer is to recommend the product.
* **Purpose:** To ensure that the product or service meets customer expectations and fosters loyalty.

### **2. Quality of Deliverables:**

* **Description:** Assesses whether the deliverables meet defined quality standards and specifications.
* **How It's Measured:**
  + Number of defects found (pre-release and post-release).
  + Compliance with standards or acceptance criteria.
* **Purpose:** To maintain product or service excellence and reduce rework.

### **3. Productivity:**

* **Description:** Measures how efficiently resources (e.g., team members, tools) are utilized to produce output.
* **How It's Measured:**
  + Work completed per unit of time (e.g., lines of code, tickets resolved).
  + Resource utilization rates.
* **Purpose:** To optimize team performance and resource allocation.

### **4. Timeliness (Adherence to Deadlines):**

* **Description:** Tracks whether tasks, milestones, or projects are completed on time.
* **How It's Measured:**
  + Percentage of projects completed within the planned schedule.
  + Average delay or deviation from estimated timelines.
* **Purpose:** To ensure smooth project execution and meet stakeholder expectations.

### **5. Financial Performance (Cost Control):**

* **Description:** Measures how well the project or organization manages its budget and expenses.
* **How It's Measured:**
  + Budget variance: Difference between planned and actual spending.
  + Return on Investment (ROI).
* **Purpose:** To maintain profitability and ensure efficient use of financial resources.

### **6. Employee Engagement and Retention:**

* **Description:** Assesses the motivation, satisfaction, and retention rate of employees involved in a project or organization.
* **How It's Measured:**
  + Employee surveys (e.g., eNPS - Employee Net Promoter Score).
  + Turnover rate and absenteeism.
* **Purpose:** To foster a positive work environment and improve team stability.

### **7. Risk Management:**

* **Description:** Evaluates how effectively risks are identified, assessed, and mitigated.
* **How It's Measured:**
  + Number of identified vs. mitigated risks.
  + Severity and impact of unaddressed risks.
* **Purpose:** To minimize project disruptions and ensure smooth execution.

**22. Difference between QA v/s QC v/s Tester**

### **1. Quality Assurance (QA):**

* QA is a proactive process focused on ensuring that the processes used to manage and develop software are effective and aligned with quality standards. It aims to prevent defects by improving processes.

### **2. Quality Control (QC):**

* QC is a reactive process that involves identifying and fixing defects in the product by evaluating the final output against the requirements.

The terms **Quality Assurance (QA)**, **Quality Control (QC)**, and **Tester** are related to the software development lifecycle but serve distinct roles and purposes. Here's a detailed comparison:

### **3. Tester:**

* A **Tester** is an individual who performs the actual testing of the software to find defects and ensure it meets the required standards.

**23. Difference between Smoke and Sanity**

Smoke testing:

Smoke Testing is a **broad and shallow testing approach** used to verify whether the critical functionalities of the software are working as expected after a new build is received. It is often referred to as a "build verification test."

Sanity testing:

Sanity Testing is a **narrow and deep testing approach** performed to verify that specific functionalities or changes introduced in the application are working correctly. It is a subset of regression testing.

**24. Difference between verification and validation**

### **1. Verification:**

* Verification is the process of evaluating whether a product, system, or component meets the specified requirements and is being developed correctly. It ensures that the software aligns with its design and specifications.

### **2. Validation:**

* Validation is the process of evaluating whether the final product meets the intended use and satisfies the business or user requirements. It ensures the software performs as expected.

**25. Explain types of Performance testing?**

Here are the types of performance testing and their details:

### **1. Load Testing**

* Tests the application's performance under expected user loads to determine how it behaves during normal and peak conditions.

### **2. Stress Testing**

* Tests the application under extreme workloads to determine its breaking point and identify its ability to recover gracefully.
* **Purpose:**

### **3. Endurance Testing (Soak Testing)**

* Tests the application's performance over an extended period to detect issues like memory leaks or resource depletion.

### **4. Spike Testing**

* Tests the system's performance under sudden and extreme spikes in user load.

### **5. Scalability Testing**

* Evaluates the system's ability to scale up or down in terms of performance, capacity, and resource usage.

### **6. Volume Testing**

* Also known as **flood testing**, it involves testing the system with a large volume of data to evaluate its capacity and performance.

### **7. Configuration Testing**

* Assesses how the system performs under various hardware or software configurations (e.g., different browsers, operating systems, or network speeds).

**26. What is Error, Defect, Bug and failure?**

### **1. Error**

* An error is a **human mistake** made during the process of coding, designing, or understanding the requirements of a software application.  
  It occurs at the developer or designer level.
* **Example:**
  + A developer uses the wrong variable in a calculation.

### **2. Defect**

* A defect is a **flaw or issue in the software** caused by an error. It represents a deviation from the expected requirements or specifications.
* **Example:**
  + A page that doesn't load properly because of incorrect implementation.

### **3. Bug**

A **bug** is a commonly used term for a defect, especially when it is identified during testing or after the software is deployed.  
It highlights an issue that causes the software to behave unexpectedly or incorrectly.

* **Example:**
  + A mobile app freezes when scrolling quickly through a list.

### **4. Failure**

* A failure occurs when the software **does not perform as expected** in a production or testing environment, resulting in an observable malfunction.
* **Example:**
  + An e-commerce website crashes during a flash sale due to high traffic.

**27. Difference between Priority and Severity**

### **1. Severity**

Severity refers to the **impact** or **seriousness** of a defect on the functionality of the software. It describes how significantly the defect affects the system's operation or the user's ability to use the software.

### **2. Priority**

Priority refers to how **soon** a defect should be **fixed**. It is determined by the business or product needs and indicates the urgency of addressing the defect based on its impact on the project, users, or stakeholders.

28. What is the Bug Life Cycle?

The **bug life cycle** (also known as the **defect life cycle**) refers to the stages that a bug or defect goes through from its identification to its resolution and closure. This life cycle helps to track and manage defects effectively, ensuring quality in software development.

29. Explain the difference bitween Functional testing and Non Functional testing

### **1. Functional Testing:**

* Focuses on verifying that the software behaves as expected and meets the functional requirements defined in the specification.
* Deals with the functionality of the software, such as input/output, data processing, and user interactions.

### **2. Non-Functional Testing:**

* Focuses on evaluating the non-functional aspects of the software, such as performance, usability, security, and scalability.
* Deals with the non-functional qualities of the system, such as efficiency, usability, reliability, and security.

\* What is the difference between the STLC and SDLC?

**STLC (Software Testing Life Cycle):**

* Focuses on the process of testing a software application to ensure its quality, functionality, and performance meet the requirements.
* The purpose is to identify defects in the software and ensure it functions as expected.

**SDLC (Software Development Life Cycle):**

* Covers the complete process of software development, from initial planning to design, development, testing, deployment, and maintenance.
* The purpose is to build the software product, ensuring it meets customer needs and business goals.

\*What is the difference between test scenarios, test case, and test script?

### **1. Test Scenario:**

* A test scenario is a high-level description of a functionality or feature of the software that needs to be tested. It outlines what will be tested but does not go into detailed steps.
* To identify and cover a broad aspect of the system, helping testers understand what areas should be verified.
* Test scenarios are used to ensure the system's critical functionalities are tested and to guide the creation of detailed test cases.

### **2. Test Case:**

* A test case is a detailed and specific set of conditions, inputs, actions, and expected results to verify if a particular aspect of the software is working correctly.
* To verify a specific functionality or feature by providing step-by-step instructions for the tester to follow.
* Test cases are written to verify whether a particular function works as expected, based on the requirements.

### **3. Test Script:**

* A test script is an automated set of instructions written in a programming or scripting language to test the functionality of the software. It is essentially an automated version of a test case.
* To automate the execution of test cases, reducing the need for manual testing and improving efficiency.
* Test scripts are written to run tests automatically, using tools like Selenium, QTP, JUnit, etc.

\*Explain what the Test Plan is? What is the information that should be covered?

A **Test Plan** is a document that outlines the strategy, scope, objectives, resources, and schedule for testing activities related to a specific software project. It is used to ensure that all aspects of the software are tested effectively and systematically to identify bugs, ensure quality, and meet project requirements.

### **Information Covered in a Test Plan:**

1. **Test Plan Identifier**
2. **Introduction/Overview**
3. **Test Objectives**
4. **Test Scope**
5. **Test Strategy/Approach**
6. **Test Criteria**
7. **Test Deliverables**
8. **Resources and Responsibilities**
9. **Test Schedule**
10. **Risk Management**
11. **Test Environment**
12. **Test Data**
13. **Communication Plan**
14. **Defect Management**

**\*What is priority?**

**Priority refers to the level of importance, urgency, or precedence assigned to a task, goal, or decision in comparison to others. It helps in determining what should be addressed first or receive more attention and resources based on its significance or impact.**

**\* What is severity?**

**Severity refers to the degree of seriousness, impact, or intensity of an issue, condition, or situation. It is often used to evaluate how critical or harmful something is and determines the level of attention or resources it requires.**

**\*Bug categories are…**

**1. Functional Bugs**

**2. Performance Bugs**

**3. Security Bugs**

**4. UI/UX Bugs**

**5. Compatibility Bugs**

**6. Logic Bugs**

**7. Data-Related Bugs**

**8. Integration Bugs**

### **9. Critical or Blocking Bugs**

### **10. Regression Bugs**

### **11. Documentation Bugs**

12. **Hardware Bugs**

**\*Advantage of Bugzilla.**

### **1. Open-Source and Cost-Effective**

* **Advantage: Bugzilla is open-source, meaning it is free to use and can be customized to meet specific needs without additional licensing costs.**

### **2. Comprehensive Bug Tracking**

* **Advantage: Provides advanced features for tracking, reporting, and managing bugs and issues.**

### **3. Customizability**

* **Advantage: Highly customizable to adapt to different workflows and processes.**

### **4. Detailed Reporting**

* **Advantage: Offers robust reporting tools and metrics to track bug trends, workload, and project progress.**

### **5. Workflow Automation**

* **Advantage: Allows for creating custom workflows and automated notifications.**

### **6. Integration Capabilities**

* **Advantage: Integrates with various development tools like version control systems (e.g., Git, SVN) and testing frameworks.**

### **7. Multi-Platform Support**

* **Advantage: Runs on multiple operating systems, including Windows, macOS, and Linux.**

### **8. Scalability**

* **Advantage: Suitable for projects of all sizes, from small teams to large enterprise-level projects.**

### **9. Security Features**

* **Advantage: Offers user authentication, access controls, and SSL support to protect sensitive data.**

### **10. Active Community and Support**

* **Advantage: Backed by an active open-source community providing regular updates, plugins, and support.**

### **11. Email Integration**

* **Advantage: Built-in email notifications for updates on bug status or new comments.**

### **12. Historical Tracking**

* **Advantage: Maintains a detailed history of each bug, including all changes, comments, and resolutions.**

### **13. Access Control**

* **Advantage: Fine-grained user permissions allow administrators to control access to specific features or projects.**

**\*Difference between priority and severity**

1. **Severity focuses on the technical impact, while priority is determined by business needs.**
2. **A bug with high severity might not always have high priority and vice versa.**

**High Severity, Low Priority:**

* 1. **A rarely used feature in a mobile app causes the app to crash.**
  2. **Fixing it is important but not urgent as it doesn't impact most users.**

1. **Low Severity, High Priority:**
   1. **A spelling mistake in the company name on the homepage.**
   2. **Technically minor, but highly visible and critical for branding.**

**\* What are the different Methodologies in Agile Development Model?**

**Agile development encompasses a range of methodologies designed to enhance flexibility, collaboration, and iterative progress in software development. Each Agile methodology provides a framework for teams to plan, develop, and deliver products efficiently.**

**1. Scrum**

**2. Kanban**

### **3. Extreme Programming (XP)**

4. Lean Development

5. Crystal

6. Feature-Driven Development (FDD)

7. Adaptive Software Development (ASD)

8. Dynamic Systems Development Method (DSDM)

### **9. Agile Unified Process (AUP)**

10. Scaled Agile Framework (SAFe)

11. DevOps (Agile-Related)

\*Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?

Authentication:

The process of verifying the identity of a user or system.

Authorization:

The process of verifying what actions or resources a user is allowed to access.

Common Problems Faced in Web Testing:

1. Functional Issues

2. Cross-Browser Compatibility

3. Cross-Device Compatibility

4. Performance Issues

5. Security Vulnerabilities

6. Usability Issues

7. Localization and Globalization Issues

8. Integration Issues

9. Session and State Management

10. Data Integrity Issues

11. Search Engine Optimization (SEO) Problems