**Module–2(Manual Testing)**

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

Ans.Exploratory Testing is a type of software testing where testers actively explore the application, using their creativity and experience to identify defects, issues, or unexpected behavior. Unlike scripted testing, where tests are predefined and executed according to a test case, exploratory testing emphasizes simultaneous learning, test design, and test execution.

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

Ans. A **traceability matrix** is a document used in software development and project management to ensure that all requirements are addressed throughout the lifecycle of a project. It helps to track the relationships between different project elements, such as requirements, design, testing, and implementation. The primary purpose of a traceability matrix is to ensure that every requirement is properly tested, verified, and implemented and that no requirements are overlooked.

Q.3 **What is Boundary value testing?**

**Ans. Boundary Value Testing (BVT)** is a software testing technique that focuses on testing the boundaries of input values, rather than the values in the middle of input ranges. The idea is that errors often occur at the edges of input ranges or near boundary values. By testing these boundary conditions, you are more likely to uncover defects that might not be found by testing with values in the middle.

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

Ans. Equivalence partitioning testing is a software testing technique which divides the application input test data into each partition at least once of equivalent data from which test cases can be derived. By this testing method it reduces the time required for software testing.

**Q.5 What is Integration testing?**

Ans. Integration testing is a level of software testing process, where individual units of an application are combined and tested. It is usually performed after unit and functional testing.

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

Ans. The likelihood of an adverse event and the impact of the event determine the level of risk.

Q.7 **What is Alpha testing?**

Ans. Alpha testing is done to make sure a product is ready to send to potential end users of beta testing. During Alpha testing, internal tester check the product for bugs and other quality issues. These internal testers include stakeholders, team members etc. Alpha testing occurs before product launch.

**Q.8 What is beta testing?**

**Ans. Beta testing** is a phase in the software development process where a product, application, or system is tested by a group of real users outside of the development team. It comes after alpha testing, which is typically performed by internal teams to identify and fix major bugs.

Q.9 **What is component testing?**

Ans. Component testing (also known as unit testing) is a type of software testing where individual components or modules of a software application are tested in isolation from the rest of the system. The goal of component testing is to ensure that each individual unit of the software (often a single function, method, or class) works correctly according to its specified behavior.

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

**Ans.** Functional system testing is a type of software testing that focuses on verifying whether a system or application performs its intended functions correctly. It ensures that all features and functionalities work according to the specified requirements and that the system behaves as expected under normal conditions.

Q.11 **What is Non-Functional Testing?**

Ans. Non-functional testing refers to the testing of the non-functional aspects of a software application. While functional testing ensures that the system works according to specified requirements, non-functional testing focuses on how well the system performs under various conditions. This includes testing aspects such as performance, usability, security, and compatibility, which do not directly relate to the specific functionality of the software, but are crucial for ensuring the overall quality and user experience.

**Q.12 What is GUI Testing?**

**Ans. GUI testing** (Graphical User Interface testing) is a type of software testing that focuses on verifying the **user interface (UI)** of an application. The goal is to ensure that the graphical elements of the application (such as buttons, icons, menus, text boxes, and images) are displayed and function correctly from the user's perspective.

Q.13 **What is Ad hoc testing?**

**Ans. Ad hoc testing** is an informal and unstructured type of software testing where testers try to identify defects without following any predefined test cases or scripts. The main goal of ad hoc testing is to discover unexpected issues in the software by testing it in an intuitive, random, or exploratory manner. It is typically done without detailed planning or documentation and is based on the tester's understanding of the application, experience, and creativity.

**Q.14 What is load testing?**

**Ans. Load testing** is a type of **performance testing** that evaluates how a system or application behaves under a specific, expected load of users or transactions. The main goal is to determine whether the system can handle a predefined number of simultaneous users or requests without performance degradation or failure.

**Q.15 What is stress Testing?**

Stress testing is a type of performance testing that evaluates how a system or application performs under extreme conditions, beyond its normal operational capacity. The goal of stress testing is to determine the system's breaking point, identify how it behaves under stress, and assess how it recovers from failure once it exceeds its limits.

**Q.16 What is white box testing and list the types of white box testing?**

Ans. White box testing technique involves selection of test cases based on an analysis of the internal structure (Code coverage, branches coverage, paths coverage, condition coverage etc.) of a component or system. It is also known as Code-Based testing or Structural testing.

Different types of white box testing are

1. Statement Coverage

2. Decision Coverage

**Q.17 What is black box testing? What are the different black box testing techniques?**

Ans. Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program. This testing is usually done to check the functionality of an application. The different black box testing techniques are

1. Equivalence Partitioning

2. Boundary value analysis

3. Cause effect graphing

**Q.18 Mention what are the categories of defects?**

Ans. Mainly there are three defect categories

* **Wrong:** When requirement is implemented incorrectly
* **Missing:** It is a variance from the specification, an indication that a specification was not implemented or a requirement of the customer is not met
* **Extra**: A requirement incorporated into the product that was not given by the end customer. It is considered as a defect because it is a variance from the existing requirements.

**Q.19 Mention what big bang testing is?**

**Ans. Big Bang Testing** is a type of software testing strategy where all components or modules of a system are integrated and tested simultaneously, rather than being tested individually or incrementally. In this approach, developers or testers combine all the system components into a single environment and test them as a whole, without conducting individual unit tests or integration tests first.

Big Bang Testing is generally not preferred for large-scale or complex projects, as it can lead to inefficiencies and challenges in managing defects. More commonly, testing is done incrementally (such as with unit tests, integration tests, and system tests), which is more structured and manageable.

**Q.20 What is the purpose of exit criteria?**

Ans. The purpose of exit criteria is to define when a test level is completed.

**Q.21 When should "Regression Testing" be performed?**

Ans. After the software has changed or when the environment has changed Regression testing should be performed.

**Q.22 What is 7 key principles? Explain in detail?**

**Ans.**  **7 Key Principles of Software Testing**:

**1. Testing Shows the Presence of Defects**

The primary purpose of testing is to find defects (bugs) in the software. However, it's important to recognize that testing does not prove the absence of defects; it only provides evidence of defects that are present. The goal of software testing is not to guarantee a bug-free product but to identify and fix as many issues as possible before release

**2. Exhaustive Testing is Impossible**

It's practically impossible to test every possible combination of inputs, states, and behaviors in a complex software system. Software testing typically involves selecting a subset of tests that provide maximum coverage of critical areas, rather than testing everything. This is known as risk-based testing.

**3. Early Testing**

The earlier testing begins in the software development life cycle (SDLC), the more effective it is. Identifying and addressing defects early in the development process is more cost-effective than finding and fixing them later. This principle emphasizes **shifting left**, where testing is done as early as possible (even during the requirement and design phases).

**4. Defect Clustering**

This principle states that defects tend to cluster in specific areas of a software application, rather than being evenly distributed. A small number of modules or features typically contain the majority of the defects. By identifying and focusing on these defect-prone areas, testers can prioritize their efforts.

**5. Absence of Errors is a Fallacy**

This principle reminds us that finding and fixing defects is important, but it’s also important to focus on whether the software is doing the **right thing**. Absence of errors in the software does not necessarily mean the software is meeting customer needs. A product might be defect-free but still fail to deliver the required functionality or performance.

**6. Testing is Context Dependent**

The approach to testing depends on the context of the software being tested. Factors such as the type of software (e.g., web, mobile, embedded), its complexity, the risk involved, and the resources available all influence the testing approach. There is no "one-size-fits-all" approach to testing.

**7. Pesticide Paradox**

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

**Q.23 Difference between QA v/s QC v/s Tester**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Quality Assurance** | **Quality Control** | **Testing** |
| 1. | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2. | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| 3. | Process oriented activities. | Product oriented activities. | Product oriented activities. |
| 4. | Preventive activities. | It is a corrective process. | It is a preventive process. |
| 5. | It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

**Q.24 Difference between Smoke and Sanity?**

Ans.



|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality / bugs have been fixed |
| The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of Regression testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |

**Q. 25 Difference between verification and Validation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Verification** | **Validation** |
| Definition | The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| Objective | To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements. | To ensure that the product actually meets the user’s needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment. |
| Evaluation Items | Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| Activities | • Reviews  • Walkthroughs  • Inspections | • Testing |

**Q.26 Explain types of Performance testing.**

**1. Load Testing**

* **Purpose**: To determine how the system behaves under normal and expected load conditions.
* **Description**: In load testing, the system is tested with a specific volume of concurrent users or transactions to ensure it can handle the expected load without performance degradation.
* **Example**: A website is tested with 500 simultaneous users to check if it can handle the load without crashing or slowing down.

**2. Stress Testing**

* **Purpose**: To determine the system's stability and behavior when subjected to stress or excessive load.
* **Description**: Stress testing involves pushing the system beyond its expected capacity, often to the point of failure, to observe how it recovers or fails. The goal is to find the system's breaking point and understand its limits.
* **Example**: A web application is subjected to 10 times the expected traffic to observe if it crashes and how it recovers.

**3. Spike Testing**

* **Purpose**: To observe how the system reacts to sudden spikes in load.
* **Description**: In spike testing, a sudden, large increase in load (users or requests) is introduced in a short period to see how the system responds to rapid changes. It's similar to stress testing but focuses more on sudden surges.
* **Example**: A sudden spike in users visiting an e-commerce site during a flash sale to ensure the site doesn’t go down.

**4. Endurance Testing (Soak Testing)**

* **Purpose**: To determine how the system performs over an extended period under a normal load.
* **Description**: Endurance testing tests the system's ability to handle a continuous load over a prolonged period (days, weeks, etc.) to check for memory leaks, performance degradation, or any other long-term issues.
* **Example**: Running a system with a steady stream of users over a 24-hour period to see if it can maintain consistent performance.

**5. Scalability Testing**

* **Purpose**: To evaluate the system's ability to scale up or down as demand increases or decreases.
* **Description**: Scalability testing assesses how well the system can handle an increasing number of users, transactions, or data. It also looks at the system's ability to scale down when demand decreases.
* **Example**: Testing a cloud-based application to see if adding more servers can accommodate an increasing number of users without sacrificing performance.

**6. Volume Testing**

* **Purpose**: To check the system’s behavior when it handles a large volume of data.
* **Description**: Volume testing focuses on testing how well the system performs when it processes a large amount of data (e.g., database records). It helps identify issues like slow query responses or data processing problems.
* **Example**: Testing an application’s database performance when handling millions of records to ensure fast and efficient querying.

**Q.27 What is Error, Defect, Bug and failure?**

**1. Error**

* **Definition**: An error is a human mistake or incorrect action in the software development process. It typically refers to a fault made by a developer or a team member, such as a logical mistake, a coding typo, or a miscalculation during design or implementation.
* **Cause**: Errors usually occur during development, coding, or configuration.
* **Example**: A developer writing code with a logical flaw or using an incorrect algorithm.

**2. Defect**

* **Definition**: A defect is a flaw or issue in the software that causes it to behave unexpectedly or incorrectly. It typically arises from an error in the code or design. A defect is often identified after the software is written and during the testing phase.
* **Cause**: Defects are the result of errors made by developers that were not detected or addressed before the software was released for testing.
* **Example**: A function in a web application not performing its intended task correctly due to a coding mistake.

**3. Bug**

* **Definition**: A bug is a specific type of defect that causes a program to function incorrectly or produce an unintended result. The term "bug" is often used to describe a technical issue that prevents the software from working as expected. It’s a more informal term than "defect," but they are essentially the same.
* **Cause**: Bugs arise from defects in the code or logic that prevent the system from performing as designed.
* **Example**: A button in a mobile app that, when clicked, crashes the application — this is a bug in the code.

**4. Failure**

* **Definition**: A failure is when the software does not perform its intended function or meet the expected requirements during execution. It is the manifestation of a defect or bug during real-world use or testing. A failure occurs when the software behaves incorrectly, causing it to be unusable or inefficient.
* **Cause**: A failure happens when the system encounters a defect or bug that impacts the end-user experience or system performance.
* **Example**: A user trying to submit a form on a website, but the form doesn't process due to a bug in the backend code, causing a failure in completing the task.

**Q.28 Difference between Priority and Severity**

**Ans.**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Severity** | **Priority** |
| Definition | Impact or seriousness of the defect on the application | Urgency or importance of fixing the defect |
| Focus | Technical impact on functionality and user experience | Business needs and release schedule |
| Impact | Affects system functionality or performance | Affects how soon the defect needs to be fixed |
| Example | System crash or data corruption due to a bug | Fixing a UI issue in a non-critical feature |
| Levels | Critical, Major, Minor, Trivial | High, Medium, Low |
| Decision Maker | Typically determined by developers or QA teams | Typically determined by product managers or business teams |

**Q.29 What is Bug Life Cycle?**

Ans. **Bug (Defect) Life Cycle**

“A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”

The duration or time span between the first times defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’.

1. New:

The bug is reported by a tester or user. It is not yet verified by the development team. This is the initial state.

2. Assigned:

The bug is assigned to a developer or a team for investigation and resolution. The developer will begin analyzing and fixing the issue.

3. Open:

The developer starts working on the bug. At this point, the bug is being actively investigated or fixed.

4. Fixed:

The developer has resolved the issue and submitted the fix. The bug is marked as "fixed," and the testing team will now verify whether the issue has been correctly addressed.

5. Pending Retest:

After the bug fix, the tester will verify the fix to ensure that the issue is resolved and that no new issues have been introduced.

6. Retested:

The bug fix is verified during testing. If the issue is fixed successfully, the bug moves to the next stage. If the issue persists, it may be sent back to the developer for further investigation.

7. Closed:

Once the bug is fixed and verified, it is closed. The issue is marked as resolved, and no further action is needed.

8. Reopened (if applicable):

If the bug persists even after being marked as fixed or if the fix was not implemented properly, the tester can reopen the bug. This often leads to further investigation and fixing by the development team.

9. Deferred (optional):

In some cases, the bug is considered low-priority or irrelevant to the current release. It is postponed for a future release and marked as "deferred" or "won't fix."

10. Rejected (optional):

Sometimes, the reported bug may be a misunderstanding or not an actual defect. In such cases, it is rejected by the development team or test team.

**Q.30 Explain the difference between Functional testing and Non-Functional testing**

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. | Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first | Non-functional testing should be performed after functional testing |
| Manual testing or automation tools can be used for functional testing | Using tools will be effective for this testing |
| Functional testing describes what the product does | Nonfunctional testing describes how good the product works |
| Easy to do manual testing | Tough to do manual testing |
| Types of Functional testing are  • Unit Testing  • Smoke Testing  • Sanity Testing  • Integration Testing  • White box testing  • Black Box testing  • User Acceptance testing  • Regression Testing | Types of Nonfunctional testing are  • Performance Testing  • Load Testing  • Volume Testing  • Stress Testing  • Security Testing  • Installation Testing  • Penetration Testing  • Compatibility Testing  • Migration Testing |

**Q.31 What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

**Ans.**

|  |  |
| --- | --- |
| **SDLC (Software Development Life Cycle)** | **STLC (Software Testing Life Cycle)** |
| SDLC is the process of developing software from inception to deployment. | STLC is the process of testing the software to ensure it meets the requirements. |
| Focuses on **designing, developing**, and **deploying** software. | Focuses on **testing** the software for quality assurance. |
| To build a high-quality software product. | To identify defects and ensure the software is of high quality. |
| Includes phases like **requirement gathering, design, coding**, testing, and deployment. | Includes phases like **test planning, test design, test execution**, and defect reporting. |
| Entire software development process, including design, development, and maintenance. | Testing-focused activities, mainly aimed at identifying and resolving defects. |
| SDLC starts from project initiation and continues until the software is delivered and maintained. | STLC occurs after the development phase begins, but can overlap with development in some cases. |

**Q.32 What is the difference between test scenarios, test cases, and test script?**

Ans. Difference between test scenarios and test cases is that

**Test Scenarios:** A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.

**Test Cases:** It is a document that contains the steps that has to be executed, it has been planned earlier.

**Test Script:** It is written in a programming language and it's a short program used to test part of functionality of the software system. In other words, a written set of steps that should be performed manually.

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **Test Scenario** | **Test Case** | **Test Script** |
| Definition | A high-level description of what needs to be tested. | A detailed step-by-step procedure for testing a specific feature. | A set of instructions or code used for automating the execution of a test case. |
| Purpose | To identify what functionality needs to be tested. | To provide detailed steps, inputs, and expected results for testing. | To automate the execution of test cases. |
| Detail Level | High-level, abstract. | Detailed, specific to functionality. | Code-based, automates execution of the test case. |
| Example | "Test the login functionality." | "Verify that entering correct username and password logs the user in." | Code that automates login functionality testing. |
| Used By | Test managers, and sometimes business analysts. | Testers (manual or automated). | Test automation engineers. |

**Q.33 Explain what Test Plan is? What is the information that should be covered.**

Ans. A **Test Plan** is a comprehensive document that outlines the strategy, approach, scope, objectives, resources, and schedule for testing activities within a software development project. It serves as a blueprint for the testing process, providing direction and guidelines to ensure that the testing is structured, systematic, and aligned with project goals.

**Test Plan Identifier**- Unique reference for tracking the test plan.

**Introduction-** Overview of the testing effort and background information about the project.

**Test Objectives-** Clear goals of the testing effort.

**Test Scope-** Defines the areas of the application that will and won’t be tested.

**Test Approach-** The testing strategy, including manual or automated testing and testing types.

**Test Criteria-** Entry and exit conditions for starting and completing the testing.

**Test Deliverables-** List of test documents, reports, and outputs.

**Testing Schedule-** Timeline and milestones for testing activities.

**Resource Requirements-** Identification of resources (personnel, tools, hardware, software).

**Test Environment-** Details on the hardware, software, and configurations needed for testing.

**Risks and Contingencies-** Potential risks and the strategies for mitigating them.

**Test Case Design and Execution-** Guidelines for test case creation, execution, and tracking.

**Approval-** Stakeholder names and approval of the test plan.

**Assumptions and Dependencies-** Assumptions made and external dependencies that might affect testing.

**Q.34 What is priority?**

**Ans.**

* **Definition**: Priority refers to the **urgency** or **importance** of fixing a defect based on factors like business needs, user impact, and release schedules. It determines the order in which defects should be fixed.
* **Focus**: Priority focuses on the **business needs** or **release requirements** rather than the technical impact.
* **Classification**: Priority is typically classified into levels such as:
  + **High Priority**: The defect needs to be fixed immediately, as it directly affects the users or critical functionality.
  + **Medium Priority**: The defect should be fixed soon but is not as urgent as high-priority issues.
  + **Low Priority**: The defect is not urgent and can be fixed later, as it has a minimal impact on the user or the business.

**Q.35 What is severity?**

**Ans**

* **Definition**: Severity refers to the **impact** or **seriousness** of a defect or bug on the functionality or performance of the software. It indicates how much the defect affects the system, its features, or the user experience.
* **Focus**: Severity is primarily concerned with the **technical impact** of the defect on the application.
* **Classification**: Severity is generally classified into levels such as:
  + **Critical/Blocker**: The defect prevents the software from functioning entirely or crashes the system.
  + **Major**: The defect significantly affects functionality but does not cause a complete breakdown of the system.
  + **Minor**: The defect has a small impact on functionality, but it does not prevent the system from working.
  + **Trivial/Low**: The defect has minimal impact on the software, often cosmetic or related to non-essential features.

**Q.36 Bug categories are…**

Ans.

|  |  |
| --- | --- |
| **Bug Category** | **Description** |
| Functional Bugs | Related to issues in functionality as per the requirements. |
| Performance Bugs | Issues related to the speed, scalability, or responsiveness of the application. |
| Security Bugs | Vulnerabilities or weaknesses in the software's security. |
| UI Bugs | Visual issues in the design or layout of the software interface. |
| Compatibility Bugs | Failures to work across different devices, browsers, or operating systems. |
| Regression Bugs | Bugs that arise when previously working functionality is affected by new code changes. |
| Database Bugs | Issues related to the database queries, integrity, or data manipulation. |
| Integration Bugs | Problems arising from the interaction between different modules or external systems. |
| Usability Bugs | Bugs that affect the user experience or make the software harder to use. |
| Localization Bugs | Bugs related to the software’s functionality in different languages or regions. |
| Critical/Blocker Bugs | Serious issues that prevent key operations or cause the system to stop working. |
| Non-Critical Bugs | Minor issues that do not affect the overall functioning of the application. |
| Memory Leaks | Issues where the software fails to release memory that is no longer in use. |
| Concurrency Bugs | Problems that occur in multi-threaded environments when processes interfere with each other. |
| Installation Bugs | Issues that arise during the installation, uninstallation, or updating of the software. |

**Q.37 Advantage of Bugzilla.**

Ans. Bugzilla is an open sources bug tracking tool that has many advantages, including:

**Improved product quality:** Bugzilla can help improve the quality of your product.

**Enhanced communication:** Bugzilla can help improve communication between developers and testers.

**Increased productivity:** Bugzilla can help increase productivity.

**Advanced search:** Bugzilla has a Google-like search that can search the full text of a bug. You can also create custom searches, such as time-based searches.

**Time tracking:** Bugzilla has a time tracking feature that allows you to log time spent on bug fixing.

**Deadlines:** You can set deadlines for fixing bugs to ensure timely delivery of your project.

**Assigning bugs:** You can assign bugs to specific team members to promote accountability and efficient task allocation.

**Moving bugs:** You can move a bug from one Bugzilla installation to another, even across versions.

**Collaborative environment:** Bugzilla provides a collaborative environment between developers and testers.

**Email notifications:** Bugzilla provides email notifications related to bugs if any changes occur.

**Q.38 What are the different Methodologies in Agile Development Model?**

Ans. There are currently seven different agile methodologies are:

1. Extreme Programming (XP)

2. Scrum

3. Lean Software Development

4. Feature-Driven Development

5. Agile Unified Process

6. Crystal

7. Dynamic Systems Development Model (DSDM)

**Q.39 Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?**

**1. Authentication:**

* **Definition:** Authentication is the process of verifying the identity of a user. It ensures that the person attempting to access the system is who they claim to be.
* **How it works:** During authentication, users typically provide credentials like usernames, passwords, biometric data, or authentication tokens (such as from two-factor authentication).
* **Example:** A user logs into a web application with their username and password. The system checks if the credentials match what is stored in its database.

**2. Authorization:**

* **Definition:** Authorization occurs after authentication and is the process of determining what actions a user is allowed to perform and what resources they can access.
* **How it works:** Based on the authenticated user's role, permissions, and other policies, the system grants or denies access to specific resources or actions.
* **Example:** After logging in, a user may be granted access to a dashboard but not to sensitive financial data, depending on their role (e.g., admin, user, manager).

**Common Problems Faced in Web Testing:**

Web testing involves testing various aspects of a web application to ensure its functionality, security, and performance. Common problems encountered include:

1. **Cross-browser Compatibility Issues:**
   * Web applications may work fine on one browser but may encounter layout or functionality issues on others. This includes differences in rendering, support for features, and JavaScript behavior.
2. **Security Vulnerabilities:**
   * Issues like SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) are common in web applications and need thorough testing.
   * Insecure authentication (e.g., weak passwords or insecure session handling) and improper authorization (e.g., privilege escalation) can also pose security risks.
3. **Session Management Issues:**
   * Problems like session fixation, session hijacking, and the improper handling of session timeouts can affect the security and user experience of the application.
4. **Performance and Load Issues:**
   * Web applications may slow down or crash under high traffic. Load testing and performance testing are crucial to identify bottlenecks and optimize the application.
5. **Broken Links and Missing Resources:**
   * errors (broken links) and missing images, scripts, or other resources can negatively affect the user experience and functionality of a website.
6. **Inadequate Input Validation:**
   * Failure to validate input data properly can lead to security vulnerabilities and application crashes. This is especially important for forms, file uploads, and URL parameters.
7. **Usability Issues:**
   * Problems related to the user interface (UI), such as poor navigation, unclear error messages, or lack of mobile responsiveness, can affect the overall user experience.
8. **Data Integrity and Synchronization:**
   * Issues can arise when data is not properly synchronized between the client-side and server-side, or when users encounter race conditions or inconsistent data.
9. **Mobile Responsiveness:**
   * As mobile traffic increases, ensuring the web application works seamlessly across various mobile devices and screen sizes is a common challenge.
10. **Third-party Integrations:**
    * Many web applications rely on third-party services (e.g., payment gateways, social media login, APIs). Testing these integrations can be challenging, especially if the third-party service experiences downtime or changes their API.

**Q.39 When to used Usability Testing?**

**Ans.** Usability Testing is conducted to evaluate how user-friendly and intuitive a software application, product, or system is. It focuses on the user experience (UX) and aims to identify areas of the product that might be difficult or confusing for users. This type of testing helps ensure that the product meets the expectations of its target audience and is easy to use.

Early stages of development to gather initial user feedback.

Major design changes or feature updates to ensure new changes don’t negatively impact usability.

Pre-launch testing to validate the product is user-friendly.

Post-launch to ensure continuous improvement and user satisfaction.

For complex features to check if users can easily understand and use them.

For accessibility to ensure the product is usable by all, including disabled users.

When user interaction is key to the product’s success.

**Q.40 What is the procedure for GUI Testing?**

**1. Layout and Visual Testing**

* + Verify GUI Elements: Check the presence, alignment, and sizing of GUI elements (e.g., buttons, labels, text boxes).
  + Check Color Scheme and Fonts: Verify the application's color scheme, font styles, and font sizes.
  + Validate Images and Icons: Check the presence, quality, and alignment of images and icons.

**2. Functionality Testing**

* + Test Menu Options: Verify that menu options are functional and behave as expected.
  + Validate Button and Link Functionality: Test buttons and links to ensure they perform the expected actions.
  + Test Form Validation: Verify that form fields are validated correctly, including error messages and feedback.
  + Test Data Entry and Editing: Test data entry and editing functionality, including data validation and feedback.

**3. Usability Testing**

* + Test Navigation: Verify that navigation is intuitive and easy to use.
  + Test Error Handling: Test error handling, including error messages, feedback, and recovery options.
  + Test Accessibility: Verify that the application is accessible, including keyboard navigation, screen reader support, and high contrast mode.

**4. Compatibility Testing**

* + Test Browser Compatibility: Verify that the application works correctly across different browsers and versions.
  + Test Operating System Compatibility: Test the application on different operating systems, including Windows, macOS, and Linux.
  + Test Device Compatibility: Verify that the application works correctly on different devices, including desktops, laptops, tablets, and smartphones.

**5. Security Testing**

* + Test Authentication and Authorization: Verify that authentication and authorization mechanisms are secure and functioning correctly.
  + Test Data Encryption: Test data encryption, including data at rest and data in transit.
  + Test Input Validation: Verify that input validation mechanisms are secure and functioning correctly.

**Q.41 To create HLR & Test Case of (Instagram, Facebook) first page and chat functionality**

**Q.42 Facebook Login Page :** [**https://www.facebook.com**](https://www.facebook.com)

**Q.43 To create HLR & Test Case of Web Based (WhatsApp web) 1. WhatsApp Web :** [**https://web.whatsapp.com**](https://web.whatsapp.com)

**Q.44 Create Test Cases on What’s app Group Chat.**

**Q.45 Write a scenario of only What’s app chat messages**

**Q.46 Write a Scenario of Pen**

**Q.47 Write a Scenario of Pen Stand**

**Q.48 Write a Scenario of Door**

**Q.49 Write a Scenario of ATM**

**Q.50 Write a scenario of Microwave Owen**

**Q.51 Write a scenario of Coffee vending Machine**

**Q.52 Write a scenario of chair**

**Q.53 To Create Scenario (Positive & Negative)**

**1.Create Test Cases on Compose Mail Functionality.**

**2. Online shopping to buy product (flip cart)**

**Q.54 Write a Scenario of Wrist Watch**

**Q.55Write a Scenario of Lift (Elevator)**

**Q.56 Write a Scenario of What’s app payment**