ASSIGNMENT MODULE 2

Q 1. What is Exploratory Testing?

Ans. Exploratory Testing is an unscripted, hands-on approach to software testing where testers dynamically design, execute, and adapt tests in real-time based on their intuition, experience, and observations. Unlike scripted testing (where predefined test cases are followed), exploratory testing emphasizes learning, creativity, and flexibility to uncover unexpected defects.

Q 2. What is traceability matrix?

Ans. A Traceability Matrix is a table that connects different project items (like requirements, test cases, and bugs) to ensure nothing is missed. It helps track if all requirements are tested and if all tests cover the right requirements.

Q 3. What is Boundary value testing?

Ans. Boundary Value Testing is a software testing technique where test cases are designed to check the behaviour of a system at the edges (boundaries) of input ranges.

Q 4. What is Equivalence partitioning testing?

Ans. Equivalence Partitioning is a software testing technique where input data is divided into groups (partitions) that are expected to behave the same way. Instead of testing every possible input, you test one representative value from each group to save time while maintaining coverage.

Q 5. What is Integration testing?

Ans. Integration Testing is a software testing phase where individual modules (or components) of an application are combined and tested together to ensure they work correctly as a group.

Q 6. What determines the level of risk?

Ans. Risk is a factor that could result in future negative consequences; usually expressed as impact and likelihood.

Type of Risk

1. Project Risk
2. Product Risk

Q 7. What is Alpha testing?

Ans. Alpha Testing is an early real-world testing phase where the software is tested internally (by developers or QA teams) before releasing it to external users. It happens in a controlled environment, often at the developer’s site.

Q 8. What is beta testing?

Ans. Beta Testing is a real-world testing phase where a near-final version of the software is released to a limited group of external users (outside the company) to gather feedback before the official launch.

Q 9. What is component testing?

Ans. Component Testing (also called Unit Testing or Module Testing) is a type of testing where individual parts (components) of a software application are tested in isolation to ensure they work correctly before integration.

Q 10. What is functional system testing?

Ans. Functional System Testing is a type of testing where the entire software system is tested as a whole to ensure it meets all specified functional requirements

Functional Testing Types:

1. Black Box Testing
2. White Box Testing
3. Experience based Testing
4. Smoke Testing
5. Sanity Testing
6. End to End Testing

Q 11. What is Non-Functional Testing?

Ans. Non-Functional Testing checks how well a system performs, rather than what it does. It focuses on aspects like speed, reliability, security, and usability—ensuring the software is efficient, stable, and user-friendly under real-world conditions.

Non Functional Testing Types:

1. Usability Testing
2. Compatibility Testing
3. GUI Testing
4. Security Testing
5. Performance Testing
6. Stress Testing
7. Load Testing

Q 12. What is GUI Testing?

Ans. GUI Testing (Graphical User Interface Testing) checks the visual elements of a software application to ensure they look, behave, and interact correctly with users. It focuses on buttons, menus, layouts, fonts, colours and responsiveness.

Q 13. What is Adhoc testing?

Ans. Adhoc Testing is an informal, unplanned testing approach where testers randomly check the software without predefined test cases or documentation. It relies on the tester's experience, intuition, and creativity to find hidden bugs.

Q 14. What is load testing?

Ans. Load Testing is a type of performance testing that checks how a system behaves under expected or high user traffic. The goal is to identify bottlenecks (like slow responses or crashes) before real users experience them.

Q 15. What is stress Testing?

Ans. Stress Testing is a type of performance testing that pushes a system beyond its normal limits (e.g., extreme user traffic, high data volumes, or limited resources) to see how it handles overload and recovers from failure.

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

Ans. White Box Testing (also called Clear Box or Structural Testing) is a software testing method where the tester examines the internal code, structure, and logic of an application to design test cases. Unlike black-box testing (which focuses on functionality), white-box testing requires knowledge of programming and implementation details.

White Box Testing List:

1. Unit Testing
2. Static Code Analysis
3. Statement Coverage
4. Branch Coverage
5. Path Coverage
6. Loop Testing
7. Mutation Testing
8. Integration Testing (White-Box Approach)

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

Ans. Black Box Testing is a software testing method where the tester evaluates the functionality of an application without knowing its internal code or structure. The focus is on inputs and outputs—testing whether the system behaves as expected based on requirements.

Black Box Testing List:

1. Equivalence Partitioning
2. Boundary Value Analysis (BVA)
3. Decision Table Testing
4. State Transition Testing

Q 18. Mention what are the categories of defects.

Ans. Based on Severity

1. Critical Defect
2. Major Defect
3. Minor Defect
4. Trivial Defect

Based on Priority

1. High Priority
2. Medium Priority
3. Low Priority

Q 19. Mention what big bang testing is?

Ans. Big Bang Testing is an integration testing approach where all or most modules of a system are combined at once and tested as a single unit, *without* incremental integration.

Example:

A team builds a basic calculator app and tests all buttons (add, subtract, etc.) together instead of checking each operation step-by-step.

Q 20. What is the purpose of exit criteria?

Ans. Exit Criteria are predefined conditions or standards that must be met before testing can be formally concluded. They ensure that the software is thoroughly validated and ready for release.

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

Ans. Regression Testing ensures that new code changes (like bug fixes, updates, or features) do not break existing functionality.

It should be performed in these scenarios:

1. After Code Changes
2. During Integration
3. Environment/Config Updates
4. Release Cycles
5. Periodic/Scheduled Runs

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

Ans. These principles guide testers in designing effective test strategies and avoiding common pitfalls.

1. Testing shows the presence of Defects

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

1. Exhaustive Testing is impossible

Testing everything including all combinations of inputs and preconditions is not possible.

1. Early Testing

Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

1. Defect Clustering

A small number of modules contain most of the defects discovered during pre-release testing or are responsible for the most operational failures.

1. Pesticide Paradox

If the same tests are repeated over again, eventually the same set of test cases will no longer find any new defects.

1. Testing is context dependent

Testing is context-dependent. Testing is done differently in different contexts. Different kinds of sites are tested differently.

1. Absence of errors fallacy

If the system built is unusable and does not fulfil the users needs and expectations then finding and fixing defects does not help.

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

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **QA** | **QC** | **Tester** |
| Focus | Process-oriented | Product-oriented | testing functionality and behavior |
| Nature | Proactive and preventive | Reactive and corrective | Active testing |
| Objective | Ensures the process and standards are followed to prevent defects | Identifies and fixes defects in the final product | Identifies bugs and ensures quality of the product |
| Scope | Involves the entire software development process | Focuses on final product quality validation | Executes tests on the product |
| Activities | Process audits, reviews, risk management, standards creation | Test case execution, defect identification and reporting | Writing and executing test cases, reporting defects |
| Examples | Creating testing strategies, defining processes | Functional testing, performance testing | Verifying features, running test cases, reporting bugs |

Q 24. Difference between Smoke and Sanity?

Ans.

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| To verify whether the basic functionalities of the application are working correctly. | To verify if a specific functionality or bug fix works as expected after changes. |
| Broad testing of the core features of the application to check for critical failures. | Narrow testing to focus only on the changed or fixed features. |
| After receiving a new build, before deeper testing begins | After minor fixes or enhancements, typically before further detailed testing. |
| Involves a shallow and high-level check of the application's basic features | nvolves testing specific functionalities or areas impacted by recent changes. |
| Checking if the login page, homepage, and a few core functionalities are working after a build. | Verifying that a bug fix, such as the password reset function, works correctly after a recent patch. |

Q 25. Difference between verification and Validation

Ans.

|  |  |  |
| --- | --- | --- |
| **No.** | **Verification** | **Validation** |
| 1 | Verification is a process which is performed at development level | Validation is a process which is performed at testing level |
| 2 | Verification is a static testing | Validation is a dynamic testing |
| 3 | Bugs can be found during the process of development | Bugs can only be found after the process of development |
| 4 | It is used to prevent errors. | It is used to detect errors |
| 5 | Verification activities are Reviews and Inspections. | Validation activity is Testing. |
| 6 | Verification can be achieved by asking “Are you building a product right?” | Validation can be achieved by asking “Are you building a right product?” |

Q 26. Explain types of Performance testing.

Ans. Performance testing is a type of software testing that focuses on evaluating how a system performs under specific workloads. It ensures the software application is stable, scalable, and responsive. There are several types of performance testing, each designed to test different aspects of system performance:

1. Load Testing

Purpose: To check how the system behaves under an expected load.

Example: Testing a website with 1,000 users logging in at the same time.

Goal: Identify performance bottlenecks under normal conditions.

2. Stress Testing

Purpose: To determine the system's breaking point by pushing it beyond normal load limits.

Example: Doubling the expected user load to see if the system crashes.

Goal: Understand how the system behaves under extreme conditions and recoverability.

3. Spike Testing

Purpose: To test how the system reacts to sudden, large spikes in user load.

Example: Suddenly increasing the number of users from 500 to 5,000 in seconds.

Goal: Evaluate the system's ability to handle sharp changes in traffic.

4. Endurance Testing (Soak Testing)

Purpose: To test the system's performance over an extended period under a normal load.

Example: Running a test with 500 users for 24 hours.

Goal: Detect issues like memory leaks, resource exhaustion, or degradation over time.

5. Volume Testing (Flood Testing)

Purpose: To test the system’s ability to handle a large volume of data.

Example: Inputting a huge database file into the application.

Goal: Ensure the system performs well with high data volume.

6. Scalability Testing

Purpose: To determine how well the system scales when resources (like users, data, or hardware) are increased.

Example: Gradually increasing the number of users to see how system performance changes.

Goal: Find the system’s maximum capacity and optimal resource usage.

7. Configuration Testing

Purpose: To evaluate performance under different system configurations.

Example: Changing the number of CPUs or memory size and testing performance.

Goal: Identify the best configuration for optimal performance.

Q 27. What is Error, Defect, Bug and Failure?

Ans. A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then is called bug, build does not meet the requirements then it is failure.

Q 28. Difference between Priority and Severity

Ans.

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Verifies that the software functions as expected and meets business requirements. | Verifies the system's attributes such as performance, security, and usability. |
| Focuses on functional requirements. | Focuses on non-functional requirements. |
| Ensure that specific functions, features, and business logic are working correctly. | Overall system performance, security, scalability, etc. |
| Unit, Integration, System, Acceptance Testing | Performance Testing, Security Testing, Usability Testing, Compatibility Testing |
| Validates "what the system does" | Validates "how the system performs" |

Q 29. What is Bug Life Cycle?

Ans. The Bug Life Cycle describes the stages a defect goes through from discovery to resolution. It ensures proper tracking and closure of bugs in a systematic way.

Stages of the Bug Life Cycle

1. New

A bug is reported for the first time by a tester/end-user.

Example: "Login button fails on mobile devices."

2. Assigned

The bug is assigned to a developer for fixing.

3. Open

The developer starts analyzing/fixing the bug.

4. Fixed

The developer resolves the issue and marks it as fixed.

5. Retest

The tester verifies if the bug is truly fixed.

Verified → Moves to "Closed."

Not Fixed → Reopened.

6. Reopened

If the bug still exists, it goes back to the developer.

7. Closed

The bug is fully resolved and no longer exists.

8. Duplicate

If the bug is reported multiple times, it’s marked as a duplicate.

9. Rejected

The developer may reject the bug if:

It’s not reproducible.

It’s not a defect (expected behaviour).

10. Deferred

The bug is postponed (e.g., for a future release).

Q 30. Explain the difference between Functional testing and non-functional testing

Ans. Functional Testing

Functional testing focuses on what the system does. It verifies that the software functions according to the defined business requirements. This type of testing checks individual features like login, signup, form submissions, and other user interactions to make sure they work correctly.

* Purpose: To ensure that each function of the software operates as expected.
* Focus Area: User interface, APIs, database operations, user authentication, and overall application behaviour.
* Example Scenarios:

1. Verifying that users can successfully log in with correct credentials.
2. Checking if a user can reset their password.
3. Ensuring a search feature returns the correct results.

* Tools Used: Selenium, JUnit, TestNG, QTP, etc.
* Type: Usually black-box testing, where the tester does not need to know the internal code.

Non-Functional Testing:

Non-functional testing focuses on how the system performs under certain conditions. It evaluates the application's quality attributes such as performance, usability, security, and scalability.

* Purpose: To ensure the system is reliable, efficient, and user-friendly under various conditions
* Focus Area: System speed, response time, resource usage, reliability, load handling, and user experience.
* Example Scenarios:

1. Checking how the system behaves with 1,000 simultaneous users.
2. Measuring the response time of a web page.
3. Verifying if the application is secure from unauthorized access.

* Tools Used: JMeter, LoadRunner, Postman (for performance APIs), Burp Suite (for security), etc.
* Type: Can be black-box or involve specialized tools and knowledge.

Q 31. To create HLR & TestCase of

1) (Instagram , Facebook) only first page

2) Facebook Login Page : <https://www.facebook.com/>

Ans. Answer in excel.

Q 32.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)** |
| A process used for developing software systems from start to finish. | A process followed for testing the software to ensure quality. |
| To build a fully functional software product. | To ensure the software is bug-free and meets requirements. |
| Covers the entire software development process. | Focuses only on the testing phase of software development. |
| Requirements gathering, design, coding, testing, deployment, and maintenance. | Test planning, test case development, execution, defect logging, and test closure. |
| Business analysts, developers, testers, project managers, etc. | Testers and QA professionals. |
| When a business need or idea is identified. | When the software requirements are defined and ready. |
| The software is deployed and maintained. | Testing is completed and the product is ready for release. |
| Working software system. | Verified and validated product, with defect reports and test metrics. |

Q 33.What is the difference between test scenarios, test cases, and test script?

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **Test Scenario** | **Test Case** | **Test Script** |
| **Level** | High-level | Medium to detailed | Low-level (automation code) |
| **Format** | Plain language | Structured steps | Programming/scripting code |
| **Purpose** | Identify areas to test | Describe how to test those areas | Automate the execution of test cases |
| **Who writes it** | Testers, BAs | Testers | Automation engineers |

Q 34.Explain what Test Plan is? What is the information that should be covered.

Ans.

**Test plan**: A Test Plan is a detailed document that outlines the strategy, objectives, scope, resources, schedule, and activities needed to perform testing for a software project. It acts as a blueprint for how testing will be conducted and managed.

* **Scope**: Define what will and will not be tested, including in-scope and out-of-scope items.
* **Objectives**: Clearly state what the testing effort aims to achieve, such as validating functionality and ensuring requirements are met.
* **Approach**: Outline the testing strategy, methodologies, and tools to be used.
* **Resources**: Identify the necessary resources, including personnel, tools, and equipment, for testing.
* **Schedule**: Establish a timeline for testing activities, including milestones and deadlines.
* **Deliverables**: List the documents and artifacts to be produced during testing, such as test cases and reports.
* **Dependencies**: Identify any dependencies that might affect the testing process.
* **Test Environment**: Describe the hardware, software, and network configurations required for testing.
* **Risk Management**: Identify potential risks and outline mitigation strategies.
* **Roles and Responsibilities**: Define the roles and responsibilities of team members involved in testing.
* **Communication Plan**: Establish how information will be communicated during the testing process.

Q 35.What is priority?

Ans. Priority refers to how soon a bug or test case should be fixed or executed, based on business or project urgency.

Q 36.What is severity?

Ans. Severity describes the extent to which a defect affects the system. It is a technical measure of how badly the system is broken because of the bug.

Q 37. Bug categories are…

Ans.

* **Functional Bugs:** These occur when a software component fails to perform its intended function correctly. For example, a button might not submit a form as expected.
* **Logical Bugs:** These result from flaws in the design or implementation of the software's logic, leading to incorrect behavior.
* **Usability Bugs:** These affect the user experience, making it difficult or confusing for users to interact with the application.
* **Security Bugs:** These pose risks to the security of the software and its data, potentially leading to unauthorized access or data breaches.
* **Performance Bugs:** These impact the speed, efficiency, or responsiveness of the software.
* **System-Level Integration Bugs:** These occur when different software components or modules fail to work together seamlessly.
* **Unit-Level Bugs:** These occur within individual code units or modules of the software.
* **Syntax Bugs:** These arise from errors in the syntax or structure of the software code.
* **Workflow Bugs:** These relate to errors in the sequence of actions a user performs in the software.
* **Compatibility Bugs:** These arise when the software does not function correctly with certain operating systems, browsers, or other software.
* **Concurrency Bugs:** These occur when concurrent programs run without proper synchronization, potentially leading to deadlocks or other issues.

Q 38.Advantage of Bugzilla.

Ans.

* **Deadlines:** To fix the bugs, deadlines can be established.
* **Types:** It reports in a variety of formats and types.
* **Request System:** You can use the 'request system' provided by Bugzilla to ask other users to evaluate codes, provide information and other things.
* **Flexible:** Bugzilla is quite flexible, so you can modify it to fit your unique process and requirements.
* **Bug tracking tool:** Bugzilla is extremely good at monitoring and handling bugs and issues.

Q 39.Difference between priority and severity

Ans.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Severity** | **Priority** |
| **Definition** | How serious the impact of a defect is on functionality or performance. | How urgent it is to fix the defect. |
| **Focus** | Technical impact of the bug. | Business urgency or importance. |
| **Who decides?** | Usually set by testers or QA team. | Usually set by product managers or clients. |
| **Affects** | Stability, functionality, or performance. | Development or release timeline. |
| **Fix Order** | May not be fixed immediately if priority is low. | Fixes may happen quickly even if severity is low. |
| **Examples** | App crashes on login → High Severity | Typo on homepage of a major site → High Priority |

Q 40.What are the different Methodologies in Agile Development Model?

Ans.

* **Scrum:** A framework that uses short cycles (sprints) to manage software development, emphasizing teamwork and continuous improvement.
* **Kanban:** A visual workflow management system that focuses on limiting work in progress and optimizing flow to enable continuous delivery.
* **Extreme Programming (XP):** An agile methodology that emphasizes technical practices like pair programming, test-driven development, and continuous integration.
* **Feature-Driven Development (FDD):** An iterative approach where developers focus on delivering features in short cycles.
* **Lean Software Development:** An approach inspired by Lean Manufacturing that aims to reduce waste and deliver value quickly.
* **Dynamic Systems Development Method (DSDM**)**:** A structured approach to agile project management that emphasizes user involvement and frequent delivery.
* **Behaviour-Driven Development (BDD):** A collaborative approach that focuses on defining and testing software based on specific scenarios and user stories.
* **Adaptive Software Development (ASD):** A framework that emphasizes adaptability and flexibility in response to changing requirements.
* **Crystal:** A family of agile methodologies that adapts to different project types, team sizes, and criticality levels.

Q 41.Explain the difference between Authorization and Authentication in Web testing.What are the common problems faced in Web testing?

Ans.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Authentication** | **Authorization** |
| **Definition** | The process of verifying the identity of a user. | The process of verifying what actions or resources the user has permission to access. |
| **Question it answers** | *"Who are you?"* | *"What are you allowed to do?"* |
| **Occurs When** | During login (e.g., username and password check). | After authentication, when accessing specific resources or features. |
| **Example** | Entering correct credentials to log in. | Accessing the admin panel only if you're an admin. |
| **Failure Result** | Login denied or invalid credentials message. | Access denied or 403 Forbidden error. |
| **Testing Focus** | Login functionality, session management, password security. | Role-based access control, page/feature restrictions. |

**1. Cross-Browser and Device Compatibility:**

* Web applications need to function consistently across various browsers (Chrome, Firefox, Safari, Edge) and devices (desktops, tablets, mobile phones).
* This requires careful testing to ensure layout, functionality, and responsiveness are maintained across different platforms.

**2. Dynamic Content:**

* Web applications often utilize dynamic content, which can be difficult to test due to its changing nature and reliance on backend services.
* Testers need to ensure that dynamic elements render correctly and that data is displayed accurately.

**3. Performance and Load Testing:**

* Web applications must handle traffic and user load effectively without performance degradation.
* Performance testing helps identify bottlenecks and optimize load times.
* Load testing simulates real-world traffic to assess how the application behaves under stress.

**4. Security Vulnerabilities:**

* Web applications are susceptible to various security threats, including cross-site scripting (XSS), SQL injection, and unauthorized access.
* Security testing is crucial to identify and mitigate these vulnerabilities.

5. UI/UX Issues:

* Ensuring a good user experience (UX) is essential for web applications.
* This includes testing aspects like usability, accessibility, and visual design consistency.
* UI testing focuses on ensuring that the user interface works as expected and is user-friendly.

**6. Integration Testing:**

* Web applications often integrate with external services, databases, and other systems.
* Integration testing verifies that these components interact correctly and exchange data seamlessly.

**7. Responsive Design:**

* Web applications need to adapt to different screen sizes and devices.
* Responsive design testing ensures that the layout and functionality are optimized for various devices.

**8. Data Privacy:**

* Web applications handling user data must comply with privacy regulations.
* Testing ensures that data is stored, processed, and transmitted securely, and that users' privacy is protected.

**9. Compliance and Standards:**

* Web applications may need to comply with industry standards and regulations.
* This includes testing aspects like accessibility (WCAG) and security standards (OWASP).

Q 42. To create HLR & TestCase of WebBased (WhatsApp web)

Ans In excel.

Q 43. Create TestCases on Whatsapp Group Chat.

Ans In excel.

Q 44.Write a scenario of only Whatsapp chat messages.

Ans.

Q 45.Write a Scenario of Pen

Ans.

Q 46.Write a Scenario of Pen Stand

Ans.

Q 47.