Software Testing Assignment

Module–2(Manual Testing)

What is Exploratory Testing?

Exploratory testing is a software testing approach where testers actively explore the application without strict predefined test cases

 What is traceability matrix?

A traceability matrix is a document, often in the form of a table, used to map the relationships between different project artifacts, such as requirements, test cases, and design elements.

 What is Boundary value testing?

Boundary Value Analysis (BVA) is a software testing technique where test cases are designed to target the boundaries of input ranges, where errors are most likely to occur. It focuses on testing values at the edges of valid and invalid input ranges to ensure the system handles edge cases correctly.

 What is Equivalence partitioning testing?

Equivalence partitioning is a black-box testing technique where input data is divided into groups (partitions) based on expected behavior

Example:

Let's say you're testing a login form with a username field. You could create the following partitions:

* **Valid Input:** Alphanumeric strings (e.g., "user123")
* **Invalid Input (Empty):** Empty string (e.g., "")
* **Invalid Input (Special Characters):** String with special characters (e.g., "@#$")
* **Invalid Input (Spaces):** String with spaces (e.g., "user name")

 What is Integration testing?

Integration testing in software development verifies how different components or modules of a software application work together

 What determines the level of risk?

The level of risk is determined by combining the likelihood of an event occurring and the potential impact or severity of that event. Essentially, it's a measure of how probable something bad is to happen, and how damaging it would be if it does.

Here's a more detailed breakdown:

1. Likelihood (Probability):

* This refers to how likely the risk event is to actually happen.
* It's often expressed as a percentage or a qualitative term like "low," "moderate," or "high".
* Examples: A data breach, while potentially devastating, might be considered a low-probability event for a company with robust security measures.

2. Impact (Severity):

* This refers to the consequences or damages that would result from the risk event occurring.
* It can be measured by considering factors like financial loss, damage to reputation, physical harm, etc.
* Examples: A severe data breach could have a high impact, leading to significant financial losses, legal repercussions, and loss of customer trust.

 What is Alpha testing?

Alpha testing is a crucial stage in software development where the internal team rigorously tests the product, typically in a controlled environment, to identify and resolve bugs before it's released to external users.

Alpha Testing is done within the organization, while Beta Testing is done in the user's environment

 What is beta testing?

beta testing in software testing is a pre-release phase where a select group of real users, outside of the development team, test the software in a real-world environment before its official launch

 What is component testing?

Component testing, also known as module or program testing, is a software testing method where individual components or modules of a software system are tested independently, without integrating them with other parts of the system.

 What is functional system testing?

Functional system testing is a type of software testing that verifies if a system functions according to its specified requirements and expected behavior

 What is Non-Functional Testing?

Non-functional testing focuses on evaluating a software system's performance, reliability, security, usability, and other non-functional aspects rather than its core features

 What is GUI Testing?

GUI testing, also known as UI testing, focuses on verifying the functionality and usability of a software application's graphical user interface (GUI). This includes ensuring that elements like buttons, menus, and text boxes work as expected, and that the overall design is user-friendly. GUI testing aims to ensure a smooth and positive user experience

 What is Adhoc testing?

Ad hoc testing is an unstructured, informal type of software testing where testers explore the application randomly, without a formal test plan or predefined steps. It relies on the tester's intuition, experience, and creativity to identify defects and issues that more formal testing methods might miss. Ad hoc testing is often used after formal testing to find any lingering problems or unexpected behavior.

 What is load testing?

Load testing is a type of performance testing that simulates real-world user traffic to see how a system or application performs under different levels of load. It helps identify performance bottlenecks, response times, and system limitations before they affect real users. The goal is to ensure the system remains stable and efficient, even during peak usage

 What is stress Testing?

Stress testing is a form of performance testing that pushes a system, device, or software application beyond its normal operational capacity to identify weaknesses and vulnerabilities. The goal is to determine the system's breaking point, assess its robustness, and evaluate its ability to handle extreme or abnormal conditions. This helps developers ensure the system can recover gracefully from failures and maintain stability under stress.

 What is white box testing and list the types of white box testing?

White box testing, also known as glass box or clear box testing, is a software testing method where the tester has knowledge of the internal structure and workings of the software. It involves testing the code, logic, and data flow to ensure the software functions as expected.

Types of White Box Testing:

* **Unit Testing:** Testing individual units or components of the software in isolation.
* **Integration Testing:** Verifying the interactions between integrated units or components.
* **Regression Testing:** Ensuring that recent code changes haven't negatively impacted existing functionality.
* **Mutation Testing:** Evaluating test case effectiveness by introducing small changes (mutations) to the code.
* **Static Code Analysis:** Analyzing the code without executing it to identify potential issues.
* **Dynamic Code Analysis:** Analyzing the code while it's running to observe its behavior.
* **Statement Coverage:** Verifying that each statement in the code is executed at least once during testing.
* **Branch Coverage:** Ensuring that all branches of the code (e.g., if-else statements) are executed at least once.
* **Path Testing:** Verifying that all possible paths through the code are executed.
* **Loop Testing:** Focusing on checking the validity of loop constructs in the code.
* **Security Testing:** Identifying vulnerabilities and weaknesses in the software.
* **Memory Leak Testing:** Checking for memory leaks that can cause performance issues.

 What is black box testing? What are the different black box testing techniques?

Black box testing is a software testing method where testers evaluate functionality without knowing the internal code structure, algorithms, or implementation details. It focuses on the external behavior of the software by examining inputs, outputs, and responses to user actions, treating the software as a "black box". Different black box testing techniques include equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.

Elaboration:

* **Black Box Testing:**

Testers treat the software as a "black box," meaning they don't have access to the internal workings or code.

* **Focus on External Behavior:**

The main focus is on verifying that the software behaves as expected based on user inputs, outputs, and responses.

* **No Internal Knowledge Required:**

Testers don't need to understand the code or implementation details, making it accessible to testers with varying levels of technical expertise.

* **Testing Techniques:**
  + **Equivalence Partitioning:** Dividing inputs into categories (e.g., valid, invalid) to test the software's behavior within each category.
  + **Boundary Value Analysis:** Focusing on the boundaries of valid and invalid input ranges to identify potential errors.
  + **Decision Table Testing:** Using decision tables to create test cases based on different conditions and their corresponding actions.
  + **State Transition Testing:** Testing the software's behavior as it transitions between different states, ensuring it handles transitions correctly.

 Mention what are the categories of defects?

Defects can be categorized by their nature (functional, usability, performance, etc.), severity (critical, major, minor), and priority (urgent, high, medium, low). Common defect types include functional defects, performance defects, usability defects, security defects, and compatibility defects.

Defect Categories by Nature:

* **Functional Defects:** Failures to meet the specified requirements or functions.
* **Performance Defects:** Issues related to the speed, efficiency, or responsiveness of the software or product.
* **Usability Defects:** Problems with the user interface, navigation, or ease of use.
* **Security Defects:** Vulnerabilities that could be exploited to cause harm.
* **Compatibility Defects:** Issues with the software or product's ability to run on different environments, operating systems, or devices.
* **Interface Defects:** Problems with the user interface (UI) such as broken buttons, incorrect layouts, or navigation issues.
* **Design Defects:** Flaws in the product's design that make it inherently dangerous or unreliable.
* **Manufacturing Defects:** Errors that occur during the production process, leading to defects in the final product.
* **Labeling Defects:** Problems with the product's labels or instructions, such as inadequate warnings or unclear information.

 Mention what bigbang testing is?

Big Bang integration testing is a method where all modules of a software system are integrated and tested simultaneously, rather than individually. This approach is often used when all modules are ready and can be combined without needing stubs or mockups to simulate incomplete components.

 What is the purpose of exit criteria?

Exit criteria in software testing define the conditions that must be met before a testing phase or process is considered complete. They serve as a benchmark for ensuring that sufficient testing has been done, and that the software is ready for its next stage, such as release or moving to the next testing phase

 When should "Regression Testing" be performed?

Regression testing should be performed whenever there are changes to the software codebase, including adding new features, fixing bugs, or making performance improvements. It's also crucial before releases and after bug fixes to ensure that existing functionalities are not negatively affected by the changes.

 What is 7 key principles? Explain in detail?

The 7 key principles of software testing, outlined by the International Software Testing Qualifications Board (ISTQB), guide effective testing strategies and robust software development. These principles emphasize the importance of early defect detection, risk-based testing, defect clustering awareness, test diversification, context-dependent testing, and the need for comprehensive testing.

Here's a breakdown of each principle:

1. **1. Testing shows the presence of defects, not their absence:**

Testing can demonstrate that defects exist in the software, but it cannot prove that there are no defects. The goal of testing is to find and fix defects, not to guarantee a bug-free product.

1. **2. Exhaustive testing is impossible:**

Testing every possible input and condition is impractical and unrealistic, especially for complex systems.

1. **3. Early testing saves time and money:**

Finding defects early in the software development lifecycle (SDLC) is more efficient and cost-effective than addressing them later.

1. **4. Defects cluster together:**

A small number of modules or areas often contain a disproportionate number of defects.

1. **5.  pesticide paradox:**

Repeatedly executing the same test cases may not uncover new defects as testers may become accustomed to the system and lose their "fresh" perspective, or the software may have already become robust against the existing test cases,

1. **6. Testing is context-dependent:**

The specific testing approach and techniques used depend on the software being tested, the target audience, and the business context.

1. **7.  the absence-of-errors fallacy:**

Even if a software product is highly tested and has no known defects, it may still fail to meet user needs or expectations.

 Difference between QA v/s QC v/s Tester

In the context of quality management, QA comes first because it involves establishing processes and systems to prevent defects from occurring, while QC comes after to detect and correct defects that may have occurred despite the QA efforts and testing is a subset of QC used to detect defects and verify functionality.

 Difference between Smoke and Sanity?

In the context of software testing, "smoke testing" and "sanity testing" are distinct types of testing, both focusing on the initial stability of a build, but with different scopes. Smoke testing verifies the basic functionality of the application to ensure it's stable enough for further testing, while sanity testing focuses on validating specific changes or fixes within a stable build to ensure they haven't introduced new issues

 Difference between verification and Validation

Verification and validation are two distinct processes in software development that ensure quality. Verification focuses on whether the product is being built correctly, while validation focuses on whether the right product is being built. Verification is a static process that checks the internal aspects like design and code against requirements, while validation is a dynamic process that checks the end product against user needs and expectations through actual execution

 Explain types of Performance testing.

**Types of Performance Testing**

There are primarily six types of performance testing in software testing, which are explained below.

* **Load testing –** checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* [**Stress testing**](https://www.guru99.com/stress-testing-tutorial.html)**–** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing –** tests the software’s reaction to sudden large spikes in the load generated by users.
* **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
* **Scalability testing**– The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

 What is Error, Defect, Bug and failure?

In software development, a defect (or bug) is a flaw in the software that causes it to deviate from its intended behavior. A bug is a more informal term for a defect, often used when a tester or user reports an issue. Errors are mistakes made by developers during coding, and failures occur when the software doesn't perform as expected due to a bug or other issue.

 What is Bug Life Cycle?

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the [software testing](https://www.guru99.com/software-testing-introduction-importance.html) remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

 Explain the difference between Functional testing and NonFunctional testing

Key Differences:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Functional Testing** | **Non-Functional Testing** |
| Focus | Functionality, features, and user interactions | Performance, reliability, security, usability, etc. |
| Goal | Meets specified functional requirements | Meets user expectations in terms of quality |
| Test Cases | Based on functional requirements | Based on non-functional requirements |
| Timing | Often conducted before non-functional testing | Often conducted after functional testing |
| Examples | Unit testing, integration testing | Performance testing, security testing |
| Scope | Verifies if the software "does what it's supposed to" | Verifies how well the software "does it" |

To create HLR & TestCase of

1)(Instagram , Facebook) first page and

chat functionality

2) Facebook Login Page

**High-Level Requirements (HLR)**

**A. Instagram First Page**

**HLR:**

* Display login and signup forms.
* Option to log in with Facebook.
* Instagram branding, carousel of app images.
* Links for app downloads (App Store & Google Play).
* Language selection.

**B. Facebook First Page**

**HLR:**

* Display login form (Email/Phone & Password).
* “Create new account” option.
* “Forgotten password?” link.
* Promotional text and images.
* Footer links (About, Careers, Privacy, etc.)
* Language selection.

**C. Instagram & Facebook Chat Functionality (post-login)**

**HLR:**

* View chat list with recent messages.
* Real-time sending and receiving of messages.
* Chat search functionality.
* Emojis, GIFs, media support (images, videos).
* Read receipts and typing indicators.
* Video/audio calling (Facebook Messenger).
* Message reactions and replies.

**Test Case for Instagram First Page**

| **TC ID** | **Test Case Description** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| IG\_TC\_001 | Verify login with valid credentials | Enter username/password → Click Login | User logged in |
| IG\_TC\_002 | Verify signup link | Click “Sign up” | Redirects to registration |
| IG\_TC\_003 | Verify Facebook login | Click “Log in with Facebook” | Redirects to Facebook login |
| IG\_TC\_004 | App download buttons | Click iOS/Android icons | Redirects to app store |
| IG\_TC\_005 | Language selector | Change language | Page language updates |

**Test Case for Facebook First Page**

| **TC ID** | **Test Case Description** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| FB\_TC\_001 | Verify login with valid credentials | Enter Email/Phone & Password → Login | User redirected to home |
| FB\_TC\_002 | Verify invalid login attempt | Enter wrong credentials | Shows error |
| FB\_TC\_003 | Forgot password functionality | Click “Forgotten password?” | Redirects to recovery page |
| FB\_TC\_004 | Verify “Create New Account” button | Click on button | Opens registration form |
| FB\_TC\_005 | Change display language | Select different language | Page updates accordingly |

**Test case for Chat Functionality(Instagram & facebook)**

| **TC ID** | **Test Case Description** | **Steps** | **Expected Result** |
| --- | --- | --- | --- |
| CHAT\_TC\_001 | Send and receive a text message | Open chat → Send message | Message is sent and appears in chat |
| CHAT\_TC\_002 | Send emoji or sticker | Use emoji/sticker panel | Appears in chat |
| CHAT\_TC\_003 | Send media (image/video) | Click attach → Choose file | Media appears in chat |
| CHAT\_TC\_004 | Real-time message delivery | Two users send/receive | Messages sync instantly |
| CHAT\_TC\_005 | Typing indicator | Start typing in chat | Other user sees “Typing…” |
| CHAT\_TC\_006 | Read receipt | Send message → Wait for read | Shows double-tick or “Seen” |
| CHAT\_TC\_007 | Start voice/video call | Click call icon | Call initiates |
| CHAT\_TC\_008 | Search chats | Type keyword in search | Matching messages or contacts appear |
| CHAT\_TC\_009 | React to a message | Hover on message → React | Reaction is shown below message |

 What is the difference between the STLC (Software Testing Life Cycle) and SDLC(Software Development Life Cycle)?

The main difference between STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle) is that STLC is a part of the SDLC, focusing specifically on the testing phase to ensure software quality. SDLC encompasses all stages of software development, from planning and design to deployment and maintenance. STLC, on the other hand, focuses on the process of verifying and validating that the software meets the required quality standards and is free of defects.

 What is the difference between test scenarios, test cases, and test script?

A test case is a document with instructions on testing the specific functionality of an application. Test Script is a program that runs various test data on the functionality of an application. Test scenarios serve as an outline for writing test cases.

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

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”

 What is priority?

In software testing, priority refers to the urgency or order in which a bug should be fixed. It indicates how soon a bug needs to be addressed, based on its impact on the business and the overall software product. Priority is often determined by factors like severity, user impact, and business objectives.

Levels: Urgent, High, Medium, Low

 What is severity?

In software testing, severity refers to the impact a bug or defect has on the functionality or performance of the application. It's essentially a measure of how critical the bug is and how much it affects the user experience or business objectives. Severity helps determine the order in which bugs should be addressed

 Severity Categories:

* **Critical (S1):** A bug that prevents the application from functioning or renders core features unusable.
* **Severe (S2):** A bug that significantly affects the performance or functionality of the application.
* **Major (S3):** A bug that impacts functionality, but users may be able to work around it.
* **Minor (S4):** A bug that has minimal impact on the application's functionality.
* **Trivial (S5):** A bug that has a very small or negligible impact on the application's functionality.

 Bug categories are…

Three main types of bugs in software testing are functional bugs, logical bugs, and performance bugs. Functional bugs affect the intended functionality of the software, logical bugs disrupt the program's workflow due to incorrect reasoning, and performance bugs relate to issues with speed, stability, and resource consumption.

Here's a more detailed breakdown:

1. Functional Bugs: These bugs relate to the functionality of a specific software component. For example, a button might not submit a form, or a search function might not respond to user input.

2. Logical Bugs: These bugs disrupt the intended workflow of the software and cause it to behave incorrectly. They often arise from misinterpreted logic in the code.

3. Performance Bugs: These bugs affect the speed, responsiveness, and stability of the software. They can lead to delays, lags, and other issues that negatively impact the user experience.

 Advantage of Bugzila .

Bugzilla's main advantage is being a robust and free, open-source bug tracking system that allows developers and testers to effectively manage and track issues in their projects. It offers features like effective bug tracking, automation, and robust reporting, making it a valuable tool for software development. Bugzilla also facilitates communication and collaboration among team members, and its scalable architecture can handle projects of various sizes

 Difference between priority and severity

In bug tracking and software testing, priority determines how urgently a bug should be addressed, while severity indicates the impact or seriousness of the bug on the system. Priority considers factors like business impact and deadlines, while severity focuses on the technical impact on the software's functionality or user experience

 What are the different Methodologies in Agile Development Model?

Agile development is a flexible approach to software development that emphasizes iterative development, collaboration, and customer feedback. Several methodologies fall under the Agile umbrella, each with its own unique approach to managing projects and delivering value. Key Agile methodologies include Scrum, Kanban, Extreme Programming (XP), Lean Software Development, Feature-Driven Development (FDD), and Crystal.

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

In Web testing, authentication verifies a user's identity (who they are), while authorization determines what they can access (what they are allowed to do). Common problems in Web testing include cross-browser/device compatibility, responsiveness, and security issues like weak authentication/authorization.

Elaboration:

* **Authentication:**

This process confirms a user's identity, typically through a login with credentials like a username and password. It ensures that the person accessing the system is who they claim to be.

* **Authorization:**

Once a user is authenticated, authorization checks their permissions and determines what specific resources or actions they are allowed to access. This is based on their role or assigned access rights.

* **Example:**

Imagine logging into a web application. Authentication verifies that you are the user you claim to be based on your username and password. Authorization then determines if you have permission to view your profile, access specific data, or perform certain actions within the application.

* **Common Problems in Web Testing:**
  + **Cross-Browser and Device Compatibility:** Ensuring a website functions correctly across different browsers (Chrome, Firefox, Safari, etc.) and devices (desktops, tablets, smartphones) is a major challenge.
  + **Responsiveness:** Web applications must adapt seamlessly to various screen sizes and resolutions, which can be tricky to test.
  + **Security Issues:** Problems related to weak authentication and authorization mechanisms, such as easily guessable passwords or inadequate access control, can lead to security vulnerabilities. Other security challenges include cross-site scripting (XSS), SQL injection, and session hijacking.
  + **Performance Issues:** Slow loading times, unresponsive interactions, and other performance bottlenecks can negatively impact the user experience and require careful testing.
  + **Usability:** Ensuring the website is easy to navigate, understand, and use effectively is crucial for a positive user experience and requires thorough usability testing.
  + **Integration Testing:** Testing the interactions between different components of a web application, including the server, database, and third-party services, can be complex.

**Write a Scenario of Pen**

**Scenario: Verify the Functionality of a Ballpoint Pen**

**Title: Writing with the Pen**

**Objective:**

To ensure the pen writes smoothly and consistently on a standard piece of paper.

**Preconditions:**

* The pen is new or has ink.
* Standard A4 paper is available.
* The testing environment has sufficient lighting.

**Test Steps:**

1. Remove the pen cap or click to expose the tip.
2. Hold the pen in a standard writing position.
3. Write a few lines on A4 paper using moderate pressure.
4. Check ink consistency and line quality.
5. Try writing at different angles (vertical, slanted).
6. Leave the pen uncapped for 5 minutes and test again.
7. Attempt to write on different surfaces (e.g., cardboard, glossy paper).

**Expected Results:**

* Pen starts writing immediately.
* Ink flow is smooth and uninterrupted.
* No blotting or smudging.
* Writing remains legible on all tested surfaces.
* Pen resumes writing even after short idle periods.

**Write a Scenario of Pen Stand**

**Test Scenario: Verify the Usability of a Pen Stand**

**Title: Storing and Retrieving Pens from the Pen Stand**

**Objective:**

To verify that the pen stand can hold multiple pens securely and allows easy access without damage.

**Preconditions:**

* Pen stand is placed on a flat, stable surface.
* Multiple standard-sized pens and markers are available.

**Test Steps:**

1. Place the pen stand on a desk.
2. Insert 5–10 pens, one by one.
3. Check for stability after each addition.
4. Remove and insert pens repeatedly to check ease of use.

**Expected Results:**

* Pen stand holds all items without tipping over.
* Pens stay upright and organized.
* No damage or scratches on pens from contact with stand.

Write a Scenario of Door

**Test Scenario: Verify the Opening and Closing Door**

**Title: Functional Test of Door Opening and Closing**

**Objective:**

To ensure the door opens and closes smoothly, securely, and safely under normal usage conditions.

**Preconditions:**

* Door is properly installed .
* Door handle and lock are in place.

**Test Steps:**

1. Attempt to open the door by pulling the handle.
2. Fully open the door to check range and ease.
3. Close the door with normal force.
4. Try opening and closing the door from both sides.
5. Lock and unlock the door
6. Apply slight pressure to check for Open door or close.
7. verify the gap between door and frame for alignment.

**Expected Results:**

* Door opens and closes smoothly with minimal force.
* Lock works properly.
* No unusual gaps or misalignment.
* Handle/knob works properly and Smoothly

Write a Scenario of ATM

**Test Scenario: Cash Withdrawal from an ATM**

**Title: Verify Successful Cash Withdrawal from an ATM**

**Objective:**

To ensure users can successfully withdraw cash using their ATM/debit card

**Preconditions:**

* User has a valid ATM/debit card.
* Sufficient balance is available in the account.
* ATM machine is powered on and functioning.
* ATM has cash available for dispensing.

**Test Steps:**

1. Insert ATM/debit card into the machine.
2. Select preferred language (if prompted).
3. Enter correct 4-digit PIN.
4. Choose "Withdrawal" as the transaction type.
5. Select the account type (e.g., Savings or Current).
6. Enter the withdrawal amount
7. Confirm the amount and wait for processing.
8. Collect cash from the dispenser.
9. Collect printed receipt .
10. Retrieve ATM card when prompted.

**Expected Results:**

* Card is read correctly.
* User is prompted for PIN and transaction options.
* Transaction is processed without error.
* Exact cash amount is dispensed.
* ATM screen displays “Thank You” or confirmation message.

 When to used Usablity Testing?

Usability testing should be conducted throughout the entire product lifecycle, from early conceptual stages to post-launch iterations. It's crucial for identifying and addressing usability issues early on, improving the user experience, and optimizing product functionality.

 What is the procedure for GUI Testing?

GUI testing is a software testing procedure focused on verifying the functionality, layout, and overall user experience of a Graphical User Interface (GUI). It ensures the interface behaves as expected, looks visually appealing, and is user-friendly across different platforms and devices.

Here's a breakdown of the procedure:

1. Planning and Requirements:

* **Define Scope:**

Determine the scope of GUI testing, including which features, modules, and platforms will be covered.

* **Develop Test Strategy:**

Outline the testing approach, including manual and automated testing techniques, and prioritization of test cases.

* **Gather Requirements:**

Collect all necessary information about the GUI, including its design specifications, functionality requirements, and user interface guidelines.

2. Test Case Design:

* **Create Test Cases:**

Design detailed test cases that cover various aspects of the GUI, such as layout, functionality, usability, and compatibility.

* **Include Different Scenarios:**

Develop test cases for both positive (valid data) and negative (invalid data) scenarios.

* **Consider Edge Cases:**

Test boundary conditions, error conditions, and other edge cases that may not be explicitly covered in requirements.

* **Prioritize Test Cases:**

Prioritize test cases based on their importance, complexity, and susceptibility to defects.

3. Test Execution:

* **Perform Manual Testing:**

Conduct manual testing to evaluate the GUI's layout, design, and overall user experience.

* **Utilize Automated Tools:**

Use automated testing tools to simulate user interactions and identify bugs or issues.

* **Document Findings:**

Thoroughly document all test results, including any defects or issues identified, and provide details about the test conditions and expected results.

4. Defect Reporting and Tracking:

* **Report Defects:** Report any defects found during testing to the development team.
* **Track Defects:** Track the status of reported defects and ensure they are addressed and resolved.
* **Verify Fixes:** Verify that reported defects have been fixed and that the GUI behaves as expected.

5. Iteration and Improvement:

* **Review and Analyze:**

Review and analyze the results of each testing round and identify areas for improvement.

* **Iterate on Tests:**

Continue to iterate on the test cases and testing process to ensure that the GUI meets the required quality standards.

* **Continuous Testing:**

Implement continuous GUI testing as part of the software development lifecycle to ensure that the GUI remains functional and user-friendly over time

Write a scenario of Microwave Owen

**Scenario: Microwave Oven**

**Title: Verify the Basic Heating Function of the Microwave Oven**

**Objective:**

To ensure the microwave oven heats food properly, operates safely, and responds accurately to user input.

**Preconditions:**

* Microwave is plugged in and switched on.
* A microwave-safe container with food (e.g., a bowl of rice or soup) is ready for use.
* The microwave door is in proper working condition.

**Test Steps:**

1. Open the microwave door.
2. Place the container with food inside the oven.
3. Close the door securely.
4. Set the heating time (e.g., 2 minutes) and power level (e.g., Medium/High).
5. Press the "Start" button.
6. Observe whether the microwave begins operation (light on, plate rotating, humming sound).
7. Wait for the timer to count down and microwave to stop automatically.
8. Carefully open the door and check the food temperature.
9. Remove the container using caution (heat protection if necessary).

**Expected Results:**

* Microwave starts immediately after pressing "Start."
* Timer counts down correctly.
* Food is hot .
* No sparks.
* Microwave stops exactly when the timer ends.
* Door can be opened safely after use

Write a scenario of Coffee vending Machine

**Test Scenario: Coffee from a Vending Machine**

**Title: Verify Successful Coffee Dispensing from Vending Machine**

**Objective:**

To ensure the coffee vending machine dispenses coffee correctly upon user selection and payment.

**Preconditions:**

* Machine is powered on and in service.
* Coffee ingredients (powder, water, milk, sugar) are sufficiently loaded.
* Disposable cups are loaded into the machine.
* User has coins, tokens, or a card for payment.

**Test Steps:**

1. Check the display screen for “Ready” status.
2. Insert the required coins/token/card into the payment slot.
3. Wait for the machine to acknowledge payment.
4. Select “Coffee” from the menu (could be “Espresso”, “Black”, “With milk” etc.).
5. Wait for the machine to start the dispensing process.
6. Observe cup dropping (if automated) and coffee being poured.
7. Collect the filled cup from the dispenser.
8. Taste-test (optional) to verify correct sugar and milk levels.

**Expected Results:**

* Machine accepts the payment and confirms it.
* Coffee is dispensed into a clean cup.
* No machine error occurs.
* Coffee is warm and tastes as per selected option.
* Machine returns change (if applicable).

Write a scenario of chair

**Test Scenario: Sitting on a Chair**

**Title: Verify the Stability and Comfort of a Standard Chair**

**Objective:**

To ensure the chair is stable, safe, and comfortable when used by an average adult for normal sitting purposes.

**Preconditions:**

* Chair is placed on a flat, stable surface.
* No visible damage or missing parts (legs, backrest, screws, etc.).
* Weight capacity is known (e.g., 100 kg).

**Test Steps:**

1. Visually inspect the chair for any structural damage.
2. Apply slight pressure to the seat with your hand to check balance.
3. Sit on the chair slowly, using normal.
4. Shift your weight slightly side-to-side and lean back gently.
5. Sit for 5–10 minutes to assess comfort.
6. Stand up from the chair and observe stability during the motion.

**Expected Results:**

* Chair remains stable with no wobbling or tilting.
* Legs and joints do not creak or flex under weight.
* Backrest provides proper support.
* Seat cushion (if any) retains shape and comfort.
* Chair returns to original state after standing up.
* No injuries while sitting or standing.

To Create Scenario (Positive & Negative)

**Test Scenarios for a Chair**

**Positive Test Scenarios (Expected behavior)**

| **Scenario ID** | **Scenario Description** |
| --- | --- |
| POS\_001 | Verify that a person can sit on the chair without it wobbling. |
| POS\_002 | Verify the chair supports the stated maximum weight (e.g., 120 kg). |
| POS\_003 | Verify that all four legs rest evenly on a flat surface. |
| POS\_004 | Verify the chair is stable when a person stands up from it. |
|  |  |
| POS\_005 | Verify the chair’s material (metal/wood/plastic) doesn’t get damaged under normal use. |
|  |  |

**Negative Test Scenarios (Unexpected or failure behavior)**

| **Scenario ID** | **Scenario Description** |
| --- | --- |
| NEG\_001 | Sit on the chair with weight exceeding the max limit |
| NEG\_002 | Sit on the chair with one leg resting on an. |
| NEG\_003 | Remove one screw or bolt from the leg — check structural safety. |
|  |  |
|  |  |

Write a Scenario of Wrist Watch

**Test Scenario: Display Functionality of a Wrist Watch**

**Title: Verify that the wrist watch displays the correct time**

**Objective:**

To ensure the wrist watch shows accurate time and remains functional over a period of use.

**Preconditions:**

* The wrist watch is fully charged (if digital/smart)
* Time is set accurately before starting the test.
* The watch is worn on the wrist under normal conditions.

**Test Steps:**

1. Set the watch to the current time.
2. Leave the watch running for 24 hours under normal usage.
3. Check the time displayed on the watch after 24 hours.
4. Compare it with the actual time using a reliable source.
5. Observe the second/minute/hour hand movement (for analog).
6. If digital/smart, check if the time auto-syncs

**Expected Results:**

* The watch should display the correct time after 24 hours (± a small margin).
* No delay, fast-forward in the movement.
* Time format (12hr/24hr) should match the chosen setting (for digital watches).
* Watch continues to function without interruptions.

Write a Scenario of Lift(Elevator)

**Test Scenario: Verify the Functionality of the Elevator (Lift)**

**Title: Verify whether the elevator is functioning correctly or not.**

**Objective:**

To ensure that the elevator is operational and performs all required functions, including movement, door operation, and responding to button presses.

**Preconditions:**

* The elevator is powered on.
* The elevator control panel is working.
* The building has a valid maintenance check record for the elevator.

**Test Steps:**

1. **Press the Call Button:**
2. **Verify Elevator Arrival:**
3. **Check Door Functionality:**
4. **Check Elevator Movement:**
5. **Floor Arrival and Door Opening:**
6. **Verify Button Functionality:**
7. **Check for Error Alerts or Malfunctions:**
8. **Overload Test:**

**Expected Results:**

* The elevator should arrive promptly at the requested floor when the call button is pressed.
* The doors should open and close smoothly without obstruction.
* The elevator should move to the selected floor without any jerky or irregular movement.
* The elevator should stop at the correct floor as indicated by the control panel and open its doors.
* All floor selection buttons should function correctly, with no skipping of floors or failure to respond.
* Error or malfunction alerts should trigger when there is a failure, and the system should respond appropriately (e.g., emergency stop function).
* The elevator should prevent movement if the weight capacity is exceeded and notify the users.

Write a Scenario of Whatsapp payment

**Test Scenario: WhatsApp Payment**

**Title: Verify the WhatsApp payment functionality for sending and receiving money**

**Objective:**

To ensure that the WhatsApp payment feature is functioning correctly for both sending and receiving money, including integration with the linked bank account, security features, and transaction accuracy.

**Preconditions:**

* WhatsApp is installed and running on a mobile device.
* User has linked a valid bank account or payment method (e.g., UPI, debit/credit card).
* The recipient also has WhatsApp payment activated and linked to a valid bank account.
* User has sufficient funds in their linked payment method.

**Test Steps:**

1. **Check WhatsApp Payment Setup:**
2. **Initiate Payment:**
3. **Verify Payment Confirmation:**
4. **Check Payment Receipt:**
5. **Test Payment Failure (Negative Scenario):**
6. **Verify Transaction History:**
7. **Check Security (OTP/Authentication):**
8. **Test Multiple Payment Methods:**
9. **Test Refund/Transaction Cancellation (Optional):**

**Expected Results:**

* Payment should be initiated and completed successfully when there are sufficient funds.
* The recipient should receive the payment notification and see the updated balance.
* Payment failure scenarios (such as insufficient funds) should trigger appropriate error messages.
* The transaction history should show accurate details of both sent and received payments.
* Security features (authentication via PIN/fingerprint) should function as expected to protect transactions.
* In case of an issue, the system should inform the user and provide guidance on resolving the problem.
* Payments should be processed and displayed accurately without any delay or errors.