***Name: Tabammum Haque Pranty***

***SQA Assignment-03***

**Answer to the Ques No: 01**

**Boundary Value Analysis (BVA):** Boundary Value Analysis (BVA) is a testing technique used to identify defects at the boundaries of input ranges rather than within the range itself.

For Example: Consider a system that accepts numbers between 1 and 100 as input. The boundary values here are:

Lower boundary: 1

Upper boundary: 100

Using BVA, we would test:

Just below the lower boundary: 0

At the lower boundary: 1

Just above the lower boundary: 2

Just below the upper boundary: 99

At the upper boundary: 100

Just above the upper boundary: 101

|  |  |  |
| --- | --- | --- |
| Invalid (min-1) | Valid (min, min+1, nominal, max-1, max) | Invalid (max+1) |
| 0 | 1, 2, 50, 99, 100 | 101 |

**Table-1:** Boundary Value Analysis (BVA)

**Equivalence Class Partitioning (ECP):** Equivalence Class Partitioning (ECP) is a method used to divide input data into partitions or classes where the system is expected to behave similarly. The idea is that testing one condition within a class is enough to represent all other conditions in that class.

For Example: Using the same range of input (1–100), we can divide this into equivalence classes:

Invalid Class (below 1): [-∞ to 0]

Valid Class: [1 to 100]

Invalid Class (above 100): [101 to ∞]

For testing, we can choose one representative from each class:

Invalid input: 0 (from the invalid class below 1)

Valid input: 50 (from the valid class between 1 and 100)

Invalid input: 101 (from the invalid class above 100)

|  |  |  |
| --- | --- | --- |
| Invalid | Valid | Invalid |
| 0<1 | 1-100 (50,45 | 101>100 |

**Table-2:** Equivalence Class Partitioning (ECP)

**State Transition Testing:** State Transition Testing is a technique used for systems where outputs depend on the sequence of inputs and the current state of the system. It evaluates how the system transitions from one state to another depending on the input and predefined rules.

For Example: Consider a simple login system with the following states:

State 1: User is logged out.

State 2: User enters correct credentials and logs in.

State 3: User enters incorrect credentials and remains logged out.

Possible transitions:

From State 1 to State 2: User enters correct username and password.

From State 1 to State 3: User enters incorrect credentials.

**Decision Table Based Testing:** Decision Table Based Testing is a technique that uses a decision table to map various combinations of inputs and their respective outputs.

For example: Consider a discount system for an online store where:

Condition 1: Is the customer a member? (Yes/No)

Condition 2: Is the purchase amount greater than 10000 BDT? (Yes/No)

|  |  |  |
| --- | --- | --- |
| Condition 1 (Member) | Condition 2 (Amount > 10000) | Discount (%) |
| Yes | Yes | 20% |
| Yes | No | 10% |
| No | Yes | 5% |
| No | No | 0% |

**Table-3:** Decision Table Based Testing

**Answer to the Ques. No: 02:**

**Manual Testing:** Manual Testing is the process of manually executing test cases without using any automation tools. Testers play the role of end-users and verify that the software behaves as expected by following predefined test steps.

**For Example:** Consider a web application for an e-commerce platform. In manual testing, a tester would:

- Open the web application.

- Manually navigate through the site to check for UI responsiveness, correct product details, and proper checkout functionality.

- Verify if a discount is correctly applied during checkout by testing with various discount codes.

- Check for error messages when invalid details are entered in the form.

**Automation Testing:** Automation Testing uses specialized tools and scripts to execute test cases without human intervention. The goal is to speed up testing by automating repetitive tasks.

**For Example:** In the same e-commerce web application, automation testing could involve writing scripts using an automation tool (e.g., Selenium) to:

- Automatically navigate through the website, add items to the cart, and proceed to checkout.

- Enter predefined discount codes and check if the correct discounts are applied.

- Simulate different user logins and place orders.

- Perform these tests repeatedly across multiple browsers and devices.

**Sanity Testing**: Sanity testing is a narrow regression test focused on validating specific functionality after changes or fixes. It determines whether a small section of the application still works correctly after modifications.

**Example:**  
If a bug is fixed in the user profile section of a web app, a sanity test would verify that the profile page loads correctly and the specific bug is resolved without thoroughly testing the rest of the application.

**Smoke Testing:** Smoke testing is a preliminary testing strategy performed on a new software build to ensure its basic functionality works before more rigorous testing occurs. It acts as a "sanity check" to see if the build is stable.

**Example:**  
For a mobile app, smoke testing might involve checking if the app opens correctly, if the login page loads, and if basic navigation works. If these tests fail, deeper testing is halted until the core functionality is fixed.

**Regression Testing**: Regression testing ensures that recent code changes have not adversely affected the existing functionality of the software. This strategy is vital after bug fixes, updates, or new features.

**Example:**  
If a developer fixes a bug in the search function of an application, regression tests ensure that the search still works and that other functionalities, like sorting and filtering, haven't been broken by the change.

**Acceptance Testing:** Acceptance testing is the final phase of testing, where the system is tested for acceptability. The goal is to verify whether the software meets business requirements and is ready for deployment.

**Example:**  
For a payroll system, acceptance testing might involve generating pay slips for a set of employees to ensure that calculations (salaries, deductions, bonuses) align with actual business rules.

**Risk-Based Testing:** Risk-Based Testing is a strategy where the testing effort is focused on areas of the application that pose the highest risk of failure. This involves identifying the most critical functions, modules, or features, and ensuring that these high-risk areas are thoroughly tested.

**Example:**  
In an e-commerce application, the payment gateway is crucial. A failure in this component can lead to significant financial losses. Therefore, risk-based testing would prioritize testing of the payment gateway, ensuring that all possible scenarios (like successful payment, failed payment, multiple payment options, etc.) are covered before moving on to less critical features, like product reviews or user profile settings.

**Usability Testing:** Usability Testing evaluates how easily users can interact with the application. It focuses on the user interface, navigation, and overall user experience. The goal is to ensure that the application is intuitive and user-friendly.

**Example:**  
Suppose a new mobile banking app is developed. Usability testing involves having users from the target audience (e.g., non-technical users) perform tasks like transferring money, checking balances, and viewing transaction history. Observers would note any difficulties, confusion, or delays encountered by the users and use that feedback to improve the app’s design and user interface.

**Exploratory Testing:** Exploratory Testing is an unscripted, dynamic testing approach where testers actively explore the system to find defects. Instead of following predefined test cases, testers use their experience and intuition to discover potential issues in the software.

**Example:**  
Imagine testing a social media platform where users can post, like, and comment on posts. Instead of sticking to a test script, a tester might explore by randomly posting images, deleting comments, blocking users, or uploading files to see if the application behaves unexpectedly.

**Incremental Testing:** Incremental Testing is a strategy where individual modules or components of a system are tested incrementally, one after the other. After testing each module, the next module is integrated and tested together with the previous ones.

**Example:**  
Consider a project management tool that includes separate modules for “Task Management,” “Time Tracking,” and “Report Generation.” In incremental testing:

* First, test the Task Management module in isolation.
* Then, integrate Time Tracking and test it with Task Management.
* Finally, add Report Generation to ensure it works correctly with the other two.

**Ad-hoc Testing / Monkey Testing:** Ad-hoc Testing is an informal, unstructured approach to testing where the tester explores the application without any specific plan or documentation. Monkey Testing is a specific form of ad-hoc testing where random inputs are given to the application to see how it behaves.

**Example(Ad-hocTesting):**  
In a file-sharing application, the tester may randomly upload files, delete them, and change sharing permissions without following any set guidelines. The goal is to find unexpected behavior or crashes by exploring the system’s limits.

**Example(MonkeyTesting):**  
In a mobile app, random touches, swipes, and inputs are made in a disorganized manner to see if the app crashes or behaves unexpectedly.

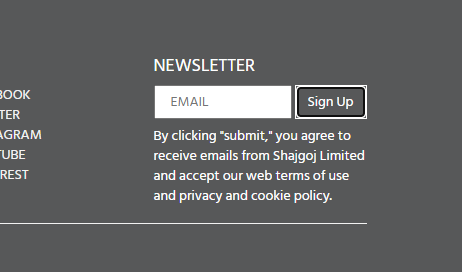
**A/B Testing:** A/B Testing is a strategy where two versions of a feature, interface, or design (Version A and Version B) are tested simultaneously with users to determine which one performs better based on predefined metrics like user engagement, conversion rates, or user satisfaction.

**Example:**  
A web-based marketing company might test two versions of a website's landing page:

* **Version A** has a simple design with minimal text.
* **Version B** has a more complex design with detailed information. A/B testing would be conducted by splitting the website traffic between the two versions and then measuring which version leads to higher customer signups.

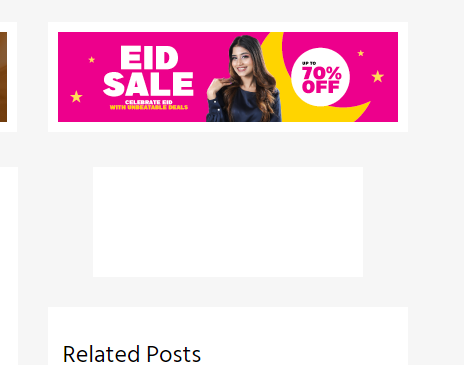
**Ans to the ques no: 03:**

**Bug Finding for a Website:**

****

In this picture, the signup button doesn’t work. It should be responsive.

**URL:** [**https://www.shajgoj.com/**](https://www.shajgoj.com/)

****

In this picture, the white box is not needed here.

**URL:** [**https://www.shajgoj.com/all-about-lactic-acid/**](https://www.shajgoj.com/all-about-lactic-acid/)