Q.1 Explain smoke and sanity testing. Also write advantages and disadvantages.

Smoke Testing

Definition:

Smoke testing is a preliminary test conducted to check whether the basic functionalities of a software build are working properly. It's often referred to as "build verification testing." The primary purpose is to ensure that the critical functionalities of the application work fine without going into deeper details.

Advantages:

1. Quick Identification of Issues: Detects major issues early in the development process.

2. Saves Time and Effort: Ensures that the build is stable enough for further testing, preventing testers from wasting time on faulty builds.

3. Simple and Fast: Requires minimal effort and time since it focuses on the critical paths of the application.

4. Helps Prevent Rework: Reduces the risk of finding basic issues later in the testing cycle, where it would require more time and effort to fix.

Disadvantages:

1. Not Comprehensive: Smoke tests do not cover all scenarios, only the critical parts of the application.

2. Misses Deep Bugs: May overlook issues in less critical or deeper parts of the application.

3. Manual in Some Cases: Although automated, smoke testing may sometimes require manual effort, especially in complex environments.

Sanity Testing

Definition:

Sanity testing is a subset of regression testing conducted when a small section of the software is modified. It verifies that the changes or fixes have worked properly and have not affected any other areas of the application. It ensures that the software is rational and stable after minor changes.

Advantages:

1. Quick Confirmation: Provides a fast check to validate that the recent code changes are functioning correctly.

2. Targets Specific Areas: Focuses on testing specific components affected by the recent changes, which saves time and resources.

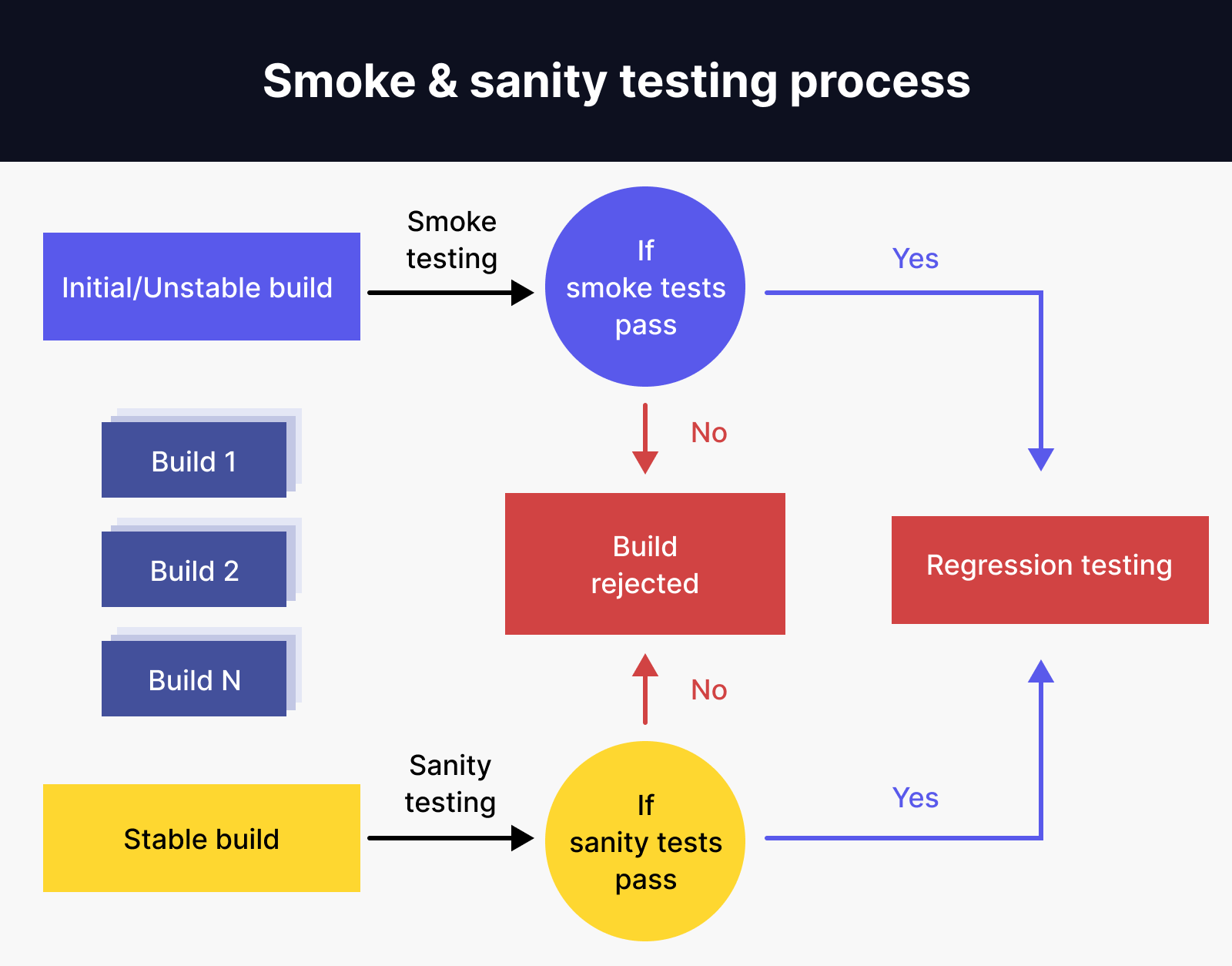
3. Prevents Major Failures: Prevents further defects from entering the application after updates or bug fixes.

Disadvantages:

1. Limited Coverage: Only tests the areas of the application that were changed, which may miss issues in other parts of the software.

2. Not Comprehensive: Similar to smoke testing, it does not perform a thorough check, as it's only concerned with the changes.

3. No Detailed Testing: It’s a high-level test, so it cannot be used to ensure that every functionality of the software works perfectly.



Q.2 Difference Between Functional and Non-Functional Testing

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Functional Testing** | **Non-functional Testing** | | --- | --- | | It verifies the operations and actions of an application. | It verifies the behavior of an application. | | It is based on requirements of customer. | It is based on expectations of customer. | | It helps to enhance the behavior of the application. | It helps to improve the performance of the application. | | Functional testing is easy to execute manually. | It is hard to execute non-functional testing manually. | | It tests what the product does. | It describes how the product does. | | Functional testing is based on the business requirement. | Non-functional testing is based on the performance requirement. | | **Examples:**  **1.** [Unit Testing](https://www.geeksforgeeks.org/unit-testing-software-testing/) **2.** [Smoke Testing](https://www.geeksforgeeks.org/smoke-testing-software-testing/) **3.** [Integration Testing](https://www.geeksforgeeks.org/software-engineering-integration-testing/) **4.** [Regression Testing](https://www.geeksforgeeks.org/software-engineering-regression-testing/) | **Examples:**  **1.** [Performance Testing](https://www.geeksforgeeks.org/performance-testing-software-testing/) **2.** [Load Testing](https://www.geeksforgeeks.org/software-testing-load-testing/) **3.** [Stress Testing](https://www.geeksforgeeks.org/stress-testing-software-testing/) **4.** [Scalability Testing](https://www.geeksforgeeks.org/software-testing-scalability-testing/) | |

Q.3 Define software quality assurance. Also explain SQA activities.

Software Quality Assurance (SQA) is a systematic process to ensure that the software being developed meets the required quality standards and works as expected. It involves monitoring the software development processes and methodologies to ensure they adhere to predefined standards, and it is aimed at preventing defects in the final product rather than just detecting them.

Key SQA Activities:

1. Requirements Gathering and Analysis

2. Test Planning

3. Code Reviews

4. Test Execution

5. Defect Tracking

6. Process Monitoring

7. Configuration Management

8. Risk Management

9. Audits and Reviews

Importance of SQA:

Prevention of Defects: By establishing processes to prevent defects, SQA reduces the time and cost associated with fixing issues later in the development process.

Improved Software Quality: Ensures that the software meets the desired quality standards, resulting in higher user satisfaction.

Consistency: Maintains a consistent approach to software development and testing, ensuring that each release meets the same high-quality standards.

Compliance: Ensures that the software and the development process comply with relevant regulations, standards, and policies.

Cost Savings: Identifying and addressing issues early in the development lifecycle helps save costs associated with rework and late-stage bug fixes.

Q.4 An OTP number which contains only 6 digits less or more than 6 digits will not have expected and the application will redirect the user to the error page. Using Equivalence partitioning write valid and invalid test cases.

Test Plan

1.Only 6-digit OTP number is allowed

2.No blank spaces are allowed

3.No alphabets are allowed

4.No special characters are allowed

UI

OTP :

Cancel

Ok

Test Case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Case Name | Purpose of Objective | Steps | Expected Results | Actual Results | Pass/Fail |
| 1 | OTP | To validate OTP | 1.More than 6-digits  2.Click ok | Success | Invalid | Fail |
| 2 | OTP | To validate OTP | 1.Less than 6-digits  2.Click ok | Success | Invalid | Fail |
| 3 | OTP | To validate OTP | 1.Alphabet instead of number  2.Click ok | Success | Invalid | Fail |
| 4 | OTP | To validate OTP | 1.Only 6-digts OTP  2.Click ok | Success | Success | Pass |
| 5 | OTP | To invalidate OTP | 1.Empty/Blank spaces  2.Click ok | Invalid | Invalid | Pass |
| 6 | OTP | To invalidate OTP | 1.Special Character  2.Click ok | Invalid | Invalid | Pass |

Q.5 What is Black Box Testing? Explain various Black Box Testing.

Black Box Testing

Definition:

Black Box Testing is a software testing method in which the tester evaluates the functionality of an application without having any knowledge of the internal workings, code, or structure. The tester interacts with the software through its user interface and checks if the outputs are correct based on the given inputs. The focus is on verifying what the software does rather than how it achieves that.

Key Characteristics of Black Box Testing:

1. No Knowledge of Code: Testers do not need to know the internal code, logic, or structure of the software.

2. Functional Testing: Primarily concerned with verifying that the software meets user requirements and functions correctly.

3. Test Inputs and Outputs: Tests are based on input-output relationships. The tester provides input and observes whether the output matches the expected result.

Types of Black Box Testing:

1. Functional Testing:

Functional testing ensures that the software behaves according to its functional specifications. It verifies individual features or functionalities of the software by providing input and validating the corresponding output.

2. Non-Functional Testing:

Non-functional testing evaluates aspects of the software that are not related to specific functionalities but are important for its overall quality. This includes performance, usability, reliability, and security.

3. Regression Testing:

Regression testing checks whether recent code changes or updates have not adversely affected existing functionality. It ensures that old code still works after new updates or bug fixes.

4. Boundary Value Testing (Boundary Value Analysis - BVA):

This technique focuses on testing at the boundaries between partitions. Boundary values are the extreme ends of the input domain, and these are the points where the software is most likely to fail.

5. Equivalence Partitioning:

Equivalence Partitioning divides the input data into different partitions or classes where all the values within each partition are expected to behave the same way. Only one value from each partition is tested, as it represents the entire class.

6. Decision Table Testing:

Decision Table Testing is used when a system’s output depends on multiple conditions. A decision table lists all possible inputs (conditions) and the corresponding outputs (actions), allowing testers to cover various input combinations.

7. State Transition Testing:

State Transition Testing is used for systems that change states based on certain conditions or events. It verifies that the system transitions from one state to another correctly based on input events.

8. Error Guessing:

Error Guessing is an intuitive technique where testers use their experience to guess where defects might exist in the application. Testers anticipate potential errors based on past experiences or common mistakes made in software development.

Q.6 Write detail test plan for online registration & examination application. Candidate will register through valid email ID & will book date & time slot from available dates & time slots. Candidate will select subject from list of subjects given. Application will send user ID & password to the candidate's email ID. On the date & time, candidate will login with given user ID & Password Application will display multiple choice questions & will accept choice from candidate. At the end of given time, application will display the marks

**Test Plan ID:** TP\_ORE\_01  
**Application Name:** Online Registration and Examination Application  
**Prepared By:** [Your Name]  
**Date:** [Date]  
**Version:** 1.0

**1. Introduction**

This test plan covers the testing activities for the **Online Registration and Examination Application**. The application allows candidates to register for an exam using a valid email, book a date and time slot, select a subject, and take an online examination. After the exam, the application will display the marks obtained by the candidate. The goal of this test plan is to ensure the application works according to the functional and non-functional requirements.

**2. Objectives**

The objectives of this test plan are to:

* Verify that the registration process works with valid and invalid email IDs.
* Ensure that the date and time slot booking system functions properly and only allows valid selections.
* Test the subject selection process and ensure only valid subjects are displayed and selected.
* Validate the email system to ensure user ID and password are sent correctly to the candidate.
* Confirm that the login process using user ID and password works.
* Test the examination interface for displaying multiple choice questions (MCQs) and capturing the candidate’s answers.
* Ensure that the examination timer works and the application calculates the final marks correctly.

**3. Scope**

**Functional Testing**

**Non-Functional Testing**

**4. Testing Approach**

**4.1 Manual Testing**

**4.2 Automated Testing**

**5. Test Items**

The following items will be tested:

* **Registration Module**
  + Email ID validation.
  + Date and time slot booking system.
  + Subject selection.
  + Email generation with user ID and password.
* **Examination Module**
  + Login using user ID and password.
  + Displaying MCQs with choices.
  + Capturing and recording candidate’s answers.
  + Exam timer functionality.
  + Displaying marks at the end of the exam.

**6. Test Environment**

* **Hardware Requirements:**
  + Windows/Linux/Mac systems.
  + Browser compatibility for Chrome, Firefox, Safari, and Edge.
* **Software Requirements:**
  + Web-based application hosted on a server.
  + Access to a database to store user details, exam information, and results.

**7. Entry Criteria**

* The application must be fully developed and deployed on the testing environment.
* Test data, including test email addresses, subjects, time slots, and MCQ question sets, should be available.
* All required hardware and software for testing should be ready.

**8. Exit Criteria**

* All critical and major defects must be fixed and verified.
* 95% of test cases must be executed successfully.
* No open critical defects at the time of release.
* The application should pass both functional and non-functional requirements.

**9. Test Deliverables**

* Test cases (manual and automated).
* Test results and reports.
* Bug reports and logs.
* Final sign-off document.

**10. Test Schedule**

| **Test Phase** | **Start Date** | **End Date** |
| --- | --- | --- |
| Test Plan Creation | [Start Date] | [End Date] |
| Test Case Preparation | [Start Date] | [End Date] |
| Test Environment Setup | [Start Date] | [End Date] |
| Test Execution | [Start Date] | [End Date] |
| Test Result Analysis | [Start Date] | [End Date] |

**11. Test Case Design**

**Module 1: Registration**

| **Test Case ID** | **Test Description** | **Test Steps** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_REG\_01 | Verify registration with valid email. | 1. Open registration page. 2. Enter valid email. 3. Select valid time slot. 4. Select subject. 5. Submit form. | Registration successful, email with user ID and password sent to email. | Pending |
| TC\_REG\_02 | Verify registration with invalid email. | 1. Open registration page. 2. Enter invalid email (e.g., missing domain). 3. Submit form. | Registration fails with an appropriate error message. | Pending |
| TC\_REG\_03 | Verify available date and time slot selection. | 1. Open registration page. 2. Check available date and time slots. 3. Select a slot. 4. Submit form. | Only valid date and time slots should be available, and selected slot should be reserved. | Pending |

**Module 2: Login**

| **Test Case ID** | **Test Description** | **Test Steps** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_LOGIN\_01 | Verify login with correct user ID and password. | 1. Enter valid user ID and password sent via email. 2. Submit. | User is successfully logged in. | Pending |
| TC\_LOGIN\_02 | Verify login with incorrect credentials. | 1. Enter invalid user ID or password. 2. Submit. | Login fails with an appropriate error message. | Pending |

**Module 3: Examination**

| **Test Case ID** | **Test Description** | **Test Steps** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_EXAM\_01 | Verify MCQ display and selection. | 1. Login to exam. 2. View and answer the MCQs. 3. Submit answers. | MCQs displayed correctly, choices accepted. | Pending |
| TC\_EXAM\_02 | Verify exam timeout functionality. | 1. Login to exam. 2. Wait for the time to elapse. 3. Check exam submission after timeout. | Exam submits automatically after time runs out. | Pending |
| TC\_EXAM\_03 | Verify marks display after exam completion. | 1. Complete exam. 2. Submit answers. 3. View marks displayed after submission. | Marks are correctly displayed based on answers. | Pending |

**12. Risks**

* Network or server issues affecting exam login and performance.
* Issues with email delivery leading to candidates not receiving their credentials.
* Performance degradation with multiple candidates registering or taking exams simultaneously.

**13. Approvals**

* [Your Name], Test Lead
* [Stakeholder Name], Project Manager

Q.7 Short Notes

a. Walkthrough

A Walkthrough is an informal process of reviewing a document, code, or process to identify any potential issues, discrepancies, or improvements. It involves team members or stakeholders going through the material to understand and provide feedback.

Key Characteristics:

Led by the author or the person responsible for the material being reviewed.

Attended by developers, testers, business analysts, and other stakeholders.

Focus is on knowledge sharing, finding defects, and improving quality.

No formal documentation or sign-off is required.

Purpose:

Early identification of defects in design documents, code, test cases, etc.

Improve communication among team members.

Gather feedback from multiple perspectives.

b. TestNG and JMeter

TestNG:

TestNG is a testing framework inspired by JUnit but with additional features, making it suitable for a wide range of testing needs, including unit, integration, and end-to-end testing.

Key Features:

Supports annotations like `@Test`, `@BeforeMethod`, `@AfterMethod`, etc.

Parallel execution of tests.

Flexible test configurations (groups, dependencies).

Data-driven testing (using `@DataProvider`).

Reports generation in HTML and XML format.

Use Cases:

Testing Java applications.

Integration testing for verifying the interaction between various components.

JMeter:

JMeter is an open-source tool primarily used for performance testing, load testing, and functional testing of web applications.

Key Features:

Simulates heavy loads on servers, groups of servers, or networks.

Supports various protocols like HTTP, HTTPS, FTP, JDBC, and more.

Provides real-time graphs, charts, and statistical reports.

Can be integrated with CI/CD pipelines.

Use Cases:

Load testing a web application to evaluate its performance under different load conditions.

Stress testing to determine the maximum capacity of the server before failure.

c. V Model

The V Model (or Verification and Validation Model) is an extension of the waterfall model that emphasizes the relationship between development and testing phases. It represents a sequential process where development and testing activities are mapped in a "V" shape.

Key Components:

1. Left Side (Development Phases):

Requirements Analysis: High-level requirements are gathered and analyzed.

System Design: Overall architecture and system design are created.

Detailed Design: Detailed internal design for each module or component.

Implementation: Actual coding of the system takes place.

2. Right Side (Testing Phases):

Unit Testing: Testing individual units or components.

Integration Testing: Testing the interaction between integrated modules.

System Testing: Testing the entire system as a whole.

Acceptance Testing: Verifying the system against user requirements.

Advantages:

Testing is planned parallel to development.

Defects are detected at an early stage, reducing cost and effort.

Clear documentation at each stage.

Disadvantages:

Rigid and less flexible for iterative projects.

Not suitable for projects with frequently changing requirements.

d. Defect Life Cycle

The Defect Life Cycle (also known as Bug Life Cycle) represents the journey of a defect from its identification to resolution in the software development process. It helps track the status of a defect and ensures that it gets addressed efficiently.

Stages of Defect Life Cycle:

1. New: The defect is identified and logged by the tester or user.

2. Assigned: The defect is assigned to a developer or relevant team member to fix.

3. Open: The developer starts working on fixing the defect.

4. Fixed: The defect is fixed by the developer and sent back for retesting.

5. Retest: The tester re-examines the defect to verify if the fix is successful.

6. Verified: If the defect is fixed, it is marked as verified.

7. Closed: The defect is considered closed if it no longer exists in the system.

8. Reopen: If the defect still persists after being marked fixed, it is reopened.

Other Possible States:

Deferred: The defect is not fixed immediately and will be addressed in a future release.

Duplicate: The defect is similar to another already logged.

Rejected: The defect is invalid or not considered a defect.

Importance:

Helps track the status and priority of each defect.

Improves communication between testing and development teams.

Ensures proper documentation and resolution of issues.