Quality of software can be accessed using 6 quality factors such as Functionality, Reliability, Efficiency, Maintainability, Portability, Usability.

**Static Testing**: - Analysis of code, documentation without executing the program. Defects identified in static testing are less expensive to fix. It involves both developers and testers.

1. **Informal Reviews**: - Reviewing the documents and providing informal comments.
2. **Technical Reviews**: - Technical specification document such as test plan.
3. **Walkthrough**: - Explanation of a product by the developer to the whole team.
4. **Inspection**: - Formal reviews to find out defect. Reviewers find out the defects and inform the development team.
5. **Static Code Review**: - Code syntax and optimization.

**Dynamic Testing**: - Execution of software and validating the output with expected result. Black & White box testing. It can be performed all the stages.

|  |  |
| --- | --- |
| STATIC TESTING | DYNAMIC TESTING |
| Without executing code | With executing code |
| Verification process | Validation process |
| Prevention of defects | Finding and fixing of defects |
| Before compilation | After compilation |
| Cost of defect is less | Cost of defect is high |
| ROI is very high at it occurs at early stage | ROI is low at it occurs at late stage |

**Black Box Testing:** - Examines the application functionality and ignores the internal mechanism of system and concentrates on the generated output. Tester has no access of source code. The tester is conscious about “What the software is supposed to do” but not aware of “how it does it”.

|  |  |
| --- | --- |
| **ADVANTAGES** | **DISADVANTAGES** |
| Well suited and efficient | Limited coverage |
| Code access not required | Inefficient testing |
| Separated users and developers perspective | Test cases are difficult to design |

**While Box Testing:** - Analyze the internal structure and working of a program. The tester is required to have programming skills and knowledge of internal structure of a program.

|  |  |
| --- | --- |
| ADVANTAGES | DISADVANTAGES |
| Due to source code access it is easy to identified test data | Skilled tester is required |
| It help is optimizing the code | Not possible to look into each corner |
| Extra lines of code can be removed which bring hidden defects. | Costly tools are required like debuggers |
|  |  |

**Unit Testing:** - Done by developers. It comes under white box testing. Development of stubs and drivers. It separates each part of a program and verify proper functionality of individual modules. **Ex: -** Calling functionality of google duo.

**Integration Testing:** - It test the functionalities of various modules of an application combined together. It mostly used in distributed systems. It comes under white and black box testing. It helps in identifying defects in interfaces. Two types of integration testing: - Top-Down and Bottom-Up. **Ex: -** Sign Up of Instagram using Facebook/Gmail.

**Functional Testing:** - It done on complete and integrated system to ensure that specified functionality required in the system works. It comes under black box testing. Focuses on testing the application against SRS, Test cases. It tests the functionality of each and every section of application. **Ex: -** Message sending functionality of WhatsApp. It should be send to the correct recipient without any dropping of messages etc.

**System Testing:** - It done on complete and integrated system. It comes under black box testing. Implemented by dedicated testing team which ensures that application meets functional and technical specification along with quality standards. Test environment is similar to production environment. **Ex: -** How system is performing after giving various types of inputs in different conditions.

**Smoke Testing:** - Initial build of software to ensure that crucial functionalities of the system working fine. The main objective is to ensure that the major functionalities are working and build is stable to go for functional testing. **Ex: -**  After deployment of any web application check whether URL is working fine or not before doing functional testing.

**Sanity Testing:** - It performs to ensure defects have been fixed and no further defects have been introduced due to minor changes in code. It is a subset of Regression testing. The main objective is to do quick checkup of software when defects are fixed before started detailed Regression Testing.

**Regression Testing:** - There are various ways for introduction of defects in application.

1. Defects has been introduced when a new functionality interact with existing one.
2. Fixing of defect in application may introduced more defects.

It mainly focuses on detecting such defect and ensure application is functioning properly. It provides repetitive and consistence validation of every new product released. Automation is preferred during regression. **Ex: -** After deployment of v1.1 on test environment check v1.0 is working as expected or not.

**Acceptance Testing:** - It ensures the software has met the required and performs the way customer expect in production. It helps in defect spelling mistake, cosmetic errors.

1. **Alpha Testing:** - Performed by an independent team within the company. Build confidence before goes for Beta Testing.
2. **Beta Testing:** - Performed by end users. It detects unexpected errors.

This testing helps in detecting real life issues in the product. Fixing these issues before releasing would improve quality of production and customer satisfaction.

**Performance Testing:** - Non-functional requirements like response time, throughput is all accordance with SLA (Service Level Agreement).

1. **Load Testing:** - Ability of an application to work under a particular user load. It helps in identifying bottleneck in application.
2. **Stress Testing:** - Increase the load till it reaches to the saturation point called breakpoint. It ensures application handled load spikes when it goes live.
3. **Endurance Testing:** - Expected load over a long period of time. It helps in uncovering bottleneck like memory leak etc.
4. **Volume Testing:** - Large amount of data populated in DB. It helps in estimates impact of data on application such as response time and DB strength.

**Functional Technique:** -

1. **Equivalence Partitioning:** - The whole range of input is split into set of equivalence classes such that a single value act as a sample for each equivalence classes. **For ex: -** A program needs to be tested within input range between 1000 & 2000 so this can be divided into 3 equivalence classes such as <1000, >2000 and between 1000 & 2000 etc.
2. **Boundary Value Analysis:** - This technique consists by developing of test cases and data that focuses on input and output boundaries of a given functionality as these are more prone to errors. **For ex:** - A program needs to be tested with input range between 1000 & 2000 so it will have **Lower Boundary (999 & 1001), Upper Boundary (1999 & 20001), On Boundary (1000 & 2000).**
3. **OATS (Orthogonal Array Testing Strategy): -** Structural approach to test pair wise integrations. It is helpful in integration level.

**Structural Technique: -** It comes under white box testing. Focuses on coverage of code.

1. **Statement Coverage:** - Every statement of code executed at least once.
2. **Branch Coverage:** - Test cases are designed to exercise branches.
3. **Path Coverage:** - Test cases are executed in such a way that every path is executed at least once.

**Characteristic of Software Required:** -

1. **Unambiguous:** - A good requirement should be stated in such a way that multiple readers must interpret same meaning out of it. It must be simple, straight forward and clear.
2. **Complete:** - It should contains required needs and conditions expected from the system relating to functionalities, performance etc.
3. **Testable:** - Software requirement is said to be testable if there exist some finite cost effective process that enables a tester to check if the requirement is met in software.
4. **Non-Redundant:** - It clearly points to 1 behavior and function. If more than 1 requirement point to the same function, then it became redundant.

**Software Testing Life Cycle:** -

**Requirement Capture and Analysis**

**Test Cycle Closure**

**Test Closure**

Re

**Test Planning and Design**

**Test Execution**

1. **Requirement Capture and Analysis:** - It is the most significant stage in STLC. During this phase the QA team interacts with different stakeholders, developers, client etc. and tries to understand requirement from testing point of view. They identified testable requirements.
2. **Test Planning and Design:** - Test strategy for complete testing process. Cost and Efforts estimated. It specifies scope, approach, resources, roles and responsibilities of various testers.

**Test Design**: - Test cases/scripts ae designed, reviewed and reworked. Identification of test data and RTM Preparation also done.

1. **Test Execution:** - Test team test the readiness of environment which was set up in test design phase. Testers also execute the test cases as per test plan. The defects are reported back to development team. Once the defects are fixed the system is tested again to ensure software is defect free.
2. **Test Cycle Closure:** - The testing team meets and evaluate criteria for cycle completion based on time, scope, coverage, cost, quality etc.
3. **Test Closure:** - The testing process for the project is evaluated and lessons learnt from the process are documented. The process that are implemented in future is identified.

**Software Defect:** - A software defect is an error or fault in software that leads to an unexpected behavior of system. The defects could be caused due to time pressure, miscommunication of requirements, Coding skills etc.

1. **Severity:** - It is the extent to which a defect can affect the software.
2. **Priority:** - The urgency with which defect should be addressed.

**Urgent** **Low**

**Critical** Key feature does not work Feature that is rarely used does not work

**Non-Critical** Company logo in wrong color Image caption is in wrong format

1. **New:** - Logged defect for the first time.
2. **Assigned:** - When defect is assigned to the development team to fix.
3. **Open:** - Developer started working on it.
4. **Fixed:** - Developer make code change to rectify defect.
5. **Retest:** - The tester rechecks the defect whether it has been fixed or not.
6. **Deferred:** - When the defect is expected to be fixed in upcoming releases.
7. **Duplicate:** - When 2 defects points to the same functionality.
8. **Rejected:** - If defect is not genuine then developer reject the defect.
9. **Reopen:** - If defect is not fixed then it re-opens.
10. **Closed:** - Defect does not exist in the software then status made as closed.

**Need of RTM (Requirement Traceability Matrix):** - The client who ordered for the product specifies the requirement to the development team and process of development get started. But maintaining a track of all the required documents and checking whether all the requirements are met by end product or not is a very cumbersome process. So, the remedy for this is RTM.

1. Captures the complete user and system requirements.
2. RTM captures all the requirements in a single document.
3. Changes in the requirements also tracked by RTM.
4. The RTM is maintained throughout the life cycle of system.
5. Track, Time, Change, Risk management is also done by RTM.

Forward, Backward are the types of Requirement Traceability Matrix.

**Test Data:** -

1. **Static Data:** - It is permanent data and related to DB. The information that changes little over time. **Ex: -** Country
2. **Master Data:** - Rarely updated but often read. **Ex:** - Name/email
3. **Configurable Data:** - Data which drives the application. **Ex: -** User Credentials
4. **Transactional Data:** - Business transactions in Application Under Test. **Ex: -** Order ID**.**

**Challenges in Software Testing:** -

1. Difficult to test an entire application completely.
2. Regression Testing.
3. Lack of skilled testers.
4. Time constraint.
5. Understanding the requirement.
6. Which testes to execute first.

**Entry Criteria: -**

1. Business requirement should be clear and available.
2. Good understanding of requirement must be done.
3. Test environment must be up and stable for detailed testing.
4. Proper sign-off must be received for test cases from BA before starting detailed testing.

**Exit Criteria: -**

1. Acceptance test should be executed and all the tests should be pass.
2. No critical/major defects left open. All the defects should be fixed and verified.
3. Test cases should be signed off.
4. Go/No-Go decision for the product.

**Testers start creating test cases/scenarios from the user story and get proper sign-off from BA**

**Basic Structure: -**

**BA creates user stories based on requirements and upload it on JIRA/TFS**

**Business Analyst (BA) try to understood requirement from Developers /Testers perspective from business persons**

**Business Person have Requirements for Product**

**ELSE**

**IF**

**Test results are shared with BA/Business Person for Signed Off.**

**After code fix retest done on the same and if it has been fixed then that defect should be closed.**

**If any defect has occurred, then it assigned back to developer to fix**

**Once development done tester assigned to that story and work of testing will start based on test cases of that user story.**

**Developers gets assigned to the user stories and development works start**

**Developers gets assigned to the user stories and development works start**

**Testing Flow: -**

**Code Deploy**

**Final PROD ENV. If any issues occurred in PROD, then OPS Team will review and get in touch with either Testing/Development Team.**

**Once Deployment done on TEST ENV process of testing gets started where it starts with Smoke Testing, Then Functional Testing, if any defects has been fixed then Sanity Testing.   
  
After detailed Regression Testing and getting proper signed-off for test results. Code will deploy on PROD ENV on GO-LIVE date.**

**Code Deploy**

**Developers start working on user stories in DEV ENV. After Unit Testing done on DEV ENV they deploy their code on TEST ENV for Functional Testing purpose.**

**PROD (Production)  
Environment**

**TEST (ST/SIT/UAT)  
Environment**

**DEV Environment**

**Login Page Test Cases: -**

1. **Functional Test Cases: -**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Scenario** | **Test Case ID** | **Test Case Description** | **Pre-Requisite** | **Steps** | **Expected Result** |
| Login Functionality Of Facebook Page | TC\_Login\_001 | User should be able to Login after valid credentials | User must present on Login Page | Enter valid USERID on User Name field | USERID should be entered |
|  |  |  |  | Enter valid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | User must be on Homepage. |
|  | TC\_Login\_002 | User should not be able to Login after entering in- valid credentials | User must present on Login Page | Enter invalid USERID on User Name field | USERID should be entered |
|  |  |  |  | Enter invalid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | User should get alert as “Invalid User ID or Password” |
|  | TC\_Login\_003 | User should get Single Sign On (SSO) Error | User’s previous session must be ON anywhere. | Enter valid USERID on User Name field | USERID should be entered |
|  |  |  | User must present on Login Page | Enter valid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | Due to previous ON session. User must get SSO Error as “Invalid Session due to Single Sign On Policy”. |

1. **Non - Functional Test Cases: -**
2. Performance of application after user load on system.
3. Should not take more time to login.
4. Response Time, Throughput of system etc.
5. Is there Bottleneck problem like Memory leak etc.