Software Testing and Quality Assurance.

Unit 1: Fundamentals of software testing.

Introduction :-

Software testing can be stated as the process if verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability.

Software testing can be divided into two steps:-

1. Verification: it refers to the set of tasks that ensure that the software correctly implements a specific function.
2. Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Basics of Software Testing:-

1. Same definition as above
2. Importance of Software Testing:-
   1. Quality Assurance:- Testing helps ensure that the software is free from critical defects, making it reliable for end-users.
   2. Cost-efficiency:- Detecting and fixing issues during testing is less expensive than addressing them after the software is deployed.
   3. Customer Satisfaction :- thorough testing leads to higher user satisfaction as it results in a more reliable and user-friendly product.
   4. Risk Reduction:- Testing mitigates the risk of software failures, which can be costly and damaging.

Approaches to testing:-

The test approach defines the testing methodology, tools, techniques, and strategies that will be used to test the software application. It outlines the scope of testing, the test objectives, and the expected outcome of the testing process. The test approach also includes the roles and responsibilities of the testing team, the test environment, and the test data requirements.

1. Analytical - This approach looks for problems by carefully examining the software and its documents, like reading a book to find errors without actually using it.
2. Model Based - Testers create drawings or diagrams that show how the software should work and use those drawings to plan their tests.
3. Methodical - Testers follow a structured plan like a step-by-step recipe to make sure they test everything in an organized way.
4. Process or Standard Compliant - Testers follow rules and guidelines set by experts to make sure the software meets certain quality standards, like following traffic rules on the road.
5. Dynamic or Heuristic - Testers use their experience and instincts to come up with tests on the spot, like a chef who doesn't use a recipe but knows how to make a delicious dish.
6. Consultative - Testers work together and talk to other people involved in the project to make decisions and share ideas, like a team brainstorming to solve a problem.
7. Regression Averse. - This approach tries to avoid introducing new problems when making changes to the software, like being careful not to break a toy while fixing it.

Testing during Development Life Cycle:-

The primary goal is to detect and address issues early to ensure the final product is of high quality and meets its intended requirements.

1. **Requirements Gathering and Analysis:**

* **Testing Activities**: Review and validate the initial project requirements and specifications.
* **Purpose**: Ensure that the requirements are clear, complete, and feasible. Identify potential issues or ambiguities early.

**2. System Design:**

* **Testing Activities**: Review system design documents and architecture diagrams.
* **Purpose**: Confirm that the design aligns with the requirements and that potential design flaws are identified and addressed.

**3. Coding and Unit Testing:**

* **Testing Activities**: Developers perform unit testing, where they test individual code components (units) in isolation.
* **Purpose**: Verify that each piece of code functions correctly and adheres to coding standards.

**4. Integration Testing:**

* **Testing Activities**: Test the interactions between different units or modules of code.
* **Purpose**: Ensure that integrated components work together as expected and that interfaces are functioning correctly.

**5. System Testing:**

* **Testing Activities**: Test the complete software system as a whole, including all its components and features.
* **Purpose**: Verify that the entire system meets the specified requirements, functions correctly, and detects any integration issues.

**6. User Acceptance Testing (UAT):**

* **Testing Activities**: End-users or stakeholders validate the software against their requirements and use cases.
* **Purpose**: Ensure that the software aligns with user expectations and can be used effectively in real-world scenarios.

**7. Regression Testing:**

* **Testing Activities**: After each code change or enhancement, run regression tests to ensure that new modifications do not introduce new defects into existing functionality.
* **Purpose**: Maintain the stability and reliability of the software as it evolves.

**8. Release and Deployment:**

* **Testing Activities**: Conduct final testing on the production environment to confirm that the software works correctly in the actual deployment setting.
* **Purpose**: Ensure a smooth transition from development to production and minimize the risk of issues affecting end-users.

**9. Post-Release Testing and Maintenance:**

* **Testing Activities**: Continue monitoring and testing the software in the production environment, addressing any issues that arise.
* **Purpose**: Identify and resolve issues that may not have been detected during earlier testing phases.

Essential of Software testing:-

· **Clear and Detailed Requirements**:

* · A well-defined set of requirements is essential for effective testing. Testers need a clear understanding of what the software is supposed to do to create relevant test cases.

· **Test Planning**:

* · A comprehensive test plan outlines the testing approach, scope, objectives, resources, and schedules. It serves as a roadmap for the testing process.

· **Test Design**:

* · Creating detailed test cases, test data, and test scripts based on the requirements and test plan is crucial. Well-designed tests ensure thorough coverage of the software.

· **Test Environment Setup**:

* · Preparing a test environment that mimics the production environment is necessary to simulate real-world conditions and execute tests accurately.

· **Test Execution**:

* · Running test cases, recording results, and comparing actual outcomes with expected results to identify defects. This phase is at the heart of testing.

· **Defect Reporting and Management**:

* · Documenting and prioritizing defects, including clear descriptions and steps to reproduce, is essential for efficient defect resolution.

· **Regression Testing**:

* · Repeating tests to ensure that new changes or features do not introduce new defects into the existing functionality. Regression testing maintains software stability.

· **Traceability**:

* · Establishing traceability links between requirements, test cases, and defects ensures that every requirement is tested and defects are tracked to their source.

· **Test Automation**:

* · Automating repetitive and time-consuming tests using scripts and testing tools can improve efficiency, accuracy, and coverage.

· **Test Data Management**:

* · Managing test data, including its creation, storage, and maintenance, is critical for effective testing, especially in data-intensive applications.

· **Continuous Testing**:

* · Integrating testing into the development process, often referred to as continuous testing, ensures faster feedback and early defect detection.

· **Test Documentation**:

* · Properly documenting test artifacts, such as test plans, test cases, test reports, and defect logs, is essential for knowledge sharing and audit purposes.

· **Metrics and Reporting**:

* · Collecting and analyzing testing metrics, such as test coverage, defect density, and test execution progress, provides insights into the quality of the software.

· **User Acceptance Testing (UAT)**:

* · Involving end-users or stakeholders in UAT ensures that the software meets their requirements and is ready for deployment.

· **Test Maintenance**:

* · Updating and maintaining test cases and scripts to reflect changes in the software is essential to keep the testing process relevant.

· **Test Closure**:

* · Summarizing test results, generating test closure reports, and making a decision about whether the software is ready for release.

Question Bank ISE 1

Q1) Principles of Testing?

Ans->

Testing is a crucial phase in the software development process that helps identify and address defects or issues in a software application. The principles of testing provide a framework for conducting effective and efficient testing activities.

Following are the some of the principles of the software testing:-

1. Testing shows the presence of defects, not their absence
   1. This means that when we test software, we can find mistakes or problems, but even if we don't find any, it doesn't mean there aren't any more.
2. Exhaustive testing is impossible
   1. Testing every possible way a software can work is impossible. So, we focus on the most important parts.
3. Early testing saves time and money
   1. If you check your work as you go along, you can catch mistakes early, when they're easier and cheaper to fix.
4. Defects cluster together
   1. Sometimes, problems in software tend to gather in certain areas or features. So, when you find one problem in a part of the software, there might be more nearby.
5. Beware of pesticide paradox
   1. This means that if you use the same tests over and over again, they might stop finding new problems because they become too familiar. So, it's important to update and change your tests regularly.
6. Testing is context dependent
   1. Different projects and software have different needs and situations. So, the way we test should fit what we're working on.
7. Absence-of-errors is a fallacy
   1. Just because you didn't find any problems doesn't mean everything is perfect.

Q2) White Box Testing.

Ans->

* + 1. This is also called as structural testing.
    2. In this type of testing tester will check all the statements, control structures, conditional statements inside the program.
    3. In this testing method, the design and structure of the code are known to the tester.
    4. Programmers of the code conduct this test on the code.
    5. White Box Testing Techniques :-
       1. Control-flow testing - The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
       2. Data-flow testing - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

Q3) Black Box Testing?

Ans->

* + 1. This is also called as Functional Testing.
    2. Here tester only deals with inputs and output values.
    3. In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.
    4. Functionality of the software tested in this method.
    5. Black box Testing Techniques :-
       1. Equivalence class - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
       2. Boundary values - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
       3. Cause-effect graphing - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
       4. Pair-wise Testing - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
       5. State-based testing - The system changes state on provision of input. These systems are tested based on their states and input.

Q4) Error and Defect?

Ans->

|  |  |
| --- | --- |
| Error | Defect |
| 1. An ****Error**** is a mistake made in the code; that's why we cannot execute or compile code. | 1)The ****Defect**** is the difference between the actual outcomes and expected outputs. |
| 1. The ****Developers and automation test engineers**** raise the error | 2) The ****Testers**** identify the defect. And it was also solved by the developer in the development phase or stage. |
| Different type of Error is as below:   * Syntactic Error * User interface error * Flow control error * Error handling error * Calculation error * Hardware error * Testing Error | Different type of Defects are as follows: Based on ****priority****:   * High * Medium * Low   And based on the severity:   * Critical * Major * Minor * Trivial |
| Enhance the software quality with system review and programming. Detect the issues and prepare a suitable mitigation plan. Validate the fixes and verify their quality and precision. | Implementing several innovative programming methods. Use of primary and correct software development techniques. Peer review It is executing consistent code reviews to evaluate its quality and correctness. |

Q5) Difference between white box and black box testing?

Ans->



Q6) Testing Approaches?

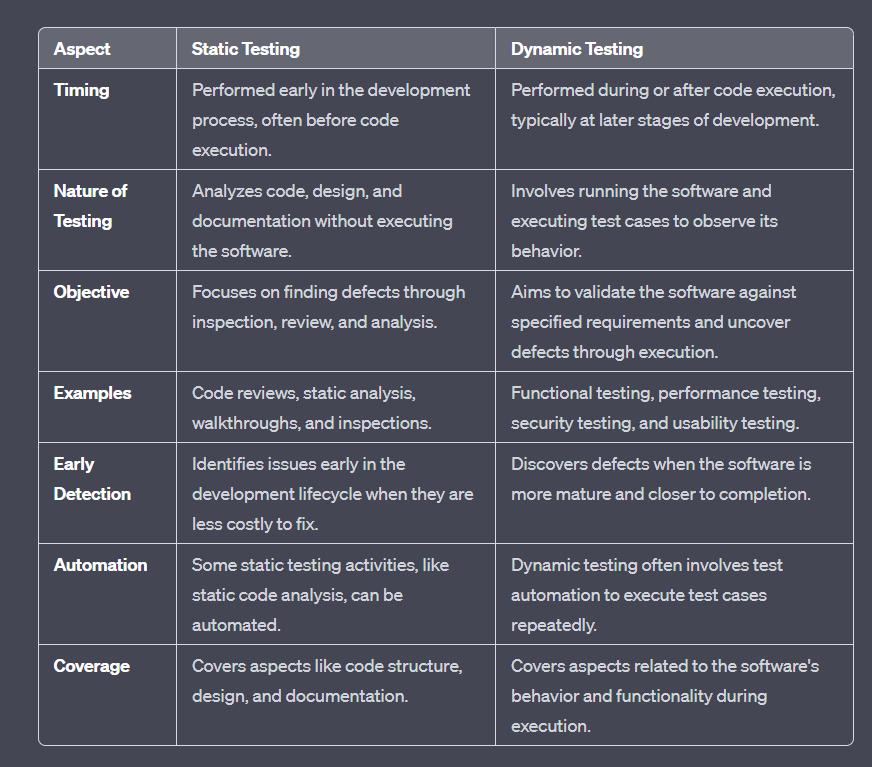
Ans->

The test approach defines the testing methodology, tools, techniques, and strategies that will be used to test the software application. It outlines the scope of testing, the test objectives, and the expected outcome of the testing process. The test approach also includes the roles and responsibilities of the testing team, the test environment, and the test data requirements.

1. Analytical - This approach looks for problems by carefully examining the software and its documents, like reading a book to find errors without actually using it.
2. Model Based - Testers create drawings or diagrams that show how the software should work and use those drawings to plan their tests.
3. Methodical - Testers follow a structured plan like a step-by-step recipe to make sure they test everything in an organized way.
4. Process or Standard Compliant - Testers follow rules and guidelines set by experts to make sure the software meets certain quality standards, like following traffic rules on the road.
5. Dynamic or Heuristic - Testers use their experience and instincts to come up with tests on the spot, like a chef who doesn't use a recipe but knows how to make a delicious dish.
6. Consultative - Testers work together and talk to other people involved in the project to make decisions and share ideas, like a team brainstorming to solve a problem.
7. Regression Averse. - This approach tries to avoid introducing new problems when making changes to the software, like being careful not to break a toy while fixing it.

Q7) Difference between static and dynamic testing?

Ans->

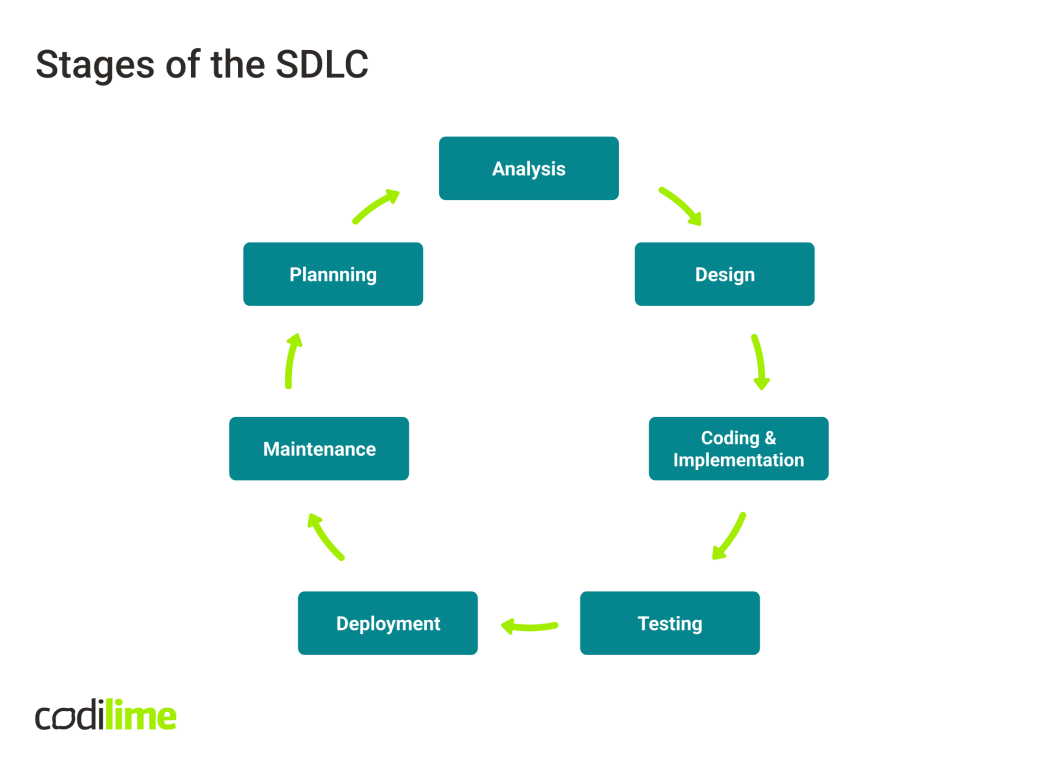


Q8) Describe white box and black box testing?

Ans-> Abhi tak jo samjha vo likh lo upar se

Q9) SDLC Model?

Ans->



1. Planning-
   1. The planning phase is where project objectives, scope, and requirements are defined. It involves setting clear goals, estimating resources, establishing timelines, and creating a project roadmap. This phase ensures that the project's direction is well-defined and aligns with the organization's strategic goals.
2. Analysis-
   1. During the analysis phase, a comprehensive examination of the system's requirements takes place. It involves gathering and documenting user needs, functional and non-functional requirements, and constraints.
3. Design-
   1. Design is the phase where the high-level architecture and system structure are planned. It includes creating detailed technical specifications, defining the software's components and interfaces, and determining how data will flow within the system.
4. Coding and Implementation-
   1. In this phase, developers write the actual code for the software based on the design specifications. It involves translating the design into a functional system using programming languages and development tools. The focus is on writing clean, efficient, and maintainable code.
5. Testing-
   1. The testing phase is where the software is systematically evaluated to identify and rectify defects and ensure that it meets the specified requirements. Different testing types, such as unit testing, integration testing, system testing, and user acceptance testing, are conducted to validate the software's correctness and quality.
6. Deployment-
   1. Deployment involves making the software available for users. It includes activities like installing the software on production servers, configuring it for optimal performance, and ensuring that it is accessible to end-users or clients.
7. Maintenance-
   1. Maintenance is the ongoing phase where the software is monitored, updated, and enhanced. It includes addressing issues reported by users, applying patches and updates, improving system performance, and incorporating new features or functionalities.