Software Testing

Program Testing

Definition: Testing is intended to show that a program does what it is intended to do and to discover program bugs before it is put into use.

When testing software, you execute a program using artificial data ,the results ensure the software meets requirements and behaves correctly.Testing reveals the presence of errors, not their absence.

Validation Testing & Defect Testing

Validation testing ensures the system performs correctly with expected test cases
Defect testing uncovers issues using unconventional test cases.

Verification vs Validation

Verification: Ensuring that we are building the product correctly by conforming to its specification.

Validation: Ensuring that we are building the right product by meeting the user's actual requirements.

V & V Confidence

The aim of verification and validation (V & V) is to establish confidence that the system is **"fit for purpose."** Confidence depends on the following things:

I. **Software purpose:** The level of confidence depends on how critical the software is to an organization.
II. **User expectations:** Users may have low expectations of certain kinds of software.
III. **Marketing environment:** Getting a product to market early may be more important than finding defects in the program.

Inspections vs Testing

Inspections: Analyzing software without executing it (e.g., code review).It is also called Static Testing.
Testing: Running the software with test data to observe its behavior. It is also called Dynamic Testing.

Stages of Testing

1. **Development testing:** Testing carried out by the team developing the system
    - Unit Testing
    - Component Testing
    - System Testing
2. **Release Testing:** Performed by a separate team to ensure the system is ready for deployment.
    - Requirements-based Testing
    - Performance Testing
3. **User Testing:** Actual users test the system in their own environment.
    - Alpha Testing
    - Beta Testing
    - Acceptance Testing

Example: A software team tests a new feature (development testing), a QA team tests a release candidate (release testing), and beta testers provide feedback (user testing).

Development testing

- **Unit Testing:** Testing individual components (functions, methods, classes) in isolation to ensure they function correctly.
  Example: Testing a function that calculates the sum of two numbers.

Automated testing frameworks like **JUnit** enable the automation of unit tests, providing generic test classes and reporting mechanisms. Unit test cases should cover both normal operation and specialized scenarios to ensure the component functions correctly and expose potential defects.

- **Component Testing:** Testing groups of related units integrated to form larger components. Focuses on verifying interactions and data transfer between units.
  Example: Testing a component that handles user login, involving units for input validation, database interaction, and session management.

- **System Testing:** Testing the entire integrated system as a whole. Focuses on validating overall system functionality, interactions between components, and emergent behavior.
  Example: Testing the entire weather station system, including data collection, processing, transmission, and user interface interactions.

Release Testing

Performed by a separate team before releasing the software to users. Aims to validate that the system meets requirements and is ready for use.

- **Requirements-based Testing:** Examining each requirement and developing specific tests to ensure it is met.
  Example: Creating tests to verify that allergy warnings are issued for patients with known allergies to specific medications.
- **Performance Testing:**: Evaluating the system's performance characteristics, such as response time, resource usage, and throughput under various load conditions. Here we do Stress Testing in which we intentionally overload the system to test its behavior under extreme conditions and identify its breaking point.
  Example: Flooding the system with an unusually high number of requests to observe how it handles the load and whether it recovers gracefully.

User Testing

Conducted by users in their own environment to provide feedback and identify real-world issues.

- **Alpha Testing:** Users work with the development team at the developer's site to test early versions of the software. Provides direct feedback and identifies usability issues early on.
- **Beta Testing:** A release of the software is made available to a wider group of users to experiment with and report problems. Helps gather feedback from a diverse user base and identify environment-specific issues.
- **Acceptance Testing:** Customers test the system to decide whether it meets their requirements and is ready for deployment.

8.1. Explain how the number of known defects remaining in a program at the time of delivery affects product support.

The number of known defects in a program at delivery time affects product support in several key ways:

More Support Requests: More defects mean more users facing problems, leading to a higher volume of support requests.
Customer Unhappiness: Defects can frustrate users and harm the company’s reputation.
Higher Costs: Fixing defects after release is more expensive and takes more time.
Lower Productivity: Defects can interrupt users' work, especially in business applications.
Security Risks: Some defects can be security vulnerabilities, leading to potential data breaches.

In short, delivering software with many defects can overload support, increase costs, frustrate users, and pose security risks. It's important to minimize defects before release to keep customers happy and maintain a good reputation.

8.2: Testing is meant to show that a program does what it is intended to do. Why may testers not always know what a program is intended for?

Testers may not always know what a program is intended for because:

- Incomplete Documentation: Sometimes, the program's requirements and specifications are not fully documented or are unclear.
- Communication Gaps: There may be poor communication between the development team and the testers, leading to misunderstandings about the program's purpose.
- Complexity: The program might be complex or have many features, making it hard for testers to understand all its intended uses.
- Changing Requirements: The intended functionality of the program might change during development, and testers may not always be updated on these changes.

8.3: Some people argue that developers should not be involved in testing their own code but that all testing should be the responsibility of a separate team. Give arguments for and against testing by the developers themselves.

For Testing by Developers:

- **In-depth Knowledge:** Developers have a deep understanding of the code and can identify potential issues quickly.
- **Immediate Feedback:** Developers can test their code as they write it, allowing for immediate detection and correction of errors.
- **Efficiency:** It can be more efficient and faster since developers are already familiar with the codebase.

Against Testing by Developers:

Bias: Developers might overlook errors in their own code due to familiarity and bias.
Lack of Objectivity: A separate testing team can provide an unbiased perspective and catch issues that developers may miss.
Specialized Skills: Testers often have specialized skills in creating comprehensive test cases and scenarios that developers might lack.

In summary, while developers testing their own code can be efficient and quick, a separate testing team offers objectivity and specialized skills that can lead to more thorough testing.

8.5: What is regression testing? Explain how the use of automated tests and a testing framework such as JUnit simplifies regression testing.

Regression Testing is the process of re-running functional and non-functional tests to ensure that previously developed and tested software still works after changes.

Automated Tests and JUnit:

1. **Efficiency:** Automated tests can be run quickly and frequently with minimal human intervention.
2. **Consistency:** Automated tests provide consistent results and eliminate human error.
3. **Early Detection:** Automated tests can catch regressions early in the development cycle.

4. **JUnit:** A framework like JUnit simplifies regression testing by providing a structured way to write and run automated tests, making it easier to integrate into the development process.

8.6: The Mentcare system is constructed by adapting an off-the-shelf information system. What do you think are the differences between testing such a system and testing software that is developed using an object-oriented language such as Java?

Off-the-Shelf System (Mentcare):

1. **Limited Control:** Less control over the source code, making it harder to fix issues.
2. **Integration Testing:** Focus on how well the system integrates with existing infrastructure.
3. **Customization:** Testing the customizations and configurations made to the off-the-shelf system.

Custom Software (Java):

1. **Full Access:** Complete control over the source code for detailed testing.
2. **Unit Testing:** More emphasis on unit testing individual components.
3. **Flexibility:** Easier to make and test changes in response to detected issues.

8.7. Write a scenario that could be used to help design tests for the wilderness weather station system.

Scenario:

1. **Setup:** The weather station is installed in a remote location.
2. **Initialization:** The station initializes and starts collecting weather data.
3. **Data Collection:** The station collects temperature, humidity, and wind speed data every minute.

4. **Data Transmission:** The station transmits collected data to the central server every hour.
5. **Power Failure:** The station experiences a power failure and resumes operation with data integrity upon recovery.

8.8 : What do you understand by the term stress testing? Suggest how you might stress-test the Mentcare system.

Stress Testing involves testing the system under extreme conditions to ensure stability and reliability.

Stress-Testing Mentcare:

1. **Load Simulation:** Simulate a high number of concurrent users accessing the system.
2. **Data Volume:** Input a large volume of patient data to test database performance.
3. **Network Conditions:** Test under poor network conditions to check system response.
4. **Resource Limitation:** Limit system resources (CPU, memory) to evaluate performance.

8.9 : What are the benefits of involving users in release testing at an early stage in the testing process? Are there disadvantages in user involvement?

Benefits:

1. **Real-World Feedback:** Users provide practical insights and identify usability issues.
2. **Increased Acceptance:** Early involvement can lead to higher user acceptance and satisfaction.

3. **Requirement Validation:** Ensures that the software meets user needs and expectations.

Disadvantages:

1. **Scope Creep:** Users may request additional features, leading to scope changes.
2. **Conflicting Feedback:** Different users may provide conflicting suggestions.
3. **Resource Allocation:** Involving users early can require additional resources and time.