Introduction to Software Testing

Introduction

- **Software testing** is a process to evaluate the functionality of a software system with an objective to find out whether the developed software meets the specified requirements or not and to identify the defects to ensure that the software is error free.
- To summarize:
    1. Reliability.
    2. Free from any bugs.
    3. Clients requirement.
    4. Used to check if product is user friendly.
    5. It is not possible to create software with zero error without incorporating software testing in the development cycle.

Basics of Software Testing

- **Benefits of Software Testing:**
    1. Cost Effective
    2. Security
    3. Product Quality
    4. Customer Satisfaction
- **Testing is classified into three categories:**
    1. Functional Testing:
        - Validates the software system against the functional requirements.
        - This testing check user interface, APIs, Database, Security, Client/Server communication and other functionality of the application under test. The testing can be done either manually or using automation.
    2. Non-functional Testing or Performance Testing:

- These requirements include the performance output that is expected from the application or the software under test. This basically includes the time taken by the software to operate a particular system.
- A non-functional requirement should be captured as:
    a. User/Technical issues/stories
    b. Acceptance criteria
    c. In artifacts

3. Maintenance:
    - Performance testing after it is released is known as maintenance testing.
    - At times we need to correct or upgrade the software during its run time which involves maintenance, planned enhancement, changes of the operational environment, patches for defects and vulnerabilities and request from the end user to add new features.
    - Types of Maintenance Testing:
        i. Confirmation Testing: Testing the modified functionality.
        ii. Regression Testing: Testing the existing functionality.

- Testing Strategies:
    1. Unit Testing
    2. Integration Testing
    3. System Testing
    4. Program Testing
- A **fault** is a state that causes the software to unable to perform its required function.
- An **error** is the variance between the actual output and the expected output.
- A **failure** is the inability of a system or its components to perform the required task according to its specification.
- **Professional Software Testing practices include:**
    1. Planning and design of testing
    2. Acceptance test criteria defined by users.

3. System tests identified during requirements analysis.
4. Unit test cases identified.
5. Execution of unit tests, integration tests, system tests and acceptance tests.
6. Evaluation of test effectiveness.

Testing Objective

1. Verification
2. Validation
3. Defects
4. Providing Information
5. Preventing Defects
6. Quality Analysis
7. Compatibility
8. Optimum User Experience
9. Verifying Performance and Functionality

Principles of Testing

1. Testing shows the presence of defects and not their absence
2. Exhaustive testing is impossible
3. Early testing saves time and money
4. Defects cluster together
5. Beware of the pesticide paradox
6. Testing is context-dependent
7. Absence of errors is a fallacy

Testing and Debugging

Testing	Debugging
It is a process to find bugs and errors in the software.	It is the process to correct the bugs or errors found during testing process.

Testing	Debugging
It is the process to identify the failure of implemented code in the software.	It is the process to identify and fix error.
Displays the errors.	It is a logical process.
It is done by the software tester.	It is done by the programmer.
Design knowledge is not required in the testing process.	It requires proper design knowledge.
Can be manual or automated.	It is always manual, can’t be automated.

Verification and Validation

- **Verification** checks if the work products are complete and consistent as per the user’s needs.
- **Validation** process assesses the quality of a software system in its operating environment.

Verification	Validation
This process includes checking documents, design, codes and programs.	This process includes testing and validating the actual product.
It is static testing.	Validation is dynamic testing.
It does not include the execution of the code.	It includes the execution of the code.
It can find bugs in the early stage.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.

Testing Life Cycle

1. Decision to enter the test phase
2. Introducing the test process
3. Test plan preparation
4. Development of test case
5. Execution and management of test plan

6. Test process evaluation and improvement

Myths

1. Testing is too expensive
2. Testing is time consuming
3. Only fully developed products are tested
4. Complete testing is possible
5. Tested software is bug-free
6. Missed defects are due to testers
7. Testers are responsible for quality of product
8. Test automation should be used wherever possible to reduce time
9. Anyone can test a software application
10. A tester’s only task is to find bugs

Levels of Testing

Four levels of Testing/Test Strategy

- **Unit Testing:** Checks if software component is fulfilling functionality or not.
- **Integration Testing:** Checks the data flow between one module to the other.
- **System Testing:** Evaluate functional and non-functional needs.
- **Acceptance Testing:** Checks if product meets user requirements.

Unit Testing

- Unit testing is the lowest level of testing that is conducted by development team.
- Unit testing tests individual units of the software.
- This is carried out during the development of an application.

Integration Testing

- Main objective is to test the software modules of application interact in a correct, stable and coherent way.
- Development team conducts integration testing.
- Modules are combined and tested in the integration testing.
- Goal is to check the correctness of communication among all the modules in the software.

Top-Down Integration

- It is a type of incremental testing which is done by integrating two or more modules by moving from top to bottom.
- This testing is used in order to simulate the behavior of the lower level modules that are not yet integrated.

Bottom-Up Integration

- It is a type of incremental testing which is done by integrating two or more modules by moving from bottom to up.

- Each component on lower hierarchy is tested individually and then the components that rely upon these components are tested.

System Testing

- This testing is done to establish confidence that the application will be accepted by the users.
- During system testing, it is necessary that truly independent observation of the testing process be performed.
- **Steps to perform System Testing:**
    1. Test Environment Setup
    2. Create Test Case
    3. Create Test Data
    4. Execute Test Case
- **Types of System Testing:**
    1. **Performance Testing**: User to test performance benchmarks like speed, scalability, stability and reliability of the software.
    2. **Load Testing**: Test the application under extreme load.
    3. **Stress Testing**: Used to check robustness of the system under varying loads.
    4. **Scalability Testing**: Used to check if the system will be able to scale up or down.

Acceptance Testing

- This testing is done to confirm that the system meets its requirements before it is formally handed over to the user.
- Acceptance testing is performed by nominated user representative under the guidance and supervision of the testing team.

Alpha Testing

- This testing is conducted by a representative group of end users at the developer’s side and sometimes by an independent team of testers. It comes after unit testing, integration testing, etc.

- This can be a type of white box or black box testing.
- **Process for Alpha Testing:**
    1. Requirement Review
    2. Test Development
    3. Test Case Design
    4. Logging Defects
    5. Bug Fixation
    6. Retesting
- **Two Phases of Alpha Testing:**
    1. **First Phase:** In house developers do the first phase of testing. In this phase, the testers uses hardware debuggers in order to catch the bugs.
    2. **Second Phase:** This involves the quality assurance staffs who performs the alpha testing by involving black box and white box techniques.
- **Reasons for using Alpha Testing:**
    1. Refines the software product.
    2. Test the software in a real world environment.
    3. Ensure the success of the software product.
    4. Alpha testing validates quality, functionality and effectiveness before it is released in the real world.
- **Features of Alpha Testing:**
    1. Type of acceptance testing.
    2. Happens at the stage of the completion of the software.
    3. Done in labs in controlled environment.
    4. Performed by in-house testers and developers.
    5. Done to gain confidence in the user acceptance.
    6. Ensures the maximum possible quality of the software before releasing.
    7. Done after unit testing, integration testing, system testing but before beta testing.
- **Advantages of Alpha Testing:**

1. Reduces the delivery time of the project.
2. Provides a complete test plan and test cases.
3. Feedback helps to improve software quality.
4. Provides a better observation of the software’s reliability and accountability.

- **Disadvantages of Alpha Testing:**
  1. Does not involve in-depth testing.
  2. Difference between tester’s data and customer’s data may result in discrepancy in the functioning.
  3. Lab cannot furnish all the requirements of the real environment.

Beta Testing

- Beta testing is the term used to describe the external testing process where the software is used by real users in a real time environment.
- This testing is carried out at the client’s site.
- This is the final phase of testing before the product delivery.
- This testing is done at the end of the software testing life cycle.
- Beta version of software is released to limited number of users to obtain feedback.
- **Features of Beta Testing:**
  1. Used in real environment at the user’s site.
  2. Testing is performed by client, stakeholders as well as end users.
  3. Done after alpha testing and before the software release into the market.
  4. Type of black box testing.
- **Types of Beta Versions:**
  1. **Closed Beta Version:** Also known as private beta which is released to a group of selected and invited people. Those people will test the software and evaluate their features and specifications.
  2. **Open Beta Version:** Also known as public beta which is open to public. Any user as a tester can assess the beta version to provide the relevant feedback and reviews.

- **Process of Beta Testing:**
    1. Planning
    2. Participants recruitment
    3. Product launch
    4. Collect and evaluate feedback
    5. Closure
- **Types of Beta Testing:**
    1. **Open Beta Testing:** This version lets large number of people test the software before the final release. Organization make the product open to public before the release.
    2. **Closed Beta Testing:** It is performed by the selective and limited number of people.
    3. **Traditional Beta Testing:** A software product is delivered to the target market and feedback is collected.
    4. **Public Beta Testing:** Similar to open beta testing. Here product is open to world wide users to try and give feedback.
    5. **Technical Beta Testing:** This testing involves delivering the software product to the internal groups of the organization.
    6. **Focused Beta Testing:** This is focused on monitoring and evaluating a specific feature of the software.
    7. **Post-Release Beta Testing:** The product is delivered to the market for end-users and their feedback, reactions and experience are collected for the future release.
- **Checklist before final release of the product:**
    1. All the components of the product are ready.
    2. Documentation is ready.
    3. All the functionality is in good condition.
    4. Procedure to collect bugs, feedback, etc. should be identified before releasing it.

Performance Testing

- **System performance measuring parameters:**
    1. System response time.
    2. External interface response time.
    3. CPU utilization.
    4. Memory utilization.
- This testing is done to develop effective and acceptable system performance.
- It is an information gathering and analyzing process which is used to predict when load levels will consume system resources.

Regression Testing

- This testing ensure the the application functions correctly even after carrying the required modification to the systems.
- This type of testing can be used in the context of system and acceptance testing.

Load/Stress Testing

- This is a non-functional testing in which the performance of software application is tested under a specific expected capacity.
- It determines how the application behaves while being accessed by multiple users at once.
- **This testing usually identifies:**
    1. Maximum operating capacity.
    2. If current infrastructure is sufficient to run the application.
    3. Sustainability of application.
    4. Number of concurrent users that an application can support.

Security Testing

- This uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders.
- Security testing of any system focuses on finding all possible loopholes and weakness of the system.
- **Goals of Security Testing:**

1. Identify the threats.
2. Measure the potential vulnerabilities.
3. Detecting every possible security risks.
4. Help developers in fixing the security problems.

- **Principle of Security Testing:**
  1. Confidentiality
  2. Integrity
  3. Authentication
  4. Authorization
  5. Availability
- **Focus area of Security Testing:**
  1. Network security
  2. System software security
  3. Client-side application security
  4. Server-side application security
- **Types of Security Testing:**
  1. Vulnerability Scanning
  2. Security Scanning
  3. Penetration Testing
  4. Risk Assessment
  5. Security Auditing
  6. Ethical Hacking
  7. Posture Assessment

Configuration and Compatibility Testing

Configuration Testing

- This testing helps in finding out best combination of software and hardware for peak performance without any flaws.

- **Objectives:**
    1. Whether the software fulfills the configurability requirements.
    2. Identify the defects that were not found during different testing processes.
    3. Determine an optimal configuration.
    4. Do analysis of the performance.
    5. Do analysis of the system efficiency.
- **Types:**
    1. **Software Configuration Testing**: It is done over the application under test with various OS versions and various browser versions.
    2. **Hardware Configuration Testing**: It is typically performed in labs where physical machines are used with various hardware connected to them.
- **Further Classified into:**
    1. **Client Level Testing:** This is associated with the usability and functionality testing.
    2. **Server Level Testing:** This is carried out to determine the communication between the software and the external environment.

Compatibility Testing

- This comes under non-functional testing category.
- It is performed on an application to check its compatibility on different platforms.
- **Types:**
    1. **Software:**
        a. Testing the compatibility of an application with various OS.
        b. Testing compatibility on databases like Oracle SQL server, MongoDB server, etc.
        c. On different devices like mobile phones, computers.
    2. **Hardware:**
        a. RAM
        b. ROM
        c. Hard disk

    d. Memory cards
    e. Processor
    f. Graphics card
3. **Smartphones:** Checking compatibility with android, iOS, etc.
4. **Network:**
    a. Bandwidth
    b. Operating speed
    c. Capacity

- **Advantages:**
    1. Ensures complete customer satisfaction.
    2. Provides service across multiple platforms.
    3. Identifying bugs during development process.
- **Disadvantages:**
    1. Use of variety of user interface.
    2. Changes the look with respect to font size.
    3. Alignment issues.
    4. Issues arising due to broken frames.
    5. Issues arising due to overlapping of content.

Testing Web Applications

- **Web Application Testing Techniques:**
  1. Functionality testing
  2. Usability testing
  3. Interface testing
  4. Compatibility testing
  5. Performance testing
     a. **Load testing**: It is the simplest form of testing conducted to understand the behavior of the system under a specific load.
     b. **Stress testing**: It is performed to find the upper limit capacity of the system and to know how the system performs if the current load goes well above the expected maximum limit.
     c. **Soak testing**: This is also known as endurance testing, is performed to determine the system parameters under continuous predictable load.
     d. **Spike testing**: It is performed by increasing the number of users abruptly by a very large amount and measuring the performance of the system.
  6. Security testing

Dimensions of Quality

Sr. No.	Quality Dimensions	How to Measure
1	Maintainability	1. Lines of code 2. Peer Review 3. Level of complexity
2	Portability	1. Setting up realistic testing environments 2. Automating tests of installation/uninstallation
3	Functionality	1. Regression tests 2. Integration tests 3. Acceptance tests
4	Performance	1. Stress testing 2. Soak testing
5	Compatibility	1. Cross browser testing 2. UI automation
6	Usability	1. User acceptance testing 2. Real user monitoring 3. Customer satisfaction

Sr. No.	Quality Dimensions	How to Measure
7	Reliability	1. Production defects 2. Unit tests 3. Integration tests
8	Security	1. Number of incidents 2. Number of breaches

Error within a WebApp Environment

- **Challenges faced by web developers:**
  1. Integration
  2. Interoperability
  3. Security
  4. Performance
  5. Usability
- **Web Testing Checklist**
  1. Functionality Testing
  2. Compatibility Testing
  3. Usability Testing
  4. Interface Testing
  5. Security Testing
  6. Performance/Load Testing
- **Importance of Test Plans**
  1. Help individuals outside the QA teams to understand exactly how the website or app will be tested.
  2. They offer a clear guide for QA engineers to conduct their testing activities.
  3. Test scope, test estimation, strategy can be collated into a single document which makes it easier to be reviewed by management.
- **Components of a Test Plan**
  1. Scope
  2. Schedule
  3. Resource allocation

4. Environment
5. Tools
6. Defect management
7. Risk management
8. Exit parameters

- **Creating a Test Plan**
  1. Product analysis
  2. Designing test strategy
  3. Defining objectives
  4. Establish test criteria
     a. Suspension criteria
     b. Exit criteria
  5. Planning resource allocation
  6. Planning setup of test environment
  7. Determine test schedule and estimation
  8. Establish test deliverables
     a. Deliverables required during testing
        i. Test scripts
        ii. Simulators or emulators
        iii. Test data
        iv. Error and execution logs
     b. Deliverables required after testing
        i. Test results
        ii. Defects reports
        iii. Release notes
- **The Testing Process**
  1. Functionality testing of a website

2. Usability testing
3. Interface testing
4. Database testing
5. Compatibility testing
6. Performance testing
7. Security testing
8. Crowd testing

Agile Testing

- Done at the beginning with incorporation between development and testing.
- **Advantages of Agile Testing:**
    1. Saves our time and money.
    2. Less documentation.
    3. Regular feedback.
    4. Regular interactions help to determine the issues.
- **Principles of Agile Testing:**
    1. Testing is not a phase.
    2. Testing move the project forward.
    3. Everyone tests.
    4. Shortening Feedback Response Time.
    5. Clean Code.
    6. Reduce Test Documentation.
    7. Test Driven.

Traditional Testing

- **Features:**
    1. Project manager manages.
    2. Client’s involvement is required.
    3. Executed with predefined standard steps.
- **Advantages:**
    1. Full coverage of the software.
    2. Allows one to detect maximum number of defects.
    3. Ensure quality and effectiveness.
- **Disadvantages:**
    1. Requires excessive time and effort.

2. Dependent on documentation.
3. Changes are implemented only at the end of testing.
4. No interactions between various software testers.

Agile Testing

- **Features:**
  1. Flexible to changes.
  2. Requires minimum documentation.
  3. Collaboration with the end users.
  4. Performed with the assistance of automated tools.
- **Advantages:**
  1. Involves rigorous planning, analysis and testing.
  2. Offers effective and efficient risk management.
  3. Promotes interactions with testers, client and end users.
  4. Feature driven.
  5. Rapid product delivery, along with optimum quality.
- **Disadvantages:**
  1. Difficult to specify and communicate certain design or testing components.
  2. Assessing the efforts is difficult at the beginning of the project.

Traditional Testing	Agile Testing
Top-Down approach.	Iterative approach.
Performed when development is completed.	Testing and debugging are done in each sprint.
Different teams tests different modules.	Teams collaborate in an open workspace.
Requirements are concrete and are hardly modified.	Flexible and easily adapts.

12 Agile Principles

1. Satisfy customers through early and continuous delivery.
2. Welcome changing requirements.

3. Deliver value frequently.
4. Break the silos of your project.
5. Build project around motivated individual.
6. Face-to-Face communication.
7. Working software is the primary measure of progress.
8. Maintain a sustainable working pace.
9. Continuous excellence enhances agility.
10. Simplicity is essential.
11. Self-organizing teams generate most value.
12. Regularly reflect and adjust your way of work to boost effectiveness.

4 Agile Values

1. Individuals and Interactions
2. Working software
3. Customer collaboration
4. Responding to change

Agile Testing Quadrants

1. Quadrant 1 : Technology-facing tests that support the team
2. Quadrant 2 : Business-facing tests that support the team
3. Quadrant 3 : Business-facing tests that critique the product
4. Quadrant 4 : Technology-facing tests that critique the product