Module 2

1. What is Exploratory Testing ?

* In exploratory testing tester focuses more on how the software actually works, testers do minimum planning and maximum execution of the software by which they get in -depth idea about the software functionality, once the tester starts getting insight into the software he can make decisions on what to test next. Exploratory testing is mostly used if the requirements are incomplete and time to release the software is less.

Key Characteristics:

1. No predefined test cases: Testers don't create test cases beforehand.

2. Simultaneous learning and testing: Testers learn about the software while testing.

3. Dynamic test design: Testers design and execute tests on the fly.

4. Focus on exploration: Testers investigate the software's behavior, rather than following a script.

5. Emphasis on critical thinking: Testers use their expertise and judgment to identify potential issues.

Benefits:

1. Flexibility: Allows testers to adapt to changing requirements or software behavior.

2. Creativity: Encourages testers to think critically and outside the box.

3. Time-efficient: Reduces test preparation time, as no test cases need to be written.

4. Effective for complex systems: Suitable for testing complex, dynamic, or hard-to-script systems.

5. Improved test coverage: Can uncover issues that scripted testing might miss.

Challenges:

1. Requires skilled testers: Testers need expertise, experience, and critical thinking skills.

2. Difficult to reproduce issues: Without clear test steps, reproducing issues can be challenging.

3. Lack of documentation: Testers may not document their testing process.

4. Hard to measure progress: Difficulty tracking test coverage and progress.

2. What is traceability testing ?

* Traceability in software testing is the ability to track tests and their results throughout the development lifecycle, both forward and backward. It's often done using a traceability matrix, which is a document that shows the relationship between requirements and other artifacts.
* Traceability testing is also known as Requirement traceability Matrix.
* 1. Every requirement has at least one test case associated with it.
* 2. Test cases adequately cover the requirements.
* 3. Requirements are accurately implemented in the software.

Goals of Traceability Testing:

* 1. Ensure requirements are testable and verifiable.
* 2. Verify that software meets specified requirements.
* 3. Identify missing or incomplete requirements.
* 4. Reduce defects and rework.
* 5. Improve testing efficiency and effectiveness.
* \*Types of Traceability:\*
* 1. Forward Traceability : Mapping requirements to test cases.
* 2. Backward Traceability : Mapping test cases to requirements.
* 3. Bi-Directional Traceability : Combining forward and backward traceability.
* \*Benefits:\*
* 1. Improved test coverage and effectiveness.
* 2. Reduced risk of requirement-related defects.
* 3. Enhanced collaboration between developers, testers, and stakeholders.
* 4. Simplified impact analysis for changes or updates.
* 5. Compliance with regulatory requirements.
* \*Challenges:\*
* 1. Maintaining accurate and up-to-date requirements documentation.
* 2. Ensuring test cases adequately cover complex requirements.
* 3. Managing traceability across multiple testing levels (unit, integration, system).
* 4. Scalability issues with large, complex projects.
* \*Best Practices:\*
* 1. Establish clear, concise requirements.
* 2. Create a traceability matrix.
* 3. Regularly review and update requirements and test cases.
* 4. Automate traceability where possible.
* 5. Involve stakeholders in traceability activities.
* \*Traceability Matrix:\*
* A table or spreadsheet mapping requirements to test cases, typically including:
* | Requirement ID | Requirement Description | Test Case ID | Test Case Description |
* | --- | --- | --- | --- |

3. What is Boundary value testing ?

* Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams.
* This is one of the software testing technique in which the test cases are designed to include values at the boundary. If the input data is used within the boundary value limits, then it is said to be Positive Testing. If the input data is picked outside the boundary value limits, then it is said to be Negative testing.
* BVA example :

A system can accept the numbers from 0 to 10 numeric values. All other numbers are invalid values. Under this technique, boundary values 0 , 10 and -10 will be tested.

4. What is Equivalence partitioning testing ?

* In this method the input domain data is divided into different equivalence data classes. This method is typically used to reduce the total number of test cases to a finite set of testable test cases, still covering maximum requirements. In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing. Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:
* 1) One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be the same. So one test case for valid input data should be sufficient.
* 2) Input data class with all values below lower limit. I.e. any value below 1, as a invalid input data test case.
* 3) Input data with any value greater than 1000 to represent the third invalid input class. Equivalence partitioning uses fewest test cases to cover maximum requirements

5. What is Integration testing ?

* Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
* The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
* Integration testing is done by a specific integration tester or test team.
* There are two levels of Integration Testing

1. Component Integration Testing
2. System Integration Testing

* Need of integration testing :

A Module in general is designed by an individual software developer who understanding and programming logic may differ from other programmers. Integration testing becomes necessary to verify the software modules work in unity.

At the time of module development, there wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence integration testing becomes necessary.

7. What is Alpha testing?

* It is always performed by the developers at the software development site.
* Sometimes it is also performed by Independent Testing Team.
* Alpha Testing is not open to the market and public
* It is conducted for the software application and project.
* It is always performed in Virtual Environment.
* It comes under the category of both White Box Testing and Black Box Testing.
* During this phase, the following will be tested in the application:

1. Spelling Mistakes

2. Broken Links

3. Cloudy Directions

* Alpha Testing is always performed at the time of Acceptance Testing when developers test the product and project to check whether it meets the user requirements or not.
* Unit testing, integration testing and system testing when combined are known as alpha testing.

8. What is beta testing ?

* It is always performed by the customers at their own site.
* It is not performed by Independent Testing Team.
* Beta Testing is always open to the market and public.
* It is usually conducted for software product.
* It is performed in Real Time Environment.
* It is always performed outside the organization.
* Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
* It is only a kind of Black Box Testing.
* Beta Testing is always performed at the time when software product and project are marketed.
* Beta testing can be considered “pre-release” testing.
* Pilot Testing is testing to product on real world as well as collect data on the use of product in the classroom.

9. What is component testing ?

* A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
* Component Testing – The testing of individual software components
* TWO types of integration testing :

1. Big bang integration testing
2. Incremental integration testing

* Integration Testing is performed after Unit Testing and before System Testing.
* Either Developers themselves or independent Testers perform Integration Testing.

6. What determines the level of risk ?

* Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’
* Types of risks :

1. Project Risks
2. Product Risk

* Example of Project risk is Senior Team Member leaving the project abruptly.

Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10.

10. What is functional testing system ?

* Testing based on an analysis of the specification of the functionality of a component or system.
* ‘Function’ – what the system does
* Functional test – based on the Functions and features – may be applied at all Test levels (e.g. Component Test, System Test etc.)
* Considers the external (not internal) behaviour of the software. Black- Box testing.
* Functional testing verifies that each function of the software application operates in conformance with the requirement specification.
* This testing mainly involves black box testing and it is not concerned about the source code of the application.
* This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application under Test. The testing can be done either manually or using automation.

11. What is non functional testing ?

* Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability.
* It is the testing of “how” the system works. Non-functional testing may be performed at all test levels.
* Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g. performance test scaling
* Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.
* Hence load testing is carried out to check systems performance at different loads i.e. number of users accessing the system

12. What is GUI testing ?

* Graphical User Interface involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.
* What we check in GUI testing ?
* Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
* Check you can execute the intended functionality of the application using the GUI.
* Check Error Messages are displayed correctly.
* Check for Clear demarcation of different sections on screen.
* Check Font used in the application is readable.
* Check the alignment of the text is proper.
* Check the Color of the font and warning messages are aesthetically pleasing.
* Check that the images have good clarity.
* Check that the images are properly aligned.
* Check the positioning of GUI elements for different screen resolution.

13. What is Adhoc testing ?

* Adhoc testing is an informal testing type with an aim to break the system.
* It does not follow any test design techniques to create test cases.
* Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
* Main aim of this testing is to find defects by random checking.
* Adhoc testing can be achieved with the testing technique called Error Guessing.
* Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.
* It also saves a lot of time because of the assumptions and guessing made by the experienced testers to find out the defects which otherwise won’t be able to find.
* Using experience to postulate errors.
* TYPES OF Adhoc testing:

1. Buddy testing : Two buddies mutually work on identifying defects in the same module.
2. Pair testing : Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings.
3. Monkey testing : Randomly test the product or application without test cases with a goal to break the system.

14. What is load testing ?

* Load testing is a kind of performance testing which determines a system’s performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.
* This testing usually identifies –
* The maximum operating capacity of an application
* Determine whether current infrastructure is sufficient to run the application
* Sustainability of application with respect to peak user load
* Number of concurrent users that an application can support, and scalability to allow more users to access it.
* It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.
* Need of load testing : Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes.
* Load testing gives confidence in the system & its reliability and performance.
* Load testing gives excellent protection against poor performance and accommodates complementary strategies for performance management and monitoring of a production environment.
* Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.
* Example : Popular toy store Toysrus.com, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.
* An Airline website was not able to handle 10000+ users during a festival offer.

15. What is stress testing ?

* System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
* Stress testing is done to make sure that the system would not crash under crunch situation.
* Stress testing is also known as endurance testing.
* Example : The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives ‘Not Responded’ error message.
* Resource reduction such as a disk drive failure.
* Need of stress testing :

1. During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
2. System failure under extreme conditions could result in enormous revenue loss
3. Displaying appropriate error message when the system is under stress.
4. It is better to be prepared for extreme conditions by executing Stress Testing.

* The main purpose of stress testing is to make sure that the system recovers after failure which is called as recoverability.
* Types of stress testing :

1. Application stress testing
2. Transactional stress testing
3. Systemic stress testing
4. Exploratory stress testing

16. What is white box testing and list the types of white box testing ?

* White Box testing, also known as clear box testing, glass box testing, or code-based testing is a method of software testing that involves testing the internal structures or workings of an application.
* Types of white box testing :

1. Unit testing - test individual code units ( Functions/ methods)
2. Integration testing : Test the interaction between integrated units or components.
3. Static analysis : Involves examining the code without executing it.

* Helps in identifying potential errors, code quality issues and security vulnerabilities.

1. Dynamic Analysis : Involves executing the code and analyzing its behavior.

* Helps in identifying runtime errors and performance issues.

17. What is black box testing ? What are the different black box testing techniques ?

* Testing software based on output only without any knowledge of its internal code or logic. The testers have no knowledge of how the system or component is structured inside the box.
* Advantages :
* Code Access not required.
* Well suited and efficient for large code segments.
* Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.
* Disadvantages :
* Limited Coverage since only a selected number of test scenarios are actually performed.
* Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
* The test cases are difficult to design.
* Techniques of black box testing :
* Equivalence partitioning : This is a software testing technique which divides the input date into many partitions .
* Boundary value analysis : This is one of the software testing technique in which the test cases are designed to include

values at the boundary.

* Decision tables
* State transition testing
* Use-case Testing
* Other Black Box Testing

Syntax or Pattern Testing

18. Mention what are the categories of defects ?

* Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software.
* Types :

1. Low: The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
2. Medium: The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
3. High: The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.
4. Critical: Extremely urgent, resolve immediately

19. Mention what bigbang testing is ?

* In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
* Advantages: Convenient for small systems.
* Disadvantages: Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.

20. What is the purpose of Exit criteria ?

* Successful Testing of Integrated Application.
* Executed Test Cases are documented
* All High prioritized bugs fixed and closed
* Technical documents to be submitted followed by release Notes.

21. When should “Regression Testing” be performed ?

* when the system is stable and the system or the environment changes.
* when testing bug-fix releases as part of the maintenance phase
* It should be applied at all Test Levels
* It should be considered complete when agreed completion criteria for regression testing have been met
* Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation

22. What are 7 key principles ? Explain in detail ?

* Testing shows presence of Defects : Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
* Exhaustive Testing is Impossible! : Testing everything including all combinations of inputs and preconditions is not possible. We have learned that we cannot test everything.
* Early Testing : Testing activities should start as early as possible in the development life cycle. Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
* Defect Clustering : A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures. Defects are not evenly spread in a system, They are ‘clustered’.
* The Pesticide Paradox : If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
* To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
* Testing is Context Dependent : Testing is done differently in different contexts.

Different kinds of sites are tested differently.

* Absence of error fallacy : If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help. Even after defects have been resolved it may still be unusable and/or does not fulfill the users.

23. Difference between QA v/s QC v/s Tester.

| QA   1. Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. 2. QA mostly focuses on processes and procedures rather than conducting actual testing on the system. 3. Process oriented activities. 4. It has preventive activities. | QC   1. Activities which ensure the verification of developed software with respect to documents. 2. QC focuses on actual testing by executing software intended to identify bugs/defects through implementation of process and procedures. 3. Product oriented activities. 4. It is a corrective process. | Tester  1.Activities which ensure the identification of bugs/errors/defects in the Software.  2. Tester focuses on actual testing.  3. Product oriented activities.  4. It is a preventive activity. |
| --- | --- | --- |

24. Difference between smoke and sanity testing.

| Smoke Testing   1. Testing technique which examines all the basic components of a software system to ensure that they work properly. 2. This testing is performed by Developers or testers. 3. Smoke testing is Documented and scripted. 4. Smoke testing is like general health check up. | Sanity Testing   1. Testing technique which determines if a new software version is performing well enough to accept it for a major testing effort. 2. Sanity testing is performed by testers. 3. Sanity testing is not documented or scripted. 4. Sanity testing is like specialized health check up. |
| --- | --- |

25. Difference between verification and validation.

| Verification   1. The process of evaluating the work-product of a development phase to determine whether the meet the specified requirements for that phase. 2. Plan, design, code, test cases 3. Review, walkthroughs, inspection | Validation   1. The process of evaluating software during or at the end of the development process to determine whether it satisfies business requirements. 2. The actual product or software. 3. Testing |
| --- | --- |

26. Explain types of performance testing ?

* Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.
* Types of performance testing :

1. Load testing
2. Stress testing
3. Endurance testing
4. Spike testing
5. Volume testing
6. Scalability testing

27. What is Error, Defect, Bug and failure ?

* “A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure”
* Error : a human action that produces an incorrect result. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.
* Defect : Commonly refers to several troubles with the software products, with its external behavior or with its internal features.
* Bug : A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.
* Failure : The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

28. Difference between priority and severity ?

| Priority   1. Priority determines the **urgency** with which a defect should be fixed. In other words, it’s the order in which developers should address defects. 2. Priority types : Low, medium, high 3. **Priority** measures the **business impact** and urgency of fixing the defect. | Severity   1. Severity refers to the **degree of impact** that a defect (or bug) has on the operation of the software product. 2. Categories of severity : Critical, major, medium, Minor, Cosmetic. 3. Severity measures the technical impact of a defect on software functionality. |
| --- | --- |

29. What is Bug life cycle ?

* Bug life cycle is nothing but the various phases a bug undergoes after it is raised or reported.
* The different phases of Bug life cycle are,

- New or Opened

- Assigned

- Fixed

- Tested

- Closed

30. Explain the difference between functional and non - functional testing ?

| Functional Testing   1. Functional testing is executed first. 2. Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. 3. Manual testing or automation tools can be used for functional testing 4. Business requirements are the inputs to functional testing. 5. Functional testing describes what the product does 6. Easy to do manual testing 7. Types of Functional testing are  * Unit Testing * Smoke Testing * Sanity Testing * Integration Testing * White box testing * Black Box testing * User Acceptance testing * Regression Testing | Non functional Testing   1. Non functional testing should be performed after functional testing. 2. Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system. 3. Using tools will be effective for this testing. 4. Performance parameters like speed , scalability are inputs to non-functional testing. 5. Nonfunctional testing describes how good the product works. 6. Tough to do manual testing. 7. Types of Nonfunctional testing are - Performance Testing  * Load Testing * Volume Testing * Stress Testing * Security Testing * Installation Testing * Penetration Testing * Compatibility Testing * Migration testing |
| --- | --- |

31. To create HLR and Testcase.

1. Instagram

* [Instagram- HLR & Testcase of log in](https://docs.google.com/spreadsheets/d/1DAlTYyr193xVmvaBq4K0x5hboeNNtyKxtKI06DlCfUg/edit?usp=sharing)

1. Facebook

* [Facebook Create account HLR & Testcase](https://docs.google.com/spreadsheets/d/1U7rFCgXHtRebswm68Inx2y24ddg61rK0HTRq_07y0JM/edit?usp=sharing)

32. What is the difference between STLC & SDLC ?

| SDLC   1. SDLC is a systematic process for developing software, from planning o deployment. 2. Phases : Requirement gathering, Analysis, design, Implementation, testing, Deployment, Maintenance. 3. Goals :- Deliver quality software  * Meet customer requirements * Stay within budget and timeline  1. Software development 2. Development oriented 3. Deliver working software 4. Entire software development | STLC   1. STLC is a process for testing software, ensuring it meets requirements and works as expected. 2. Phases : Test planning, Test case development, test environment setup, test results analysis, defect reporting and tracking, test closure. 3. Goals :- Identify defects early  * Ensure software quality * Reduce testing time and costs  1. Software testing 2. Testing oriented 3. Ensure software quality 4. Testing phase |
| --- | --- |

33. What is the difference between test scenarios, test cases and test script ?

| **Test scenarios**   1. A high level idea of what to test. 2. User log in 3. ‘I want to make a pizza”. 4. Idea | **Test cases**   1. A detailed step by step plan to test a specific scenario. 2. Enter username, enter password, click login, check if login is successful 3. Recipe (step by step istructions) 4. Plan | **Test Script**   1. The actual code or steps to perform the test case. 2. Manual script : follow steps 1-4 manually  * Automated script : code to automate steps 1-4 (e.g., python, java)  1. Actual cooking (manual) or Automated pizza machine. 2. Execution |
| --- | --- | --- |

34.Explain what is test plan ? What is the information that should be covered.

* Test Plan: A test plan is a detailed document outlining the testing approach, scope, schedule, and deliverables for a software testing project.
* Purpose:

1. Ensure thorough testing

2. Define testing scope and objectives

3. Identify resources and timelines

4. Establish testing standards

* Information to Cover:

1. Test Plan Identifier (unique ID)

2. Introduction

- Project overview

- Testing objectives

3. Scope

- In-scope features/functions

- Out-of-scope features/functions

4. Test Objectives

- Functional testing

- Performance testing

- Security testing

5. Test Strategy

- Testing approach (e.g., black box, white box)

- Testing techniques (e.g., equivalence partitioning)

6. Test Schedule

- Testing timeline

- Milestones

7. Test Deliverables

- Test reports

- Test summary

8. Test Environment

- Hardware requirements

- Software requirements

- Network configuration

9. Test Data

- Test data requirements

- Data sources

10. Test Scripts and Cases

- Test script/CASE development approach

- Test script/CASE repository

11. Testing Tools and Resources

- Testing tools (e.g., Selenium, JIRA)

- Resource allocation (e.g., personnel, equipment)

12. Risk Management

- Identified risks

- Mitigation strategies

13. Test Metrics and Reporting

- Test metrics (e.g., pass/fail rate)

- Reporting requirements

14. Test Completion Criteria

- Criteria for test completion

- Exit criteria

15. Approval and Revision History

- Approval signatures

- Revision history

* Test Plan Template:

You can use the following template:

[Your Company Logo]

Test Plan for [Project Name]

[Date]

[Revision History]

[Introduction]

[Scope]

[Test Objectives]

[Test Strategy]

[Test Schedule]

[Test Deliverables]

[Test Environment]

[Test Data]

[Test Scripts and Cases]

[Testing Tools and Resources]

[Risk Management]

[Test Metrics and Reporting]

[Test Completion Criteria]

[Approval and Revision History]

* Benefits:

1. Ensures thorough testing

2. Reduces testing time and costs

3. Improves testing efficiency

4. Enhances communication among stakeholders

* Best Practices:

1. Review and update the test plan regularly

2. Involve stakeholders in test planning

3. Use a standardized test plan template

4. Ensure test plan aligns with project objectives

35. What is priority ?

* Priority: Priority refers to the level of importance or urgency assigned to a task, requirement, or defect.
* Purpose:

1. Focus on critical tasks first
2. Allocate resources efficiently
3. Meet deadlines and objectives

* Types of Priority:

1. High Priority (Critical):

- Must be addressed immediately

- Significant impact on project or business

- High risk or urgency

2. Medium Priority (Major):

- Important but not critical

- Moderate impact on project or business

- Some flexibility in timeline

3. Low Priority (Minor):

- Less important or non-essential

- Minimal impact on project or business

- Flexible timeline

* Priority Levels in Software Testing:

1. Critical (High): Showstoppers, crashes, or security issues

2. High (Major): Functionality issues, data loss, or performance problems

3. Medium (Moderate): UI issues, minor functionality problems

4. Low (Minor): Cosmetic issues, minor documentation errors

* Benefits of Prioritization:

1. Efficient resource allocation

2. Improved focus on critical tasks

3. Enhanced productivity

4. Better risk management

5. Increased customer satisfaction

36. What is Severity ?

* Severity: Severity refers to the impact or potential impact of a defect or issue on the software's functionality, performance, or user experience.
* Severity Levels:

1. Critical (High):

- Causes system crash or failure

- Loss of critical data

- Significant security vulnerability

- Impacts multiple users

2. Major (Medium-High):

- Causes significant functionality issues

- Impacts key features

- Data corruption or loss

- Performance degradation

3. Moderate (Medium):

- Causes minor functionality issues

- Affects non-critical features

- Cosmetic issues

- Minor performance impact

4. Minor (Low-Medium):

- Causes minimal impact

- Cosmetic issues

- Minor documentation errors

- Low-risk security issues

5. Low (Minimal):

- No significant impact

- Minor suggestions

- Typos or formatting issues

* Factors Influencing Severity:

1. Impact on functionality

2. Data loss or corruption

3. Security risks

4. Performance degradation

5. User experience

6. Business criticality

7. Number of affected users

* *Relationship with Priority:*

1. Severity influences priority

2. High severity often requires high priority

3. Priority considers additional factors (deadline, resources)

* Example:

Defect: "Login button not working"

Severity: Critical (High)

Priority: High

Reason: Critical functionality issue impacting all users.

* Benefits of Severity Assessment:

1. Informed decision-making

2. Effective resource allocation

3. Prioritized defect fixing

4. Improved customer satisfaction

5. Reduced business risk

37. Bug categories are …

* Here are common bug categories:
* Functional Bugs:

1. Logic Errors

2. Calculation Errors

3. Data Corruption

4. Functionality Not Working as Expected

* Non-Functional Bugs:

1. Performance Issues

2. Security Vulnerabilities

3. Usability Issues

4. Compatibility Problems

* UI/UX Bugs:

1. Visual Defects

2. Layout Issues

3. Navigation Problems

4. Responsiveness Issues

* Configuration/Installation Bugs:

1. Installation Failures

2. Configuration Errors

3. Upgrade Issues

4. Compatibility Problems

* Network/Communication Bugs:

1. Connectivity Issues

2. Data Transfer Errors

3. Network Congestion

4. API Integration Problems

* Database Bugs:

1. Data Loss

2. Data Corruption

3. Query Errors

4. Schema Issues

* Security Bugs:

1. Authentication Issues

2. Authorization Problems

3. Data Encryption Errors

4. Vulnerabilities

* Platform/Environment Bugs:

1. OS-Specific Issues

2. Browser Compatibility Problems

3. Hardware-Related Errors

4. Third-Party Library Issues

* Other Bugs:

1. Documentation Errors

2. Localization Issues

3. Accessibility Problems

4. Miscellaneous Errors

* Bug Severity Levels:

1. Critical

2. Major

3. Moderate

4. Minor

5. Low

* Bug Priority Levels:

1. High

2. Medium

3. Low

* Bug Status:

1. New

2. Assigned

3. In Progress

4. Resolved

5. Closed

* Bug categories may vary depending on the project, industry, or organization.

38. Advantage of Bugzilla ?

* *Bugzilla* is a popular, open-source bug tracking system used by developers, testers, and project managers.
* *Key Advantages:*

1. Free and Open-Source: Bugzilla is free to use, modify, and distribute.

2. Customizable: Tailor Bugzilla to suit your project's specific needs.

3. Scalable: Handles large projects with thousands of bugs.

4. User-Friendly Interface: Intuitive web-based interface for easy bug reporting and tracking.

5. Robust Search: Advanced search capabilities for quick bug identification.

6. Assignment and Notification : Assign bugs to team members and receive notifications.

7. Status Tracking: Monitor bug status (New, Assigned, Resolved, Closed).

8. Priority and Severity: Set priority and severity levels for bugs.

9. Commenting and Attachment : Add comments and attachments to bugs.

10. Integration: Integrates with various tools (e.g., GitHub, SVN).

11. Reporting : Generate reports on bugs, status, and progress.

12. Security : Access control and authentication features.

* Benefits for Teams:

1. Improved collaboration and communication.

2. Enhanced bug tracking and management.

3. Increased productivity and efficiency.

4. Better prioritization and resource allocation.

5. Reduced bugs and defects.

* Benefits for Developers:

1. Easy bug reporting and tracking.

2. Customizable workflows.

3. Integration with development tools.

4. Automated notifications.

5. Improved code quality.

* Benefits for Testers:

1. Centralized bug tracking.

2. Simplified bug reporting.

3. Easy reproduction of bugs.

4. Quick assignment and notification.

5. Enhanced testing efficiency.

* Industry Usage:

Bugzilla is widely used in various industries, including:

1. Software development

2. IT

3. Finance

4. Healthcare

5. Government

39. What are the different methodologies in the Agile Development Model ?

* Agile Development Methodologies:

Here are popular Agile methodologies:

1. Scrum

- Focus: Teamwork, iterative development (Sprints)

- Roles: Product Owner, Scrum Master, Development Team

- Artifacts: Product Backlog, Sprint Backlog, Increment

2. Kanban

- Focus: Visual workflow, continuous flow and delivery

- No specific roles

- Artifacts: Board, Cards, Columns

3. Extreme Programming (XP)

- Focus: Technical practices (pair programming, TDD)

- Roles: Customer, Developer, Tracker

- Artifacts: User Stories, Iteration Plan

4. Lean Software Development

- Focus: Eliminate waste, maximize value

- Principles: Continuous improvement, iterative development

- Artifacts: Value Stream Map, Kanban Board

5. Crystal

- Focus: Tailored Agile approach for project size/complexity

- Roles: Project Manager, Team Members

- Artifacts: Project Plan, Iteration Plan

6. Feature-Driven Development (FDD)

- Focus: Feature development and delivery

- Roles: Project Manager, Chief Architect, Development Team

- Artifacts: Feature List, Iteration Plan

7. Dynamic Systems Development Method (DSDM)

- Focus: Continuous delivery, business value

- Roles: Project Manager, Business Ambassador, Development Team

- Artifacts: Project Plan, Iteration Plan

8. Adaptive Software Development (ASD)

- Focus: Adaptability, rapid response to change

- Roles: Project Manager, Development Team

- Artifacts: Project Plan, Iteration Plan

9. Agile Unified Process (AUP)

- Focus: Hybrid Agile and Unified Process (RUP)

- Roles: Project Manager, Development Team

- Artifacts: Project Plan, Iteration Plan

* Other Methodologies:

1. Test-Driven Development (TDD)

2. Behavior-Driven Development (BDD)

3. Acceptance Test-Driven Development (ATDD)

4. Large-Scale Scrum (LeSS)

5. Scrum of Scrums

* Key Considerations:

1. Project size and complexity

2. Team size and experience

3. Customer requirements and involvement

4. Organizational culture and structure

5. Need for scalability and flexibility

Benefits:

1. Improved collaboration and communication

2. Increased flexibility and adaptability

3. Enhanced customer satisfaction

4. Faster time-to-market

5. Better quality software

40. Explain the difference between Authorization and Authentication in web testing.

| **Authentication**   1. Verifying the identity of a user, system or entity. 2. Ensure the user/entity is who they claim to use. 3. Purpose : Verify identity 4. Scope : Initial verification 5. Process : one time verification | **Authorization**   1. Controlling access to resources based on user identity, role or permissions. 2. Determine what actions an authenticated user can perform. 3. Purpose : Control access 4. Scope : ongoing access control 5. Process : Continuous evaluation. |
| --- | --- |

41. What are the common problems faced in web testing ?

* Common Problems Faced in Web Testing:

Functional Testing:

1. Broken links and images

2. Form submission issues

3. Authentication and authorization problems

4. Shopping cart functionality issues

5. Payment gateway integration problems

Performance Testing:

1. Slow page load times

2. High resource utilization

3. Database performance issues

4. Network connectivity problems

5. Scalability issues

Security Testing:

1. SQL injection vulnerabilities

2. Cross-site scripting (XSS) attacks

3. Cross-site request forgery (CSRF) attacks

4. Authentication bypass vulnerabilities

5. Data encryption issues

Usability Testing:

1. Poor navigation and menu issues

2. Inconsistent layout and design

3. Difficulty with search functionality

4. Inadequate error handling

5. Inaccessible content for disabled users

Compatibility Testing:

1. Browser compatibility issues (e.g., IE, Chrome, Firefox)

2. Operating system compatibility problems (e.g., Windows, macOS)

3. Mobile device compatibility issues (e.g., iOS, Android)

4. Screen resolution and display issues

5. Third-party plugin compatibility problems

Configuration and Infrastructure:

1. Server configuration issues

2. Database configuration problems

3. Network configuration issues

4. Firewall and security configuration problems

5. Content delivery network (CDN) issues

Testing Challenges:

1. Testing complex business logic

2. Handling dynamic content

3. Testing for multiple user roles

4. Managing test data and environments

5. Integrating with third-party services

\*Best Practices to Overcome These Challenges:\*

1. Conduct thorough requirements gathering

2. Create detailed test plans and cases

3. Use automation testing tools

4. Perform continuous integration and testing

5. Conduct regular security audits

6. Test on various devices and browsers

7. Use cloud-based testing platforms

8. Collaborate with developers and stakeholders

42. Whatsapp web : [Whatsapp web HLR & testcase](https://docs.google.com/spreadsheets/d/1wzsWJUTgvfCyiekeRcyYkkAXzHs57L2ryH0HfWavFy4/edit?usp=sharing)

43. Art Of testing : [Art of testing HLR & testcase](https://docs.google.com/spreadsheets/d/1UT4nrvQFrM5y509nANzjkW1bySHvhH1zekNrFCF3TWY/edit?usp=sharing)

44. Scenario :

* [Scenario](https://docs.google.com/spreadsheets/d/14dVKNQl99z9MoYXm-TguR_T2rOnBNtxR3cHk0OxNmYQ/edit?usp=sharing)

45. When to used Usability Testing ?

* Usability Testing:

When to Use:

1. Early design stages: Validate design concepts and identify usability issues.

2. Prototype : Test clickable prototypes to refine interactions.

3. Pre-launch: Ensure usability before releasing a new product or feature.

4. Post-launch: Identify usability is development use and optimize user experience.

5. Redesign or revamp: Test new designs and interactions.

6. Competitor analysis: Compare usability with competitor products.

7. Accessibility testing: Ensure usability for users with disabilities.

Why Use:

1. Improve user experience and satisfaction.

2. Increase conversion rates and engagement.

3. Reduce bounce rates and errors.

4. Enhance user retention and loyalty.

5. Inform design and development decisions.

6. Identify and fix usability issues early.

7. Reduce support queries and costs.

Types of Usability Testing:

1. Moderated testing: Facilitated by a moderator.

2. Unmoderated testing: Self-directed by participants.

3. Remote testing: Conducted online.

4. In-person testing: Conducted in-person.

5. A/B testing: Compare two versions.

6. Heuristic evaluation: Expert review against usability principles.

7. User interviews: In-depth discussions.

Methods:

1. Think-aloud protocol: Participants verbalize thoughts.

2. Task-based testing: Participants complete tasks.

3. Survey and questionnaire: Gather user feedback.

4. Eye-tracking: Analyze user attention.

5. Session recording: Record user interactions.

46. To create positive and negative scenario

* [Positive & negative scenario](https://docs.google.com/spreadsheets/d/1qI0FflJdpOTI6xdqgbK8PW6mVRE5e1w_eQ1cKUfv_9k/edit?usp=sharing)