1. **WHAT IS SOFTWARE?**

* Software is a set of programs that help to run a computer system or basic hardware.

1. **EXP COMPONENTS OF SOFTWARE**

Components of s/w includes:

* User interface - the means by which a user interacts with a device or software
* User roles – The end user permissions to use the application
* Business roles- business-specific functionalities and responsibility within an organisation
* Data base- storage of all data and information
* Validations- the process of confirming the correctness
* Reports- a structured format of all the information and logs
* Logs – recorded information about the users
* Back end process – server side operations of the application
* Interfaces – to interact with the users and with other devices.

1. **EXP APPLN ARCHITECTURE. TYPES OF ARCHITECTURE**

Application s/w is a type of software that provides services to end users to perform their activities or tasks

Types:

* Mobile architecture – The document describes a three-tier architecture for mobile computing (eg WhatsApp, Twitter)
* Desktop architecture-It provides a comprehensive structure of software systems, detailing the components involved eg amazon
* Client Server architecture- request services from the server and provide with a response (eg MS Teams)
* Web Application architecture- The architecture of a web app is the design and construction of a website or web app (eg google chrome etc)

1. **EXP SERVICE MODEL OF APPLN**

a logical representation of how an application and its infrastructure components relate to each other.

Service model of application includes:

* Project- a product developed according to the customer needs
* Product- a complete product developed and deployed
* Enhancements – a feature enhanced or added to the existing version of the software
* Cots – Commercial off the shelf , used by multiple customers (SAP)
* Re engineering- building a new software again from the scratch .

1. **EXP APPLN CATEGORY WITH EG**

System S/w – It’s the basic software for a computer to work or run like an operating system eg( windows 11)

Application S/w –it’s a software that provides service to the end users like for eg MS Outlook, word etc

1. **EXP APPLN USERS**

* These are all the application users categorised:
* Community users – like a LinkedIn community or a group of companies.
* Technical users- Programmers who use the application
* Business users- like CGI or any other business enterprise

1. **WHAT IS ERROR**

It’s a human incorrect action or a mistake .

1. **WHAT IS A BUG/DEFECT**

A bug or a defect usually occurs because of the errors committed by humans. Deviation from the expected behaviour to the actual behaviour.

1. **WHAT IS FAULT**

A fault is a result of a bug occurrence/defect identification

1. **WHAT IS FAILURE**

A failure happens when the end user identifies the deviation in the application while using the system.

1. **DIFFERENCE BETWEEN ERROR,DEFECT AND FAULT**

|  |  |  |  |
| --- | --- | --- | --- |
| **S NO** | **ERROR**  **A human mistake/incorrect action** | **DEFECT**  **It’s because of the occurrence of a bug/ defect. Deviation from expected to actual behaviour.** | **FAULT**  **It’s the result of a defect/bugs** |

1. **WHAT IS SOFTWARE QUALITY**

* It’s a journey towards excellence.
* Meeting the user requirements, providing on-time delivery of the software, maintainable, bug-free, and is within budget.

1. **What is SDLC?**

* It’s a software development life cycle which involves certain phases of development like requirement analysis, design , coding , implementation and deployment .

1. **What is waterfall model**

* A sequential software development approach where each phase, such as requirements, design, implementation, testing, and maintenance, is completed before the next one begins.
* Cannot accommodate changes and is not applicable for complex projects.

1. **What is an agile model?**

* An iterative and incremental approach that emphasizes flexibility, collaboration, and continuous improvement.
* It's basically SDLC + Operation.
* Can accommodate changes, and it's in use by most business enterprises.

1. **What is Spiral model?**

* a software development lifecycle (SDLC) model used for risk management that combines the iterative development process of the Waterfall model.
* Each loop (spiral) represents a development phase and includes steps like planning, risk analysis, engineering, and evaluation.

1. **What is V model**

* The V-Model, also known as the Verification and Validation Model, is a software development model that emphasizes a correspondence between development stages and testing stages. It’s essentially an extension of the Waterfall Model, but with a strong focus on testing at every stage.
* The **V-Model** is called so because the process steps form a **"V" shape**, where:
* The **left side** of the V represents **development activities**.
* The **right side** of the V represents **testing/validation activities**.
* The **bottom** of the V is where **coding** (implementation) occurs.

1. **What is an incremental model?**

* The incremental model is an iterative approach where a software system is developed in small, manageable increments or modules.
* The changes / requirements can be made incrementally in each build.

1. **Difference between waterfall and agile model**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Waterfall** | **Agile** |
| **1** | **Sequential execution of the process** | **Incremental execution** |
| **2** | **Rigid , well structured and not easy to adapt to changes** | **Flexible and can adapt to changes.** |
| **3** | **Testing is done after the entire development of the software** | **Testing is done in the development process** |
| **4** | **Breaks it into distinct phases** | **Breaks it into sprints.** |

1. **What is DevOps?**

Development + Operation . Emphasizes communication between the development and the operations team.

1. **What is a test Scenario?**

A testscenario is a high level documentation of a functionality that needs to be tested.

1. **What is a test case?**

A set of both valid and invalid inputs given to test the functionality of the software module.

1. **What is a test script?**

A set of instructions used to verify the functionality of an application.

1. **Difference between agile and DevOps**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Agile** | **DevOps** |
| **1** | **Breaks the process into sprints** | **Continuous integration and delivery** |
| **2** | **Incremental delivery** | **Continuous delivery** |
| **3** | **Post delivery monitoring** | **Continuous monitoring** |
| **4** | **Feedback is from the end users/clients** | **Internal feedback from monitoring tools** |
| **5** | **It’s a methodology and can be used in the DevOps** | **It's not a methodology and cannot be used in agile.** |

1. **Difference between DevSecOps and SecDevOps**

|  |  |  |
| --- | --- | --- |
| **S NO** | **DevSecOps** | **SecDevOps** |
| **1** | **Development🡪Security🡪operation** | **Security🡪Development🡪operation** |
| **2** | **Main emphasis on the development** | **Main emphasis on the security** |

1. **Difference between QA and QC**

|  |  |  |
| --- | --- | --- |
| **S NO** | **QA** | **QC** |
| **1.** | **Checking the quality parameters for assurance** | **It’s the actual testing for quality** |
| **2.** | **Helps in building quality and is process-oriented** | **It’s product-oriented** |
| **3.** | **Performed in parallel with the project** | **Performed after the completion of the project( final product)** |
| **4.** | **Focuses on the prevention of defects** | **Focuses on the identification of the defects** |

1. **Difference between Static and Dynamic Testing.**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Static** | **Dynamic** |
| **1.** | **Performed at early stages** | **Performed at later stages** |
| **2.** | **Carried out without executing the software’s code** | **Carried out by executing software’s code** |
| **3.** | **Takes lesser time** | **Takes a longer time** |
| **4.** | **Has checklists to be followed** | **Generally done with test cases** |

1. **What is review?**

* Software reviews are systematic examinations of software artifacts such as code, design, documentation, test cases, or user feedback, to identify and resolve defects.
* It ensures correctness, completeness and opinions can be taken.

1. **What is a walkthrough?**

* A walkthrough is a peer review process where the author of a document or code explains it to a group to gather feedback and identify issues.
* It doesn’t have the minutes of the meeting.

1. **What is inspection?**

* A formal type of review that involves checking the documents thoroughly before a meeting and is carried out mostly by moderators.
* It is intimated beforehand to the developers through email.

1. **Explain the levels of testing**

The levels of testing are as follows:

**Unit testing**- In this testing, the individual small units of code are tested to validate the expected behaviour.

**Integration testing**- In this testing, 2 or more modules are integrated and tested together to validate the expected behaviour.

**System testing**- in this testing, the whole complete software system is tested and validated for expected behaviour.

**User acceptance testing**- It’s the final testing and tested by the end users to ensure it meets the expected standards and requirements.

1. **Difference between verification and validation**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Verification** | **Validation** |
| **1.** | **It's verifying whether we are building the right system** | **It's validating whether the system we built is right** |
| **2.** | **Static process (no code execution)** | **Dynamic process(code execution is involved)** |

1. **What is integration testing? Explain the top-down, bottom-up up and sandwich approach**

Integration Testing is a level of software testing where individual units or components are combined and tested as a group.

**Top-down approach**: In this approach, testing starts from the topmost module (usually the main module) and progresses downward, one level at a time.

**Bottom-up approach**: This approach starts testing from the lowest-level modules and works its way up to the top-level modules.

**Sandwich approach**: This is a combination of both top-down and bottom-up approaches. It tests both high-level and low-level modules simultaneously, then integrates in the middle.

1. **Difference between stub and driver.**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Stub** | **Driver** |
| **1.** | **A stub is a dummy module that simulates the behaviour of lower-level modules.** | **A driver is a dummy module that simulates the behaviour of higher-level modules.** |
| **2.** | **Top-Down Integration Testing** | **Bottom-up integration testing** |

1. **Explain the following terminology:**
   * **GUI testing-** GUI testing involves checking the graphical elements of an application—like buttons, menus, forms, text boxes, and layouts—to ensure they function and appear correctly.
   * **Functional testing**- Functional testing checks whether the software behaves according to the functional requirements or specifications.
   * **Non-functional testing**- Non-functional testing evaluates how the system performs under certain conditions rather than what it does.
   * **Usability testing-** Usability testing assesses how easy and user-friendly a software application is for end users.
2. **Explain user acceptance testing**

* User Acceptance Testing (UAT) is the final phase of software testing, where real users (or clients) test the software to ensure it meets their business requirements and is ready for production.

1. **Difference between alpha testing and beta testing**

|  |  |  |
| --- | --- | --- |
| **S no** | **Alpha** | **Beta** |
| **1.** | **Testing performed internally by the development or QA team before the product is released to external users.** | **Testing done by external users in a real-world environment to get feedback on the product's usability and functionality.** |
| **2.** | **To catch bugs before releasing to real users** | **to validate the product in real conditions and gather user feedback** |

1. **Explain the below terminology:**

* White box- Testing where the **internal structure**, logic, and code of the software are **fully visible** to the tester.
* Black box- Testing where the **internal code or logic is unknown** to the tester. Focus is only on **inputs and expected**
* Grey box- A **combination of White Box and Black Box** testing. The tester has **partial knowledge** of the internal workings of the application.

1. **Difference between white box and black box**

|  |  |  |
| --- | --- | --- |
| **S NO** | **White box** | **Black box** |
| **1.** | **Testing based on knowledge of the internal code and logic.** | **Testing without any knowledge of the internal structure.** |
| **2.** | **Requires programming knowledge** | **Does not require coding knowledge** |
| **3.** | **How the software works (code, structure, logic)** | **What the software does (input/output behavior)** |

1. **Write a checklist for GUI testing**
   * Layout & Design
   * Buttons
   * Input Fields
   * Navigation
   * Messages & Alerts
   * Images & Icons
   * Responsiveness
2. **Explain usability testing**

Usability testing is a type of software testing that evaluates how easy, efficient, and satisfying a product or system is for real users to use.

1. **Difference between ECP and BVA**

|  |  |  |
| --- | --- | --- |
| **S no** | **ECP** | **BVA** |
| **1.** | **Divides input data into valid and invalid partitions; tests one value from each partition.** | **Focuses on testing the boundary values of input ranges.** |
| **2.** | **Reduce number of test cases while ensuring coverage of input classes.** | **Catch errors that occur at the edges of valid input ranges.** |
| **3.** | **Broad input ranges** | **Specific points at the edges of the range** |
| **4.** | |  | | --- | | **Effective for general data input errors.** |  |  | | --- | |  | | **Effective for off-by-one and boundary-related errors.** |

1. **What is database testing**

* A software testing method that checks the integrity, accuracy, and consistency of data stored in a database. It ensures that data operations (like insert, update, delete, and retrieve) work correctly
* It tests all the DML,DDL and DCL operations.

1. **What is agile methodology**

Agile methodology is a modern, flexible approach to software development that emphasizes collaboration, customer feedback, and frequent delivery of small, working pieces of software.

1. **What are agile frameworks**

* Scrum
* Kanban
* Extreme Programming (XP)
* FDD (Feature Driven Development)
* AUP(Agile unified process)

1. **What is TDD?**

TDD (Test-Driven Development) is a software development approach where tests are written before writing the actual code. It follows a short, repetitive cycle to ensure the code is reliable and meets requirements.

* **Red** – Write a test for a feature that doesn't exist yet → the test fails.
* **Green** – the **minimum code** needed to pass the test.
* **Refactor** – Clean up the code while keeping the test green.

1. **What is BDD?**

BDD (Behaviour-Driven Development) is a software development approach that focuses on defining the behaviour of a system from the end-user's perspective.

1. **What are the limitations of BDD**

* Collaboration Requirement
* Learning Curve
* Time-Consuming Process
* Maintenance Overhead
* Dependency on stakeholders

1. **What are the advantages of BDD**

* Enhanced collaboration
* Focus on user behaviour
* Clearer requirements
* Early issue detection
* Automated testing
* Improved documentation
* Better code quality

1. **What are the advantages of TDD**

* Improved Code Quality
* Fewer Bugs in Production
* Faster Debugging and Easier Refactoring
* Better Code Design and Structure
* High Test Coverage
* Confidence During Changes
* Clear Requirements and Focused Development

1. **What are the limitations of TDD**

* Steep Learning Curve
* Time-Consuming Initially
* Difficult to Write Good Tests First
* Not Suitable for All Types of Projects
* Overhead of Maintaining Tests

1. **Difference between TDD and BDD**

|  |  |  |
| --- | --- | --- |
| **S NO** | **TDD** | **BDD** |
| **1** | **Focuses on code correctness** | **Focuses on system behavior from the user's perspective** |
| **2** | **Mostly, developers write this test** | **Collaboration between developers, testers, and stakeholders** |
| **3** | **Less readable for non-technical stakeholders** | **Designed to improve collaboration and communication** |
| **4** | **Primarily unit-level testing** | **Primarily acceptance level testing** |

1. **What is git?**

* It’s a distributed version control system that is used by multiple developers working in the same project simultaneously.
* It can be used both publicly and privately

1. **Why use git?**

* Tracks code changes (version control)
* Enables team collaboration
* Supports branching and merging
* Works offline (distributed system)
* Keeps code safe and backed up
* Free and open source
* Maintains commit history and logs
* Integrates with CI/CD tools (e.g., GitHub, GitLab)

1. **What is the command to check the version of git?**

* git -- version

1. **What is the command to list all folders in git?**

* ls

1. **What is GitHub?**

* GitHub is a web-based platform used for storing, managing, and collaborating on code projects using Git.
* GitHub is a code hosting platform built on Git, designed to help developers collaborate, share, and manage code efficiently, whether in small teams or large open-source communities.

1. **What is working with git?**

* Working with Git means using Git commands and workflows to manage your code, track changes, and collaborate with others during software development.

1. **Difference between centralised and distributed version control systems.**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Centralized** | **Distributed** |
| **1** | **Single central server** | **Every developer has a full local copy** |
| **2** | **Easier for beginners to learn** | **slightly more complex to learn** |
| **3** | **Centralized — all commit to the same server** | **Decentralized — users can push/pull changes selectively** |
| **4** | **if the server is down, no access** | **Any user has a full backup** |
| **5** | **Internet is needed to commit or update** | **Internet is not required and can commit, log locally** |

1. **What is version control system with eg**

* A Version Control System (VCS) is a tool that helps manage changes to source code or files over time.
* It allows multiple developers to work on a project simultaneously, tracks every modification, and makes it easy to revert to previous versions if needed.
* For eg: SVC, SVM, and Mercury.

1. **What is distributed version control system with eg**

* A **Distributed Version Control System (DVCS)** is a type of version control where **every user has a complete copy** of the project’s entire history (not just the latest version).
* Unlike centralized systems, DVCS lets you **work offline**, make commits locally, and then **push** changes to a shared remote repository when you're ready.
* For eg: Git, Mercurial and Bazaar

1. **What are adv of centralized version control system**

* Simple to use and manage
* Centralized backup of all files
* Easier collaboration and coordination
* Less disk space required on client machines
* Easy to set access control and permissions
* Consistent and unified view of the project
* Changes are always available in one place
* Better suited for small teams or controlled environments

1. **What are the limitations of CVCS.**

* Single point of failure — if the server goes down, no one can work
* Requires constant internet/network connection
* Slower performance for large teams or projects
* Risk of data loss if the central server is not backed up properly
* Limited offline access — can't commit or view history without server access
* Merging and branching are more difficult compared to DVCS
* Less fault-tolerant — all version history is stored in one place

1. **What are adv of distributed version control system**

* Full local copy of the entire repository (history + files)
* Works offline — commits, diffs, and logs can be done without internet
* Faster operations due to local processing
* No single point of failure — every clone is a backup
* Easier branching and merging
* Better collaboration with flexible workflows (pull, push, fork)
* Allows experimental development without affecting others
* Improved data integrity and security through multiple copies

1. **What are the limitations of DVCS.**

* Steeper learning curve for beginners
* Requires more local disk space (full repo copy on each machine)
* More complex setup and management for large teams
* Potentially more complicated conflict resolution in merges
* Initial clone/download can be slow for very large repositories
* Coordination needed to decide which remote repo is the “official” source
* Can be overwhelming without proper workflow discipline

1. **What are the components of git?**

* Working directory
* Staging area
* Commit
* Repository
* Branch

1. **What is repository? Exp the types of repo in git ?**
   * A **repository (repo)** in Git is a **storage space** where your project's files, history, and configuration are tracked using Git.
   * It can be **local** (on your computer) or **remote** (on platforms like GitHub, GitLab, Bitbucket)
2. **Write the best practice for using git**
   * Use Meaningful Commit Messages
   * Commit Often, But Logically
   * Use Branches Strategically
   * Pull Regularly from Remote
   * Test Before You Commit
   * .gitignore Unnecessary Files
3. **Write a command to clone the repo**
   * git clone <repo URL>
4. **Write a command to check the repo status**
   * git status
5. **Write a command to add changes in git**
   * git add <filename> and to add all changes use **git add .**
6. **Write a command to commit a change in git**
   * git commit -m “your commit msg”
7. **Write a command to create a branch in git**

* git branch <branch-name>

1. **Write a command to switch a command**

* git checkout <branch-name> or git switch <branch name>

1. **Write a command to retrieve changes in git**

* git pull

1. **Write a command for merging the branches in git**

* git merge <branch name>

1. **Write a command for pushing the changes in git**

* git push origin <branch name>

1. **What is error handling testing?**

* Error Handling Testing is a type of software testing that verifies whether an application can gracefully handle unexpected or invalid inputs, system failures, and runtime errors without crashing or misbehaving.
* The system responds properly to invalid inputs, system failures, or unexpected conditions
* Meaningful error messages are shown to users

1. **What is link’s coverage? Explain it with types**

* Link Coverage is a software testing metric used primarily in white-box testing. It measures whether all the possible control flow links (edges) in a program’s control flow graph (CFG) have been executed at least once during testing.
* Link Coverage ensures that all logical paths connecting one part of the code to another have been tested
* TYPES:
* 1. Internal Links- Ensure all pages and navigation are working within the site
* 2. External Links- Prevent user frustration if external resources are dead
* 3. Broken Links- Improve user experience and SEO by avoiding dead links.

1. **Explain cookies and sessions.**

* A cookie is a small piece of data stored on the client side (browser). It helps the server remember information between page requests or visits.
* A session stores user data on the server side, and the server keeps track of which session belongs to which user using a unique session ID (usually stored in a cookie).

1. **What is performance testing**

* Performance Testing is a non-functional testing technique used to evaluate how a system behaves under expected or high load, focusing on its speed, responsiveness, stability, and scalability.

1. **What is load testing**

* Load Testing is a type of performance testing that checks how a system behaves under expected or normal workload conditions.
* To determine if the application can handle the expected number of users or transactions efficiently.

1. **What is stress testing**

* Stress Testing evaluates how a system behaves under extreme or beyond-peak load conditions.
* To determine the system's breaking point, how it fails, and how it recovers.

1. **What is volume testing**

* Volume Testing (also called flood testing) checks how the system performs with a large amount of data.
* To assess performance and stability when the database or file size is very large.

1. **What is security testing**

* Security Testing ensures that the software system is protected against threats, unauthorized access, data breaches, and vulnerabilities.
* To identify and fix **security flaws** in the system.

1. **Difference between authentication and authorization**

|  |  |  |
| --- | --- | --- |
| **S NO** | **AUTHENTICATION** | **AUTHORIZATION** |
| **1** | **Confirms who the user is** | **Determines what the user is allowed to do** |
| **2** | **Verifies identity (login)** | **Grants access to resources** |
| **3** | **This happens first before authorization** | **This happens after authentication** |
| **4** | **Eg: Entering username and password** | **Eg, Accessing the admin dashboard or restricted pages** |

1. **What is compatibility testing? explain with types.**

Compatibility Testing is a type of non-functional testing that checks whether your software or website works correctly across different environments — including devices, operating systems, browsers, and networks.

TYPES:

* + 1. **Browser Compatibility Testing**: Checks if the application works across different web browsers and browser versions.
    2. **Operating System Compatibility Testing**: Tests how the application runs on different OS platforms (Windows, macOS, Linux, Android, iOS).
    3. **Device Compatibility Testing**: Verifies performance and appearance across mobile, tablet, desktop, and smart devices.
    4. **Network Compatibility Testing**: Assesses performance over different network types and speeds (e.g., slow mobile data vs fast Wi-Fi).
    5. **Software/Library Compatibility Testing**: Ensures the app works with different versions of browsers, frameworks, databases, drivers, etc.
    6. **Backward Compatibility Testing**: Confirms the new version of the software still works with older systems or data formats.
    7. **Forward Compatibility Testing**: Tests if the current version will work with future versions of related software (less common, harder to perform).

1. **What is recovery testing?**

Recovery Testing is a type of non-functional testing that verifies how well a system can recover from crashes, failures, hardware issues, or other unexpected problems.

To ensure the application can:

* Recover gracefully from unexpected disruptions
* Restore data and functionality after failure
* Resume operations without loss or corruption

1. **What is garbage testing?**

* Garbage Testing involves entering invalid, random, or junk input data into a system to test how well it handles unexpected or meaningless inputs.
* To check the system’s error handling, data validation, and stability under messy or nonsensical input.

1. **What is monkey testing?**

* Monkey Testing is a form of random, unstructured testing where testers (or automated scripts) provide random inputs and actions to see if the application crashes or misbehaves.
* To test the system’s robustness, especially in unusual or chaotic user behavior.

1. **What is ad hoc testing?**

* Ad Hoc Testing is informal, unplanned testing without any documentation, test cases, or structured process. It relies heavily on the tester’s experience and intuition.
* To quickly find defects that structured testing might miss, especially in time-constrained situations.

1. **Difference between positive and negative test case scenario**

|  |  |  |
| --- | --- | --- |
| **S NO** | **POSITIVE** | **NEGATIVE** |
| **1** | **Valid/Expected input is given** | **Invalid/Unexpected input is given** |
| **2** | **Ensures correct behavior** | **Ensures proper error handling** |
| **3** | **Accepts and processes input** | **Rejects or handles input gracefully** |
| **4** | **Eg: Valid login credentials** | **Eg: Empty fields, SQL injection, etc.** |

1. **What is the difference between SDLC and STLC.**

|  |  |  |
| --- | --- | --- |
| **S NO** | **SDLC** | **STLC** |
| **1** | **To build high-quality software efficiently.** | **To verify & validate software for defects and compliance.** |
| **2** | **SDLC is the broader process of software development** | **STLC is a subset of SDLC, specifically for testing.** |
| **3** | **Performed by Developers, Business Analysts, Project Managers** | **Performed by Testers,QA Engineers** |
| **4** | **The final output will be Functional software product** | **The final output will be Test reports, Bug logs** |
| **5** | **Phases in SDLC:**   1. **Requirement Gathering & Analysis** 2. **System Design** 3. **Implementation (Coding)** 4. **Testing (where STLC comes into play)** 5. **Deployment** 6. **Maintenance** | **Phases in STLC:**   1. **Requirement Analysis (Test Planning)** 2. **Test Planning (Strategy & Approach)** 3. **Test Case Development** 4. **Test Environment Setup** 5. **Test Execution** 6. **Test Cycle Closure (Reporting & Analysis)** |

1. **Explain the STLC phases with output**

The Software Testing Life Cycle (STLC) is a structured process that ensures systematic and high-quality testing of software. Each phase has specific inputs, activities, and outputs.

Phases of STLC:

* **Requirement Analysis** – To understand and gather requirements about what needs to be tested.
  + Output- List of testable requirements , Requirement Traceability Matrix (RTM)
* **Test planning**- To define the strategy, resources, and schedule for testing.
  + Output- Test Plan Document
* **Test development**- To create detailed test cases for execution.
* Output- Test data sheets, Test scripts, RTM, Test Cases Document
* **Test Environment Setup** – To Prepare the required hardware/software setup for testing.
* Output- Testable environment
* **Test execution** – To Execute test cases and log defects.
* Output - Test Execution Report, Defect logs , Test logs
* **Test Closure** - Analyze testing results and finalize reports.
* Output - Test Summary Report, Do’s and Don’ts document

1. **What are the activities we can do in requirement analysis phase?**

Objective: Understand what to test by analyzing requirements.

Key Activities:

* Review Requirements (BRD, SRS, User Stories)
* Identify Testable Requirements (Functional & Non-Functional)
* Define Scope of Testing (In-Scope/Out-of-Scope)
* Prepare Requirement Traceability Matrix (RTM)
* Identify Automation Feasibility (if automation is planned)
* Clarify Ambiguities with stakeholders (BA, Dev, Client)
* Categorize Requirements (Priority: High/Medium/Low)

1. **What are the activities we can do in test planning phase phase?**

Objective: Define how to test (strategy, resources, schedule).

Key Activities:

* Prepare Test Plan (Objectives, Scope, Approach)
* Select Testing Tools (Selenium, JIRA, TestNG, etc.)
* Estimate Effort & Cost (Time, Team Size, Budget)
* Define Roles & Responsibilities (QA Lead, Testers, Devs)
* Identify Risks & Mitigations (e.g., tight deadlines)
* Decide Test Metrics (Defect Density, Test Coverage)
* Schedule Test Cycles (Smoke, Regression, UAT)

1. **What are the activities we can do in test case development phase?**

Objective: Create detailed test cases for execution.

Key Activities:

* Write Test Cases (Step-by-Step Scenarios)
* Review Test Cases (Peer Review/BA Sign-off)
* Prioritize Test Cases (Critical Path First)
* Prepare Test Data (Dummy Data, DB Scripts)
* Develop Automation Scripts (if applicable)
* Link Test Cases to RTM (Ensure Coverage)

1. **What are the activities we can do in test environment set up phase?**

Objective: Prepare the environment for testing.

Key Activities:

* Set Up Hardware/Software (Servers, Browsers, Devices)
* Configure Test Tools (Selenium Grid, JMeter)
* Deploy Test Build (Dev/QA Environment)
* Verify Environment Stability (Smoke Test)
* Sync Test Data (DB, APIs, User Credentials)
* Document Environment Details (IPs, Configs)

1. **What are the activities we can do in test execution phase?**

Objective: Execute tests, log defects, and retest.

Key Activities:

* Run Test Cases (Manual/Automated)
* Log Defects (JIRA/Bugzilla with Screenshots/Logs)
* Retest Fixed Defects (Regression Testing)
* Update RTM (Pass/Fail Status)
* Perform Ad-hoc Testing (Unplanned Scenarios)
* Report Daily Progress (Test Execution Status)

1. **What are the activities we can do in test closure phase?**

Objective: Analyze results and conclude testing.

Key Activities:

* Evaluate Test Coverage (Requirement vs. Test Cases)
* Analyze Defects (Root Cause, Trends)
* Prepare Test Summary Report (Metrics, Findings)
* Document Lessons Learned (Improvements for Next Cycle)
* Archive Test Artifacts (Test Cases, Reports)
* Sign-off (QA Lead/Client Approval)

1. **What is bug life cycle?**

* The Bug Life Cycle (or Defect Life Cycle) is a sequence of stages a bug goes through from discovery to resolution. It ensures systematic tracking, prioritization, and fixing of defects in software testing.
* Bug Life Cycle ensures systematic defect management.
* Proper implementation improves software quality and team efficiency.
* Challenges can be mitigated with clear processes and tools.

1. **Exp the phases of bug life cycle in detail?**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Description** | **Responsible Role** |
| **1. New** | Bug is identified and logged for the first time. | Tester/QA Engineer |
| **2.Assigned** | Bug is assigned to a developer for fixing. | QALead/Project Manager |
| **3. Open** | Developer starts analyzing/fixing the bug. | Developer |
| **4. Fixed** | Developer resolves the bug and marks it as "Fixed." | Developer |
| **5. Retest** | QA verifies if the bug is resolved in the new build. | Tester |
| **6. Verified** | Bug is confirmed as fixed and closed. | QA Engineer |
| **7.Reopened** | If the bug persists, it’s reopened and sent back to the developer. | Tester |
| **8. Closed** | Bug is permanently closed after successful retesting. | QA Lead |
| **9. Duplicate** | If the bug is a duplicate of an existing one, it’s marked as such. | QA/Developer |
| **10.Rejected** | Developer may reject the bug if it’s invalid (e.g., not a bug, cannot reproduce). | Developer |
| **11.Deferred** | Bug is postponed to a future release (low priority). | Project Manager |

1. **What are the benefits of bug life cycle?**

* Structured Tracking – Ensures no bug is missed.
* Clear Accountability – Assigns roles (Dev, QA, PM).
* Prioritization – Helps focus on critical bugs first.
* Transparency – Stakeholders can track bug status.
* Quality Improvement – Reduces defect leakage.
* Historical Data – Helps in future test planning.

1. **What are the disadvantages of bug life cycle?**

* Time-Consuming – Multiple stages can delay fixes.
* Overhead – Requires documentation and tool maintenance.
* Miscommunication – Wrong status updates can cause confusion.
* Tool Dependency – Needs a bug-tracking system (JIRA, Bugzilla).
* Human Errors – Incorrect classification (e.g., marking a valid bug as "Rejected").

1. **What are the challenges faced in bug life cycle?**

* Reproducibility Issues – Bugs that can’t be replicated.
* Lack of Details – Poor bug reports slow down fixes.
* Priority Conflicts – Dev vs. QA disagreement on severity.
* Delayed Fixes – Critical bugs deferred due to deadlines.
* Tool Limitations – Some tools lack customization for workflows.

1. **What are the factors/parameters to improve the bug life cycle?**

* Clear Bug Reporting – Include steps, screenshots, logs.
* Automation – Use tools for tracking (JIRA, Azure DevOps).
* Regular Triage Meetings – Discuss priority and progress.
* Standardized Severity/Priority Levels – E.g., Critical, Major, Minor.
* Continuous Feedback – Dev-QA collaboration for faster resolution.
* Training – Teams should understand the lifecycle thoroughly.

1. **What are the key stages of bug life cycle?**

The key stages of bug life cycle are :

* **New** - A bug is identified and logged for the first time.
* **Open**- Developer acknowledges and starts working on the bug.
* **Fixed** - Developer resolves the bug and marks it as "Fixed."
* **Closed**- Bug is permanently closed after successful retesting.

1. **What is test plan? Explain the types?**

A **test plan** is a detailed document that outlines the strategy, objectives, schedule, resources, scope, and approach for testing a software product.

**Types of Test Plans:**

* **Master Test Plan** – High-level document covering all testing activities.
* **Level Test Plan** – For specific test levels (unit, integration, system, etc.).
* **Specific Test Plan** – For specialized testing (e.g., performance, security).

1. **What is master test plan?**

A **Master Test Plan (MTP)** is a high-level document that consolidates all other test plans and defines the overall testing strategy across test levels.

1. **What is the international standard for creating a master test plan**

The international standard is **IEEE 829**), which provides templates and guidelines for test documentation.

1. **What is sanity testing?**

* A **quick, narrow testing** done after receiving a software build to verify that a particular bug fix or feature works as expected and nothing else is broken.
* If sanity testing fails, the build is **rejected for further testing**.

1. **What is smoke testing?**

* A **broad, shallow testing** of the main functionalities to ensure that the build is stable enough for further testing.
* Usually performed by **QA engineers** after receiving a software build.

1. **What is end to end testing?**

* **End-to-End testing** is a **type of testing that verifies the complete flow** of an application from **start to finish**, ensuring all integrated components work together as expected.
* Performed **after system testing** and before user acceptance testing (UAT).
* Simulates **real-world user scenarios** from login to logout.

1. **What is exploratory testing?**

* An **informal, hands-on testing** where testers explore the application without predefined test cases, using creativity and domain knowledge.
* Done **without detailed test cases or scripts**. The tester learns and tests the application at the same time.
* Example: A tester randomly clicks buttons, tries invalid inputs, or navigates pages in an unusual order to find hidden bugs.

1. **What is globalization / internationalization testing?**

* Testing to ensure that the software supports **multiple languages, regions, and cultures** without code changes (e.g., currency, date formats, text expansion).
* Tests that the software **can adapt** to different local formats (date, time, currency, number, calendar, etc.).

1. **What is localization testing?**

* Testing that focuses on checking **region-specific** features like language translation, cultural appropriateness, and UI adaptation for a target locale.
* Conducted **after Globalization Testing** when the app is already internationalized.

1. **What is test suite?**

* A **Test Suite** is a **collection of test cases** that are grouped together and designed to test a specific module, functionality, or feature of a software application.
* A test suite may contain **functional, integration, regression, or smoke test cases**.

1. **What is test bed / test environment?**

* A **Test Bed** (also called a **Test Environment**) is the **complete setup of software, hardware, network, tools, and configurations** required to perform testing on a software application.
* It simulates the real-world environment where the application will run.

1. **What is defect management process?**

* The **Defect Management Process** is a structured approach to **identify, track, resolve, and close** defects (bugs or issues) found during software testing.
* It ensures that defects are **handled systematically** to maintain software quality.

1. **Diff between severity and priority ? exp with eg**

|  |  |  |
| --- | --- | --- |
| **S NO** | **SEVERITY** | **PRIORITY** |
| **1** | |  | | --- | | **How serious the bug is to the application.** |  |  | | --- | |  | | **How urgent it is to fix the bug.** |
| **2** | **Usually set by the tester.** | **Usually set by the developer or project manager.** |
| **3** | **Focuses on the Technical impact.** | **Focuses on Business or customer impact.** |
| **4** | **Types of severity: Critical, Major, Medium, Low.** | **Types of priority: High, Medium, Low.** |
| **5** | **Example: Crash on clicking "Submit" = High severity** | **Spelling mistake in company name on homepage = High priority** |

1. **What is the role of a product owner in an Agile model?**

The **Product Owner** acts as the **voice of the customer** and is responsible for:

* Managing and **prioritizing the product backlog**
* Writing **user stories** and defining **acceptance criteria**
* Ensuring the development team understands **business requirements**
* Participating in **sprint planning, reviews, and demos**
* Making decisions on **what features to build** and when

They **maximize product value** and bridge the gap between stakeholders and the development team.

1. **What is epic?**

An **Epic** is a **large user story or a high-level business requirement** that can be broken down into smaller, manageable user stories.

* Represents **bigger features or initiatives**
* Can span **multiple sprints**
* Example: "Enable user account management" (which includes stories like sign up, login, password reset, etc.)

1. **Define the below terminology**

**User story**: A short, simple description of a feature from the user's point of view. Example: *"As a user, I want to reset my password so that I can access my account."*

**Backlog**: A **prioritized list** of features, enhancements, bug fixes, and tasks to be worked on in future sprints.

**Task**: A **small unit of work** derived from a user story, often technical (e.g., "Write test cases for login feature").

**Sprint**: A **time-boxed iteration** (typically 1–4 weeks) where a set of tasks or user stories are selected and completed by the development team.

1. **Diff between test efficiency and test effectiveness?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Test Efficiency** | **Test Effectiveness** |
| **Definition** | Measures how **efficiently** testing is done. | Measures how **well** testing finds real defects. |
| **Focus** | Time, cost, and resource usage. | Defect detection and prevention. |
| **Formula** | (Test cases executed / Total time taken) | (Defects found during testing / Total defects in product) |
| **Goal** | Minimize effort and maximize output. | Maximize bug detection rate. |
| **Example** | 100 test cases run in 2 hours = efficient | Found 95% of bugs before release = effective |

1. **What is shift left?**

Shift Left is a practice of performing testing and quality activities earlier in the software development lifecycle to detect and fix issues sooner.

* Detect defects early.
* Prevent costly fixes later.
* Improve delivery speed and quality.

1. **What is CI/CD?**

CI/CD is Continuous Integration and Continuous Delivery/Deployment — automating code integration, testing, and release.

* CI: Merge and test code frequently.
* CD: Deliver tested code quickly.
* Improves efficiency and reduces manual errors.

1. **Write down the names of CI/CD tools**

Tools that automate build, test, and deployment processes.

* Jenkins
* GitLab CI
* GitHub Actions
* CircleCI
* Azure DevOps

1. **Difference between CI and CD**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **CI (Continuous Integration)** | **CD (Continuous Delivery /Deployment)** |
| **Definition** | **Practice of frequently merging code changes into a shared repository and running automated tests** | **Practice of automatically delivering tested code to staging or production environments** |
| **Focus** | **Code integration and verification** | **Code delivery and release** |
| **Goal** | **Detect integration issues early** | **Deliver software updates quickly and reliably** |
| **Key Activities** | **Code commits, build creation, unit/integration testing** | **Automated deployment, release approvals, environment configuration** |
| **Outcome** | **Stable and tested codebase in the repository** | **Updated and deployed application for end users** |
| **Frequency** | **Happens multiple times a day with each commit** | **Happens when a release is ready (could be multiple times a day in deployment)** |
| **Tools Examples** | **Jenkins, GitLab CI, GitHub Actions** | **Jenkins, GitLab CD, ArgoCD, Spinnaker** |

1. **Why shift left?**

To test and review earlier in the process for faster, cheaper, better outcomes.

* Early detection of defects.
* Lower cost of fixes.
* Reduces last-minute project delays**.**

1. **What are the adv of early testing and feedback in organisation?**

Benefits gained from testing and reviewing early in the SDLC.

* Improves collaboration between teams.
* Higher quality output.
* Faster releases.
* Less rework and reduced risk.

1. **What are the key practices in left shifting CI/CD?**

Common methods to implement early testing in CI/CD pipelines.

* Automated unit and integration testing.
* Early peer code reviews.
* Static code analysis.
* Security scanning from start.
* Frequent builds.

1. **What is continuous code analysis and static checks?**

Automated review of code to find issues without executing it.

* Detects bugs early.
* Checks style, performance, and vulnerabilities.
* Runs as part of CI pipelines.

1. **What is security vulnerability identification?**

Process of finding weaknesses in software that attackers could exploit.

* Includes outdated libraries and unsafe coding.
* Done via security scanning tools.
* Helps prevent breaches.

1. **What do you mean by automated build?**

Automatic process to compile, package, and prepare code for deployment.

* Reduces manual errors.
* Ensures consistent builds.
* Saves time for developers.

1. **What is continuous quality assurance?**

Ongoing verification that software meets quality standards at all times.

* Uses automated tests and monitoring.
* Detects issues instantly.
* Keeps product ready for release anytime.

1. **What do you mean by automated test reports**

Reports generated automatically after running tests.

* Summarize pass/fail results.
* Highlight defects and coverage.
* Track quality over time.

1. **Define the below terms:**

* **Code review-** A systematic examination of code by peers before merging.
  + Improves maintainability and readability.
  + Detects bugs early.
  + Encourages knowledge sharing**.**
* **Security alerts-** Notifications triggered when vulnerabilities or risks are detected.
  + May come from security scanning tools.
  + Allows quick action to fix issues.
  + Improves software safety.

1. **What is automation testing?**

Automation testing is the process of using software tools to execute test cases automatically, reducing the need for manual intervention.

* Saves time by running tests faster than manual execution.
* Helps in running repetitive and regression tests efficiently.

1. **What are the key components of automation testing?**

These are the essential parts required to design, run, and maintain automated tests.

* **Automation Tool** – Software used to create and run automated tests (e.g., Selenium, Cypress, UFT).
* **Test Scripts** – Code or instructions that perform the automated test steps.
* **Test Data** – Input values used during test execution, often stored externally for flexibility.
* **Test Environment** – Hardware, software, and network setup where automated tests run.
* **Test Execution Engine** – The part of the tool that runs the scripts and interacts with the application.
* **Reporting Mechanism** – Generates execution results, logs, and failure details.
* **Automation Framework** – Structured approach that defines guidelines, coding standards, and reusable components for scripts.

1. **Difference between manual and automation testing**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Manual Testing** | **Automation Testing** |
| **Execution** | **Performed by human testers without tools.** | **Performed using automation tools and scripts.** |
| **Speed** | **Slower, as it requires human effort.** | **Faster, as tests are executed by machines.** |
| **Accuracy** | **Prone to human errors.** | **More accurate and consistent once scripts are correct.** |
| **Cost (Long-Term)** | **Lower initial cost but expensive for repetitive tests.** | **Higher initial cost but cheaper for repetitive tests.** |
| **Suitability** | **Best for exploratory, usability, and ad-hoc testing.** | **Best for regression, load, and repetitive functional tests.** |
| **Maintenance** | **No script maintenance needed.** | **Requires regular script updates with application changes.** |
| **Resource Requirement** | **Needs skilled manual testers.** | **Needs skilled automation engineers and tool knowledge.** |

1. **What are the benefits and challenges of automation testing?**

**Benefits of Automation Testing**

* Faster Execution – Automated tests run much quicker than manual ones.
* Reusability – Test scripts can be reused across multiple test cycles.
* Better Coverage – Can execute a large number of test cases in less time.
* Accuracy – Eliminates human errors during repetitive tests.
* Cost-Effective in Long Run – Initial setup cost is high, but reduces cost over time.
* Supports Continuous Testing – Works well with CI/CD pipelines in Agile/DevOps.
* 24/7 Execution – Can run tests anytime without human presence.
* Data-Driven Testing – Can easily run the same test with multiple data sets.

**Challenges of Automation Testing**

* High Initial Cost – Requires investment in tools, training, and setup.
* Maintenance Effort – Scripts need regular updates when the application changes.
* Not Suitable for All Tests – Some cases (like usability testing) need human judgment.
* Tool Limitations – Some tools may not support all platforms or technologies.
* Skill Requirement – Needs skilled automation engineers to create and manage scripts.
* Time for Setup – Building a robust automation framework takes time.
* False Positives/Negatives – Scripts may produce incorrect results if poorly designed.
* Dependency on Environment – Test failures can occur due to environment issues, not the application.

1. **What are the types of automation testing?**

1. **Functional Testing**

* Definition: Verifies that the application’s functions work as expected.
* Example: Checking if the login feature accepts correct credentials and rejects wrong ones.

**2. Regression Testing**

* Definition: Ensures that existing functionalities still work after changes or updates.
* Example: After adding a new search filter, checking if the cart functionality still works.

**3. Performance Testing**

* Definition: Checks the application’s speed, scalability, and stability.
* Example: Measuring how fast a page loads with 1,000 users accessing it at once.

**4. Load Testing**

* Definition: Evaluates how the system behaves under a specific expected load.
* Example: Testing if the website can handle 500 concurrent users.

**5. Stress Testing**

* Definition: Determines the system’s breaking point by pushing it beyond normal load.
* Example: Increasing traffic until the application crashes.

**6. Smoke Testing**

* Definition: Runs basic tests to check if the major functionalities are working after a build.
* Example: Verifying login, navigation, and data saving right after deployment.

**7. Sanity Testing**

* Definition: Focused testing to check if a specific functionality works after a small change.
* Example: Testing only the “Add to Cart” feature after fixing a related bug.

**8. UI (User Interface) Testing**

* Definition: Verifies that the application’s interface elements work and look as intended.
* Example: Checking button clicks, text fields, and layout alignment.

**9. API Testing**

* Definition: Ensures APIs return correct responses and handle errors properly.
* Example: Sending a GET request and checking if the correct data is returned in JSON.

**10. Security Testing**

* Definition: Validates that the system is secure from vulnerabilities.
* Example: Checking if passwords are encrypted in the database.

1. **What is the difference between regression and performance testing**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Regression Testing** | **Performance Testing** |
| **Purpose** | **Ensures existing features still work after changes, bug fixes, or updates.** | **Measures the system’s speed, stability, and scalability under different loads.** |
| **Focus** | **Functional correctness of the application.** | **Non-functional aspects like response time, throughput, and resource usage.** |
| **When Performed** | **After code changes, enhancements, or bug fixes.** | **Usually before release, during load/stress validation, or periodically.** |
| **Tools Used** | **Selenium, QTP/UFT, Cypress, TestNG.** | **JMeter, LoadRunner, NeoLoad, Gatling.** |
| **Execution Type** | **Can be manual or automated.** | **Mostly automated due to complexity.** |
| **Metrics Measured** | **Pass/fail of functional test cases.** | **Response time, CPU/memory usage, concurrent user handling capacity.** |
| **Outcome** | **Confirms no new bugs were introduced.** | **Identifies bottlenecks and performance issues.** |

1. **How to select the right automation tool?**

Selecting the right automation tool means choosing a testing tool that best fits your project’s technology, budget, and testing needs.

Key Points (Selection Criteria)

1. **Technology Support** – Ensure the tool supports your application’s programming language, frameworks, and platforms.
2. **Ease of Use** – Choose a tool with a user-friendly interface and good documentation.
3. **Integration Capability** – Should integrate with CI/CD pipelines, bug tracking, and version control systems.
4. **Cross-Browser / Cross-Platform Support** – Must handle different browsers, devices, and operating systems if needed.
5. **Cost & Licensing** – Evaluate budget; choose open-source or paid tools based on ROI.
6. **Community & Support** – Prefer tools with active community forums and reliable customer support.
7. **Scalability** – Tool should handle future growth in testing needs.
8. **How to identify test cases for automation?**

Identifying test cases for automation means selecting the scenarios that give maximum value, efficiency, and reusability when automated.

Key Points (How to Identify)

1. **High Repetition** – Choose tests that need to be executed frequently, like regression tests.
2. **Stable Functionality** – Select areas of the application that don’t change often.
3. **Data-Driven Scenarios** – Pick tests that require multiple sets of input data.
4. **Critical Business Flows** – Automate test cases for core functionalities that must always work.
5. **Time-Consuming in Manual** – Automate cases that take a lot of time to test manually.
6. **What are the best practices of automation testing?**

* **Choose the Right Test Cases –** Automate stable, repetitive, and high-value scenarios first.
* **Use a Modular & Reusable Framework** – Break scripts into reusable components to reduce duplication.
* **Follow Naming & Coding Standards** – Keep scripts readable and consistent for all team members.
* **Regular Maintenance** – Update automation scripts when the application changes.
* **Integrate with CI/CD** – Automate execution in pipelines for continuous testing.
* **Keep Tests Independent** – Each test should run on its own without depending on another test’s result.
* **Handle Test Data Efficiently** – Store and manage test data separately from scripts for flexibility.
* **Monitor & Analyze Reports** – Review execution results regularly to track trends and issues.

1. **What are the common mistakes in an automation test project?**

* **Automating Unstable Features** – Writing scripts for features that are still changing frequently.
* **Automating Everything** – Not all tests should be automated; some require manual testing.
* **Lack of Maintenance** – Not updating scripts when the application changes.
* **Poor Test Case Selection** – Choosing low-priority or rarely used scenarios for automation.
* **Ignoring Test Data Management** – Hardcoding data in scripts instead of using external data sources.
* **Skipping Framework Design** – Writing scripts without a proper structure leads to maintenance issues.
* **Inadequate Reporting** – Not having clear, actionable reports for failures.
* **No CI/CD Integration** – Missing the opportunity for continuous testing with automation.

1. **What are the parameters of regression testing?**

* Core business flow
* Frequently used functionality
* High risk/High priority features
* Impact
* ECP/BVA related scenario
* Others(based on current business need, etc.)