www.techglobalschool.com

Soft Skills Mock-2 Questions and Answers

Software Development Life Cycle

What is SDLC?

- SDLC stands for Software Development Life Cycle, which is a systematic process for planning, designing, developing, testing, deploying, and maintaining software applications.
- The primary objective of SDLC is to ensure that the software product meets the desired quality and functionality requirements.

What are the stages of SDLC?

There are typically six main stages in SDLC:
a) Requirement Gathering & Analysis (Planning)
b) Design
c) Implementation/Development
d) Testing
e) Deployment
f) Maintenance

What is the purpose of each stage in SDLC?

a) Requirement Gathering & Analysis: To collect and analyze the client's requirements and establish the project's scope.
b) Design: To create a detailed system and software design based on the requirements.
c) Implementation/Development: To build the software based on the design.
d) Testing: To identify and fix defects in the software and ensure it meets the requirements.
e) Deployment: To deliver and install the software on the client's systems.
f) Maintenance: To provide ongoing support, enhancements, and updates to the software.

a) Requirement Gathering & Analysis:

- During this stage, the project team collaborates with stakeholders, such as clients, end-users, and subject matter experts, to gather and analyze their requirements.
- This involves understanding the business processes, user needs, and constraints of the proposed software solution.
- The output of this stage is typically a Software Requirements Specification (SRS) document, which serves as a foundation for the subsequent stages of the SDLC.

b) Design:

- In the design stage, the project team creates detailed plans for the software system, including its architecture, user interfaces, data structures, and algorithms.
- This involves translating the requirements gathered in the previous stage into a coherent and feasible design.
- The output of this stage includes design documents, such as high-level and low-level design specifications, that guide the development process.

c) Implementation/Development:

- During the implementation stage, the software developers write the actual code for the application, following the design specifications.
- They utilize programming languages, frameworks, and tools to build the software components and integrate them into a functional system.
- The output of this stage is a working software product that is ready for testing.

d) Testing:

- The testing stage involves verifying and validating the software product to ensure it meets the specified requirements and is free from defects.
- This includes various types of testing, such as unit testing, integration testing, system testing, and user acceptance testing (UAT).
- Testers create test plans, test cases, and test scripts to systematically identify and fix issues in the software.
- The output of this stage is a stable, high-quality software product that is ready for deployment.

e) Deployment:

- During the deployment stage, the software product is installed and configured on the client's systems or the target environment.
- This may involve setting up the necessary infrastructure, such as servers and databases, as well as training end-users on how to use the software.
- The output of this stage is a fully operational software product that is accessible to its intended users.

f) Maintenance:

- The maintenance stage is an ongoing process that begins after the software has been deployed.
- It involves providing support to end-users, addressing issues and bugs that may arise, and updating the software to accommodate new requirements or changes in the environment.
- Maintenance also includes enhancing the software with new features, optimizing its performance, and ensuring compatibility with evolving technologies.
- The output of this stage is a continually improved and updated software product that meets the needs of its users over time.

Can you describe the different roles and responsibilities involved in SDLC process?

Customer:

- Responsibilities:
    - Define business objectives and requirements.
    - Provide feedback and review deliverables.
    - Approve changes and project milestones.

Product Owner (PO) / Business Analyst (BA):

- **Responsibilities:**
    - Define and prioritize product backlog items.
    - Act as the liaison between the development team and stakeholders.
    - Ensure the team works on the highest-priority tasks.

Scrum Master:

- **Responsibilities:**
    - Facilitate Scrum ceremonies (stand-ups, sprint planning, retrospectives).
    - Remove impediments and ensure team productivity.
    - Shield the team from external distractions.

Software Architect:

- **Responsibilities:**
    - Define system architecture and design.
    - Create technical specifications and design documents.
    - Ensure the system is scalable, maintainable, and secure.
    - Coordinate with development teams.

Software Developer:

- **Responsibilities:**
    - Write code based on technical specifications.
    - Debug and troubleshoot software issues.
    - Collaborate with the testing team.
    - Follow coding standards and best practices.

Software Tester:

- **Responsibilities:**
    - Develop test plans and test cases.
    - Execute functional, integration, and regression tests.
    - Report and track defects.
    - Ensure software meets quality standards.

UX Designer / UI Designer:

- Responsibilities:
    - Design user interfaces and user experiences.
    - Create wireframes, mockups, and prototypes.
    - Ensure a user-friendly and visually appealing interface.

End User:

- **Responsibilities:**
    - Use the software for all their needs.
    - Work with UI and UX Designers for testing purposes.
    - Report any bugs during maintenance phase.

Project Management Methodologies

What is Agile and Waterfall, and how do they differ?

- Agile and Waterfall are two distinct software development methodologies, each with its own approach to project management, collaboration, and workflow.
- NOTE: A project management methodology is a set of principles and practices that guide you in organizing your projects to ensure their optimum performance.

Agile:

- Agile is an iterative and flexible methodology that emphasizes collaboration, adaptability, and customer feedback throughout the development process.
- It involves breaking a project into smaller increments, known as iterations or sprints, with each delivering a potentially shippable product increment.
- Agile values customer involvement welcomes changes even late in the development cycle, and focuses on delivering value to the customer early and consistently.
- Popular Agile frameworks include Scrum and Kanban.

Waterfall:

- Waterfall is a linear and sequential approach where each phase of the project must be completed before moving on to the next.
- It follows a predefined project plan with distinct phases, such as requirements gathering, design, development, testing, and deployment, in that order.
- Waterfall is often used for projects with well-defined requirements and limited changes expected during development.
- Changes in Waterfall can be costly and challenging to accommodate once a phase is completed.

Key Differences:

- **Approach:** Agile is iterative and incremental, allowing for changes and customer feedback throughout the project, while Waterfall follows a strict, linear sequence.
- **Flexibility:** Agile is highly flexible and adaptive to changes, whereas Waterfall is less accommodating to changes once a phase is completed.
- **Customer Involvement:** Agile encourages ongoing customer involvement and feedback, whereas Waterfall typically involves customers in the initial phase but has less ongoing interaction.
- **Project Delivery:** Agile delivers increments of a product throughout the project, enabling early value delivery, while Waterfall delivers the complete product at the end of the project.
- **Risk Management:** Agile provides better risk management through frequent inspections and adaptability, while Waterfall may have higher risks if requirements are not well-defined.

What is the main difference between Scrum and Kanban?

Scrum:
- Structured framework with defined roles, ceremonies, and artifacts.
- Work is organized into fixed-length iterations (sprints).
- Defines specific roles (Scrum Master, Product Owner, Development Team).
- Relies on estimation (e.g., story points) for capacity planning.
- Discourages changes to sprint scope once it begins.

Kanban:
- Flexible methodology without predefined roles or ceremonies.
- Work is pulled from a backlog as capacity allows no fixed iterations.
- No specific roles defined, promotes self-organization.
- Minimal upfront estimation: work items are pulled without rigid estimation.
- Embraces change and adapts to new priorities at any time.

Simply, Scrum is fixed with 2–4-week sprints and every team member are assigned with specific tasks to complete in the same sprint and it has meetings to synchronize team members while Kanban is not time-boxed and flexible. Team member in Kanban can take a new task whenever they finish the current tasks.

SCRUM - Roles & Ceremonies

Can you explain the Scrum framework and its key components, such as the Scrum Master, Product Owner, and Development Team?

- Scrum is an Agile framework that breaks down large projects into smaller, manageable increments called sprints.
- The key components are the Scrum Master, Product Owner, and Development Team.
- The Scrum Master facilitates the Scrum process, removes impediments, and ensures the team follows the agreed-upon practices.
- The Product Owner defines the product vision, prioritizes the backlog, and represents the stakeholders.
- The Development Team is responsible for delivering potentially shippable increments at the end of each sprint.

How does Scrum differ from other Agile methodologies?

- Scrum is an Agile framework that emphasizes teamwork, accountability, and iterative progress towards a common goal.
- It differs from other Agile methodologies in that it places a heavy emphasis on regular communication and collaboration between team members, as well as on frequent inspection and adaptation of the product and the process.

How do you differentiate between a project manager and a product owner in Agile methodology?

- In Agile, a project manager focuses on managing the project timeline, resources, and budget, ensuring that the team delivers the project as planned.
- A product owner, on the other hand, is responsible for defining the product vision, prioritizing the backlog, and ensuring that the team delivers value to the end-users and stakeholders.

What are the Scrum ceremonies, artifacts, and roles?

- The Scrum framework consists of several ceremonies, artifacts, and roles.
- Ceremonies include the Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective.
- Artifacts include the Product Backlog, Sprint Backlog, and Increment.
- Roles include the Product Owner, Scrum Master, and Development Team.

What is Epic and who is responsible for creating and managing Epics?

- **Definition:** An Epic in JIRA represents a large and high-level body of work that can be broken down into smaller, more manageable pieces of work called User Stories.
- **Responsibility:** Epics are typically created and managed by the Product Owner or a person responsible for defining high-level project objectives. Development teams may also contribute to Epic refinement.

What is User Story and who is responsible for creating and managing User Stories?

- **Definition:** A User Story is a smaller, more detailed piece of work that describes a single piece of functionality from an end user's perspective. User Stories are often used to describe the features or functionality that make up an Epic.
- **Responsibility:** User Stories are typically created and managed by the Development Team and the Product Owner collaboratively. Development teams often provide estimates for User Stories during sprint planning.

What are acceptance criteria?

- **Definition:** Acceptance Criteria are specific conditions or requirements that must be met for a User Story to be considered complete. They serve as a definition of "done" for the story.
- **Responsibility:** Acceptance Criteria are defined collaboratively by the Development Team and the Product Owner. They ensure a shared understanding of what it means for a User Story to be finished.

Can you explain the difference between a product backlog and a sprint backlog?

- The product backlog contains all the features, enhancements, and bug fixes that the team plans to deliver over the course of the product development.
- The sprint backlog is a subset of the product backlog and contains only the items that the team commits to delivering during the upcoming sprint.

How do you prioritize items in the Product Backlog, and what factors should be considered?

- Prioritization of Product Backlog items is typically based on factors such as business value, risk, dependencies, and technical complexity.
- Companies use different techniques for prioritization.

How do Product Backlog, Sprint Backlog, and Increment contribute to the Scrum framework?

- The Product Backlog drives the product's evolution and helps the Scrum Team understand what needs to be done.
- The Sprint Backlog provides a clear plan for the current Sprint and ensures that the team focuses on the highest priority items.
- The Increment represents the progress made during each Sprint and serves as a foundation for continuous improvement.

What is a Sprint?

- A sprint is a time-boxed iteration in Agile project management and development.
- It is a fundamental concept within Agile methodologies, particularly in Scrum.
- Sprints provide a structured framework for teams to work on a defined set of tasks or User Stories within a fixed time, typically ranging from one to four weeks, with two weeks being the most common duration.

What is a Sprint Goal, and how does it relate to the Sprint Backlog?

- A Sprint Goal is a short, high-level description of the desired outcome for a Sprint.
- It provides a shared understanding of the Sprint's purpose and guides the team in selecting and prioritizing the Product Backlog items for the Sprint Backlog.

What is Burndown Chart?

- A burndown chart is a graphical representation used in Agile project management, particularly in Scrum, to visualize and track the progress of work during a sprint or an iteration.
- It helps teams monitor whether they are on track to complete their planned work within the defined timeframe.

Scrum Artifacts and Other Meetings

What are the three primary Scrum artifacts, and what is the purpose of each?

- The three primary Scrum artifacts are the Product Backlog, Sprint Backlog, and Increment.
- The Product Backlog is a prioritized list of features, enhancements, and fixes that are planned for the product.
- The Sprint Backlog contains the selected Product Backlog items for the current Sprint and the plan to deliver them.
- The Increment is the sum of all completed Product Backlog items in a Sprint, resulting in a potentially releasable product.

What are Scrum Ceremonies?

- Scrum ceremonies are essential events or meetings in the Scrum framework, an Agile project management methodology.
- These ceremonies help teams work together efficiently, maintain transparency, and adapt to changing requirements.
- There are several Scrum ceremonies, each serving a specific purpose within the Scrum framework:
  - **Sprint Planning:** This ceremony marks the start of each sprint.
    - The team selects User Stories or tasks from the product backlog and decides what can be accomplished during the sprint.
    - The outcome of this ceremony is the sprint backlog, a list of items to be worked on during the sprint.
  - **Daily Stand-up (Daily Scrum):** This is a short, daily meeting where team members share progress updates, discuss challenges, and plan for the day.
    - It promotes collaboration and ensures everyone is aligned on the work.
    - **Questions:** What did you do yesterday? What will you do today? Any impediments?
  - **Sprint Review (DEMO):** At the end of each sprint, the team conducts a sprint review to demonstrate the completed work to stakeholders and gather feedback.
    - It provides an opportunity to inspect the increment and make decisions about the product's future.
  - **Sprint Retrospective**: Following the sprint review, the team holds a retrospective meeting.
    - In this meeting, team members reflect on the sprint, discuss what went well and what could be improved, and identify actions for continuous improvement.
  - **Backlog Refinement (Grooming):** This is not a single event but a continuous process.

- It involves regularly reviewing and updating the product backlog, clarifying User Stories, and ensuring that items are well-defined and prioritized for future sprints.

Can you explain the process of refining the Product Backlog, and how does the Scrum Team ensure that items are "Ready" for an upcoming Sprint?

- Refining the Product Backlog involves adding detail, estimating effort, and prioritizing items.
- The Scrum Team collaborates on this process, ensuring that items are clear, actionable, and appropriately sized. An item is considered "Ready" when it meets the team's **Definition of Ready**, which typically includes having clear acceptance criteria, dependencies resolved, and a reasonable estimate of effort.
- **Estimation:** To estimate the effort required to complete each backlog item, often using techniques like story points or time-based estimates.

What is the definition of "Done" in Scrum, and how does it impact the Increment?

- The **Definition of Done (DoD)** is a shared understanding of the criteria that a Product Backlog item must meet to be considered complete.
- The DoD ensures consistent quality and provides a clear goal for each Increment.
- An Increment is considered potentially releasable when all its items meet the DoD.

What is Three-Amigos Meeting?

- The 3-Amigos Meeting is a collaborative and informal meeting that typically involves three key roles in Agile development:
  - **Business Analyst or Product Owner**: Represents the business or customer perspective and provides information about the desired functionality and user requirements.
  - **Developer**: Represents the technical perspective and can clarify how the functionality will be implemented from a technical standpoint.
  - **Tester:** Represents the quality assurance perspective and focuses on understanding how the functionality will be tested and validated.
- The purpose of the 3-Amigos Meeting is to foster shared understanding and collaboration among these three essential roles.
- During the meeting, they review and discuss User Stories or features to ensure that they are well-defined, feasible, and testable.

- This collaboration helps identify potential issues or ambiguities early in the development process, reducing the likelihood of misunderstandings, rework, and defects.
- Key elements of a 3-Amigos Meeting include:
    - Reviewing User Stories or requirements.
    - Asking questions and seeking clarification.
    - Discussing technical considerations.
    - Defining acceptance criteria.
    - Exploring potential test scenarios.

What is Parking Lot Meeting?

- A Parking Lot Meeting is a term often used in project management, particularly in Agile and Scrum environments.
- It refers to a meeting or discussion where team members or stakeholders set aside topics or issues that are not directly related to the current agenda or sprint but need to be addressed in the future.

Software Testing Life Cycle

What is STLC?

- STLC is a fundamental part of the Software Development Life Cycle (SDLC), but STLC consists of only the testing phases.
- STLC starts as soon as requirements are defined, or software requirement document is shared by stakeholders.
- STLC yields a step-by-step process to ensure quality software.
- In the initial stages of STLC, while the software product or the application is being developed, the testing team analyzes and defines the scope of testing, entry and exit criteria, and also test cases.
- It helps to reduce the test cycle time and also enhances product quality.
- As soon as the development phase is over, the testing team is ready with test cases and starts the execution.
- This helps in finding bugs in the early phase.

What are the stages of STLC?

There are typically six main stages in STLC:
a) Requirement Analysis
b) Test Planning
c) Test Case Development
d) Test Environment Setup

e) Test Execution
f) Test Cycle Closure

What is the purpose of each stage in STLC?

a) Requirement Analysis: Analyze project requirements to understand what needs to be tested. They identify testable requirements and create a test plan.
b) Test Planning: Develop a comprehensive test strategy and plan. This plan outlines the testing scope, objectives, resources, schedule, and test environment requirements. It also defines the test techniques and methods to be used.
c) Test Case Development: Test cases and test scripts are created based on the requirements and test plan. Test data, if necessary, is also prepared. The goal is to ensure that testing covers all aspects of the software's functionality.
d) Test Environment Setup: The test environment, which includes hardware, software, network configurations, and test tools, is set up to simulate the production environment in which the software will be used.
e) Test Execution: Testers execute the test cases in the specified test environment. They record test results, report defects, and monitor the testing process to ensure that it follows the test plan.
f) Test Cycle Closure: Close the testing activities. It includes evaluating whether the testing goals and objectives have been met, archiving test assets, and conducting a review to identify process improvements.

a) Requirement Analysis

- Requirement Analysis is the first step of the Software Testing Life Cycle (STLC).
- In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then the quality assurance team meets with the stakeholders to better understand the detailed knowledge of requirements.
- The activities that take place during the Requirement Analysis stage include:
  - Reviewing the software requirements document (SRD) and other related documents
  - Identifying any ambiguities or inconsistencies in the requirements
  - Identifying any missing or incomplete requirements
  - Identifying any potential risks or issues that may impact the testing process.
  - Creating a requirement traceability matrix (RTM) to map requirements to test cases.
- At the end of this stage, the testing team should have a clear understanding of the software requirements and should have identified any potential issues that may impact the testing process.
- This will help to ensure that the testing process is focused on the most important areas of the software and that the testing team is able to deliver high-quality results.

b) Test Planning

- Test Planning is the most efficient phase of the software testing life cycle where all testing plans are defined.
- In this phase manager of the testing team calculates the estimated effort and cost for the testing work.
- This phase gets started once the requirement-gathering phase is completed.
- The activities that take place during the Test Planning stage include:
    - Identifying the testing objectives and scope
    - Developing a test strategy
    - Identifying the testing environment and resources needed
    - Identifying the test cases that will be executed and the test data that will be used
    - Assigning roles and responsibilities to the testing team
    - Reviewing and approving the test plan
- At the end of this stage, the testing team should have a detailed plan for the testing activities that will be performed, and a clear understanding of the testing objectives, scope, and deliverables.
- This will help to ensure that the testing process is well-organized and that the testing team is able to deliver high-quality results.

c) Test Case Development

- The test case development phase gets started once the test planning phase is completed.
- In this phase testing team notes down the detailed test cases.
- The testing team also prepares the required test data for the testing.
- When the test cases are prepared then they are reviewed by the quality assurance team.
- The activities that take place during the Test Case Development stage include:
    - Identifying the test cases that will be developed
    - Writing test cases that are clear, concise, and easy to understand
    - Creating test data and test scenarios that will be used in the test cases
    - Identifying the expected results for each test case
    - Reviewing and validating the test cases
    - Updating the requirement traceability matrix (RTM) to map requirements to test cases
- At the end of this stage, the testing team should have a set of comprehensive and accurate test cases that provide adequate coverage of the software or application.
- This will help to ensure that the testing process is thorough and that any potential issues are identified and addressed before the software is released.

d) Test Environment Setup

- Test environment setup is a vital part of the STLC.
- Basically, the test environment decides the conditions on which software is tested.
- This is independent activity and can be started along with test case development.
- In this process, the testing team is not involved.
- Either the developer or the customer creates the testing environment.

e) Test Execution

- After the test case development and test environment setup test execution phase gets started.
- In this phase testing team starts executing test cases based on prepared test cases in the earlier step.
- The activities that take place during the test execution stage of the Software Testing Life Cycle (STLC) include:
    - Test data is prepared and loaded into the system for test execution
    - The test cases and scripts created in the test design stage are run against the software application to identify any defects or issues.
    - The results of the test execution are analyzed to determine the software's performance and identify any defects or issues.
    - Any defects or issues that are found during test execution are logged in a defect tracking system, along with details such as the severity, priority, and description of the issue.
    - Any defects that are identified during test execution are retested to ensure that they have been fixed correctly.
    - Test results are documented and reported to the relevant stakeholders.
- It is important to note that test execution is an iterative process and may need to be repeated multiple times until all identified defects are fixed and the software is deemed fit for release.

f) Test Cycle Closure

- Test closure is the final stage of the Software Testing Life Cycle (STLC) where all testing-related activities are completed and documented.
- The main objective of the test closure stage is to ensure that all testing-related activities have been completed and that the software is ready for release.
- At the end of the test closure stage, the testing team should have a clear understanding of the software's quality and reliability, and any defects or issues that were identified during testing should have been resolved.
- The test closure stage also includes documenting the testing process and any lessons learned so that they can be used to improve future testing processes.
- Test closure is the final stage of the Software Testing Life Cycle (STLC) where all testing-related activities are completed and documented.
- It is important to note that test closure is not just about documenting the testing process, but also about ensuring that all relevant information is shared, and any lessons learned are captured for future reference. The goal of test closure is to ensure that the software is ready for release and that the testing process has been conducted in an organized and efficient manner.

Bug Life Cycle

What is a Bug and who creates a Bug ticket?

- **Definition:** A Bug/Defect represents an issue or problem in the software that needs to be addressed and fixed.
    - Bugs can be identified during testing or by end users.
- **Responsibility:** Anyone on the team can create a Bug ticket, but it's often testers or quality assurance professionals who identify and report them.
- The responsibility for fixing the Bug typically falls on the Development Team.

What is Bug Life Cycle?

- The Bug Life Cycle, also known as the Defect Life Cycle, is a systematic process that describes the various stages a software defect or bug goes through, from its identification to its resolution.
- The Bug Life Cycle is an integral part of software testing and quality assurance, ensuring that identified defects are properly managed and tracked throughout their lifecycle.
- The specific stages of the Bug Life Cycle can vary depending on the organization and the issue-tracking system in use, but the following are common stages:
    - **New:** This is the initial stage, where a tester or team member identifies and reports a defect. At this point, the bug is labeled as "New" and enters the tracking system.
    - **Assigned:** Once a bug is reported, it is assigned to the appropriate team or individual responsible for its resolution, typically a developer or development team. The bug's status is updated to "Assigned."
    - **Open:** When the developer begins working on the bug, the status is changed to "Open." This indicates that the issue is actively being addressed.
    - **In Progress:** The bug may be further classified as "In Progress" to signify that development or code changes are actively being made to fix the issue.
    - **Fixed:** After the developer believes they have resolved the bug, they mark it as "Fixed." The fix is typically committed to the code repository.
    - **Ready for Testing:** In this stage, the issue is handed over to the testing team. Testers verify the fix and validate whether the bug has been successfully addressed.
    - **Reopened:** If the testing team finds that the issue still exists or the bug is not completely resolved, they mark it as "Reopened" and provide additional details on why the issue persists.
    - **Retest:** The developer makes further fixes to the bug based on the testing team's feedback. After these changes, the issue is marked for retesting.
    - **Verified:** Once the testing team confirms that the issue has been successfully fixed and no longer exists in the software, they change the bug's status to "Verified."

- **Closed:** The final stage is "Closed." This indicates that the bug has been resolved and validated. It is now considered officially closed, and the issue-tracking system is updated accordingly.

What needs to be included in a Bug ticket?

- A clear and concise summary of the issue.
- A detailed description of the problem, including steps to reproduce.
- Priority and severity levels.
- Attachments (screenshots, logs, etc.) for better understanding.
- Information on the environment where the issue was found.

What is the difference between priority vs severity of a bug?

1. **Severity:**
    a. **Definition:** Severity refers to the impact or seriousness of a bug on the software's functionality and end-users.
    b. **Focus:** It assesses how critical the bug is in terms of its potential to cause harm or disruption to the system and its users.
    c. **Classification:** Severity is typically categorized into levels such as "Critical," "Major," "Minor," and "Cosmetic" to indicate the bug's impact on the application.
    d. **Example:** A bug that causes a system crash would have a "Critical" severity, while a minor cosmetic issue like a typo might have a "Minor" severity.
2. **Priority:**
    a. **Definition:** Priority indicates the order in which a bug should be fixed or addressed based on factors like business needs, deadlines, and resource availability.
    b. **Focus:** It considers how soon the issue needs to be resolved and the urgency of the fix.
    c. **Classification:** Priority levels can range from "Immediate" or "Urgent" for issues that require immediate attention to "Low" for less critical issues that can be addressed later.
    d. **Example:** Even though a bug may have a "Critical" severity, it might have a "Medium" priority if it doesn't need an immediate fix because it doesn't impact critical business processes.

Why do we need Bug Tracking Tools?

- Bug tracking tools (e.g., Jira, Bugzilla) play a crucial role in managing the bug life cycle.
- These tools help teams track, prioritize, and communicate about bugs.

Test Cases

What is a Test Case and what are Test Case components?

- Test cases are detailed steps that testers follow to verify the functionality of a user story.
- They include preconditions, actions, expected results, and sometimes, test data.
- They help testers systematically assess whether the software behaves as intended and meets the defined requirements.
- **Components of a Test Case:** A typical test case comprises several components:
    - **Test Case ID:** A unique identifier for the test case.
    - **Test Case Title/Name:** A descriptive title or name that provides an overview of the test case's purpose.
    - **Test Objective:** A brief description of the test case's objective, including what it aims to test or achieve.
    - **Preconditions:** Conditions or requirements that must be satisfied before the test case can be executed.
    - **Test Steps:** A step-by-step sequence of actions or interactions to be performed to execute the test. This includes input data, specific operations, and expected outcomes.
    - **Expected Results:** The anticipated results, including what the software should display or how it should behave after the test case is executed.
    - **Actual Results:** A space to record the actual outcomes observed during test execution.
    - **Status:** The current status of the test case, which may be "Pass," "Fail," "Not Executed," or similar.

What are differences between Test Scenario, Test Case and Test Script?

- **Test Scenario:** A test scenario is a high-level test objective or goal.
    - For example, a test scenario might be "User should be able to login,".
- **Test Case:** A test case is a detailed set of instructions specifying a single test scenario, including the specific inputs, actions, and expected outcomes.
- **Test Script:** A test script is a set of executable instructions or code used for automated testing. Test scripts are often derived from test cases and are used to automate the execution of test cases.

What are the benefits of writing Test Cases?

- Test cases ensures that all relevant scenarios are considered.
- Test cases allow testers to reproduce specific scenarios consistently.
- Test cases help testers work more efficiently by providing clear instructions.
- Test cases can uncover defects early in the development process.
- Test cases help trace test coverage back to specific requirements.

What are positive and negative Test Cases?

- **Positive Test Cases:** These test cases validate that the software behaves correctly under normal or expected conditions. They confirm that the software meets its functional requirements.
    - For a login system, a positive test case would be when a user enters the correct username and password and successfully logs into the application.
- **Negative Test Cases:** Negative test cases are designed to check how the software handles unexpected or incorrect inputs or scenarios. They aim to expose flaws or vulnerabilities in the software.
    - In the same login system, a negative test case would be when a user enters an incorrect password and receives an error message indicating a failed login attempt.

What is Happy Path Testing?

- Happy Path Testing is a testing approach that focuses on testing a software application under normal and expected conditions.
- It involves executing positive test cases to ensure that the software follows the intended flow and functions correctly when everything goes well.

What is Boundary or Edge Test Cases?

- Boundary or Edge Test Cases involve testing the limits of software functionality, including the boundary conditions or the extreme values of input data.
- They are designed to identify issues related to data validation, range, and limits.
- Examples;
    - If a system requires a minimum password length of 8 characters, a boundary test case would involve testing with a password of exactly 8 characters to ensure it's accepted.
    - In a file upload feature, if the maximum allowed file size is 10 MB, boundary testing would involve attempting to upload a file that is exactly 10 MB to verify it's accepted.
    - If a numerical input field has a valid range of 1 to 100, boundary tests would include entering the values 1 and 100 to check if both are accepted.

- In a text input field, if the maximum character limit is 100 characters, a boundary test case would include entering exactly 100 characters to verify they are accepted.
- If a website has an age verification process, testing would include entering an age just below and just above the legal age limit to confirm that it allows or denies access appropriately.

What is Requirement Traceability Matrix (RTM)?

- A Requirement Traceability Matrix (RTM) is a structured document that serves as a critical link between the requirements of a software project and the corresponding test cases designed to verify or validate those requirements.
- The RTM helps ensure that each requirement is adequately tested, and it provides a clear understanding of which requirements have been covered by testing.

Software Testing Types

What is Software Testing and what are the different types of Software Testing?

- Software Testing is a process of evaluating a software application to find defects and ensure its quality.
- There are various types of software testing, including Functional Testing, Non-functional Testing (Performance, Security, etc.), Manual Testing, Automated Testing, and more.

What are white-box and black-box testing? Give some examples.

- White-box testing involves examining the internal structure and code of the software. Testers need knowledge of the code to design test cases.
- Black-box testing, on the other hand, focuses on testing the software's functionality without knowledge of its internal code.
- Examples of white-box testing include Unit Testing and Code Review.
- Examples of black-box testing include Functional Testing and User Acceptance Testing.

What is Unit Testing and Integration Testing and what are their differences?

- Unit Testing checks individual components or functions of the software in isolation.
- Integration Testing verifies how these components work together.
- Unit Testing focuses on small units, while Integration Testing ensures that the integrated parts function correctly as a whole.
- Unit Testing is performed by developers while Integration Testing is performed by testers.

What is Smoke Testing and Regression Testing and what are their differences?

- Smoke Testing is a quick check to ensure that the most critical functions of the software work after changes or new features are added.
- Regression Testing, on the other hand, retests the entire application to ensure that new changes have not affected existing functionalities.

What is Sanity Testing and when to perform it?

- Sanity Testing is a narrow form of regression testing.
- It checks specific areas of the software that were modified or fixed after a defect is found.
- It's performed when minor changes are made to ensure that those changes haven't introduced new issues.

What is Performance Testing and why is it needed? What are different types of Performance Testing?

- Performance Testing assesses the software's speed, responsiveness, and stability under different conditions.
- It's needed to ensure that the software meets performance requirements.
- Types of Performance Testing include Load Testing, Stress Testing, Spike Testing, Endurance Testing and Scalability Testing, among others.

What is End-to-End Testing and what are the benefits it provides?

- End-to-End Testing checks the entire flow of a software application, from start to finish, including all interconnected components.
- It verifies the software's readiness for real-world usage and ensures that all parts work together.

What is User Acceptance Testing (UAT) and when to perform it?

- User Acceptance Testing (UAT) is conducted by end-users to ensure that the software meets their requirements and works as intended.
- It's performed in the final stages of testing, just before the software's release.

What is Alpha Testing and Beta Testing and who performs them?

- Alpha Testing is conducted by the development team within the organization.
- Beta Testing is performed by external users or customers.
- Both test phases help identify issues before a public release.
- They are both part of User Acceptance Testing (UAT).

What is A/B Testing and why is it needed?

- A/B Testing compares two versions of a web page or application to determine which one performs better.
- It's used to make data-driven decisions for optimizing user experience, increasing conversions, and improving product features.

What is Monkey (Ad Hoc) Testing and what are the benefits it provides?

- Monkey (Ad Hoc) Testing is informal, exploratory testing without predefined test cases.
- Testers interact with the software randomly to discover issues that might not be covered by formal testing.
- It helps uncover unexpected defects and assess the software's robustness.

What is Security Testing and what are the benefits it provides?

- It is a type of testing performed by a special team. Any hacking method can penetrate the system.
- Security Testing is done to check how the software, application, or website is secure from internal and/or external threats.
- This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are.
- It also checks how software behaves for any hacker's attack & malicious programs and how software is maintained for data security after such a hacker attack.