1) **List down the models of SDLC?**

The Software Development Life Cycle (SDLC) consists of various models or methodologies that guide the process of developing software. Here are some of the most used SDLC models:

1. **Waterfall Model:**

This is a linear and sequential model where each phase must be completed before moving on to the next.

Phases: Requirements, Design, Implementation, Testing, Deployment, Maintenance.

2. **Agile Model:**

Agile is an iterative and incremental approach that focuses on collaboration and flexibility.

Common frameworks include Scrum, Kanban, and Extreme Programming (XP).

3. **Scrum:**

A specific Agile framework that emphasizes small, cross-functional teams and time-boxed iterations (sprints).

4. **Kanban:**

A visual workflow management method that emphasizes continuous delivery and optimization of processes.

5. **Iterative Model:**

In this model, software is developed in smaller increments or iterations, with each iteration building on the previous one.

6. **Spiral Model:**

This model combines elements of the Waterfall and Iterative models and includes risk analysis and prototyping in each cycle.

7. **V-Model (Validation and Verification Model):**

Like the Waterfall model but with a focus on testing at each stage of development.

8. **Big Bang Model:**

In this model, there is no defined process or structure, and development occurs in an ad-hoc manner.

9. **RAD (Rapid Application Development):**

Emphasizes rapid prototyping and quick feedback from end-users.

10. **Incremental Model:**

Software is built in small, manageable parts, with each part being developed and delivered separately.

11. **DevOps:**

Although not a traditional SDLC model, DevOps emphasizes collaboration between development and operations teams to automate and streamline the software delivery process.

12. **Lean Software Development:**

Focuses on delivering value to customers while eliminating waste in the development process.

13. **Feature-Driven Development (FDD):**

A model that breaks down software development into specific features or functionalities.

14. **Unified Process (UP):**

A flexible, iterative model that focuses on the architecture-centric development of software systems.

15. **Prototype Model:**

Involves creating a basic working model of the software to gather feedback and refine requirements.

16. **Evolutionary Model:**

Software is developed in stages, with each stage building upon the previous one based on changing requirements.

17. **3-Tier Architecture:**

A model that divides the software into three tiers: presentation, application logic, and data storage.

18. **Component-Based Development (CBD):**

Involves building software by assembling pre-made components or modules.

19. **Rational Unified Process (RUP):**

An iterative software development process framework that includes guidelines, templates, and best practices.

20. Agile Unified Process (AUP):

- An agile version of the Unified Process, combining iterative development with agile principles.

**2)** **What is STLC? Explain the stages of STLC?**

STLC, or Software Testing Life Cycle, is a systematic process for planning, designing, executing, and managing software testing activities throughout the development of a software application. The phases of STLC are as follows:

1. **Requirement Analysis Phase:**

: The primary goal of this phase is to understand the project requirements and determine the testing scope.

Activities:

Review and analyze project documentation, including the requirements specification, design documents, and user stories.

* Identify testable requirements and create a traceability matrix to link requirements to test cases.
* Identify risks and dependencies that may affect testing.
* Develop a high-level test strategy and test plan outline.

2. **Test Planning Phase:**

: Create a detailed test plan that outlines the scope, s, resources, schedule, and test deliverables.

Activities

* Define test s, entry and exit criteria, and test deliverables.
  + Develop a comprehensive test strategy and test plan.
  + Identify the testing environment, including hardware, software, and data requirements.
  + Allocate resources, including test team members and testing tools.
  + Define the test schedule and milestones.
  + Obtain approval for the test plan from relevant stakeholders.

3. **Test Design Phase:**

: Create detailed test cases and test scripts based on the test plan and requirements.

Activities:

* + Identify test scenarios and create test cases that cover various aspects of the application, such as functionality, usability, performance, and security.
  + Design test data and test scripts.
  + Specify test input values and expected outcomes.
  + Review and validate test cases with stakeholders to ensure completeness and accuracy.
  + Create test data and prepare the test environment.

4. **Test Environment Setup Phase:**

: Prepare the testing environment, including hardware, software, and data, to execute test cases.

Activities:

* + Install and configure the necessary hardware and software.
  + Set up test databases and test data.
  + Verify that the test environment matches the production environment as closely as possible.
  + Perform smoke tests to ensure the environment is stable and ready for testing.

5. **Test Execution Phase:**

: Execute test cases and document the results.

Activities:

* Execute test cases as per the test plan.
* Record test results, including pass/fail status and any defects found.
* Report defects to the development team using a defect tracking system.
* Retest defects after they are fixed.
* Conduct regression testing to ensure that new changes haven't affected existing functionality.
* Monitor and manage test execution progress.

6. **Defect Reporting and Tracking Phase:**

: Document, report, and manage defects throughout the testing process.

Activities:

* Report defects in a structured format, including details like steps to reproduce, severity, and priority.
* Assign defects to the development team for resolution.
* Track the status of defects and ensure they are fixed and verified.
* Communicate defect status to stakeholders.

7. **Test Closure Phase:**

Formalize the testing process and assess the overall quality of the software.

Activities:

* + Summarize and analyze test results.
  + Prepare test summary reports and metrics.
  + Evaluate whether the testing s have been met.
  + Obtain formal sign-off from stakeholders to conclude the testing phase.
  + Archive test artifacts for future reference.

8. **Post-Implementation Review:**

Evaluate the effectiveness of the testing process and identify areas for improvement.

Activities:

* Conduct a review meeting to discuss the testing process and gather feedback from the team.
* Identify lessons learned and best practices.
* Update test processes and documentation based on the review

**3) As a test lead for web-based applications, your manager has asked to identify and explain the different risk factors that should be included in the test plan. Can you provide the list of the potential risks and their s that would be included in the test plan?**

**Scope Creep:**

: This risk involves uncontrolled changes or additions to the project's scope. It can lead to resource constraints and affect testing timelines. To mitigate this risk, clearly define the project scope and establish a change control process.

**Incomplete Requirements:**

: Insufficient or unclear requirements can result in incomplete testing and missed defects. Ensure that all requirements are well-documented and understood by the testing team.

**Resource Constraints:**

: Limited availability of testing resources, such as skilled testers, test environments, or testing tools, can impact the test execution and coverage. Allocate resources adequately and plan for contingencies.

**Data Security:**

: Security breaches can lead to data loss or unauthorized access. Identify potential security vulnerabilities and conduct security testing to protect sensitive information.

**Browser and Device Compatibility:**

: Different browsers and devices may render web pages differently, causing issues in user experience. Test on various browsers and devices to ensure compatibility.

**Performance and Scalability:**

: Slow page load times or system crashes under heavy user load can result in a poor user experience. Conduct performance testing to ensure the application can handle expected traffic.

**Third-party Integrations:**

: Dependencies on third-party APIs or services can introduce risks related to their availability and reliability. Have contingency plans for when third-party services fail.

**Usability and Accessibility:**

: Inadequate usability or accessibility can alienate users and result in a loss of customers or legal issues. Include usability and accessibility testing in your plan.

**Regression Testing:**

: Frequent code changes can introduce new defects or break existing functionality. Implement a robust regression testing strategy to catch these issues early.

**Network and Connectivity Issues:**

: Users may have varying network conditions, which can affect the application's performance. Test for usability and functionality under different network scenarios.

**Data Integrity:**

: Ensure that data input, storage, and retrieval processes maintain data integrity, preventing corruption or loss of critical information.

**Compliance and Legal Risks:**

: Failure to comply with legal regulations and industry standards can lead to lawsuits and fines. Verify that the application adheres to all relevant compliance requirements.

**Documentation Gaps:**

: Insufficient test documentation can lead to confusion and misunderstandings. Maintain comprehensive test documentation to facilitate efficient testing and troubleshooting.

**User Adoption and Training:**

: Lack of user training and support can result in user dissatisfaction and decreased adoption. Plan for user training and provide adequate support resources.

**Change Management:**

: Poorly managed changes to the application can introduce unexpected issues. Implement a change management process to control and document changes effectively.

**Vendor or Technology Dependencies:**

: Reliance on specific vendors or technologies can pose risks if they become obsolete or unreliable. Assess these dependencies and have contingency plans in place.

**Geographical and Cultural Factors:**

: Different regions and cultures may have unique requirements or expectations. Consider these factors in your testing to ensure a global user base is satisfied.

**Backup and Recovery:**

: Data loss or system failures can occur. Establish backup and recovery procedures to minimize downtime and data loss.

**Budget and Cost Overruns:**

: Exceeding the testing budget can impact project timelines and resources. Monitor and manage the budget throughout the testing process.

**Human Error:**

: Tester errors, miscommunications, or oversights can lead to defects going unnoticed. Implement peer reviews and thorough testing processes to minimize human error.

**4)What is the difference in responsibilities between QA and QC?**

QA (Quality Assurance):

QA focuses on preventing defects and ensuring that processes are in place to deliver products or services that meet predefined quality standards.

It is a proactive approach aimed at improving processes and preventing issues from occurring in the first place.

QC (Quality Control):

QC focuses on detecting and correcting defects in the final product or service.

It is a reactive approach aimed at identifying and addressing issues after they have occurred but before the product or service is delivered to the customer.

**Responsibilities:**

QA (Quality Assurance):

Developing and implementing quality management systems and processes.

Defining quality standards and guidelines.

Conducting audits and assessments to ensure compliance with quality standards.

Identifying areas for process improvement.

Training and educating team members on quality standards and best practices.

QC (Quality Control):

Inspecting and testing the final product or service to identify defects or deviations from quality standards.

Taking corrective actions when defects are found, which may include rework or rejection of the product.

Conducting inspections at various stages of production or service delivery.

Maintaining records of inspection results and defect reports.

Ensuring that products or services meet customer requirements before delivery.

**Timing:**

QA (Quality Assurance):

QA activities are typically performed throughout the entire product or service development lifecycle, from planning and design to execution and delivery.

QA is a continuous and ongoing process.

QC (Quality Control):

QC activities are usually performed at specific checkpoints or stages of production or service delivery, such as before shipment or customer delivery.

QC is event-driven and occurs at specific points in the process.

**Focus:**

QA (Quality Assurance):

Focuses on process improvement, standardization, and prevention of defects.

Aims to reduce the likelihood of defects occurring in the first place.

QC (Quality Control):

Focuses on product or service inspection and defect detection.

Aims to identify and address defects before they reach the customer.

**5) Difference between Manual and Automation testing?**

* In manual testing, a human performs the tests step by step, without test scripts.
* In Automation testing, tests are executed automatically via test automation frameworks along with other software tools.