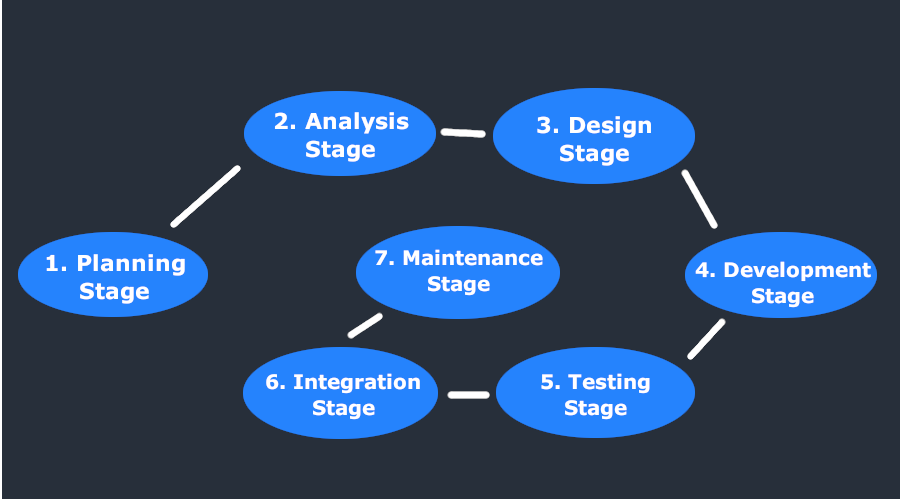
**STQA ASSIGNMENT**

1. **What are the phases of SDLC?**

The software development process is normally long and tedious. But project managers and system analysts can leverage software development life cycles to outline, design, develop, test, and eventually deploy information systems or software products with greater regularity, efficiency, and overall quality.

A system development life cycle of SDLC is essentially a project management model. It defines different stages that are necessary to bring a project from its initial idea or conception all the way to deployment and later maintenance.



There are seven primary stages of the modern system development life cycle. Here’s a brief breakdown:

1. Planning Stage
2. Feasibility or Requirements of Analysis Stage
3. Design and Prototyping Stage
4. Software Development Stage
5. Software Testing Stage
6. Implementation and Integration
7. Operations and Maintenance Stage

**Planning Stage**: The planning stage (also called the feasibility stage) is exactly what it sounds like the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems. By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development and help to secure the funding and resources they need to make their plan happen.‍

**Analysis Stage**: The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes. Developers may:

* Define any prototype system requirements
* Evaluate alternatives to existing prototypes
* Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document. This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

**‍Design Stage**: The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects, such as its:

* User interfaces
* System interfaces
* Network and network requirements
* Databases

They’ll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward.

**Development Stage**: The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications. This is where Static Application Security Testing or SAST tools come into play. In theory, all of the prior planning and outlined should make the actual development phase relatively straightforward. Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters.

**‍Testing Stage**: During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested. It’s important that the software overall ends up meeting the quality standards that were previously defined in the SRS document. Depending on the skill of the developers, the complexity of the software, and the requirements for the end-user, testing can either be an extremely short phase or take a very long time.

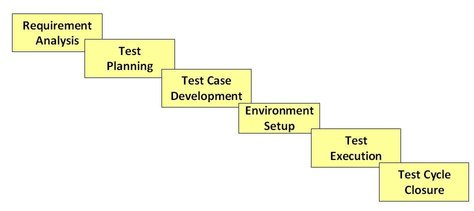
**‍Implementation and Integration Stage**: After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects. The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

**‍Maintenance Stage**: The SDLC doesn’t end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users. Furthermore, developers are responsible for implementing any changes that the software might need after deployment. This can include handling residual bugs that were not able to be patched before launch or resolving new issues that crop up due to user reports.

1. **What are the phases of STLC?**

Software Testing Life Cycle (STLC) is a testing strategy that can help you efficiently meet software quality standards. STLC enforces systematic testing, which is performed in phases.

Software Testing Life Cycle (STLC) is a process used to test software and ensure that quality standards are met. Tests are carried out systematically over several phases. During product development, phases of the STLC may be performed multiple times until a product is deemed suitable for release.



Software Testing Life Cycle consists of 6 phases, each with defined entry and exit criteria and associated activities and deliverables.

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Test Environment Setup
5. Test Execution
6. Test Cycle Closure

Entry and exit criteria define when a phase can start and when it can end. Activities and deliverables define what actions are performed and what the expected result is. Some of these phases can be performed simultaneously while others require previous phases to be completed first.

**Requirement Analysis**: During this phase, feature requirements collected in the SDLC process are evaluated to identify testable aspects. If necessary, testing teams may need to consult with stakeholders to clarify requirements. These requirements can either be functional or non-functional, defining what a feature can do or its characteristics respectively. The ability to automate testing is also evaluated during this phase.

* Entry Criteria: Documented requirements, acceptance criteria, and intended product architecture.
* Exit Criteria: Approved requirement traceability matrix (RTM) and automation feasibility report.

**Test Planning**: During this phase, the test strategy is outlined in a test plan document. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing.

* Entry Criteria: Requirement analysis, RTM, and automation feasibility report.
* Exit Criteria: Approved test plan including timelines and risk/cost analysis.

**Test Case Development**: During this phase, test cases are created. Each case defines test inputs, procedures, execution conditions, and anticipated results. Test cases should be transparent, efficient, and adaptable. Once all test cases are created, test coverage should be 100%. Any necessary automation scripts are also created during this phase.

* Entry Criteria: Approved test plan including timelines and risk/cost analysis.
* Exit Criteria: Approved test cases and automation scripts.

**Test Environment Setup**: During this phase, testing environments are configured and deployed. This phase may include a variety of testing tools, including TestComplete, Selenium, Appium, or Katalon Studio. Sometimes, this phase also includes setting up test servers. Once environments are deployed, smoke tests are performed to ensure that environments are working as expected with all intended functionality.

* Entry Criteria: System design and project architecture definitions.
* Exit Criteria: A fully functional test environment and approved test cases.

**Test Execution**: During this phase, features are tested in the deployed environment, using the established test cases. Expected test results are compared to actual and results are gathered to report back to development teams.

* Entry Criteria: All exit criteria from previous steps.
* Exit Criteria: All tests are performed and results are documented.

**Test Cycle Closure**: This is the last phase of the STLC, during which a test result report is prepared. This report should summarize the entire testing process and provide comparisons between expected results and actual. These comparisons include objectives met, time taken, total costs, test coverage, and any defects found.

* Entry Criteria: Test results and logging from all previous phases.
* Exit Criteria: Delivered and approved test closure report.

1. **What is the difference between Testing and Debugging?**

**Testing**: Testing is the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

**Debugging**: Debugging is the process of fixing a bug in the software. It can be defined as the identifying, analyzing and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software.

| **Testing** | **Debugging** |
| --- | --- |
| Testing is the process to find bugs and errors. | Debugging is the process to correct the bugs found during testing. |
| It is the process to identify the failure of implemented code. | It is the process to give the absolution to code failure. |
| Testing is the display of errors. | Debugging is a deductive process. |
| Testing is done by the tester. | Debugging is done by either programmer or developer. |
| There is no need for design knowledge in the testing process. | Debugging can’t be done without proper design knowledge. |
| Testing can be done by an insider as well as an outsider. | Debugging is done only by insiders. Outsiders can’t do debugging. |
| Testing can be manual or automated. | Debugging is always manual. Debugging can’t be automated. |
| It is based on different testing levels i.e. unit testing, integration testing, system testing etc. | Debugging is based on different types of bugs. |
| Testing is a stage of software development life cycle (SDLC). | Debugging is not an aspect of software development life cycle, it occurs as a consequence of testing. |
| Testing is composed of validation and verification of software. | While the debugging process seeks to match symptoms with cause, by that it leads to the error correction. |
| Testing is initiated after the code is written. | Debugging commences with the execution of a test case. |
| Testing process based on various levels of testing-system testing, integration testing, unit testing, etc. | Debugging process based on various types of bugs is present in a system. |