Q1 -> ***The most important models of Software Development Life Cycle (SDLC):***

1. **Waterfall model in SDLC**

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases of SDLC. In this SDLC model, the outcome of one phase acts as the input for the next phase. This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

1. **Incremental Model in SDLC**

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC life cycle process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

1. **V-Model in SDLC**

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases of SDLC on the side and the validation phase on the other side. V-Model joins by Coding phase.

1. **Agile Model in SDLC**

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

1. **Spiral Model**

The spiral model is a risk-driven process model. This SDLC testing model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

1. **Big bang model**

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come. This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

Q2 ->

***What is SDLC?***

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Life Cycle and is also referred to as the Application Development life-cycle.

***Why SDLC?***

Here, are prime reasons why SDLC is important for developing a software system.

It offers a basis for project planning, scheduling, and estimating

Provides a framework for a standard set of activities and deliverables

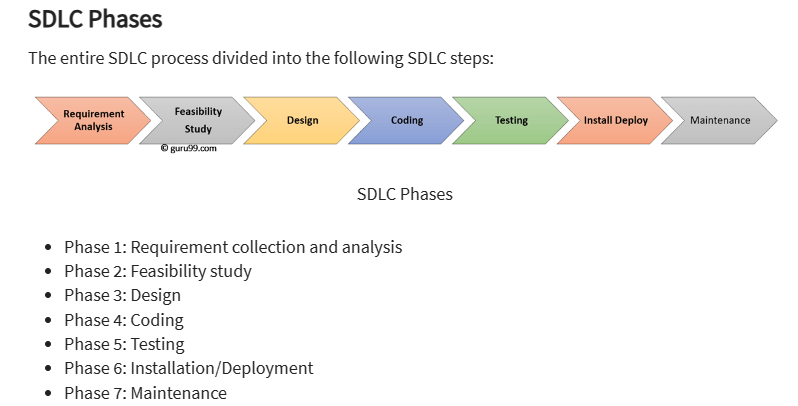
It is a mechanism for project tracking and control

Increases visibility of project planning to all involved stakeholders of the development process

Increased and enhance development speed

Improved client relations

Helps you to decrease project risk and project management plan overhead



***Phase 1: Requirement collection and analysis***

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and reorganization of the risks involved is also done at this stage. This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project. Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

***Phase 2: Feasibility study***

Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of ‘Software Requirement Specification’ document also known as ‘SRS’ document. It includes everything which should be designed and developed during the project life cycle.

There are mainly five types of feasibilities checks:

* Economic: Can we complete the project within the budget or not?
* Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
* Operation feasibility: Can we create operations which is expected by the client?
* Technical: Need to check whether the current computer system can support the software
* Schedule: Decide that the project can be completed within the given schedule or not.

***Phase 3: Design***

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture. This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

**High-Level Design (HLD)**

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

**Low-Level Design (LLD)**

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

***Phase 4: Coding***

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

***Phase 5: Testing***

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement. During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

***Phase 6: Installation/Deployment***

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

***Phase 7: Maintenance***

Once the system is deployed, and customers start using the developed system, following 3 activities occur

* Bug fixing – bugs are reported because of some scenarios which are not tested at all
* Upgrade – Upgrading the application to the newer versions of the Software
* Enhancement – Adding some new features into the existing software

***Q3 -> Risks related to Web based application testing to be included in test plan***

1. Injection – This attack involves the exploiter breaking out of a data context and switching into a code context by using special coding characters.
2. Cross-Site Scripting (XSS) – This attack is a form of injection, with the browser being used to bury the attack.
3. Broken Authentication and Session Management – This attack involves the exploiter stealing or assuming the identity of the unprotected authentication credentials of a user.
4. Insecure Direct Object References – A risk of being exposed can occur when there is a reference to an objects in a URL (an object such as a file, directory, database record or key) or form parameter because an exploiter could change these direct object references and attempt to access a different, unauthorized file, database record or key.
5. Cross-Site Request Forgery (CSRF) – is where the user’s browser is tricked into logging into a site with someone else’s credentials.
6. Security Misconfiguration – Misconfigurations of the application server as well as the database and database platform underlying the application can be exploited, especially when implemented with the known default settings. The exploits can be extremely varied due to the many configurations that could be misconfigured.
7. Insecure Cryptographic Storage – Exploiters may take or change data that is unprotected such credit card information, Social Security Numbers or authentication credentials if there is no strong encryption or hashing utilized.
8. Failure to Restrict URL Access – Applications that do not have access control checks each time a page is accessed may allow attackers to forge URLs to access pages that are thought to be hidden.
9. Insufficient Transport Layer Protection – When applications do not authenticate, encrypt or protect the confidentiality and integrity of sensitive network traffic by proper use of certificates the traffic may be intercepted by an untrusted party.
10. Invalidated Redirects and Forwards – This allows attackers to redirect the user to a site they did not intend to go to and perhaps ask for personal information by tricking the user into thinking they are at a valid, familiar, trusted site.

***Q4 -> Difference between Quality Control and Quality Assurance***

**What is Quality Assurance?**

Quality Assurance is known as QA and focuses on preventing defect. Quality Assurance ensures that the approaches, techniques, methods and processes are designed for the projects are implemented correctly. Quality assurance activities monitor and verify that the processes used to manage and create the deliverables have been followed and are operative. Quality Assurance is a proactive process and is Prevention in nature. It recognizes flaws in the process. Quality Assurance has to complete before Quality Control.

**What is Quality Control?**

Quality Control is known as QC and focuses on identifying a defect. QC ensures that the approaches, techniques, methods and processes are designed in the project are following correctly. QC activities monitor and verify that the project deliverables meet the defined quality standards. Quality Control is a reactive process and is detection in nature. It recognizes the defects. Quality Control has to complete after Quality Assurance.

|  |  |
| --- | --- |
| **Quality Assurance (QA)** | **Quality Control (QC)** |
| It is a procedure that focuses on providing assurance that quality requested will be achieved | It is a procedure that focuses on fulfilling the quality requested. |
| QA aims to prevent the defect | QC aims to identify and fix defects |
| It is a method to manage the quality- Verification | It is a method to verify the quality-Validation |
| It does not involve executing the program | It always involves executing a program |
| It’s a Preventive technique | It’s a Corrective technique |
| It’s a Proactive measure | It’s a Reactive measure |
| It is the procedure to create the deliverables | It is the procedure to verify that deliverables |
| QA involves in full software development life cycle | QC involves in full software testing life cycle |
| In order to meet the customer requirements, QA defines standards and methodologies | QC confirms that the standards are followed while working on the product |
| It is performed before Quality Control | It is performed only after QA activity is done |
| It is a Low-Level Activity, it can identify an error and mistakes which QC cannot | It is a High-Level Activity, it can identify an error that QA cannot |
| Its main motive is to prevent defects in the system. It is a less time-consuming activity | Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity |
| QA ensures that everything is executed in the right way, and that is why it falls under verification activity | QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity |
| It requires the involvement of the whole team | It requires the involvement of the Testing team |
| The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC) | The statistical technique applied to QC is known as SQC or Statistical Quality Control |

***Q5 -> Difference between Manual and Automation Testing***

**Manual Testing:**

Manual testing is a type of testing in which we do not take the help of any tools (automation) to perform the testing. In this testing, testers make test cases for the codes and test the software and give the final report about that software. Manual testing is a time consuming testing because it is done by humans and there is a chance of human errors.

**Automation Testing:**

Automation testing is a type of testing in which we take the help of tools (automation) to perform the testing. It is faster than manual testing because it is done by some automation tools. There is no chance of any human errors.

***Manual Testing Vs. Automation Testing***

| **Parameter** | **Automation Testing** | **Manual Testing** |
| --- | --- | --- |
| Definition | Automation Testing uses automation tools to execute test cases. | In manual testing, test cases are executed by a human tester and software. |
| Processing time | Automated testing is significantly faster than a manual approach. | Manual testing is time-consuming and takes up human resources. |
| Exploratory Testing | Automation does not allow random testing | Exploratory testing is possible in Manual Testing |
| Initial investment | The initial investment in the automated testing is higher. Though the ROI is better in the long run. | The initial investment in the Manual testing is comparatively lower. ROI is lower compared to Automation testing in the long run. |
| Reliability | Automated testing is a reliable method, as it is performed by tools and scripts. There is no testing Fatigue. | Manual testing is not as accurate because of the possibility of the human errors. |
| UI Change | For even a trivial change in the UI of the AUT, Automated Test Scripts need to be modified to work as expected | Small changes like change in id, class, etc. of a button wouldn’t thwart execution of a manual tester. |
| Investment | Investment is required for testing tools as well as automation engineers | Investment is needed for human resources. |
| Cost-effective | Not cost effective for low volume regression | Not cost effective for high volume regression. |
| Test Report Visibility | With automation testing, all stakeholders can login into the automation system and check test execution results | Manual Tests are usually recorded in an Excel or Word, and test results are not readily/ readily available. |
| Human observation | Automated testing does not involve human consideration. So it can never give assurance of user-friendliness and positive customer experience. | The manual testing method allows human observation, which may be useful to offer user-friendly system. |
| Performance Testing | Performance Tests like Load Testing, Stress Testing, Spike Testing, etc. have to be tested by an automation tool compulsorily. | Performance Testing is not feasible manually |
| Parallel Execution | This testing can be executed on different operating platforms in parallel and reduce test execution time. | Manual tests can be executed in parallel but would need to increase your human resource which is expensive |
| Batch testing | You can Batch multiple Test Scripts for nightly execution. | Manual tests cannot be batched. |
| Programming knowledge | Programming knowledge is a must in automation testing. | No need for programming in Manual Testing. |
| Set up | Automation test requires less complex test execution set up. | Manual testing needs have a more straightforward test execution setup |
| Engagement | Done by tools. Its accurate and never gets bored! | Repetitive Manual Test Execution can get boring and error-prone. |
| Ideal approach | Automation testing is useful when frequently executing the same set of test cases | Manual testing proves useful when the test case only needs to run once or twice. |
| Build Verification Testing | Automation testing is useful for Build Verification Testing (BVT). | Executing the Build Verification Testing (BVT) is very difficult and time-consuming in manual testing. |
| Deadlines | Automated Tests have zero risks of missing out a pre-decided test. | Manual Testing has a higher risk of missing out the pre-decided test deadline. |
| Framework | Automation testing uses frameworks like Data Drive, Keyword, Hybrid to accelerate the automation process. | Manual Testing does not use frameworks but may use guidelines, checklists, stringent processes to draft certain test cases. |
| Documentation | Automated Tests acts as a document provides training value especially for automated unit test cases. A new developer can look into a unit test cases and understand the code base quickly. | Manual Test cases provide no training value |
| Test Design | Automated Unit Tests enforce/drive Test Driven Development Design. | Manual Unit Tests do not drive design into the coding process |
| Devops | Automated Tests help in Build Verification Testing and are an integral part of DevOps Cycle | Manual Testing defeats the automated build principle of DevOps |
| When to Use? | Automated Testing is suited for Regression Testing, Performance Testing, Load Testing or highly repeatable functional test cases. | Manual Testing is suitable for Exploratory, Usability and Adhoc Testing. It should also be used where the AUT changes frequently. |