SE

1. **Functional and non functional requirements**

   A functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation.

   A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system.

2. **What is generic process model**

   There are five generic process framework activities:
   1. Communication:
   The software development starts with the communication between customer and developer.
   2. Planning:
   It consists of complete estimation, scheduling for project development and tracking.
   3. Modeling:
   Modeling consists of complete requirement analysis and the design of the project like
   algorithm, flowchart etc.
   The algorithm is the step-by-step solution of the problem and the flow chart shows a
   complete flow diagram of a program.
   4. Construction:
   Construction consists of code generation and the testing part.
   Coding part implements the design details using an appropriate programming language.
   Testing is to check whether the flow of coding is correct or not.

Testing also check that the program provides desired output.
5. Deployment:
Deployment step consists of delivering the product to the customer and take feedback from
them.
If the customer wants some corrections or demands for the additional capabilities, then the
change is required for improvement in the quality of the software.

3. **What is CMM model and define it's steps**
   The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process. The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes.
   CMM is not a software process model. It is used as a benchmark to measure the maturity of an organisation's software process

CMM's Five Maturity Levels of Software Processes

- At the *initial* level, processes are disorganized, even chaotic. Success is likely to depend on individual efforts, and is not considered to be repeatable, because processes would not be sufficiently defined and documented to allow them to be replicated.
- At the *repeatable* level, basic project management techniques are established, and successes could be repeated, because the requisite processes would have been made established, defined, and documented.
- At the *defined* level, an organization has developed its own standard software process through greater attention to documentation, standardization, and integration.

- At the *managed* level, an organization monitors and controls its own processes through data collection and analysis.
- At the *optimizing* level, processes are constantly being improved through monitoring feedback from current processes and introducing innovative processes to better serve the organization's particular needs.

4. **What is SRS**
   A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

   ## Qualities of SRS:

   - Correct
   - Unambiguous
   - Complete
   - Consistent
   - Ranked for importance and/or stability
   - Verifiable
   - Modifiable
   - Traceable

5. **White box vs black box testing? Which is the best**

   # What is Black Box testing?

   In Black-box testing, a tester doesn't have any information about the internal working of the software system. Black box testing is a high level of testing that focuses on the behavior of the software. It involves testing from an external or end-user perspective. Black box testing can be

applied to virtually every level of software testing: unit, integration, system, and acceptance.

Testers perform testing only on the functional part of an application to make sure the behavior of the software is as expected. It is also known as input-output testing.
Techniques for : Equivalence partitioning,Boundary Value Analysis, Decision table,State transition

What is White Box testing?

White-box testing is a testing technique which checks the internal functioning of the system.In this method, testing is based on coverage of code statements, branches,paths or conditions.White-Box testing is considered as low-level testing.It is also called glass box, transparent box, clear box or code base testing.The white-box Testing method assumes that the path of the logic in a unit or program is known.

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.

It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred as outer or external software testing.	It is the inner or the internal software testing.
It is functional test of the software.	It is structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.

It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.

6. **What is RMMM plan**

   A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

Risk Mitigation :

It is an activity used to avoid problems (Risk Avoidance).

Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

Risk Monitoring :

It is an activity used for project tracking.

It has the following primary objectives as follows.

1. To check if predicted risks occur or not.
2. To ensure proper application of risk aversion steps defined for risk.
3. To collect data for future risk analysis.
4. To allocate what problems are caused by which risks throughout the project.

Risk Management and planning :

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

7. What is COCOMO MODEL

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).

Determine a set of 15 multiplying factors from various attributes of the project.

Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

8. Agile method

AGILE is a methodology that promotes continuous iteration of development and testing throughout the software development life cycle of the project. Both development and testing activities are concurrent unlike the Waterfall model.
The agile software development emphasizes on four core values.
Individual and team interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

9. Increment model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.
various phases of incremental model:
1. Requirement analysis:
2. Design & Development:
3. Testing:
4. Implementation:

Uses:
When the requirements are superior.
A project has a lengthy development schedule.
When Software team are not very well skilled or trained.
When the customer demands a quick release of the product.

10. Alpha and beta testing

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testing.

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the type of User Acceptance Testing.

Alpha Testing	Beta Testing
Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Alpha testing is performed at developer's site.	Beta testing is performed at end-user of the product.
Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Alpha testing may require long execution cycle.	Beta testing requires only a few weeks of execution.

Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from beta testing will be implemented in future versions of the product.

11. What is architecture design

IEEE defines architectural design as 'the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.'
This framework is established by examining the software requirements document and designing a model for providing implementation details. These details are used to specify the components of the system along with their inputs, outputs, functions, and the interaction between them.

12. What are different estimation techniques

There are many different types of project estimation techniques used in Project Management with various streams like Engineering, IT, Construction, Agriculture, Accounting, etc. A Project manager is often challenged to align mainly six project constraints - Scope, Time, Cost, Quality, Resources, and Risk in order to accurately estimate the project. The common questions that come into the mind of a project manager at the start of the project are–

- How much work is to be estimated (scope).
- How to estimate the project (techniques).
- How much time it will require to complete the project (Schedule).
- Who will be doing the project (resources)?
- What is the budget required to deliver the project (cost)?
- Any intermediary dependencies that may delay or impact the project (Risks).

13. Phases of software development

Known as the '**software development** life cycle,' these six **steps** include planning, analysis, design, **development** & implementation, testing & deployment and maintenance.

The phases of SDLC are as follows

Requirement gathering and analysis.

Design.

Coding and Implementation.

Testing.

Deployment.

Maintenance.

14. Spiral model, RAD model etc and their advantages and disadvantages

RAD is a Rapid Application Development model.
Using the RAD model, software product is developed in a short period of time.
The initial activity starts with the communication between customer and developer.
Planning depends upon the initial requirements and then the requirements are divided into
groups.
Planning is more important to work together on different modules.
The RAD model consist of following phases:
1. Business Modeling
Business modeling consist of the flow of information between various functions in the
project.
For example what type of information is produced by every function and which are the
functions to handle that information.
A complete business analysis should be performed to get the essential business information.
2. Data modeling
The information in the business modeling phase is refined into the set of objects and it is
essential for the business.
The attributes of each object are identified and define the relationship between objects.
3. Process modeling

The data objects defined in the data modeling phase are changed to fulfil the information
flow to implement the business model.
The process description is created for adding, modifying, deleting or retrieving a data object.
4. Application generation
In the application generation phase, the actual system is built.
To construct the software the automated tools are used.
5. Testing and turnover
The prototypes are independently tested after each iteration so that the overall testing time is
reduced.
The data flow and the interfaces between all the components are are fully tested. Hence,
most of the programming components are already tested.

Evolutionary Process Models in Software Engineering
Evolutionary Process Models
Evolutionary models are iterative type models.
They allow to develop more complete versions of the software.

The Spiral model

Spiral model is a risk driven process model.
It is used for generating the software projects.
In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then
alternate solutions are suggested and implemented.
It is a combination of prototype and sequential model or waterfall model.
In one iteration all activities are done, for large projects; the output is small.
Advantages of Spiral Model
It reduces high amount of risk.

It is good for large and critical projects.
It gives strong approval and documentation control.
In spiral model, the software is produced early in the life cycle process.
Disadvantages of Spiral Model
It can be costly to develop a software model.
It is not used for small projects.

Software testing

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

15. Verification and Validation

Software Validation

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes on user requirements.

Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications ?"
- Verifications concentrates on the design and system specifications.

Target of the test are -

Errors - These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, is considered as an error.

Fault - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.

Failure - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

16. Which Model Is Best?

17. Explian SDLC

SDLC is the acronym of Software Development Life Cycle.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

18. What Is Software Engineering

Software engineering is defined as a process
of analyzing user requirements and then designing, building,
and testing software application which will satisfy those requirements.tt

19. Why Waterfall Model Is Called Waterfall Model?

The developer must complete every phase before the next phase begins. This **model is named "Waterfall Model"**, because its diagrammatic representation resembles a cascade of **waterfalls**.

20. What Is FTR

Formal Technical Review (FTR) is a software quality control activity performed by software engineers. Objectives of formal technical review (FTR): ... To ensure that software is represented according to predefined

standards. It helps to review the uniformity in software that is development in a uniform manner.

21. **CMM Principles And Aur Kuch Pucha**
22. **Requirement models and SRS and what is problem statement**

Requirements modeling comprises several stages, or 'patterns': scenario-based modeling, data modeling, flow-oriented modeling, class-based modeling and behavioral modeling. Each of these stages/patterns examines the same problem from a different perspective.

Software Requirement Specification (SRS)

A software requirements specification (SRS) is a detailed description of a software system to
be developed with its functional and non-functional requirements. The SRS is developed
based the agreement between customer and contractors. It may include the use cases of how
user is going to interact with software system. The software requirement specification
document consistent of all necessary requirements required for project development. To
develop the software system we should have clear understanding of Software system. To
achieve this we need to continuous communication with customers to gather all requirements.

23. **Explain XP (Agile Ke Ques Ke Baad Hee Pucha)**

Extreme programming (XP) is one of the most important software development framework of Agile models. It is used to improve software quality and responsive to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

The XP process comprises four framework activities:

1. Planning

Planning starts with the requirements gathering which enables XP team to understand the

rules for the software.
The customer and developer work together for the final requirements.
2. Design
The XP design follows the 'keep it simple' principle.
A simple design always prefers the more difficult representation.
3. Coding
The coding is started after the initial design work is over.
After the initial design work is done, the team creates a set of unit tests which can test each
situation that should be a part of the release.
The developer is focused on what must be implemented to pass the test.
Two people are assigned to create the code. It is an important concept in coding activity.
4. Testing
Validation testing of the system occurs on a daily basis. It gives the XP team a regular
indication of the progress.
'XP acceptance tests' are known as the customer test.
Applications of Extreme Programming (XP): Some of the projects that are suitable to
develop using XP model are given below:
Small projects: XP model is very useful in small projects consisting of small teams
as face to face meeting is easier to achieve.
Projects involving new technology or Research projects: This type of projects face
changing of requirements rapidly and technical problems. So XP model is used to
complete this type of projects.

24. Last Scrum or Scrum

Scrum is the type of Agile framework. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses Iterative process.

Silent features of Scrum are:

25. Scrum is light-weighted framework
26. Scrum emphasizes self-organization
27. Scrum is simple to understand

28. Water fall model ? Design, difference,Where it is used ? Disadvantage

Defination:

The waterfall model is a sequential model in which the next phase starts only after the first phase is completed. For example, the testing phase will start only after the development phase is complete, the maintenance phase will start only after the testing phase is complete.

Below are the various phases involved in the waterfall model. Please note that the number of phases and sequences of phases may vary from one project to another.

- Requirements
- Design
- Coding
- Testing
- Maintenance

a) Requirements: This is the phase where the system to be developed is documented in the form of Software Requirement Specification (SRS) document. This is the most important phase of SDLC as a clear understanding of requirements from the client will reduce the rework in the following phases.

b) Design: This is the phase where the architecture of the system to be developed is finalized. Architecture can be in the form of a high-level design or a low-level design.

Architecture must also include the hardware and software specifications of the system to be developed.

c) Coding: This is the phase where the code for the system to be developed is written. Unit Testing and Integration Testing must be performed by the developers at this stage before deploying the code for testing.

d) Testing: This is the phase where the product developed is tested by an independent testing team to validate if it meets the requirements in the Software Requirement Specification (SRS). Defects raised at this phase need to be fixed before providing sign off on the product.

e) Maintenance: This phase comes once the testing phase is complete. It takes care of any production issues that may arise after the product is delivered to the customer. The duration of the maintenance phase differs from project to project and one organization to another.

Uses:

Where the tools and technology used is consistent and is not changing

When resources are well prepared and are available to use.

When the requirements are constant and not changed regularly.

A project is short

Advantages:

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.

Disadvantage:

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.

29. Definition of software engineering

Software engineering is defined as a process
of analyzing user requirements and then designing, building,
and testing software application which will satisfy those requirements.

a branch of computer science that deals with
the design, implementation, and maintenance of
complex computer programs

30. Risk

31. Reverse engineering

Reverse engineering, in computer programming, is a technique used to analyze software in order to identify and understand the parts it is composed of. The usual reasons for reverse engineering a piece of software are to recreate the program, to build something similar to it, to exploit its weaknesses or strengthen its defenses.

32. Re enginneeing

The application of technology and management science to the modification of existing systems, organizations, processes and products in order to make them more effective, efficient and responsive.
the process of changing and improving the way a company works, the way a job is done, etc.:

33. v model

V-Model stands for the verification and validation model. V-model is an addition to the waterfall model, in the sense that V-model is also a sequential model. In V-model, each phase of development is associated with a corresponding testing phase.

Explanation:

Within the V-Model, SDLC is to be interpreted from top to bottom, while STLC is to be interpreted from the bottom to the top. Initially, requirements are gathered to document the system to be developed as per the client requirements. The testing team develops the system test plan based on the requirements.

Then comes the high-level design and the detailed level design phases where the architecture of the system is prepared. The testing team prepares the Integration Test plan in these phases. Once the coding is complete on SDLC, STLC will start from unit testing, followed by integration testing and System testing.

34. What is software engineer

A practitioners of software engineering are called Software Engineers. *A software engineer applies the principles of software engineering in designing, development, maintenance and testing of software.*

Software engineers are usually specialized in computer science engineering and information technology. Good knowledge of programming languages is the key to success in this field.

35. kanban model

Kanban is a popular framework used to implement agile software development. It requires
real-time communication of capacity and full transparency of work. Work items are
represented visually on a kanban board, allowing team members to see the state of every
piece of work at any time.
When Applicable
Kanban can be used in any knowledge work setting, and is particularly applicable in situations
where work arrives in an unpredictable fashion and/or when you want to deploy work as soon as

it is ready, rather than waiting for other work items.

36. which is best test

Black box testing requires no knowledge of internal paths, structures, or implementation of the software being tested. White box testing is a testing strategy based on internal paths, code structures, and implementation of the software being tested.
Therefore white box testing is better, as the tester has knowledge of the software.

37. Agile process

The meaning of Agile is swift or versatile."Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

The 12 Agile Principles: What Are They and Do They Still Matter?

Early and Continuous Delivery of Valuable Software. ...

Embrace Change. ...

Frequent Delivery. ...

Business and Developers Together. ...

Motivated Individuals. ...

Face-to-Face Conversation. ...

Working Software. ...

Sustainable Development.

38. Cmm coupling ,cohesion

Coupling:

Coupling is the measure of the degree of interdependence between the modules. A good software will have low coupling.

Types of Coupling:

Data Coupling: If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent to each other and communicating through data. Module communications don't contain tramp data. Example-customer billing system.

Stamp Coupling In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors- this choice made by the insightful designer, not a lazy programmer.

Control Coupling: If the modules communicate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument.

External Coupling: In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.

Common Coupling: The modules have shared data such as global data structures.The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses and reduced maintainability.

Content Coupling: In a content coupling, one module can modify the data of another module or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

Cohesion: Cohesion is a measure of the degree to which the elements of the module are functionally related. It is the degree to which all elements directed towards performing a single task are contained in the component. Basically, cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.

Types of Cohesion:

- **Functional Cohesion:** Every essential element for a single computation is contained in the component. A functional cohesion performs the task and functions. It is an ideal situation.
- **Sequential Cohesion:** An element outputs some data that becomes the input for other element, i.e., data flow between the parts. It occurs naturally in functional programming languages.
- **Communicational Cohesion:** Two elements operate on the same input data or contribute towards the same output data. Example- update record int the database and send it to the printer.
- **Procedural Cohesion:** Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable. Ex- calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA.
- **Temporal Cohesion:** The elements are related by their timing involved. A module connected with temporal cohesion all the tasks must be executed in the same time-span. This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at init time.
- **Logical Cohesion:** The elements are logically related and not functionally. Ex- A component reads inputs from tape, disk, and network. All the code for these functions is in the same component. Operations are related, but the functions are significantly different.
- **Coincidental Cohesion:** The elements are not related(unrelated). The elements have no conceptual relationship other than location in source code. It is accidental and the worst form of cohesion. Ex- print next line and reverse the characters of a string in a single component.

39. **architecture**

- The complete structure of the software is known as software architecture.
- Structure provides conceptual integrity for a system in a number of ways.
- The architecture is the structure of program modules where they interact with each other in a specialized way.
- The components use the structure of data.
- The aim of the software design is to obtain an architectural framework of a system.
- The more detailed design activities are conducted from the framework.

Q #12) What is Black box testing?

Answer: Black box testing involves testing the application without the knowledge of the internal structure or code implementation. Testers would only bother about the functionality of the software in black box testing rather than data flow and code execution in the back end.

Q #13) What is White box testing?

Answer: White box testing is testing the application with the knowledge of the internal structure and code implementation. This testing is generally performed by the developer who has written the code in the form of unit tests.

Q 14)Waterfall model:

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.**
- **System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.**
- **Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.**
- **Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.**
- **Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.**
- **Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.**

Q15) UML:

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

SDLC:
Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Q16)In Software Engineering, Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes

Earned Value Analysis (EVA) is an industry standard method of measuring a project's progress at any given point in time, forecasting its completion date and final cost, and analyzing variances in the schedule and budget as the project proceeds.