1) Explain the SDLC with its phases and diagram

Software Development Life Cycle (SDLC)

- Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the
- development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.
- SDLC defines the complete cycle of development i.e., all the tasks involved in planning,
- creating, testing, and deploying a Software Product. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.

SDLC Phases

- Phase 1: Requirement collection and analysis
- Phase 2: Feasibility study
- Phase 3: Design
- Phase 4: Coding

- Phase 5: Testing
- Phase 6: Installation/Deployment
- Phase 7: Maintenance

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 recognition of the risks involved is also done at this stage.

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.

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.

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.

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
- The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

2) What is the classification of software requirements? Explain with example

Classification of software requirements

- Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. Software system requirements are often classified as functional requirements, non –functional requirements or domain requirements.
- 1) Functional requirements: In software engineering, 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, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. In some cases, the functional requirements may also explicitly state what the system should not do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organization when writing requirements.
- Functional requirements in software engineering help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is

required to perform. Functional requirements indicate what a system must do.

- 2) Non –functional requirements - 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. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. On-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000.
- Description of non-functional requirements is just as critical as a functional requirement.
- Non-functional requirements support them and determine how the system must perform.

3) Explain the term software project with example

Software Project

- A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

Need of software project management

- Software is said to be an intangible product. Software development is a kind of all new stream in world business and there ş very little experience in building software products. Most software products are tailor made to fit client ş requirements. The most important is that the underlying technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. All such business and environmental constraints bring risk in software development hence it is essential to manage software projects efficiently.

Software Project Manager

- A software project manager is a person who undertakes the responsibility of executing the software project. Software project manager is thoroughly aware of all the phases of SDLC that the software would go through. Project manager may never directly involve in producing the end product but he controls and manages the activities involved in production.
- A project manager closely monitors the development process, prepares and executes various plans, arranges necessary and adequate resources, maintains communication among all team members in order to address issues of cost, budget, resources, time, quality and customer satisfaction.

Managing People

- Act as project leader
- Liaison with stakeholders
- Managing human resources
- Setting up reporting hierarchy etc.

Managing Project

- Defining and setting up project scope
- Managing project management activities
- Monitoring progress and performance
- Risk analysis at every phase
- Take necessary step to avoid or come out of problems
- Act as project spokesperson

Software Management Activities

- Software project management comprises of a number of activities, which contains planning of project, deciding scope of software product, estimation of cost in various terms, scheduling of tasks and events, and resource management. Project management activities may include:

- Project Planning
- Scope Management
- Project Estimation

Project Planning

Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production.

Scope Management

It defines the scope of project; this includes all the activities, process need to be done in order to make a deliverable software product. Scope management is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done.

Project Estimation

For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.

4) List and explain various attributes of software

- Software Quality Attributes are features that facilitate the measurement of performance of a software product by Software Testing professionals, and include attributes such as availability, interoperability, correctness, reliability, learnability, robustness, maintainability, readability, extensibility, testability, efficiency, and portability. High scores in Software Quality Attributes enable software architects to guarantee that a software application will perform as the specifications provided by the client.

Availability

- This attribute is indicative as to whether an application will execute the tasks it is assigned to perform. Availability also includes certain concepts that relate to software security, performance, integrity, reliability, dependability, and confidentiality. In addition, top-notch availability indicates that a software-driven system will repair any operating faults so that service outage periods would not exceed a specific time value.

Interoperability

- Software-driven systems could be required to communicate and act in tandem to solve certain tasks. Interoperability describes the ability of two systems to engage in the exchange of information via certain interfaces. Therefore, Software Quality Assurance engineers must examine the interoperability attribute in terms of both syntactic and semantic interoperability.

Performance

- This attribute pertains to the ability of a software-driven system to conform to timing requirements. From a testing point of view, it implies that Software Testing engineers must check whether the system responds to various events within defined time limits. These events may occur in the form of clock events, process interruptions, messages, and requests from different users, and others.

Testability

- Software testability indicates how well a software-driven system allows Software Testing professionals to conduct tests in line with predefined criteria. This attribute also assesses the ease with which Software Quality Assurance engineers can develop test criteria for

a said system and its various components. Engineers can assess the testability of a system by using various techniques such as encapsulation, interfaces, patterns, low coupling, and more.

Security

- This attribute measures the ability of a system to arrest and block malicious or unauthorized actions that could potentially destroy the system. The attribute assumes importance because security denotes the ability of the system to protect data and defend information from unauthorized access. Security also includes authorization and authentication techniques, protection against network attacks, data encryption, and such other risks. It is imperative for Software Testing Companies and professionals to regularly conduct updated security checks on systems.

Usability

- Every software-driven system is designed for ease of use to accomplish certain tasks. The attribute of usability denotes the ease with which users are able to execute tasks on the system; it also indicates the kind of user support provided by the system. The most well-known principle for this property is KISS (Keep It

Simple Stupid). In addition, Software Quality Assurance engineers must test software to check whether it supports different accessibility types of control for people with disabilities. Usability has a critical and long-standing bearing on the commercial fortunes of a software application or package.

Functionality

- This attribute determines the conformity of a software-driven system with actual requirements and specifications. Most Software Testing professionals view this attribute as crucial and a foremost requirement of a modern application, and would therefore advocate the performance of tests that assess the desired functionality of a system in the initial stages of Software Testing initiatives.

5) Differentiate between RAD model and Spiral model

1.RAD model is a software development model where by the components or functions are developed in parallel as if they were mini projects.

1.Spiral model is a software development model and is made with features of incremental, waterfall or evolutionary prototyping models.

2.Rapid development is its main objective.

2.High assurance is its main objective.

3.RAD model requirements and early stage planning is not necessary.

3.Spiral model requirements and early stage planning is required.

4.It is necessary to have detailed documentation but in a limited manner.

4.Detailed documentation is required.

5.Requirements are specified as time boxed release
5.manner.Requirements are specified in the beginning.

6.RAD model is used between large and small project.

6.Spiral model is used for large project.

7.There is low amount risk in RAD model.

7.There is medium to high amount risk in spiral model.

8.In RAD model small team size is required.

8.In spiral model large team is required.

9.Flexibility to change in RAD model is Easy.

9.Flexibility to change in spiral model is not that difficult.

10.In RAD model overlapping of phases is possible.

10.In spiral model overlapping of phases is not possible.

11.Testing is done in RAD model after completion of coding.

11.Testing is done in spiral model at the end of the engineering phase.

12.Cost of RAD model is Low.

12.Cost of spiral model is very expensive.

6) What is requirement? What are the types of requirements explain with example

Requirements

Requirement is a condition or capability possessed by the software or system component in order

to solve a real-world problem. The problems can be to automate a part of a system, to correct

shortcomings of an existing system, to control a device.

Requirement Engineering is the process of defining, documenting and maintaining the

requirements. It is a process of gathering and defining service provided by the system users. The

requirements themselves are the descriptions of the system services and constraints that are

generated during the requirement engineering process.

User requirements –

It means the high-level abstract requirements. User requirements are

statements in natural language plus diagrams of what services the system is expected to provide

and the constraint under which it much operates.

System requirements –It sets out the system functions, services, operational constraints in

detail. System requirement means the detail description of what the system should do. The

system requirement document should be precise. It should define exactly of is to be

implemented. It may be a part of contract between system buyer and software developer.

Example: Library System

User Requirement - Library system shall keep track of all data required by copyright licensing

agencies.

System Requirement –

🕐 On making a request for a document from the library system the requester shall be

presented with a form that records details of user and system mode.

🕐 Library system request forms shall be stored in the system for 5 years from the date of

request.

🕐 All library system request forms must be indexed by user, by the name of the material

requested and by the supplier of the request.

🕐 Library system shall maintain a log of all requests that have been made to the system

7) Write the structure of SRS specified by IEEE?

- IEEE defines software requirements specification as, a`document that clearly and precisely describes each of the essential

requirements (functions, performance, design constraints and quality attributes) of the software and the external interfaces. Each requirement is defined in such a way that its achievement can be objectively verified by a prescribed method, for example, inspection, demonstration, analysis or test. ¸Note that requirements specification can be in the form of a written document, a mathematical model, a collection of graphical models, a prototype, and so on.

- Essentially, what passes from requirements analysis activity to the specification activity is the knowledge acquired about the system. The need for maintaining a requirements document is that the modeling activity essentially focuses on the problem structure and not its structural behavior. While in SRS, performance constraints, design constraints, and standard compliance recovery are clearly specified. This information helps in developing a proper design of the system.
- **Various other purposes served by SRS are listed below**.
- **Feedback**: Provides a feedback, which ensures to the user that the organization (which develops the software) understands the issues or problems to be solved and the software behavior necessary to

address those problems.

- **Decompose problem into components**: Organizes the information and divides the problem into its component parts in an orderly manner.
- **Validation**: Uses validation strategies applied to the requirements to acknowledge that requirements are stated properly.
- **Input to design**: Contains sufficient detail in the functional system requirements to devise a design solution.
- **Basis for agreement between the user and the organization**: Provides a complete description of the functions to be performed by the system. In addition, it helps the users to determine whether the specified requirements are accomplished.
- **Reduce the development effort**: Enables developers to consider user requirements before the designing of the system commences. As a result, rẽwork ˛and inconsistencies in the later stages can be reduced.
- **Estimating costs and schedules:** Determines the requirements of the system and thus enables the developer to have a rõugh ˛ estimate of the total cost and schedule of the project.

8) Describe the SCRUM process

- Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes. Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project.
- The primary objective of Scrum is to satisfy the customer s need through an environment of transparency in communication, collective responsibility and continuous progress. The development starts from a general idea of what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain.
- The history of Scrum can be traced back to 1986 in the Harvard Business Review (HBR) article titled, The New Product Development Game by Hirotaka Takeuchi & Ikujiro Nonaka. This article describes how companies such as Honda, Canon, and Fuji-Xerox produce new products worldwide using a scalable and team-based approach to product development. This approach emphasizes the importance of empowering self-organized teams.
- Scrum is precisely an evolution of Agile Management. Scrum

methodology is based on a set of very defined practices and roles that must be involved during the software development process. It is a flexible methodology that rewards the application of the 12 agile principles in a context agreed by all the team members of the product.

- Scrum is executed in temporary blocks that are short and periodic, called Sprints, which usually range from 2 to 4 weeks, which is the term for feedback and reflection. Each Sprint is an entity in itself, that is, it provides a complete result, a variation of the final product that must be able to be delivered to the client with the least possible effort when requested.
- The process has as a starting point, a list of objectives/ requirements that make up the project plan. It is the client of the project that prioritizes these objectives considering a balance of the value and the cost thereof, that is how the iterations and consequent deliveries are determined.
- On the one hand the market demands quality, fast delivery at lower costs, for which a company must be very agile and flexible in the development of products, to achieve short development cycles that can meet the demand of customers without undermining the quality of the result. It is a very easy methodology to implement and very popular for the quick results it gets.

- In Scrum, the team focuses on building quality software. The owner of a Scrum project focuses on defining what are the characteristics that the product must have to build (what to build, what not and in what order) and to overcome any obstacle that could hinder the task of the development team.

9) Explain the term software project with example

- The term software specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.
- A software process is the set of activities and associated outcome that produce a software product. Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. These activities are:
- **Software specifications:** The functionality of the software and constraints on its operation must be defined.
- **Software development:** The software to meet the requirement must be produced.
- **Software validation**: The software must be validated to ensure

that it does what the customer wants.

- **Software evolution:** The software must evolve to meet changing client needs.

The Software Process Model

- A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering. Some examples of the types of software process models that may be produced are:

- **1.A workflow model**: This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
- **2. A dataflow or activity model**: This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow

model. They may perform transformations carried out by people or by computers.

- **3. A role/action model**: This means the roles of the people involved in the software process and the activities for which they are responsible.
- There are several various general models or paradigms of software development:
- **The waterfall approach**: This takes the above activities and produces them as separate process phases such as requirements specification, software design, implementation, testing, and so on. After each stage is defined, it is "signed off" and development goes onto the following stage.
- **Evolutionary development**: This method interleaves the activities of specification, development, and validation. An initial system is rapidly developed from a very abstract specification.
- **Formal transformation**: This method is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program. These transformations are 'correctness preserving.' This means that you can be sure that the developed programs meet its specification.
- **System assembly from reusable components**: This method

assumes the parts of the system already exist. The system development process target on integrating these parts rather than developing them from scratch.

10) Write a short note on Extreme programming

- Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness 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. Good practices need to be practiced in extreme programming: Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review**: Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.

- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- **Incremental development**: Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
- **Simplicity**: Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.
- **Basic principles of Extreme programming**: XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required

system. It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors.

- Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using the XP model are given below:

- **Coding**: The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
- **Testing:** XP model gives high importance to testing and considers it to be the primary factor to develop fault-free software.
- **Listening**: The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to

be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.

- **Designing**: Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
- **Feedback**: One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.