Task: 1

What is SDLC?

Software Development Life Cycle, clear roadmap map for a structured process used by development teams to design, build, and maintain software. It's a systematic approach that breaks down software development into distinct stages to ensure high-quality software is produced efficiently

Which learned different types for method which used in company as per request .

· Waterfall

· Agile

· Spiral

· Iterative

· RAD model

· V-model

Why SDLC

Structured approach to software development.It helps ensure projects are completed on time and within budget.By defining clear phases, it improves planning and execution.It enhances software quality through regular testing and validation.Overall, With better planning and risk management, SDLC helps in delivering projects on time and within budget.

STAGES IN SDLC

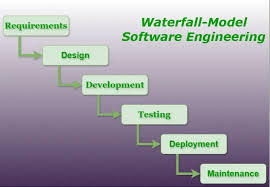
1. **Planning and Requirement Analysis**

The project's scope, objectives evaluated resources, risks, and timelines

1. **Defining Requirements** Clear, detailed documentation from end user fixing the target for project
2. **Designing** System architecture, database structure, and user interfaces are planned. This phase translates requirements into a blueprint for development.(LLD - HLD)
3. **Coding** Developers write the actual code based on the design documents. This is the execution phase where the software is built.
4. **Testing** The software is tested to identify bugs and verify functionality. It ensures the product meets user expectations.
5. ·**Deployment**  & **Maintenance** The software is released for end users & support is provided to fix issues and update the system

**Waterfall model**

The Waterfall model is a linear and sequential software development approach.It consists of distinct phases: requirement, design, implementation, testing, and maintenance.Each phase must be completed before moving to the next.It is simple to manage but not flexible to changes.



**When to Use:**

* Requirements are clearly defined and unlikely to change.
* The project is short and straightforward.
* Documentation and strict processes are important.
* The client approves each phase before moving forward.

**Application**

* Banking systems
* Payroll management software
* Manufacturing process control systems

**Advantages:**

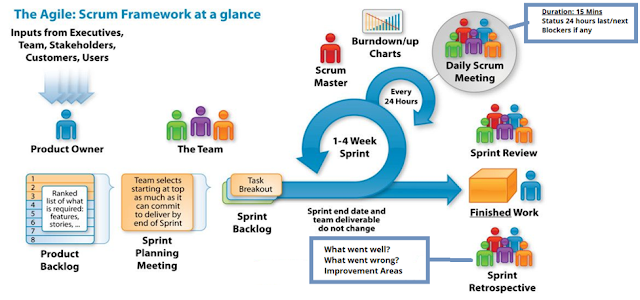
* Simple and easy to understand.
* Clear milestones and deliverables.
* Well-documented process.
* Good for projects with fixed requirements.

**Disadvantages:**

* Inflexible to changes once a phase is complete.
* Late testing can lead to undetected issues.
* Not suitable for complex or long-term projects.
* Client feedback is limited until the end.

**Agile model**

Agile is an iterative and flexible approach to software development.It breaks the project into small increments called sprints.Continuous feedback and collaboration with stakeholders are key.It allows quick adaptation to changes

during development.  


**When to Use:**

* When requirements are expected to change frequently.
* For complex or long-term projects needing flexibility.
* When early delivery of a working product is needed.
* When close collaboration with the client is possible.

**Application**

* Mobile app development
* Web applications with evolving features
* Startup MVP (Minimum Viable Product) development

**Advantages:**

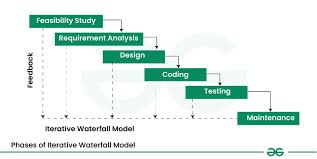
* Highly flexible and adaptable to changes.
* Continuous delivery of working software.
* Close collaboration with stakeholders.
* Faster detection and fixing of issues.

**Disadvantages:**

* Requires active user involvement.
* Less predictability in timeline and budget.
* Can be hard to scale for large teams.
* Documentation may be less detailed.

**Iterative model**

The Iterative model develops software through repeated cycles (iterations).  
Each iteration includes planning, design, coding, and testing.Feedback is used to improve the product with each cycle.It helps identify and fix issues early in development.



**When to Use:**

1. When requirements are not fully known at the start.
2. For large projects that can be broken into small parts.
3. When regular feedback and improvements are needed.
4. When early partial releases are beneficial.

**Application**

* Large enterprise software systems
* E-commerce platforms
* Complex data analytics tools

**Advantages:**

* Allows partial implementation and testing early.
* Flexible to changing requirements.
* Reduces risks by identifying issues early.
* Improves software quality with each iteration.

**Disadvantages:**

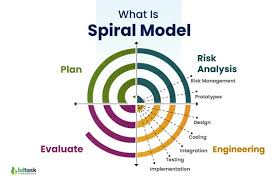
* Can be costly due to repeated cycles.
* Requires good planning and design.
* Risk of scope creep if not managed well.
* May need more resources than linear models.

**Spiral model**

The Spiral model combines iterative development with risk analysis.It goes through repeated cycles (spirals) of planning, risk assessment, development, and evaluation.Each loop of the spiral adds functionality and reduces risk.It emphasizes early identification and mitigation of risks.

**When to Use:**

1. For large, complex, and high-risk projects.
2. When frequent risk assessments are essential.
3. When requirements are unclear and likely to evolve.
4. When the client needs a structured yet flexible approach.



**Application**

* Aerospace control systems
* Defense and military applications

**Advantages:**

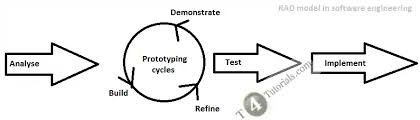
* Focus on risk assessment reduces project risks.
* Allows iterative refinement and prototype development.
* Suitable for large, complex, high-risk projects.
* Flexible to requirement changes.

**Disadvantages:**

* Can be expensive and time-consuming.
* Requires expertise in risk analysis.
* Complex to manage and implement.
* Not suitable for small projects.

**RAD (Rapid Application Development) Model:**

RAD focuses on fast development using prototypes and user feedback.It emphasizes quick iterations and reusable components.Development is done in parallel with continuous user involvement.It reduces development time and improves adaptability.



**When to Use:**

1. When a project needs to be developed quickly.
2. When users are available for frequent feedback.
3. For projects with well-defined business objectives.
4. When the system can be modularized into small parts.

**Application**

* Prototyping customer service portals
* Internal business dashboards
* Marketing campaign tools

**Advantages:**

* Fast development and delivery.
* Encourages user involvement and feedback.
* Reusable components reduce development time.
* Good for projects with clear, modularized requirements.

**Disadvantages:**

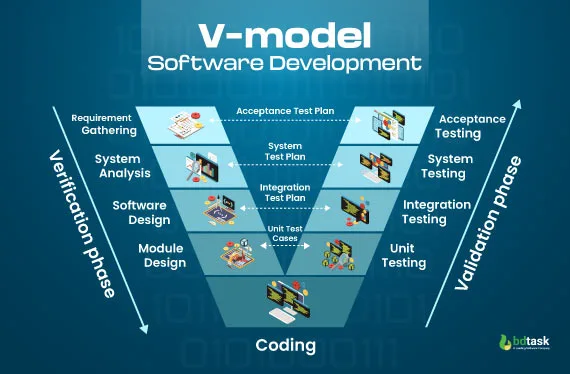
* Not suitable for very large or complex projects.
* Requires highly skilled developers and designers.
* Depends heavily on user availability.

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

The V-Model is an extension of the Waterfall model with a testing phase for each development stage.It follows a “V” shape, where development and testing activities run in parallel.Each phase must be completed before the next begins.It ensures early test planning and strict quality control.

**When to Use:**

1. For projects with clear, fixed requirements.
2. When high reliability and quality are critical.
3. In systems requiring thorough testing (e.g., medical, defense).
4. When validation at each stage is necessary.



**Application**

* Embedded systems (automotive, avionics)
* Safety-critical software (medical, nuclear plants)
* Telecommunications systems

**Advantages:**

* Simple and easy to manage due to strict phases.
* Early test planning improves defect detection.
* Works well for projects with well-defined requirements.
* Emphasizes verification and validation.

**Disadvantages:**

* Very rigid and less flexible to changes.
* Late design changes are difficult and costly.
* Not suitable for projects with unclear requirements.
* Testing begins only after implementation.

**What is Scrum ?**

Scrum is an Agile framework used to manage and complete complex projects, especially software development. It breaks work into small, manageable pieces called sprints (usually 2–4 weeks long), allowing teams to deliver parts of the product frequently. Scrum emphasizes teamwork, accountability, and iterative progress through roles like Product Owner, Scrum Master, and Development Team. It encourages continuous feedback, adaptation, and improvement to meet changing requirements and deliver high-quality results efficiently.

**What is sprint?**

A **Sprint** is a fixed time-boxed period in Scrum—usually lasting 1 to 4 weeks—during which a specific set of work or tasks (called the **Sprint Backlog**) is completed. It’s a focused cycle where the development team designs, builds, and tests a potentially shippable product increment. At the end of each sprint, the team reviews the work done, gathers feedback, and plans the next sprint, allowing for continuous improvement and adaptation. Specfic set of task at specific time

**What to do while working in a sprint:**

1. **Focus on Sprint Goals:** Work only on the tasks committed in the Sprint Backlog.
2. **Collaborate Constantly:** Communicate regularly with team members and the Scrum Master.
3. **Attend Scrum Events:** Participate actively in Daily Stand-ups, Sprint Planning, Sprint Review, and Retrospective.
4. **Deliver Increment:** Aim to complete a potentially shippable product increment by sprint end.
5. **Update Progress:** Keep task boards or tracking tools updated for transparency.

**What not to do while working in a sprint:**

1. **Avoid Scope Creep:** Don’t add new tasks or change requirements mid-sprint without team agreement.
2. **Don’t Skip Meetings:** Missing Scrum ceremonies can cause misalignment and block progress.
3. **Avoid Multitasking:** Don’t switch between unrelated tasks that can reduce focus and productivity.
4. **Don’t Ignore Blockers:** Report issues early instead of hiding problems that can delay progress.
5. **Avoid Overcommitting:** Don’t commit to more work than the team can realistically complete in the sprint.

### **What is a Backlog?**

1. A list of all tasks, features, and improvements needed in a project.
2. Managed and prioritized by the Product Owner.
3. Divided into **Product Backlog** (entire project) and **Sprint Backlog** (current sprint).
4. **Product Backlog**: Contains all high-level features, bug fixes, enhancements, and tasks for the entire product.
5. **Sprint Backlog**: A subset of the product backlog selected for a specific sprint.
6. Continuously updated as the project evolves.

### **What are User Stories?**

1. Small, specific pieces of work that describe a user’s need or feature.
2. Written from the end-user's point of view.
3. Help the team understand what to build and why.

### 

### 

### **Artifacts**

These are the tools and documents that Scrum teams use to track progress. They include:

### **What is a Product Backlog?**

1. The **Product Backlog** is a prioritized list of everything needed to improve or build the product.
2. It includes **features, bug fixes, enhancements, technical work, and research tasks**.
3. It is owned and maintained by the **Product Owner**.
4. Items in the backlog are written as **user stories** or tasks.
5. It is **dynamic**—continuously updated based on feedback, business needs, and market changes.
6. The top items are more detailed and ready for development (known as **backlog refinement**).
7. Developers pull items from the backlog during **Sprint Planning** to create the Sprint Backlog.

### **What is a Sprint Backlog?**

1. A **Sprint Backlog** is a list of tasks selected from the **Product Backlog** to be completed during a specific sprint.
2. It represents the team’s **commitment** for that sprint.
3. Contains detailed tasks, estimates, and responsibilities.
4. Managed and updated by the **development team** throughout the sprint.
5. Helps track progress and ensures the sprint goal is achieved.
6. It is **flexible internally**—the team can adjust how they work on tasks—but the sprint goal should not change.
7. Created during the **Sprint Planning** meeting at the start of each sprint.

### **What is an Increment in Scrum?**

1. **Increment** is the sum of all completed work in a sprint, added to the previous increments.
2. It is a **usable, working piece of the product** that meets the Definition of Done.
3. It must be **potentially shippable**—meaning it can be released to users if the Product Owner decides.
4. Each increment adds **real value** to the product and shows progress.
5. Delivered at the **end of every sprint**.

### **What is an Burn-Down Chart ?**

A **Burn-Down Chart** is a project management tool used to visualize progress over time in agile methodologies (like Scrum). It helps teams track how much work remains in a sprint or project and whether they are on pace to complete it by the deadline.

Networking Concepts

### **What are Ports?**

1. **Ports** are **virtual connection points** through which data enters or leaves a computer in a network.
2. They help computers understand **which service or application** the data is meant for.
3. Ports are identified by numbers from **0 to 65535**.
4. Two main types:  
   * **Well-known ports (0–1023):** Assigned to standard services.
   * **Dynamic/private ports (49152–65535):** Used for temporary or custom connections.

### **What are Protocols?**

1. **Protocols** are **rules that govern data communication** between devices.
2. They define how data is formatted, transmitted, and received.
3. Examples:  
   * **TCP (Transmission Control Protocol):** Reliable, ordered data delivery.
   * **UDP (User Datagram Protocol):** Faster but no guarantee of delivery.
   * **HTTP/HTTPS:** Used for accessing websites.
   * **FTP/SFTP:** For transferring files.
   * **DNS:** Resolves domain names to IP addresses.

### 

### **Different Types of Computer Networks**

Here are the main types of computer networks, based on size, range, and purpose:

### **1. LAN (Local Area Network)**

* Covers a **small area** like a home, school, or office.
* **Fast and inexpensive**.
* Devices are connected through Ethernet or Wi-Fi.
* Example: Office computers connected to a single router.

### **2. MAN (Metropolitan Area Network)**

* Covers a **city or large campus**.
* Larger than a LAN but smaller than a WAN.
* Often used by universities or government buildings across a city.
* Example: A city-wide network for public libraries.

### **3. WAN (Wide Area Network)**

* Covers **large geographical areas** (states or countries).
* Uses leased telecommunication lines or satellites.
* Example: The **Internet** is the largest WAN.

### **4. PAN (Personal Area Network)**

* Very small range (a few meters).
* Used for **personal devices** like smartphones, laptops, Bluetooth headsets.
* Example: Connecting your phone to your smartwatch via Bluetooth.

### **5. WLAN (Wireless LAN)**

* A **wireless version of LAN**, using Wi-Fi.
* Used in homes, cafes, and offices.
* Example: Home Wi-Fi network.

### **6. SAN (Storage Area Network)**

* A **specialized high-speed network** that provides access to data storage.
* Used in data centers.
* Example: A server accessing external storage devices.

### **7. VPN (Virtual Private Network)**

* **Secure network** built over a public network like the internet.
* Provides privacy and encryption.
* Example: Remote employees accessing company data securely.

8. **Campus Area Network (CAN)**

* **Definition**: A **Campus Area Network** (CAN) connects multiple LANs within a **university, college campus, or large organization**.
* **Scope**: Larger than a LAN, smaller than a MAN.
* **Use Case**: Ideal for connecting departments, libraries, hostels, and administrative buildings within the same campus.
* **Example**: A university network linking the engineering block, library, and administrative office under one central system

### **Types of VPN (Virtual Private Network)**

VPNs create a **secure, encrypted connection** over a public network (like the internet). There are several types of VPNs, each suited for different use cases.

### **1. Remote Access VPN**

* **Used by:** Individuals or employees working from remote locations.
* **Function:** Allows users to connect securely to a company’s internal network from anywhere.
* **Example:** Employees accessing company files from home.

### **2. Site-to-Site VPN**

* **Used by:** Large organizations with multiple offices/branches.
* **Function:** Connects entire networks to each other (e.g., Office A ↔ Office B).
* **Also known as:** Router-to-router VPN.
* **Example:** A corporate office connecting to its branch network.

### **3. Client-to-Site VPN (SSL VPN)**

* **Used by:** Remote users via a web browser.
* **Function:** Accesses company applications without full network access.
* **More secure and browser-friendly** than traditional remote access VPNs.
* **Example:** University portal accessed by students over SSL VPN.

### **4. Mobile VPN**

* **Used by:** Mobile device users (smartphones, tablets).
* **Function:** Maintains a secure connection even when switching networks (e.g., from Wi-Fi to mobile data).
* **Ideal for:** Field workers and police/military personnel.

### **5. Cloud VPN (Virtual Private Cloud VPN)**

* **Used by:** Businesses using cloud services (e.g., AWS, GCP).
* **Function:** Securely connects on-premise infrastructure to cloud environments.
* **Example:** Connecting a company’s local data center to its Google Cloud services.

### **6. IPsec VPN (Internet Protocol Security)**

* **Used for:** Securing IP communications through authentication and encryption.
* **Often used in:** Site-to-site VPNs or remote access with added security.
* **Protocol-based VPN**, often combined with L2TP or IKEv2.

**What are the types of servers ?**

###### 

### **✅ Types of Servers in Networking**

Servers are specialized computers or software that provide services to other devices (called clients) over a network. Here are the main types:

### **1. Web Server**

* Hosts websites and serves web pages to users via **HTTP/HTTPS**.
* Example: Apache, Nginx, Microsoft IIS.

### **2. File Server**

* Stores and manages **files and folders** that can be accessed over the network.
* Used in offices to share documents centrally.
* Example: Windows File Server, Samba.

### **3. Database Server**

* Stores and manages **databases**; handles data queries and transactions.
* Clients use it to read/write data.
* Example: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

### **4. Mail Server**

* Manages the **sending, receiving, and storage** of email messages.
* Example: Microsoft Exchange, Postfix, Zimbra.

### **5. Application Server**

* Hosts **business logic** and delivers applications to client devices.
* Often works with web and database servers.
* Example: Apache Tomcat, WebLogic.

### **6. DNS Server (Domain Name System)**

* Translates **domain names into IP addresses**.
* Example: BIND, Microsoft DNS Server, Google Public DNS.

### **🔹 7. Proxy Server**

* Acts as an **intermediary** between client and other servers.
* Used for **caching**, security, and content control.
* Example: Squid, Nginx as reverse proxy.

### **8. FTP Server**

* Transfers files using the **File Transfer Protocol (FTP)**.
* Example: FileZilla Server, vsftpd.

### **9. Game Server**

* Hosts multiplayer **online games** and syncs game data for all players.
* Example: Minecraft server, Steam server.

### **10. Virtual Server**

* A **software-defined server** running inside a physical server.
* Allows multiple servers to run on one machine using virtualization tools.
* Example: VMware, Hyper-V.

**What do you know about DNS? Domain Name Service**

DNS (Domain Name System or Service) is often called the “Phonebook of the Internet” because it translates human-friendly domain names into machine-friendly IP addresses.

### Key Points About DNS:

1. Purpose:  
    Converts domain names (like www.google.com) into IP addresses (like 142.250.64.100), so computers can locate each other on a network.
2. Why It's Needed:  
    Humans remember names, but computers communicate using IP addresses. DNS bridges the gap.
3. How It Works:  
   * You type a website address in your browser.
   * DNS finds the correct IP address for that domain.
   * Your browser uses the IP to connect to the correct server.
4. DNS Hierarchy:  
   * Root DNS Servers → handle top-level domains (like .com, .org)
   * TLD Servers → direct to domain name servers for specific names (like example.com)
   * Authoritative DNS Servers → provide the final IP address for the domain.
5. Common DNS Records:  
   * A record – Maps a domain to an IPv4 address
   * AAAA record – Maps a domain to an IPv6 address
   * CNAME – Alias for another domain
   * MX – Mail server for the domain
   * NS – Name server for the domain
   * TXT – Text information, often used for email verification

### Example:

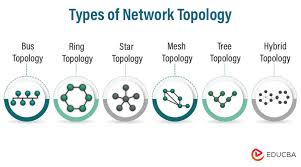
Typing www.amazon.com in a browser triggers:

1. A DNS query is sent.
2. DNS server returns the IP address (e.g., 205.251.242.103).
3. Your device connects to that IP and loads the website.

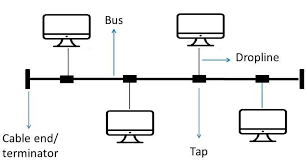
Network topology

The arrangement of nodes and connections, which dictates how data flows between devices. It's essentially the blueprint for how a network is physically and logically structured, impacting its performance, security, and scalability

Types of network topology

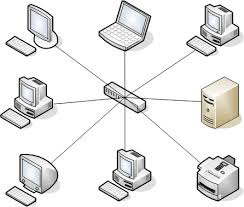
****

# 1. Bus Topology



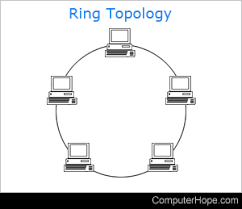
* All devices share a single communication line (backbone).
* Advantages: Easy to install, low cost.
* Disadvantages: If the main cable fails, the whole network goes down.

### 2. Star Topology



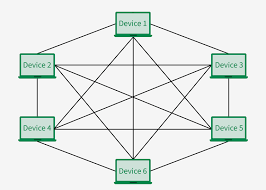
* All devices connect to a central hub/switch.
* Advantages: Easy to manage and add new devices; one device failure doesn’t affect the rest.
* Disadvantages: If the hub fails, the entire network is affected.

### 3. Ring Topology



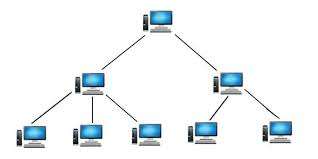
* Devices are connected in a circular fashion.
* Advantages: Data flows in one direction, reducing collisions.
* Disadvantages: A single node failure can disrupt the entire network unless a dual ring is used.

### 4. Mesh Topology



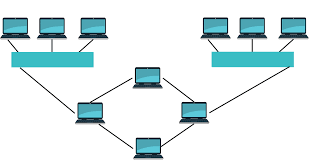
* Every device is connected to every other device.
* Advantages: High redundancy and reliability.
* Disadvantages: Expensive and complex to set up.

### 5. Tree Topology (Hierarchical)



* Combination of star and bus topology; arranged like a tree with root and branches.
* Advantages: Scalable and supports future expansion.
* Disadvantages: If the backbone fails, entire segments can go down.

### 6. Hybrid Topology

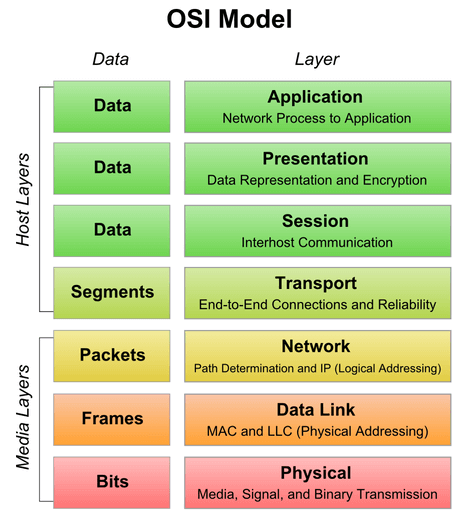


* Mix of two or more topologies (e.g., star + mesh).
* Advantages: Flexible, scalable, and customizable.
* Disadvantages: Complex design and costly implementation.

### 

### **OSI Model & Its Layers**

The OSI (Open Systems Interconnection) Model is a set of rules that explains how different computer systems communicate over a network.



## **Layers of the OSI Model**

1. **Physical Layer :** Deals with the hardware and transmission of raw data bits over physical mediums like cables or radio waves. It defines voltage levels, connector types, and physical data rates.
2. **Data Link Layer :** Responsible for node-to-node data transfer and detecting/correcting errors. It structures data into frames and handles MAC addresses.
3. **Network Layer :** Manages device addressing and routing of data packets across networks. Uses logical IP addresses to determine the best path.
4. **Transport Layer:** Ensures reliable data delivery using protocols like TCP and UDP. Handles segmentation, flow control, and error correction.
5. **Session Layer :** Establishes, maintains, and terminates connections (sessions) between applications. Ensures orderly communication and synchronization.
6. **Presentation Layer :** Translates data formats between systems (e.g., encryption, compression). Makes sure data is readable for the application layer.
7. **Application Layer :** Closest to the user, it provides network services to applications like email, FTP, and browsers. Facilitates direct interaction with the network.

### Add ons-

### 7. What is Software Configuration Management, and how does it work?

The process of tracking and regulating changes that occur during the software development lifecycle is known as software configuration management. Any modification made during the development of software must be tracked using a well-defined and controlled process. Any modifications performed during software development are regulated through a well-defined process, thanks to configuration management. Revision control and the establishment of baselines are two SCM procedures.

### 8. What do a Software Project Manager's responsibilities entail?

The Software Project Manager is in charge of seeing the project through to completion. The Software Project Manager is responsible for ensuring that the entire team follows a methodical and well-defined approach to software development. They also handle project planning, tracking project status, resource management, and risk management.

### 9. What do you know about Scrum impediments?

Obstacles or challenges that the scrum team faces slow down their work speed are referred to as impediments. An obstacle is anything that tries to prevent the scrum team from getting work "Done." Impediments can take many different forms. Some of the roadblocks include resource shortages or sick team members, technical, operational, and organisational issues, a lack of management support systems, and business issues.

### 10. Briefly explain Scrum methodology in the Agile model.

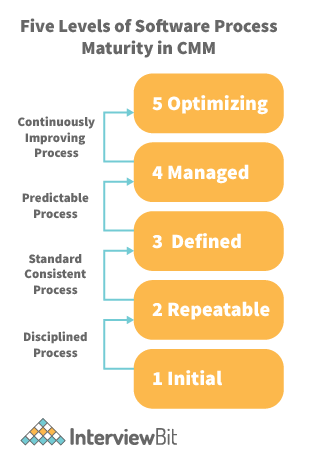
Scrum is an agile development approach based on iterative and incremental procedures that are used in the creation of software. It's an agile structure that's adaptable, rapid, flexible, and excellent at delivering value to customers throughout the project's development. Companies of all sizes employ the Agile Scrum technique because of its ability to provide high-end cooperation and efficiency for project-based work. Scrum is a sort of agile approach that breaks projects down into manageable parts known as "sprints." The Agile Scrum methodology is ideal for companies who need to complete projects fast.

### 11. What are Capability Maturity Model(CMM) levels?

Following are the five Capability Maturity Model Levels:

* Initial: The first step is to create an unstable process environment. The software development process is considered haphazard and even chaotic at times. There are few methods that have been specified, and success is based on individual effort and heroism.
* Repeatable: Work is planned and monitored, making it repeatable. To track cost, schedule, and functionality, basic project management techniques are implemented.
* Defined: This level encompasses written and defined standards that evolve over time and support consistent performance. The work is well defined at this point.
* Managed: Extensive data on the software development process and product quality are gathered. Both the software development process and the end products are quantified and managed.
* Optimized: Work is based on continuous improvement (optimization). The focus on continuously improving process performance is a significant feature of this level.

### 12. What is Capability Maturity Model?

****

The Capability Maturity Model (CMM) is a cross-discipline and technical paradigm for facilitating and refining software development processes and system improvement. This methodology is at the heart of most management systems that aim to improve the quality of all product and service development and delivery.

### 13. What is Level-0 DFD?

Context Diagram is another name for DFD Level 0. It's a high-level overview of the entire system or process that's being studied or modelled. It's meant to be a quick peek into the system, displaying it as a single high-level process with its connections to external entities. Stakeholders, business analysts, data analysts, and developers should all be able to understand it readily.

### 14. How can DDLC and SDLC work together?

The DDLC (Documentation Development Life Cycle) is a software documentation development life cycle used by technical documenters to prepare software documentation. The life cycle is followed in tandem with the SDLC, as testers and developers work on the programme at the same time. Because the documentation requires input and feedback from the various phases of the SDLC, the DDLC has stages that are comparable to the SDLC.

### 15. What are different types of prototype model?

There are four types of Prototyping models:

* Rapid Throwaway prototypes.
* Evolutionary prototype.
* Incremental prototype.
* Extreme prototype.

### 16. What is FRS document?

This document captures the user's voice from the outside, or the end user's perspective. A Business System Analyst creates it (BSA). This paper demonstrates how a system will react when a user interacts with it in order to meet the BRD and SRD standards. The key area of interest for software experts is the Functional Requirement Specification (FRS). An FRS is useful for software testers to learn the situations in which the product is intended to be tested, just as it is for developers to understand what product they are planning to produce. An FRS's ultimate purpose is to meet all of the requirements outlined in the SRS and BRS regulations.

### 17. What is the Software release process?

The Software Development Life Cycle (SDLC) release phase is historically connected with production, deployment, and post-production operations, which generally include software maintenance and support. So, release management is the process of managing, planning, scheduling, and controlling a full software development at every stage and environment, including testing and releasing software releases.

### 18. What is the use of JAD session?

JAD is a strategy for defining business system requirements that are commonly utilised in the early phases of a systems development project. JAD's goal is to bring MIS and end-users together in a structured workshop setting in order to extract outcome system needs. It allows clients and developers to swiftly agree on a project's fundamental scope, objectives, and specifications

## SDLC MCQ

1.

A feasibility study using the SDLC model is conducted to

determine whether or not the project is technically possible

determine whether the proposal is financially viable

Both a and b

None of the above

2.

A well-documented life cycle model aids in the detection of what during the development phase?

Inconsistencies

Redundancies

Omission

All of the above

3.

How many lines of code does the Build & Fix Model suit for programming exercises?

100-200

300-400

600-700

Above 800+

4.

In which life cycle does regression testing play a significant role?

Waterfall model

V model

Iterative model

All of the above

5.

What determines if the project should go forward?

feasibility assessment

opportunity identification

system evaluation

program specification

6.

What is the most significant disadvantage of employing the RAD Model?

Developers/designers that are highly specialized and skilled are required.

Component reusability is improved.

Encourages client/customer input.

Increases component reusability.

7.

Which of the following developmental models is incremental?

Prototyping, V model, Agile

Prototyping, RAD, Agile, RUP

Prototyping, V model, RAD, Agile, RUP

All of the above

8.

Which of the following is an Agile development characteristic?

Shared code ownership

Test-Driven Development

Implement the simplest solution to meet today's problem

Continual feedback from customer

All of the above

9.

Which of the following steps in the SDLC framework are valid?

Requirement Gathering

Software Design

System Analysis

All of the above

10.

Who is in charge of system development, staffing, budgeting, and reporting, as well as ensuring that deadlines are met?

Project managers

Network engineers

Graphic designers

Systems analysts