Task 1 -

The SDLC (Software Development Life Cycle is a systematic process used in software engineering to plan, design, develop, test and maintain software applications. It provides a structured approach, breaking down the software development work into smaller, manageable phases.

Task 2 -

The major goal here is to produce high quality software that meets the customer expectations and requirements. It brings order and control to software development.

Task 3 -

The SDLC typically follows seven stages:

Planning - Here we define the project's scope, objectives, and resources. It includes tasks like cost-benefit analysis, scheduling, and resource estimation.

Requirements Analysis - Here we focus on understanding the user's needs and translating them into specific, measurable, achievable, relevant, and time-bound requirements.

Design - Here we create the software's architecture and layout, ensuring it meets the defined requirements.

Coding - Here we translate the design into executable code, which is understandable by the machine.

Testing - Here we systematically check the application for defects, errors, and issues. It ensures the software functions are correct and meets the defined requirements.

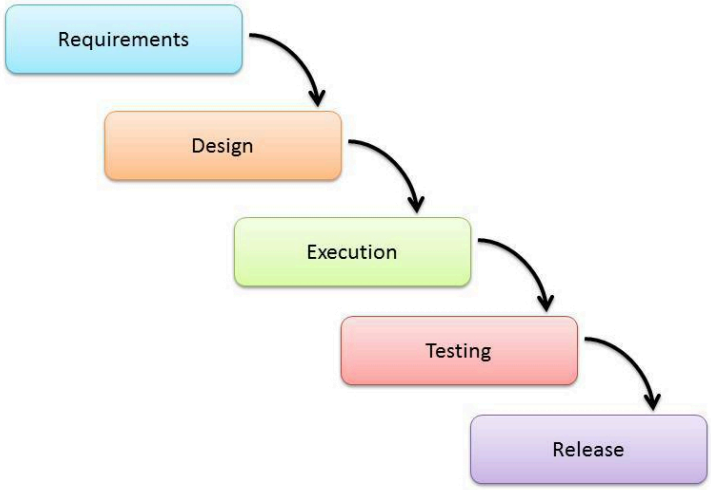
Deployment - The developed software is delivered to the customer/client for use. This includes installing the software and ensuring it works as expected in the customer's production environment.

Maintenance - Here we provide support and updates to the software, addressing any issues that arise and adapting to changing user needs.

<u>Task 4</u> -

Models of SDLC:

Waterfall Model: A step by step approach where each phase is completed before the next begins. It's suitable for projects with well-defined requirements and minimal changes. Each stage relies on information from the previous stage and has its own project plan. Waterfall is easy to understand and simple to manage. And since there is little room for revisions once a stage is completed, problems can't be fixed until you get to the maintenance stage.

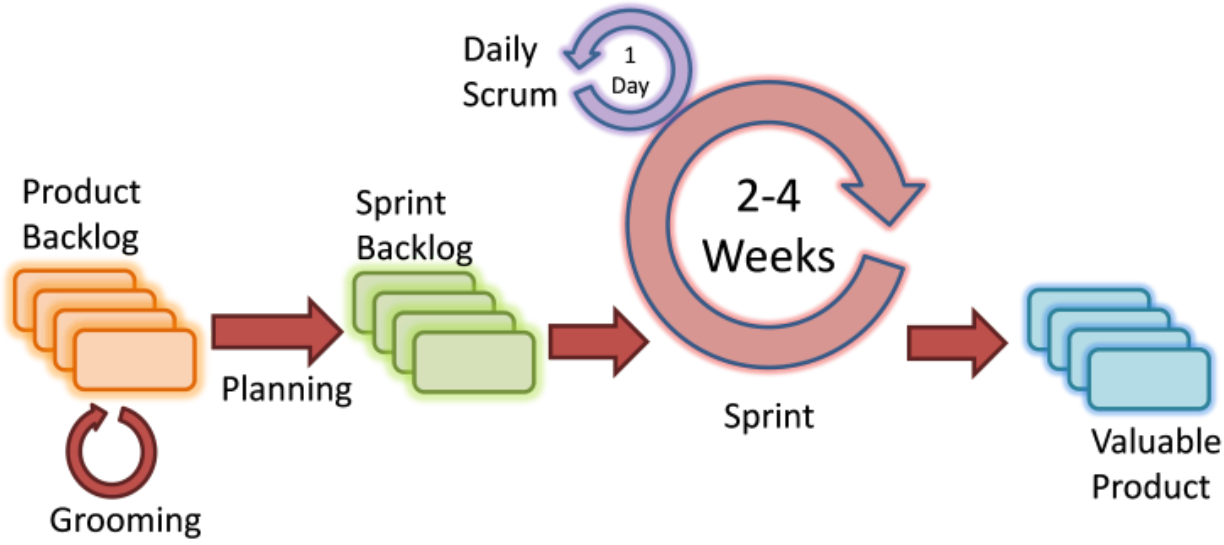


Application - This model is ideal for projects with clear, well-defined requirements, stable technology, and a strong understanding of the project scope.

Advantage - It is very easy and convenient to implement
Disadvantage - may cause some confusion if some of the changes are made at some phases, Only at the end, the customer can see the working model of the project

Agile Model: An iterative and incremental approach that focuses on flexibility. It's mostly used for projects with changing requirements and frequent client

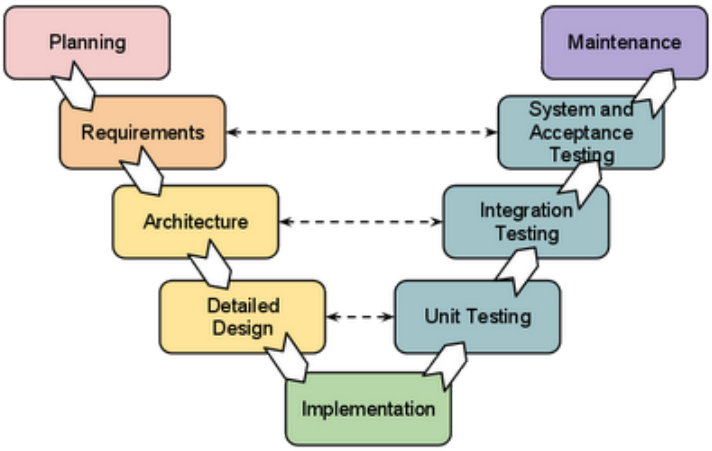
interaction. It's chosen when the focus is more on collaboration with clients. By breaking the product into cycles, the Agile model quickly delivers a working product and is considered a very realistic development approach.
But since this model depends heavily on customer interaction, the project can head the wrong way if the customer is not clear on the direction he or she wants to go.



Application - Agile is best suited for projects with changing requirements, frequent feedback, and a focus on collaboration and flexibility.

Advantage - Manage Change More Effectively, Improved Customer Engagement
Disadvantage - Reliance on Experienced Developers, Potential for Project Derailment

V-Model: An extension of the Waterfall model that emphasizes verification and validation throughout the development process. It's good for projects where testing is crucial and requirements are clear. This model is useful when there are no unknown requirements, as it’s still difficult to go back and make changes.



Application - It is best suited where there are clear and stable requirements, projects with strong testing focus, complex systems, industries with strict regulations.

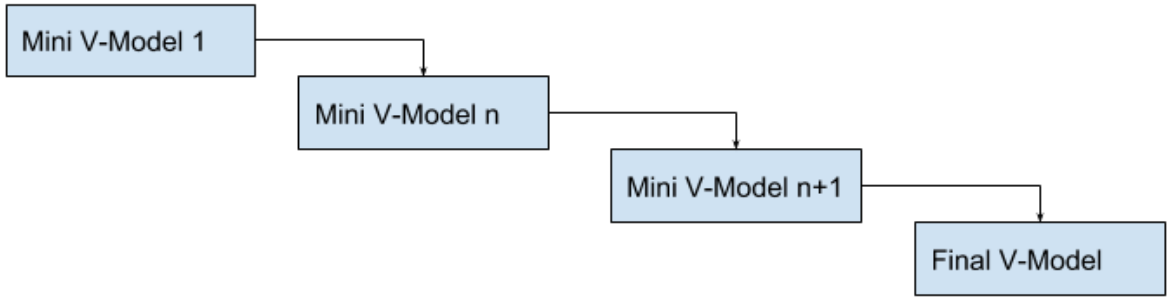
Advantage - Early Planning and Test Design, Simplicity and Ease of Use, Suitability for Small Projects
Disadvantage - Rigidity and Limited Flexibility, Documentation Overhead with Changes

Iterative Model: Breaks down the project into smaller iterations, where each iteration produces a working version of the software. It's useful for projects where requirements are not fully defined. One advantage over other SDLC models: This model gives you a working version early in the process and makes it less expensive to implement changes.

Application - This model is used when continuous improvement and testing are important, and the project involves iterative cycles of development and refinement.

Advantage - Easier to manage risk because risky pieces are identified and handled during its iteration
Disadvantage - Problems may arise about system architecture because not all requirements are gathered upfront for the entire software life cycle

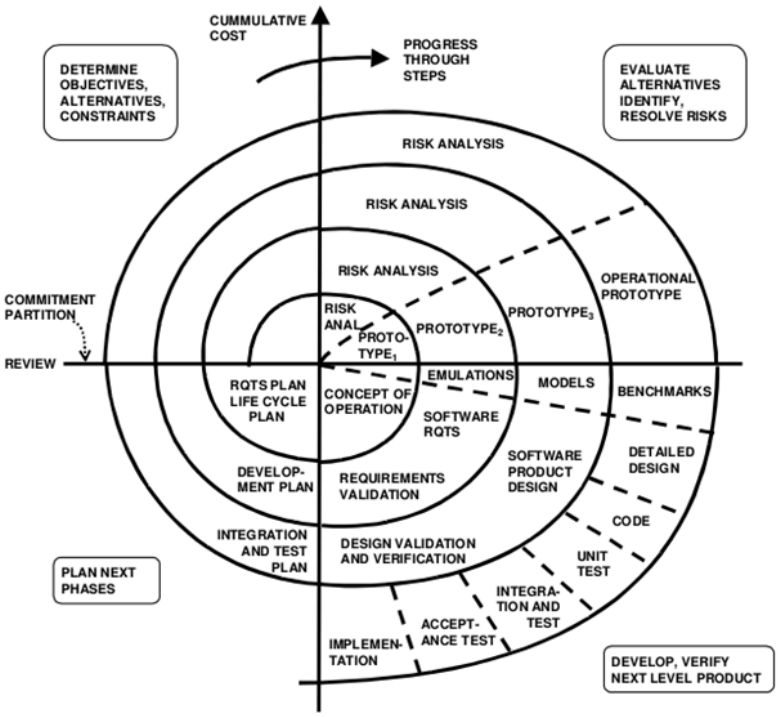
Incremental Model: Similar to the iterative model, but each increment focuses on a specific feature or module. It's good for projects where requirements are clear and the software can be delivered in smaller, manageable parts.



Application - This model is used when it's desirable to deliver software in smaller, manageable increments, allowing for early customer feedback and flexibility.

Advantage - Technical risks can be managed effectively and in a well-organized manner
Disadvantage - Highly and fully committed developers and customers are required for this project

Spiral Model: Combines elements of Waterfall and iterative models, emphasizing risk management. It's suitable for high-risk projects with evolving requirements. This model allows for the building of a highly customised product, and user feedback can be incorporated from early on in the project.



Application - This model is well-suited for projects with high-risk and complex requirements, where thorough risk assessment and iterative development are crucial.

Advantage - It is very Cost-effective and easy to maintain. Well planned and efficient project development
Disadvantage - If communication with customer is not well or proper then it results in total project failure or non-success or project that might leads to omission of the project

Task 7 -

Scrum: It is an agile framework to manage complex product development, an iterative and incremental approach, focused on delivering value in short cycles called "sprints". It provides a structured approach for teams to collaborate effectively and manage complexity particularly in environments where requirements are likely to change during the development process.

Task 8 -

Sprint: It is a short period when a scrum team works to complete a set of work items.

Task 9 -

Do's and Don'ts while working on Sprints -

Do's:
- Split Large tasks
- Involve everyone's feedback
- Learn from Previous Sprints

Don'ts:
- Forget About Quality
- Get deviated from the initial focus
- leave larger gaps between two sprints. Keep it short

Task 10 -

Stories and Backlogs -

Stories: These are short, user-focused descriptions of features that a team can commit to completing within a sprint.

Backlog: It is a prioritized list of all features, functionalities, and requirements for a software product.

Task 11 -

Scrum Artifacts:

Product Backlog:
This is a prioritized list of all the features, enhancements, and bug fixes that the team needs to consider for the product.

Sprint Backlog:

This is a subset of the Product Backlog that the team chooses to work on during a specific Sprint. It includes the Sprint Goal, the tasks that will be completed, and the estimated effort required for each task.

Increment:
This is the potentially releasable product that is delivered at the end of each Sprint. It's the result of the work completed during the Sprint and should be verified to meet the Definition of Done.

Definition of Done:
This defines the criteria that must be met for a product increment to be considered complete and potentially shippable

Task 12 -

Ports and Protocols:

Ports: These are numeric identifiers that distinguish different processes or applications running on a single computer.

Protocols: These are sets of rules that define how data is formatted, transmitted, and received across a network.

Task 13 -

Different network types:

Local Area Networks (**LAN**)
Metropolitan Area Networks (**MAN**)
Wide Area Networks (**WAN**)
Personal Area Networks (**PAN**)
Virtual Private Networks (**VPN**)

Task 14 -

Different types of servers:

Web Servers:
Host websites and web applications, delivering content to users through the internet. They respond to HTTP requests from clients (usually web browsers) and serve web pages, images, and other resources.

Application Servers:
Run and manage application software, allowing users to access and interact with these applications. It plays a significant part within the design of numerous present day program applications.

Database Servers:
Manage and store databases, providing access to organized data. They support various database management systems (DBMS) and enable applications to interact with data via queries and transactions

Mail Servers:
Handle email communication, including sending, receiving, and storing emails. Mail servers ensure proper email routing, spam filtering, and security through encryption.

Proxy Servers:
Act as an intermediary between clients and other servers, providing security and performance benefits. They provide benefits like caching commonly accessed content, which speeds up data delivery

DNS Servers:
Translate domain names into IP addresses, making it easier to access websites and services on the internet.

Virtual Servers:
Simulate physical servers within a virtual environment, enabling flexibility and scalability. It permits numerous working frameworks and applications to run on a single physical server at the same time

Task 15 -

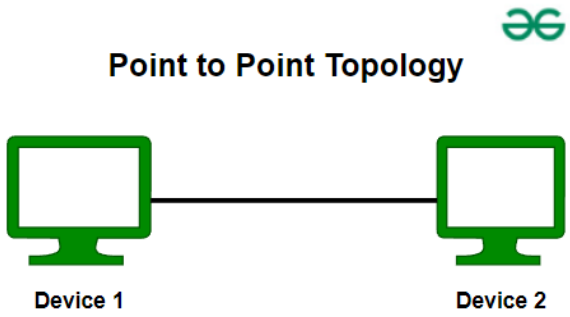
What do you know about DNS? Domain Name Service

It basically helps in translating human readable host names into machine readable IP addresses.This is in place because it helps users to remember easily readable names instead of their respective numerical IP addresses.

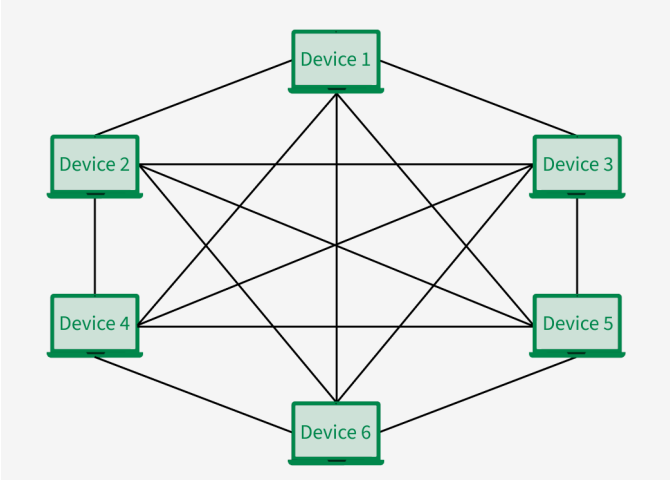
Task 16 -

Different types of Network topologies:

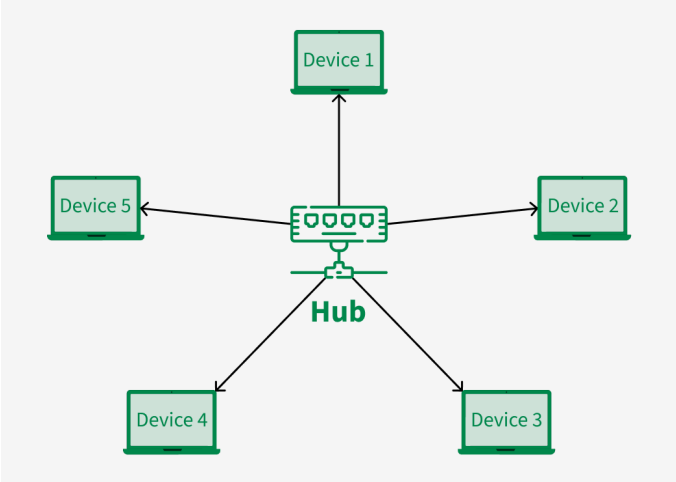
Point to Point Topology: This is a type of topology that works on the functionality of the sender and receiver. It is the one of the simplest communications between two nodes.



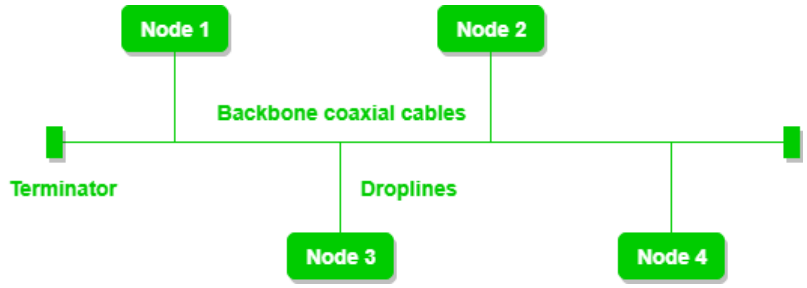
Mesh Topology: Here, every device is connected to another device via a particular channel. Every device is connected to another via dedicated channels. These channels are known as links.



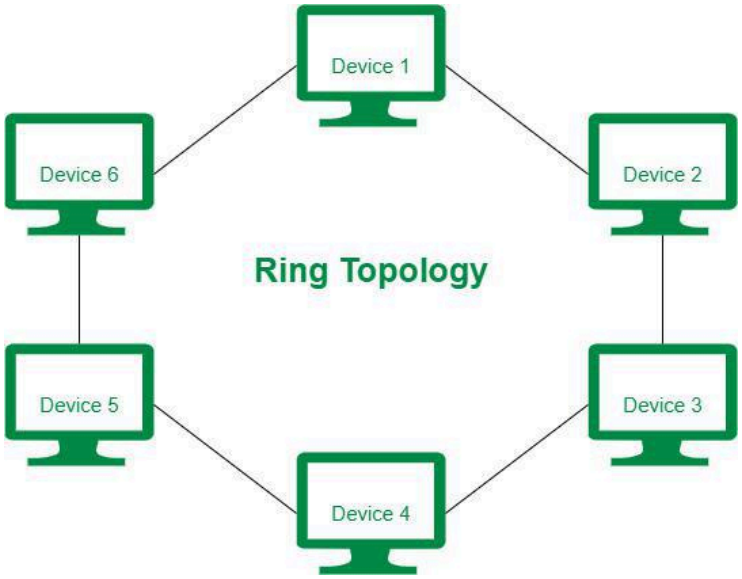
Star Topology: Here, all the devices are connected to a single hub through a cable. This hub is the central node and all other nodes are connected to the central node.



Bus Topology: This is a network type in which every computer and network device is connected to a single cable. It is bi-directional.



Ring Topology: Here, it forms a ring connecting devices with exactly two neighboring devices.



<u>Task 17</u> -

What is the OSI model and describe the layers within it?

The OSI model, or Open Systems Interconnection model, is a conceptual framework that divides network communications into seven abstract layers

1. Physical Layer:
This layer deals with the physical transmission of data, including cables, hardware, and signaling.

2. Data Link Layer:
This layer focuses on error-free transmission of data between two directly connected nodes within a network, using protocols like Ethernet.

3. Network Layer:
This layer is responsible for routing data packets between different networks, using IP addresses and routing protocols.

4. Transport Layer:
This layer provides reliable and unreliable data delivery between applications, using protocols like TCP and UDP.

5. Session Layer:
This layer manages connections (sessions) between applications, handling tasks like authentication and synchronization.

6. Presentation Layer:
This layer handles data formatting, encryption, and decryption, ensuring data is presented in a way that both the sender and receiver can understand.

7. Application Layer:
This layer provides the interface for applications to access network services, using protocols like HTTP, SMTP, and FTP.