**Task 1**

What is SDLC?

SDLC stands for software development life cycle which starts from planning and ends in maintenance.

It is a structured approach which helps in managing software development projects, minimizing risks and ensuring software meets customer expectations.

SDLC provides a framework for organizing and executing the software development process and helps in identifying and managing risks throughout the software development.

SDLC helps in understanding the clear scope for the software and helps in managing the time accordingly.

**Task 2**

Why is SDLC?

SDLC helps in managing project complexities, improves quality and time efficiency.

It increases the visibility of the process to stakeholders.

Since it is a phased and structured approach. Hence we can able to mitigate risk at every stage which will minimize the efforts.

**Task 3**

What are the stages of SDLC?

The SDLC has seven stages

* **Planning:**

Planning stage involves defining the project's scope, objectives, and resources. It includes tasks like cost-benefit analysis, scheduling, and resource estimation.

* **Requirements Analysis:**

This phase focuses on understanding the user's needs and translating them into specific, measurable and achievable requirements.

* **Design:**

The design phase involves creating the software's architecture and layout, ensuring it meets the defined requirements. It will have both low level and high level designs.

* **Coding:**

This stage translates the design into executable code.

* **Testing:**

This phase involves systematically checking the software for defects, errors, and issues with both internal and UAT testing.

* **Deployment:**

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

* **Maintenance:**

This phase involves ongoing support and updates to the software, addressing any issues that arise and adapting to changing user needs.

**Task 4:**

**SDLC Models:**

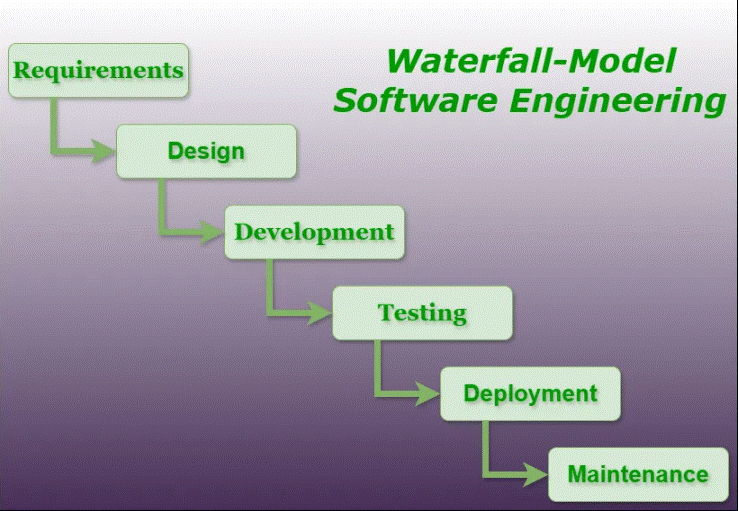
SDLC Models are frameworks that guide the development process of software applications from initiation to deployment. Various SDLC models in software engineering exist, each with its approach to the phases of development.

The different models are:

1. **Waterfall SDLC Models**

The Waterfall model is one of the oldest and most straightforward approaches to software development.The Waterfall model follows a linear and sequential approach to software development. Each phase in the development process must be completed before moving on to the next one, resembling the downward flow of a waterfall

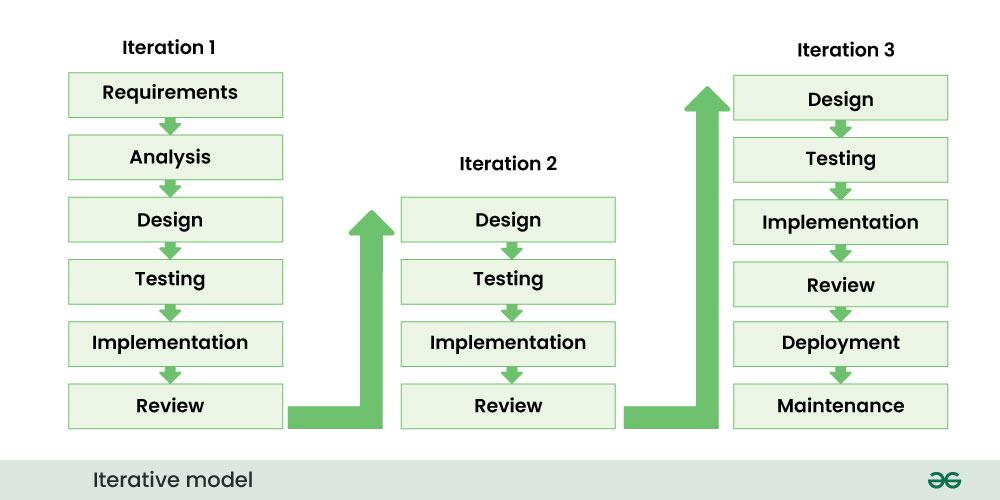
Eg: building a bridge



1. **Iterative SDLC Models**

The Iterative SDLC model stands out as a flexible and efficient methodology that promotes continuous improvement and adaptability. The Iterative model emphasizes incremental development, breaking down the project into manageable parts. This allows for quicker delivery of functional components and facilitates early user feedback.

Eg: Practicing an instrument



1. **V-models (Verification and Validation Models) in SDLC**

The V-Models, also known as the Verification and Validation models, is an extension of the traditional Waterfall models. It introduces a parallel testing phase for each corresponding development stage, forming a V-shaped diagram.

Eg; control software for aircrafts



1. **Spiral SDLC Models**

The Spiral model combines the idea of iterative development with the systematic aspects of the Waterfall model. It is based on the concept of a spiral, with each loop representing a phase in the software development process. The model is inherently risk-driven, meaning that risks are continuously assessed and addressed throughout the development life cycle.

Eg: Game development



1. **Agile SDLC Models**

Agile is not a specific methodology but rather a set of principles and values outlined in the Agile principle. The Agile principle prioritizes individuals and interactions, working solutions, customer collaboration, and responding to change over rigid processes and documentation. Several Agile methodologies, including Scrum, Kanban, and Extreme Programming (XP), have been developed to implement these principles.

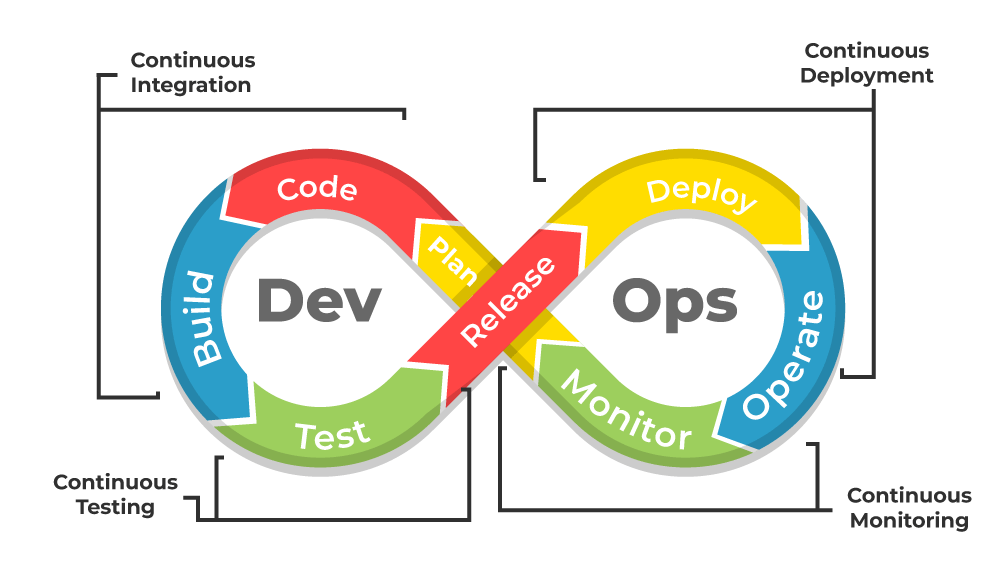
Eg: Mobile app development



1. **DevOps SDLC Models**

DevOps, comprised of "development" and "operations," represents a cultural and organizational shift in how software is developed, tested, and deployed. It emphasizes collaboration and communication between software developers and IT operations, promoting automation and continuous delivery. DevOps is not just a set of practices; it is a cultural mindset that seeks to improve collaboration and efficiency across the entire software development lifecycle.

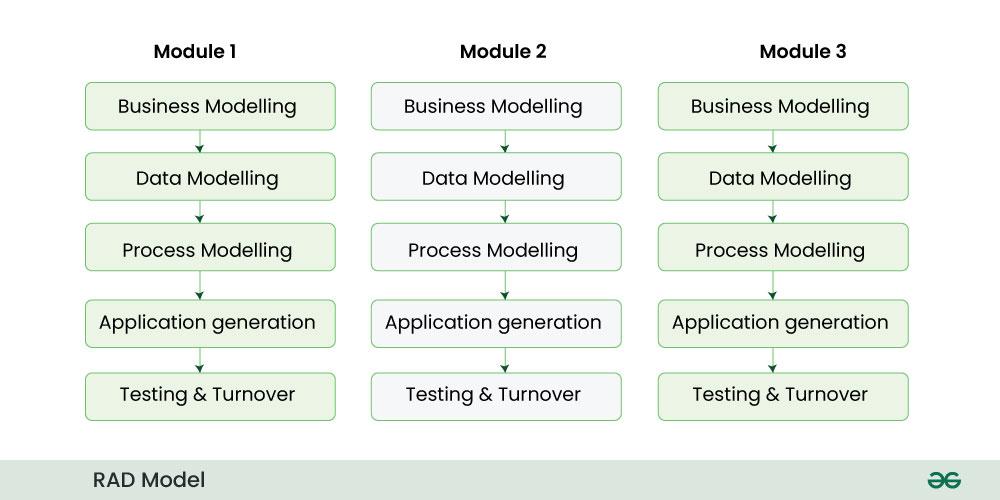
Eg: Application in online financial trading



1. **Rapid Application Development SDLC Models**

Rapid Application Development is an iterative and incremental model that prioritizes quick development and iteration cycles. It places a strong emphasis on user feedback and involvement throughout the development process. RAD aims to deliver functional prototypes rapidly, allowing stakeholders to provide feedback and guide ongoing development.

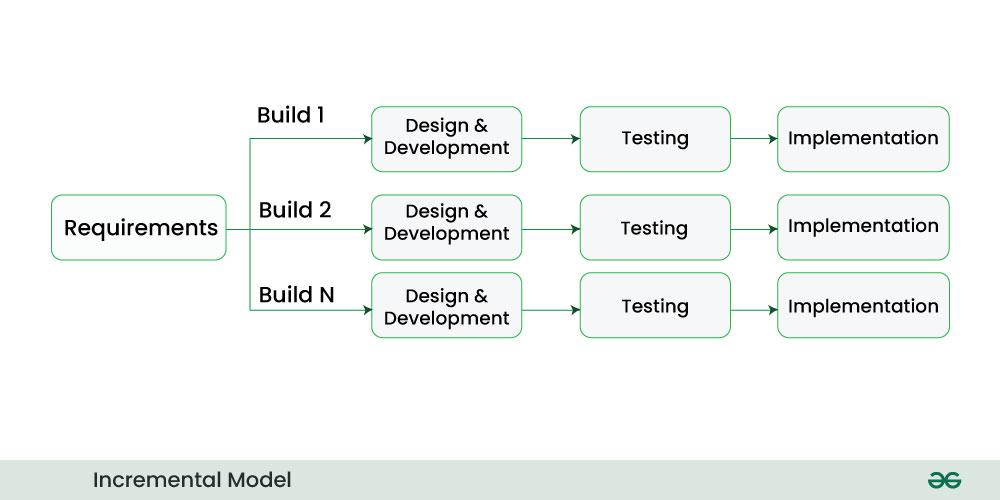
Eg: Human resource management system.



1. **Incremental SDLC Models**

The Incremental model is an iterative software development process where the product is designed, implemented, and tested incrementally until the product is finished. Each iteration represents a small part of the overall system and includes both new features and enhancements to existing ones.

Eg: Constructing a car.



**Task 5:**

**Applications:**

SDLC is used in data science , in big tech companies like Google, Amazon, In healthcare, in finance industry, in gaming industry, in building IOT devices, and in cloud applications like AWS or google cloud.

**Task 6:**

**Advantages and Limitations of SDLC Models**

Advantages of Waterfall model:

1. Simplicity: Easy to understand and implement
2. Clear documentation: Has its own set of documentation, making it easier to track progress and manage the project.

Limitations of waterfall model:

1. Rigidity: The model is highly inflexible once a phase is completed, making it challenging to accommodate changes.
2. Late Testing: Testing is performed after the implementation phase, so defects might not be discovered until late in the process.

Advantages of Iterative SDLC Models:

1. Enhanced Flexibility: The ability to adapt to changing requirements makes the Iterative models suitable for projects with evolving needs, ensuring that the final product meets user expectations.
2. Improved Quality: Continuous evaluation and testing in each iteration contribute to higher software quality. Bugs and issues are identified and addressed early, preventing them from accumulating in later stages.

Limitations of Iterative SDLC Models:

1. Clear Project Scope: Requires a clear and well defined project scope.
2. Effective communication: Open and transparent communication is very crucial.

Advantages of the V-Models:

1. Clear Design and Planning: The V-Models's structured framework facilitates clear design and planning. Well-defined tasks and deliverables at each stage contribute to effective project management.
2. Early Issue Identification: Incorporating testing early in the development process allows for the timely identification and resolution of issues. This proactive stance minimizes the likelihood of significant defects surfacing later in the project.

Limitations of the V-Models:

1. Thorough Requirements Analysis: A detailed and well-defined set of requirements is imperative for the success of the V-Models. Thorough requirements analysis ensures alignment with project goals throughout subsequent development and testing activities.
2. Effective Communication: Clear and consistent communication between development and testing teams is crucial. Regular meetings, status updates, and collaboration tools foster synchronization and enable prompt issue resolution.

Advantages of the Spiral SDLC Models:

1. Risk Mitigation: The focus on risk analysis and management allows for early identification and mitigation of potential issues, reducing the likelihood of project failure.
2. Flexibility in Requirements: Changes to requirements can be accommodated at any stage of the development process. The iterative nature of the Spiral models facilitates flexibility and adaptation.

Limitations of the Spiral SDLC Models:

1. Thorough Risk Assessment: Conduct a comprehensive risk assessment at the beginning of each iteration. Prioritize and address high-risk elements to minimize potential challenges.
2. Regular Review Meetings: Hold regular review meetings after each iteration to assess progress, evaluate the product, and plan the next steps. This continuous feedback loop is crucial for success.

Advantages of the Agile SDLC Models:

1. Early and Predictable Delivery: Agile's iterative cycles result in regular and predictable product deliveries. This allows stakeholders to see tangible progress at the end of each iteration.
2. Customer Satisfaction: Continuous customer involvement ensures that the delivered product aligns closely with customer expectations. This customer-centric approach enhances satisfaction and reduces the risk of delivering a product that does not meet user needs.

Limitations of the Agile SDLC Models:

1. Prioritization and Planning: Prioritize features and tasks based on customer value. Regular planning sessions, such as Sprint Planning in Scrum, help the team focus on high-priority items.
2. Continuous Integration and Testing: Implement continuous integration practices to ensure that code changes are integrated and tested frequently. This minimizes integration issues and helps maintain a stable codebase.

Advantages of the DevOps SDLC Models:

1. Increased Collaboration: DevOps breaks down traditional barriers between development and operations, fostering a culture of collaboration. Shared goals and responsibilities lead to improved communication and efficiency.
2. Improved Reliability: Automated testing and deployment processes enhance the reliability of software releases. DevOps practices contribute to the detection and resolution of issues early in the development lifecycle.

Limitations of the DevOps SDLC Models:

1. Cultural Transformation: DevOps is not just about tools; it requires a cultural shift. Encourage a collaborative and transparent culture where teams work together to achieve shared objectives.
2. Automation Tools: Invest in and leverage automation tools for various aspects of the development and operations lifecycle, including continuous integration, testing, deployment, and monitoring.

Advantages of Rapid Application Development (RAD) SDLC Models:

1. Speed and Time-to-Market: RAD's iterative nature and emphasis on quick prototyping contribute to faster development cycles, reducing time-to-market for applications.
2. User Satisfaction: Continuous user involvement and feedback ensure that the application meets user expectations. This user-centric approach enhances user satisfaction and adoption.

Limitations of Rapid Application Development (RAD) SDLC Models:

1. Active User Involvement: Actively involve users and stakeholders throughout the development process. Regular feedback sessions and usability testing are essential for shaping the application according to user needs.
2. Effective Communication: Maintain open and transparent communication among the development team, users, and stakeholders. This ensures that everyone is aligned on project goals and progress.

Advantages of Incremental SDLC model:

1. Integration of Increments: Increments are integrated with the existing system or increments from previous iterations. This integration ensures that the complete system evolves gradually with each increment.
2. Parallel Development: Different teams or development groups can work on different increments simultaneously. This parallel development approach contributes to faster development cycles.

Limitations of Incremental SDLC model:

1. Thorough Testing at Each Increment: Rigorous testing should be conducted at each increment to ensure that the integrated system functions correctly. This includes testing the new features and ensuring compatibility with existing ones.

**Task 7:**

Scrum is a popular framework for implementing agile project management, providing a structured approach to building software and other complex products. Agile is a broader philosophy and methodology, while Scrum is a specific framework used within Agile. Scrum helps teams manage their work in short, iterative cycles called sprints, focusing on delivering value incrementally.

**Task 8:**

The Sprint is the Scrum event that encompasses all of the other Scrum events.They are fixed length periods of work that last one month or less to create consistency and ensure short iterations for feedback in order to inspect and adapt both how work is done and what is being worked on. If cycles are longer, then the spirit of frequent feedback cycles can be lost. Longer Sprint may also get too complex and may increase risk. A new Sprint starts immediately after the conclusion of the previous Sprint.

**Task 9:**

**Dos:**

Clear Sprint Goal: Ensure the team understands the sprint's objective and why it's important.

Well-Refined Backlog: Prioritize and refine backlog items to facilitate efficient sprint planning.

Teamwork and Collaboration: Encourage self-organization and shared responsibility within the team.

**Don'ts:**

Over-Commitment: Don't overload the sprint with too many stories or tasks.

Micromanagement: Respect the team's self-organization and avoid assigning tasks directly.

Neglecting Feedback: Actively seek and incorporate feedback from stakeholders and the team.

**Task 10:**

**Stories:**

In Scrum, "stories" refer to user stories, which are the building blocks of Agile software development. These stories represent the smallest unit of work a team can commit to completing within a sprint and are focused on delivering value to the end-user.

**Backlogs:**

A backlog is a changing list of requirements based on the customer's needs. The backlog is not a to-do list; rather, it is a wish list of all the desired features for the product. The scrum team uses the backlog to prioritize features and understand which features to implement first. In the Scrum framework, backlogs are crucial for managing and prioritizing work. There are two primary types: the Product Backlog and the Sprint Backlog. The Product Backlog is a prioritized list of all desired features and requirements for the product, while the Sprint Backlog is a subset of the Product Backlog that contains the items the Scrum Team will work on during a specific Sprint

**Task 11:**

In Scrum, artifacts are key pieces of information used by the team and stakeholders to track progress, manage work, and ensure transparency throughout the development process

**Product backlog:**

A product backlog is a prioritized list of work for the development team that is derived from the product roadmap and its requirements. The most important items are shown at the top of the product backlog so the team knows what to deliver first.

**Sprint backlog:**

A sprint backlog is a subset of the product backlog and lists the work items to complete in one specific sprint. The purpose of the sprint backlog is to identify items from the product backlog that the team will work on during the sprint. This occurs during the sprint planning process

**Burn down chart:**

A burn down chart is a visual tool that helps teams track their progress on a project by showing the remaining work over time. It's a key part of Agile project management, particularly Scrum, and is used to monitor sprint progress and predict completion.

**Increment:**

In Scrum, an Increment is a valuable, usable, and concrete stepping stone towards the Product Goal. It's the sum of all completed Product Backlog items during a Sprint, along with the value of all previous Increments. Essentially, it's the progress made on a product or project during a Sprint, demonstrating tangible value and usability to stakeholders.

**Task 12:**

**Ports and protocoi:**

In networking, a port is a virtual communication channel, like a door, within a computer that's used to identify a specific process or service. A protocol is a set of rules that governs how data is transmitted and received over a network. Ports and protocols work together to enable different applications and services to communicate across a network without interfering with each other. Ports act as virtual endpoints, like numbered addresses, that applications use to identify the specific service or application they need to reach on a computer. Protocols, on the other hand, define the rules and standards for how data is formatted, transmitted, and interpreted between communicating devices

**Task 13:**

There are several types of networks, including Personal Area Networks (PANs), Local Area Networks (LANs), Wireless Local Area Networks (WLANs), Metropolitan Area Networks (MANs), Wide Area Networks (WANs), and Virtual Private Networks (VPNs). These networks are classified by their geographical scope and intended use.

**Task 14:**

Servers can be broadly categorized by their function and how they are physically implemented. Functionally, common types include web servers, database servers, mail servers, file servers, and application servers. Physically, servers are available as rackmount, blade, and tower

**By Function:**

* **Web Servers:** These servers host websites and web applications, delivering web pages to users' browsers.
* **Database Servers:** Dedicated to storing and managing data, often using database management systems (DBMS).
* **Mail Servers:** Handle the sending, receiving, and storing of email messages.
* **File Servers:** Provide centralized storage and sharing of files across a network.
* **Application Servers:** Host and manage the execution of software applications, often acting as a bridge between users and back-end systems.
* **Proxy Servers:** Act as intermediaries between clients and other servers, providing additional services like caching and security filtering.
* **DNS Servers:** Translate domain names (like google.com) into IP addresses.
* **Print Servers:** Manage printing tasks and connect printers to a network.
* **FTP Servers:** Enable file transfers between computers using the File Transfer Protocol.
* **Game Servers:** Host and manage online multiplayer gaming sessions.
* **Virtual Servers:** Created by virtualizing hardware resources on a physical server, allowing multiple operating systems and applications to run concurrently.

By Physical Form Factor:

* **Rackmount Servers:**

Designed to be mounted in standard server racks, offering space-saving and high-density deployments.

* **Blade Servers:**

Thin, modular servers that fit into a larger chassis, often used in data centers for high-density computing.

* **Tower Servers:**

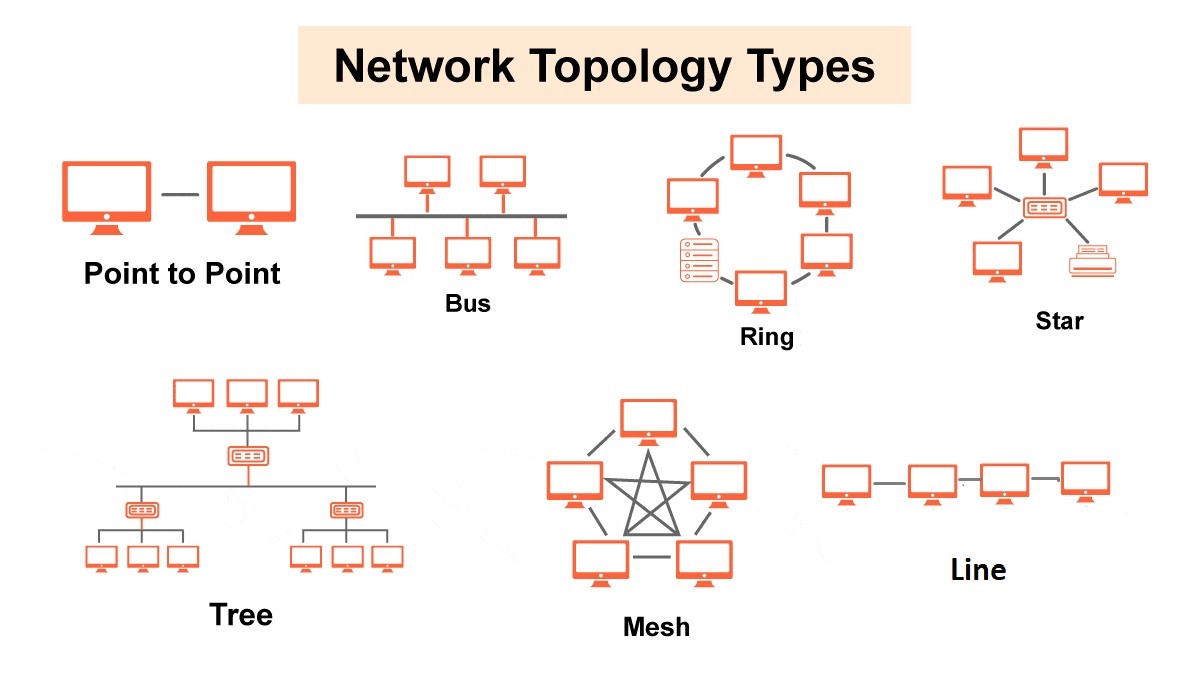
Stand-alone servers, typically resembling desktop computers, suitable for smaller environments.

**Task 15:**

The Domain Name System (DNS) is the internet's phonebook, translating human-friendly domain names (like "www.example.com") into the numerical IP addresses (like "192.168.1.1") that computers use to locate each other. Without DNS, you'd have to remember and type IP addresses to access websites, which would be a lot harder.

**Task 16:**

Network topologies define how devices are interconnected within a network, impacting data flow and performance. Common types include bus, star, ring, mesh, tree, and hybrid topologies.



Here's a breakdown of each:

1. Bus Topology:

* All devices connect to a single, shared cable (bus).
* Simple and inexpensive, but can be slow and prone to failure if the bus breaks.
* Examples include older Ethernet networks.

2. Star Topology:

* Devices connect to a central hub or switch.
* Reliable and easy to troubleshoot, but can be a bottleneck if the hub fails.
* Common in home and small office networks.

3. Ring Topology:

* Devices connect in a closed loop (ring).
* Data travels in one direction around the ring.
* Less common now, but used in older networks like FDDI.

4. Mesh Topology:

* Each device connects directly to many other devices.
* Redundant connections provide high reliability and fault tolerance.
* Examples include some wireless networks.

5. Tree Topology:

* Organized hierarchically, like a tree.
* A central root node connects to branches, which connect to further branches.
* Can be used to create a hierarchical network structure.

6. Hybrid Topology:

* Combines two or more different topologies.
* Can leverage the strengths of different topologies, such as combining star and bus for a more flexible network.
* Common in larger networks.

7. Point-to-Point Topology:

* A direct connection between two devices and Simple and efficient for dedicated connections.

8. Daisy Chain Topology:

* Similar to a bus, but devices are connected sequentially in a chain.
* Simple and cost-effective for small networks.

**Task 17:**

The OSI (Open Systems Interconnection) model is a conceptual framework that standardizes how computer systems communicate over a network. It consists of 7 layers:

1. **Physical Layer (Layer 1)**

* Deals with physical transmission of raw data bits
* Includes hardware specifications, voltage levels, cables, hubs
* Examples: Ethernet cables, fiber optics, wireless signals

1. **Data Link Layer (Layer 2)**

* Handles node-to-node data delivery
* Error detection and correction
* MAC addressing
* Examples: Ethernet switches, Network Interface Cards (NICs)

1. **Network Layer (Layer 3)**

* Manages logical addressing and routing between networks
* IP addressing and packet forwarding
* Examples: Routers, IP protocol

1. **Transport Layer (Layer 4)**

* Ensures end-to-end data delivery
* Flow control and error recovery
* Examples: TCP, UDP protocols

1. **Session Layer (Layer 5)**

* Establishes, maintains, and terminates connections
* Manages dialogue control and synchronization
* Examples: NetBIOS, RPC

1. **Presentation Layer (Layer 6)**

* Data translation and encryption
* Data formatting, compression, and encryption
* Examples: SSL/TLS, JPEG, ASCII

1. **Application Layer (Layer 7)**

* User interface and network services
* Direct interaction with applications
* Examples: HTTP, FTP, SMTP, DNS