**Task 1 –**

Q: What do you understand by SDLC?

Ans: SDLC is a structured process used for developing software system efficiently. IT outlines the steps involved the planning, creating, testing, & deploying software.

**Task 2 –**

Q: Why only SDLC?

A: SDLC helps deliver reliable, efficient, & maintainable software on time within budget.

* It ensure high-quality software
* Reduce Risks & errors
* Control project scope & budget
* Improve communication between tester & developer
* Ensure Maintainablity

**Task 3-**

Q: What are the stages of SDLC ? write 2 lines about each.

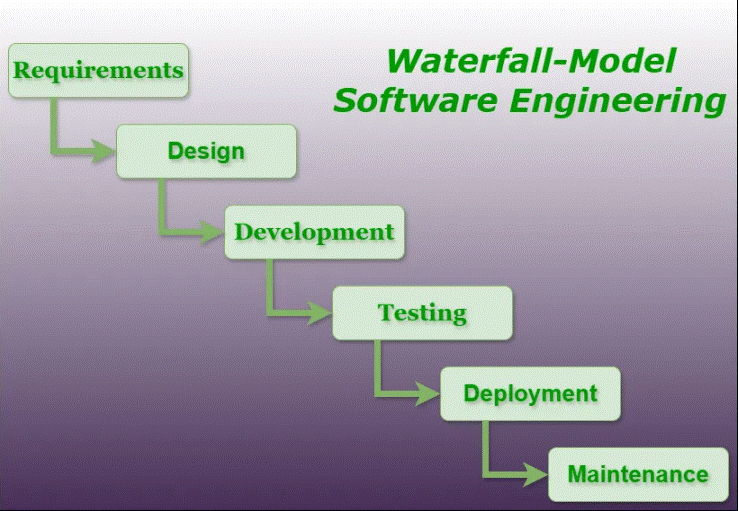
1. Requirement & Analysis ( collect the basic need & required specification document)
2. Planning ( contains scope, resources, timeline & budget & risk analysis)
3. Design ( create architecture & detailed design for system: output design)
4. Development & Implementation ( developers write codes based on doc design using programming language & frameworks)
5. Testing ( test the s/w for bugs, errors includes unit, integration user testing)
6. Deployment ( s/w is released for use)
7. Maintenance ( Ongoing support to fix bugs, updates & improve performance bases on user feedback)

**Task 4-**

Q: SDLC Models: List them , description - 4 lines min and with a image

Ans: A System Development Life Cycle (SDLC) is a framework for software development, which includes the following types:

* **Waterfall Model:** The linear sequential model should be used for projects with well-defined requirements and a limited scope. It is typically used for projects with well-defined requirements.



**Application**:

* Ideal for **small-scale projects** with well-defined and fixed requirements.
* Used in **construction, defense, and manufacturing industries** where phases must be clearly documented.
* Suitable for **regulatory-driven projects** like healthcare systems, where documentation is critical.

**Advantages**:

* Simple to understand and use.
* Easy to manage due to its rigid structure.
* Clearly defined stages and deliverables.
* Works well for smaller projects with clear requirements.

**Disadvantages**:

* No flexibility for changes once the project starts.
* Difficult to go back to any stage once it is completed.
* Poor model for long or ongoing projects.
* Testing is late in the cycle, leading to late detection of issues.
* **Agile Model**: Iterative models are best suited to projects with rapidly changing requirements and high levels of uncertainty. This model allows for flexibility and constant adaptation to changing requirements.



**Application**:

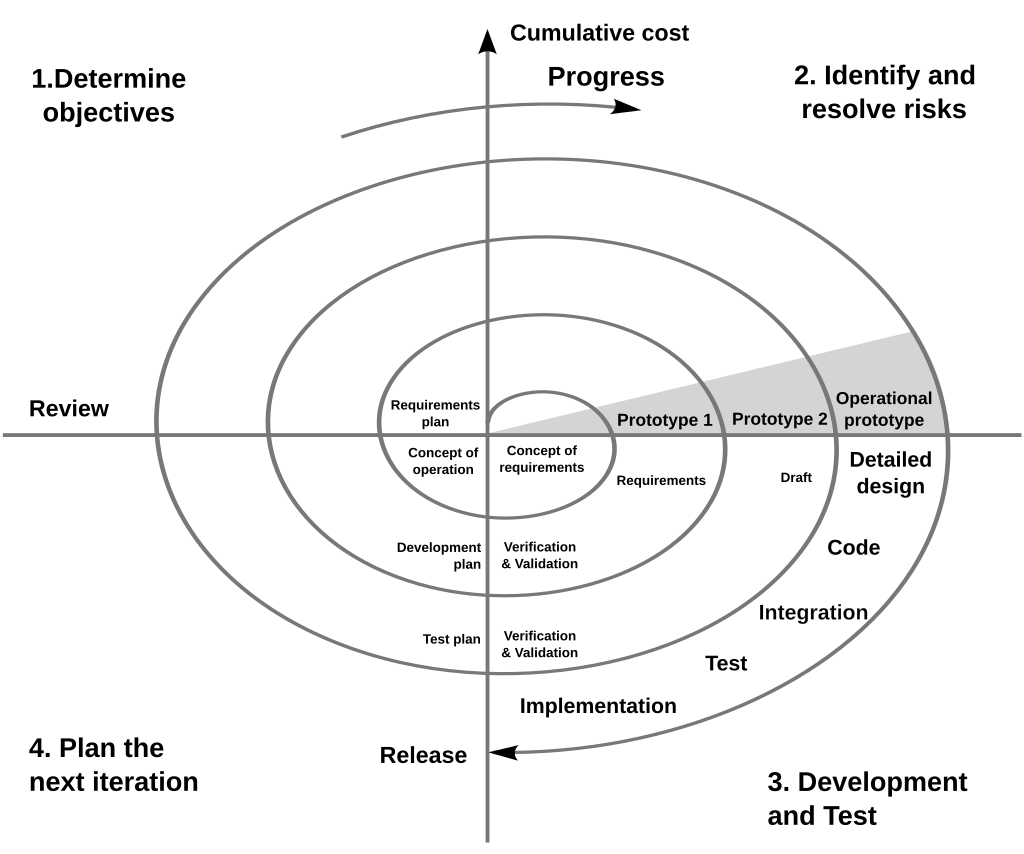
* Best for **dynamic and evolving projects**, especially in **startups** and **modern web/app development**.
* Used in **software product development** where user feedback and iterative releases are vital (e.g., SaaS platforms).
* Preferred in **DevOps** and **scrum-based team environments**.

**Advantages**:

* Flexible to changing requirements.
* Promotes collaboration and user feedback.
* Faster delivery through iterative sprints.
* High product quality due to continuous testing and integration.

**Disadvantages**:

* Requires a high degree of customer involvement.
* Difficult to predict time and cost in advance.
* Not suitable for small teams without experienced developers.
* Can lead to scope creep if not well managed.
* **Spiral Model:** Waterfall and Agile are combined in this process, where development takes place in multiple iterations. Each iteration involves risk analysis, design, and implementation.



**Application**:

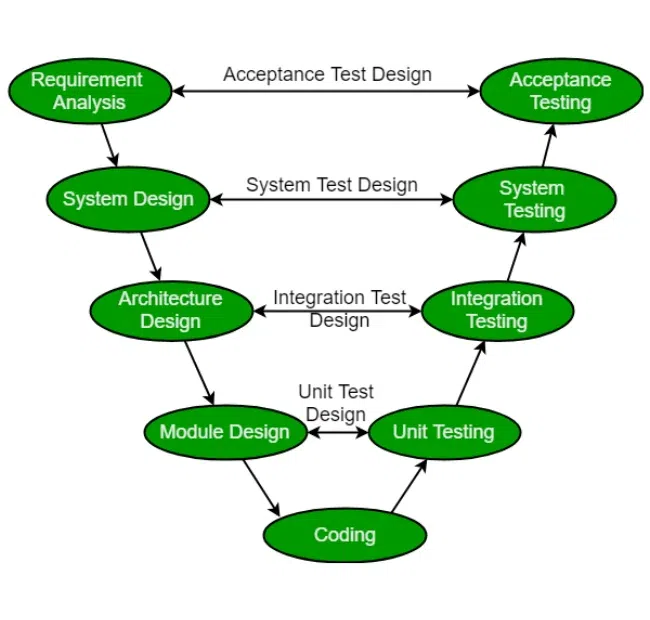
* Applied in **high-risk projects** where requirements are complex and frequently changing.
* Often used in **aerospace, defense**, and **financial systems** where risk analysis is critical.
* Suitable for **R&D projects** where multiple refinements are necessary before final deployment.

**Advantages**:

* Strong focus on risk analysis and reduction.
* Ideal for large, complex, and high-risk projects.
* Allows for changes to be implemented at later stages.
* Combines strengths of both iterative and waterfall models.

**Disadvantages**:

* Can be expensive and time-consuming.
* Requires expertise in risk assessment.
* Complex to manage; not suitable for small or low-risk projects.
* **V-Model:** Each phase of the development process is represented by a V shape in this graphical representation of the Waterfall Model. An extension of the waterfall model that emphasizes verification and validation. Each development phase has a corresponding testing phase directly associated with it.



**Application**:

* Used in **embedded systems** development like automotive or medical device software.
* Ideal for **mission-critical applications** (e.g., avionics, nuclear control software) where testing at every stage is essential.
* Suitable for **projects with strict regulatory compliance** needs.

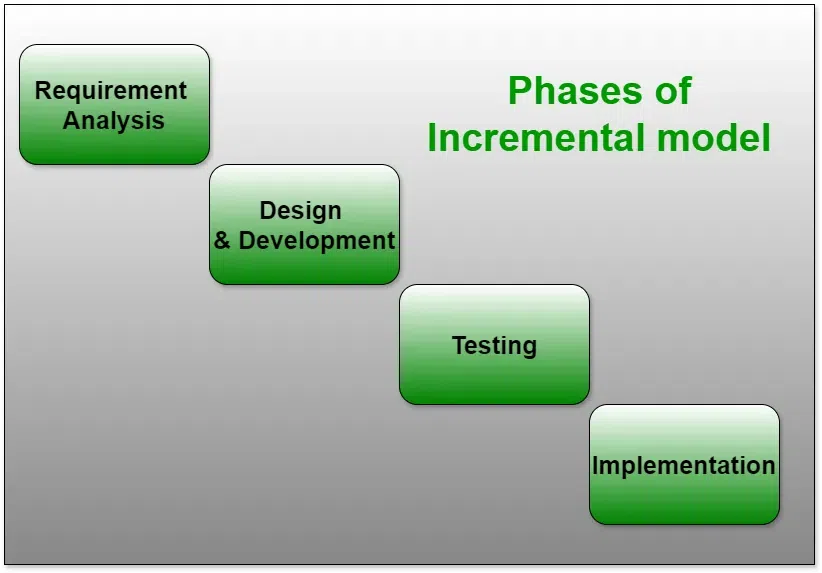
**Advantages**:

* Simple and easy to use.
* Each phase has a corresponding test phase, ensuring quality.
* Works well for smaller projects with well-understood requirements.
* Emphasizes verification and validation early.

**Disadvantages**:

* Very rigid, similar to Waterfall.
* No flexibility for changing requirements.
* High risk and uncertainty.
* Not suitable for complex or object-oriented projects.
* **Incremental Model:** A software development model is iterative, where each increment builds upon the previous one.

Iterative software development, referred to as the incremental model, entails developing software in small, manageable increments over time.



**Application**:

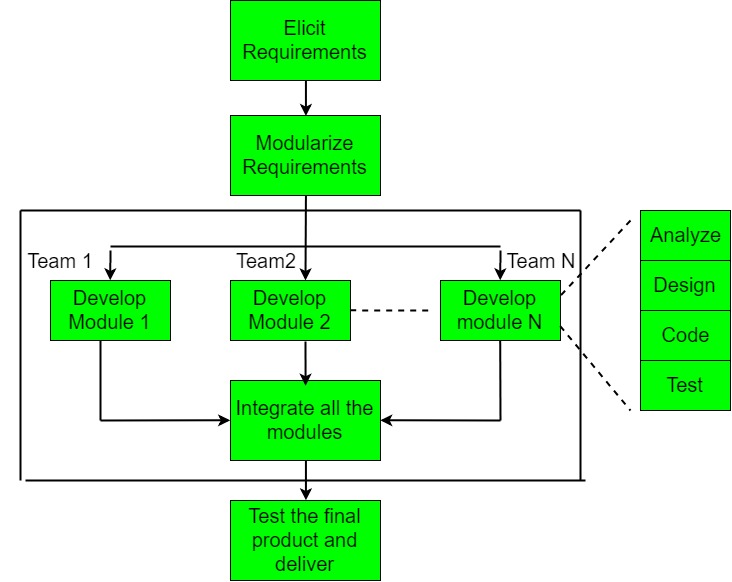
* Good for **projects with clear long-term goals** but where a product must be delivered in phases.
* Commonly applied in **banking and financial services**, where features are rolled out gradually.
* Used when **functionality can be built and released in stages**.

**Advantages**:

* Produces working software early.
* Easier to test and debug smaller iterations.
* Allows partial implementation of functionality.
* Easier to manage risk through phased delivery.

**Disadvantages**:

* Requires clear and complete planning.
* Integration can become difficult as modules are added.
* Total cost may exceed Waterfall due to repeated cycles.
* **Rapid Application Development (RAD) Model:** Rapid prototyping and iterative development are emphasized in this model for developing software quickly.As the RAD Model allows for rapid and easy software changes based on feedback from stakeholders and end users, it is best suited for projects with fast-paced and rapidly changing requirements.



**Advantages**:

* Significantly reduces development time.
* Encourages customer feedback through prototypes.
* Flexible to changing requirements.
* Emphasizes user involvement and quick iterations.

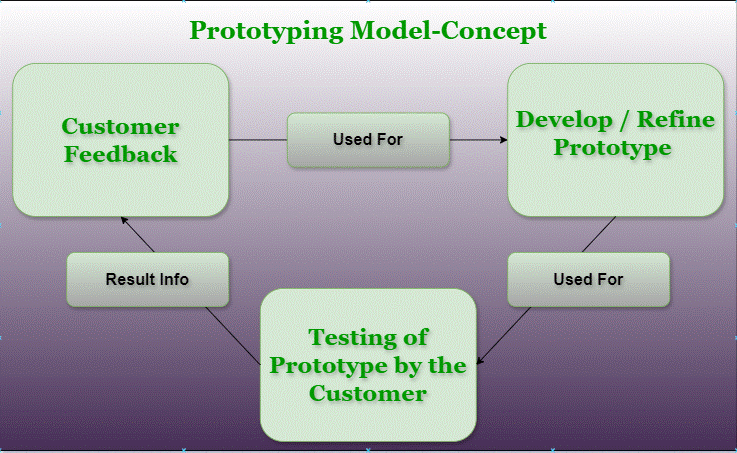
**Disadvantages**:

* Requires highly skilled developers and designers.
* Not suitable for large-scale or complex projects.
* Poor documentation due to speed.
* Depends heavily on strong team collaboration.

**Application**:

* Suitable for **projects with tight deadlines** and **high user interaction**, such as UI-heavy apps.
* Common in **prototyping** for mobile or desktop applications.
* Used in **startups** or marketing tools where **speed to market** is critical.

**Prototype Model:** In this model, a working prototype of the software is quickly developed, and then refined and updated based on feedback from users. A project’s specific needs determine which SDLC model to use, since each model has its own strengths and weaknesses. In order to create a final product, this prototype is refined and improved based on the feedback of users.As a result, SDLC’s prototype model allows for rapid prototyping and iterative development that is cost-effective and efficient.



**Application**:

* Helpful in **requirement-gathering** phases of complex or innovative systems.
* Often used in **software consulting**, **product design**, and **client-driven projects**.
* Suitable for **user interface-heavy applications** (e.g., eCommerce websites, dashboards).

**Advantages**:

* Helps clarify requirements through early visualization.
* Enhances user involvement and satisfaction.
* Reduces risk of failure through early feedback.
* Flexible to requirement changes.

**Disadvantages**:

* Can lead to misunderstanding of prototype as the final system.
* Not suitable for large systems with complex logic.
* Frequent changes can delay the final system.
* Requires extra time and effort to build prototypes.

**Task 7-**

Q- What is Scrum in Agile?

Ans: **Scrum** is an **Agile framework** used for developing, delivering, and sustaining complex products, especially software. It focuses on **iterative progress**, **collaboration**, and **continuous improvement**.

it’s a way to **work together as a team**, **divide work into small parts**, and **deliver something useful in each step**.

**Task 8-**

Q- What is Sprint?

Ans: A **Sprint** is a short, time-boxed period where a **software development team works to complete a specific set of features, bug fixes, or improvements**. It's a core concept in the **Scrum framework**, part of Agile methodology.

During which a **team works together to complete specific tasks** and deliver a **small, working part of a bigger project**.

**Task 9-**

Q- What are the do’s and don’ts of Sprint?

### Ans: ****Do’s of a Sprint:****

1. **Define a Clear Sprint Goal**  
   → Make sure everyone knows **what needs to be achieved** during the Sprint — it keeps the team focused and aligned.
2. **Hold Daily Stand-ups (Daily Scrum)**  
   → These short meetings help the team stay on track, identify blockers early, and adjust quickly if needed.
3. **Deliver a Working Product Increment**  
   → Always aim to complete something **functional** that can be shown, tested, or released — even if it's a small part.

### ****Don’ts of a Sprint:****

1. **Don’t Add New Work Mid-Sprint**  
   → Once a Sprint starts, don’t change the scope unless absolutely necessary — it disrupts focus and planning.
2. **Don’t Ignore Team Input**  
   → Everyone’s voice matters — skipping retrospectives or daily scrums reduces collaboration and learning.
3. **Don’t Overcommit**  
   → Taking on too much work leads to unfinished tasks and low morale. Be realistic during Sprint Planning.

**Task 10:**

Q: What are stories and Backlogs in Scrum World?

Ans: In **Scrum**, a **Backlog** is a prioritized list of everything that might be needed in the product. It's essentially the **to-do list** for the Scrum Team — constantly updated, refined, and prioritized by the **Product Owner**.

In **Scrum**, a **story** refers to a **User Story** — a short, simple description of a feature or functionality written from the **end-user’s perspective**.

It helps the team understand **what the user needs**, **why they need it**, and **what value it provides**.

**Task 11:**

Product Backlog-

 The **master list** of all the features, changes, bug fixes, enhancements, and tasks for a product.

 Managed by the **Product Owner**.

 Items are usually written as **User Stories**.

 It is **dynamic** — it grows and changes as the product evolves or new requirements come up.

Sprint Backlog-

 A **subset** of the Product Backlog.

 Contains only the items the team has **committed to complete during a Sprint** (usually 1–4 weeks).

 Created during **Sprint Planning**.

 Managed by the **development team**.

Burn-Down Chart-

A **Burn-Down Chart** is a simple graph used in **Scrum** to track how much work remains in a **Sprint or project**, and whether the team is on track to complete it **on time**.

**Increment** in Scrum-

An **Increment** in Scrum is the **usable, working piece of the product** that is **completed at the end of each Sprint**. It is the **sum of all the work** finished so far and must be in a **releasable** state — meaning it could be delivered to users right away, even if it’s not the full product yet.

An **Increment** is what the team **delivers at the end of a Sprint** — a small, but working, addition to the product.

**Task 12-**

Q: What Are **Ports and Protocols**?

A: In the world of **computer networking**, **ports** and **protocols** help devices like computers and smartphones **communicate with each other** over the internet or a local network.

Think of a **port** like a **door** into a computer.

* Each device (like a computer or server) has **many ports**.
* Each port is like a **channel** that handles a specific type of communication.
* Ports are identified by **numbers** (ranging from 0 to 65535).

A **protocol** is a **set of rules** that defines **how data is formatted and transmitted** between devices.

* Think of it like a **language** both computers agree to use when talking.
* Without protocols, devices wouldn’t understand each other.

**Task 13-**

Q: What are the different Network types

A

* 1. **PAN – Personal Area Network**
* 2. **LAN – Local Area Network**
* 3. **MAN – Metropolitan Area Network**
* 4. **WAN – Wide Area Network**
* 5. **WLAN – Wireless LAN**
* 6. **SAN – Storage Area Network**
* 7. **VPN – Virtual Private Network**

**Task 14-**

Q: What are the types of servers ?

A: A **server** is a powerful computer or system that provides **resources, services, or data** to other computers (called **clients**) over a network.

1. **Web Server**  
   Hosts and delivers websites to users through browsers using HTTP/HTTPS protocols.  
   Example: Apache, Nginx.
2. **Mail Server**  
   Handles sending, receiving, and storing emails over the internet.  
   Works with protocols like SMTP, IMAP, and POP3.
3. **File Server**  
   Stores and manages files so users can access, share, or back up data on a network.  
   Common in office environments.
4. **Database Server**  
   Stores and retrieves structured data for applications or websites.  
   Examples: MySQL, Oracle, SQL Server.
5. **Application Server**  
   Runs business-specific applications and processes requests from clients.  
   Connects users to backend databases or systems.
6. **DNS Server**  
   Converts domain names (like www.example.com) into IP addresses.  
   Essential for locating websites on the internet.
7. **Authentication Server**  
   Verifies usernames and passwords, managing secure access to systems.  
   Example: Active Directory.
8. **Game Server**  
   Hosts multiplayer online games, managing game data and player interactions.  
   Used in both LAN and online gaming.
9. **Cloud Server**  
   A virtual server hosted in the cloud, used for various services like storage or hosting.  
   Examples: AWS, Google Cloud, Microsoft Azure.
10. **Proxy Server**  
    Acts as a middleman between a client and another server.  
    Helps improve security, privacy, and performance.

**Task 15-**

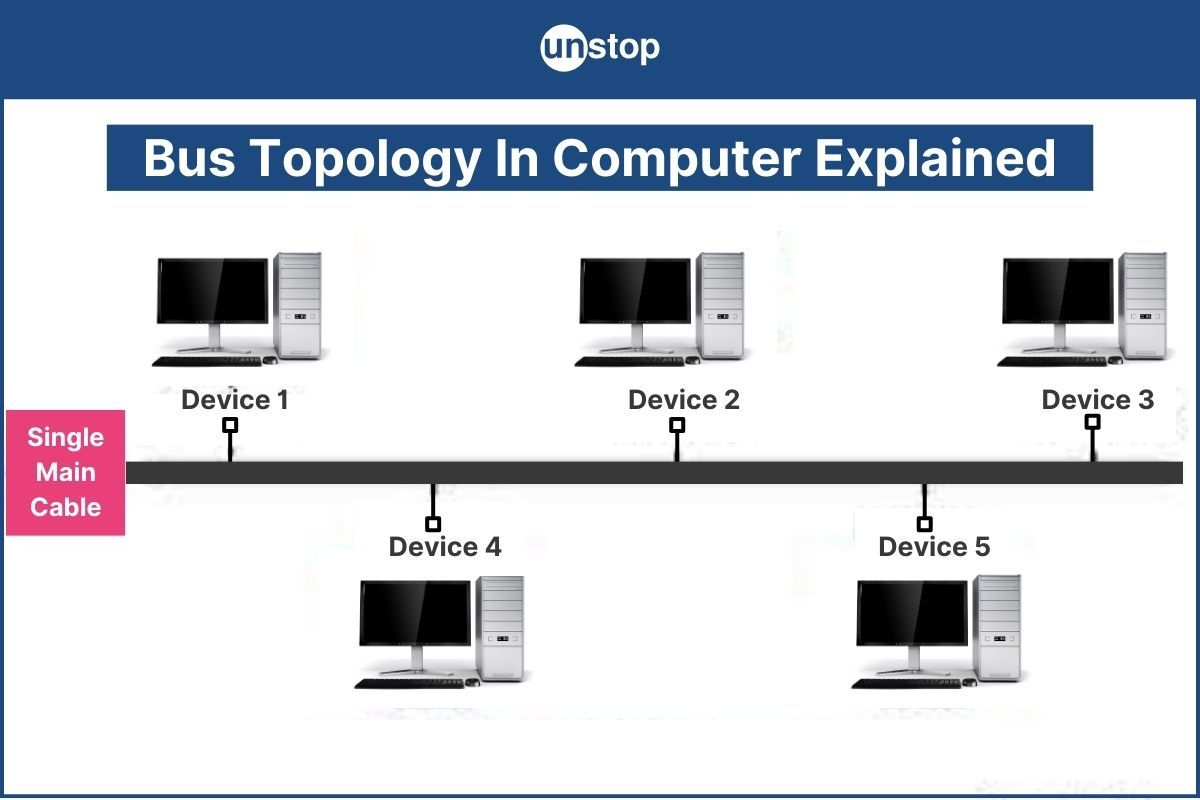
Q: What do you know about DNS? Domain Name Service

A: The **Domain Name System (DNS)** is often called the **“phonebook of the internet.”** It translates **domain names** (like www.google.com) into **IP addresses** (like 142.250.190.68) so that computers can find and communicate with each other.

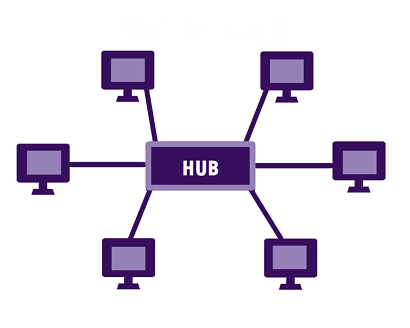
**Task 16-**

Q: What are the different Network Topologies?

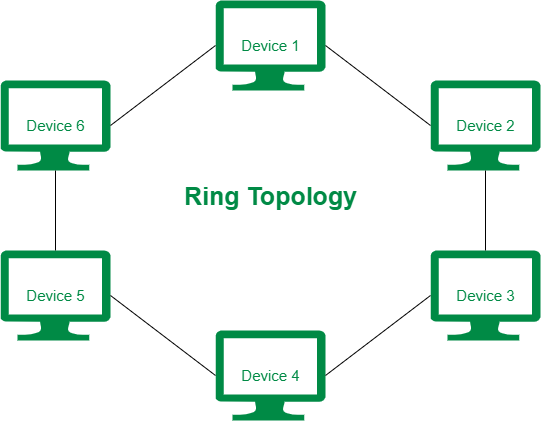
1. A: **Bus Topology**  
   All devices are connected to a **single central cable** (the bus).   
   Simple and inexpensive but difficult to troubleshoot.  
   A break in the main cable can bring down the entire network.



1. **Star Topology**  
   All devices are connected to a **central hub or switch**.  
   Easy to manage and expand, but the central device is a single point of failure.  
   Commonly used in modern LANs.

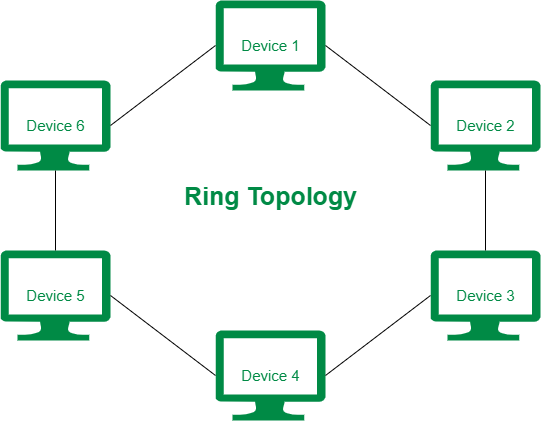


1. **Ring Topology**  
   Devices are connected in a **circular loop**, and data travels in one direction.  
   Each device has exactly two neighbors.  
   Failure in one device can disrupt the whole network unless a dual ring is used.



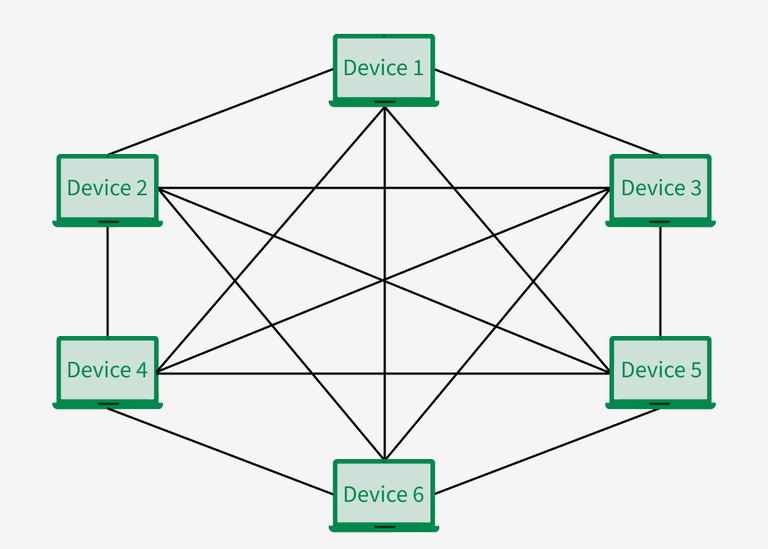
### ****4 Tree Topology****

* A **hierarchical** structure that combines characteristics of **Star** and **Bus** topologies.
* Devices are grouped into **branches** connected to a central backbone.
* Easy to manage but can get complex as the network grows.



### ****5 Mesh Topology****

* Every device is **connected to every other device** in the network.
* Provides **high redundancy and reliability**, since data can take multiple paths.
* Expensive and complex to set up, especially in large networks.



**Task 17-**

Q: What is OSI Model ? Describe the 7 layers with description.

### A: ****What is the OSI Model?****

The **OSI (Open Systems Interconnection) Model** is a framework designed to standardize and guide how network protocols interact across different systems. It is organized into **seven layers**, each with a unique function that helps in the process of communication from one device to another. By dividing this communication process into layers, the OSI Model makes it easier to understand and troubleshoot networking issues.

### ****The Seven Layers of the OSI Model****

The seven layers of the OSI Model work together to ensure smooth communication and data transfer over a network. Here's a breakdown of each layer from top to bottom:

### ****1. Physical Layer****

The **Physical Layer** deals with the **physical connection** between devices, transmitting raw bits over the network. It manages how data is sent through cables, wireless channels, or any other medium.

* **Primary Role**: Transmission of electrical signals, light signals, or radio waves across the physical medium.
* **Devices**: Cables, network adapters, hubs, and switches.
* **Example**: Ethernet cables, fiber optics.

### ****2. Data Link Layer****

The **Data Link Layer** ensures that data is reliably delivered between two directly connected devices. It packages raw bits from the Physical Layer into **frames** and handles **error detection and correction**.

* **Primary Role**: Framing data, error checking, and addressing devices using MAC (Media Access Control) addresses.
* **Devices**: Network Interface Cards (NICs), switches.
* **Example**: Ethernet frames.

### ****3. Network Layer****

This layer is responsible for **routing data** across different networks. It uses **logical addresses** (IP addresses) to ensure data travels from its source to the destination, even if they are on different networks.

* **Primary Role**: Routing and forwarding of data packets, managing IP addresses.
* **Devices**: Routers.
* **Example**: IP addresses, routers determining the best path for data.

### ****4. Transport Layer****

The **Transport Layer** ensures that data is transferred correctly between two devices, overseeing **flow control** and **error correction**. It establishes a reliable connection for data transmission.

* **Primary Role**: Provides reliable communication with **flow control**, **error correction**, and segmentation.
* **Protocols**: TCP (Transmission Control Protocol), UDP (User Datagram Protocol).
* **Example**: Breaking large messages into smaller segments, ensuring all data arrives in order.

### ****5. Session Layer****

The **Session Layer** is responsible for establishing, managing, and terminating communication sessions between devices. It ensures that data from the two devices can communicate without interference.

* **Primary Role**: Session management, synchronization, and controlling the flow of data between devices.
* **Protocols**: NetBIOS, RPC (Remote Procedure Call).
* **Example**: Managing dialogue between two applications.

### ****6. Presentation Layer****

The **Presentation Layer** translates data into a format that both the sender and receiver can understand. It also handles data encryption, compression, and translation between different data formats.

* **Primary Role**: Data translation, encryption, compression.
* **Example**: Data conversion between different character encoding formats (ASCII to Unicode), SSL/TLS encryption for secure communication.

### ****7. Application Layer****

The **Application Layer** is the topmost layer that directly interacts with the end user. It enables network services and applications to access the network, allowing users to interact with the system.

* **Primary Role**: Provides networking services directly to applications, such as file transfer or web browsing.
* **Protocols**: HTTP (HyperText Transfer Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol).
* **Example**: Web browsers, email clients, file sharing apps.