**DAY 1 202073753 PRATHYUSH NANDAN(@mpnandan)**

**Task 1. What is SDLC?**

SDLC (software development Life cycle) is the complete process of teams plan, design, build, test, and maintain software, ensuring it meets requirements and expectations. It's a systematic approach that helps organizations create high-quality software efficiently.

Benefits: cost effective, time efficient reduces risk, collaboration.

Planning—design—development—testing—deployment—maintaince—quality maintain is the systematic procedure of sdlc.

**Task 2. Purpose of SDLC**

The SDLC is the cost effective and time efficient process that development team use to design and build best quality software. The main goal is to minimize project risks forward palnning software meets client requirement during the development.

**Task 3: what are the stages of SDLC:**

1.planning- 1st stage is scope of project scope, goals, and objectives – understanding the requirements of stake holders

2.Requirement – understanding the requirements of client

3.Design- preparing the flowchart and designing

4.Implementation- building the code with necessary tools

5.testing- testing the code to reduce the errors

6.deployment- releasing the software accessible to users

7.maintainance- monitoring the realtime issues and improvisations required

**Task 4. SDLC models:**

**1. Waterfall Model**:

Requirements Gathering: Defining what the system needs to do.

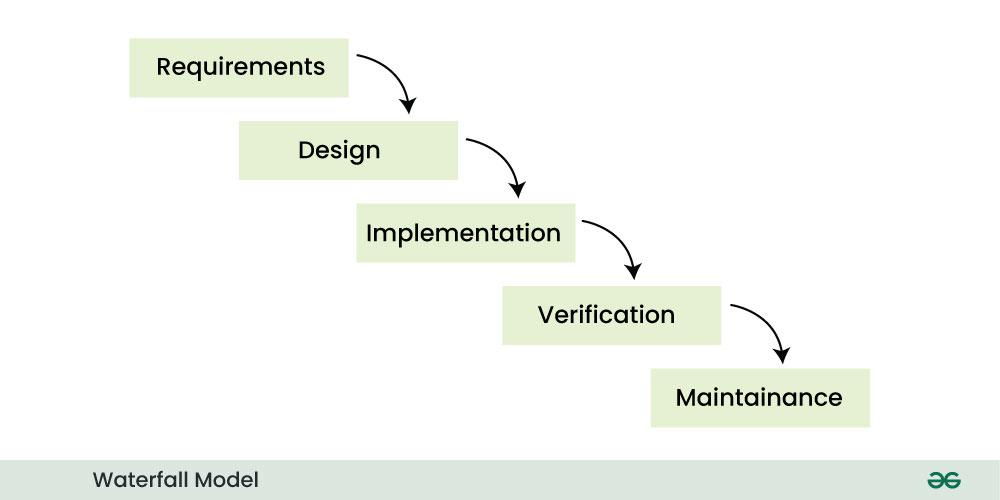
System Design: Planning the overall architecture and components of the system.

Implementation: Writing the actual code.

Testing: Verifying that the software meets the requirements.

Deployment: Making the software available to users.

Maintenance: Supporting and updating the software after deployment. - Easy to understand and manage but lacks flexibility for changes



**2. Agile Model:**

Iterative and incremental development.

Close collaboration between the development team and business stakeholders.

Responding to change over following a plan.

Continuous delivery of working software.



**3. Spiral Model:**

**Planning:** Defining objectives, alternatives, and constraints.

**Risk Analysis:** Identifying and assessing potential risks.

**Engineering:** Developing and testing the software.

**Evaluation:** Reviewing the results and planning for the next cycle.

**4. V-Model:**

Requirements Analysis ↔ User Acceptance Testing

System Design ↔ System Testing

Architectural Design ↔ Integration Testing

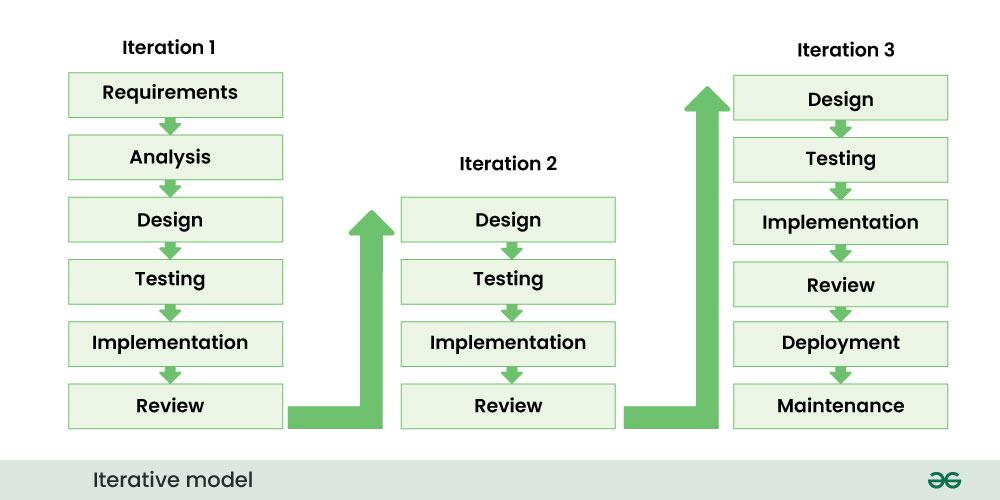
Module Design ↔ Unit Testing



**5.The Iterative model:**

.The software is developed in cycles (iterations), with each iteration building upon the previous one. Each iteration involves planning, design, implementation, and testing.

. The process starts with a small set of requirements and progressively adds more functionality in each iteration until the complete system is ready.

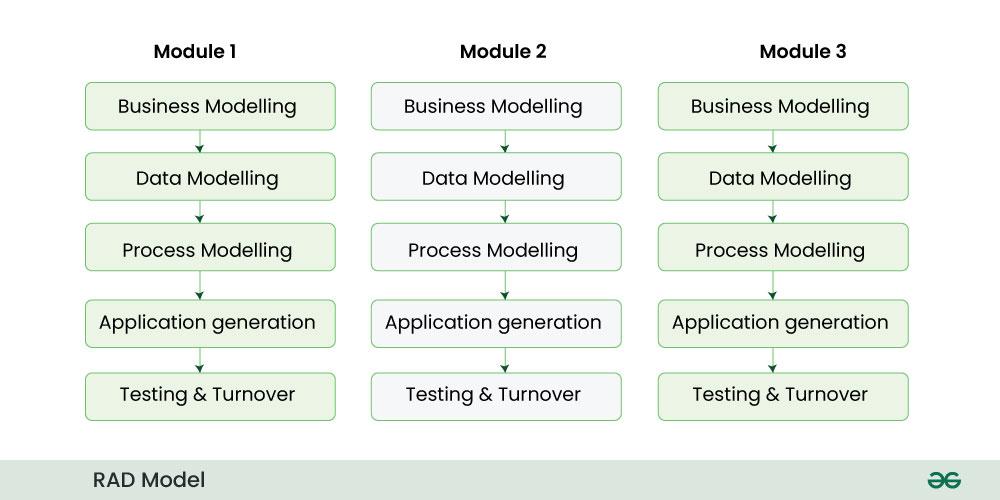
. Use Case**:** Suitable for projects where requirements may evolve over time, or when building a complex system in stages allows for feedback incorporation.

**6. RAD (Rapid Application Development)**:

Small, highly collaborative teams.

Iterative development.

Heavy use of prototyping.

-Time-boxing (fixed deadlines for each phase).

**7. Prototype Model:**

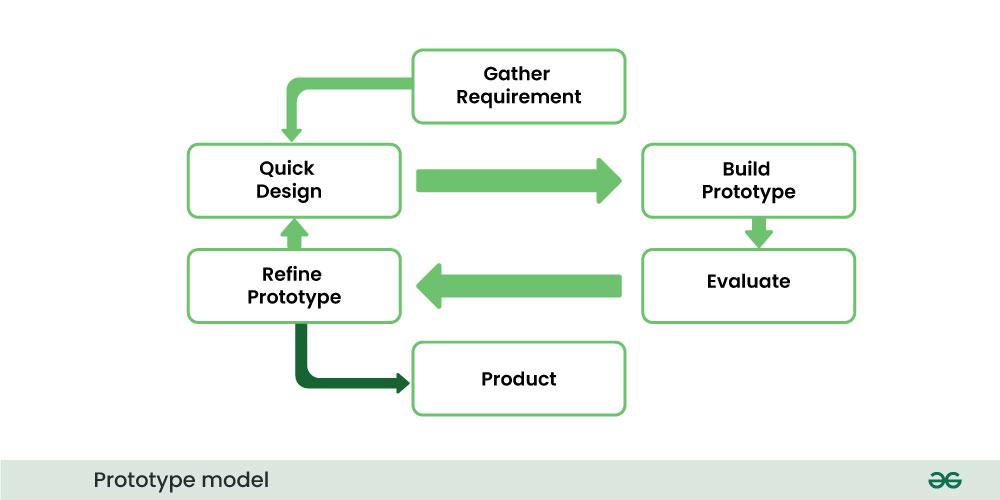
Requirements Gathering

Quick Design

Building a Prototype

Customer Evaluation

Refining the Prototype

Developing the Final Product

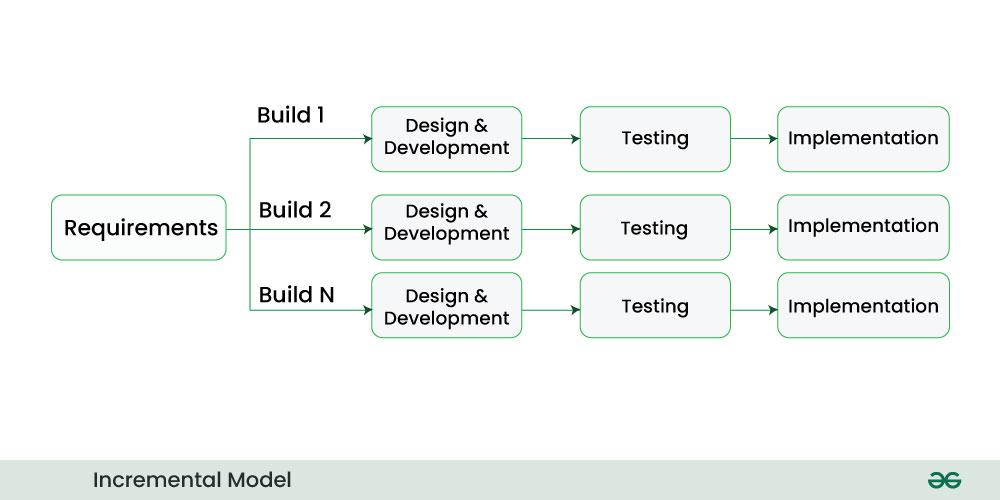
**8. Incremental Model:**

- Delivers the product in pieces rather than all at once

- Divides the product into builds with increasing functionality

- Each increment adds features to previously delivered increments

- Beneficial when staffing is limited or core features need early delivery



**Task 5. where are the models applied?**

1. Waterfall Model

* Application: Best suited for projects with very well-defined and stable requirements, where changes are unlikely and the process can flow linearly. Often used for smaller projects or in highly regulated industries.
* Examples:
  + Developing a simple data conversion program with clearly documented input and output formats.
  + Building embedded software for devices with fixed functionalities and specifications.
  + Large-scale government projects with clearly defined requirements and lengthy approval processes.

Agile Model

* Application: Highly versatile and widely used for projects with dynamic or evolving requirements, where customer collaboration and rapid feedback are crucial. Suitable for various project sizes and complexities.
* Examples:
  + Developing web applications with features added iteratively based on user feedback.
  + Creating mobile apps with frequent updates and new functionalities.
  + Developing e-commerce platforms with continuous integration and deployment.
  + Internal business systems where requirements may change as the business evolves.

Spiral Model

* Application: Best for large, complex projects with significant risks, where requirements are not well-defined initially, and risk analysis is essential at each phase.
* Examples:
  + Developing new operating systems with complex functionalities and potential technical challenges.
  + Large-scale software development for aerospace or defense industries where failure can have severe consequences.
  + Complex enterprise resource planning (ERP) system implementations with high integration risks.

V-Model

* Application: Used in projects where a strong emphasis on testing and quality assurance is required, with a clear correspondence between development and testing phases. Often applied in safety-critical or regulated domains.
* Examples:
  + Developing medical device software that requires rigorous validation and verification.
  + Building software for nuclear power plants with stringent safety standards.
  + Developing banking or financial systems where data integrity and security are paramount.

Iterative Model

* Application: Suitable when requirements are not fully clear at the outset, allowing for the development of initial versions and refinement through repeated cycles based on feedback.
* Examples:
  + Developing a new customer relationship management (CRM) system where features are rolled out in phases based on user needs.
  + Building a complex data analytics platform with iterative improvements to algorithms and visualizations.
  + Creating a new module for an existing software system, allowing for feedback after initial implementation.

RAD (Rapid Application Development) Model

* Application: Focuses on speed and rapid prototyping, often used for projects where time is a critical factor and requirements are reasonably well-defined but need flexibility. Requires strong user involvement.
* Examples:
  + Developing internal business applications with active participation from end-users in prototyping sessions.
  + Creating a specific module or extension for an existing system with a tight deadline.
  + Building proof-of-concept applications to quickly demonstrate feasibility and gather feedback.

Prototype Model

* Application: Primarily used to clarify unclear requirements or to visualize the system's functionality for the user before full-scale development. Often serves as a precursor to other SDLC models.
* Examples:
  + Designing the user interface (UI) and user experience (UX) of a new application to get early user feedback.
  + Developing a preliminary version of a key module to test its feasibility or performance.
  + Creating a mock-up of a website or application to demonstrate the intended look and feel.

Incremental Model

* Application: Involves building the system in small, functional increments, with each increment being developed and delivered. Similar to iterative but with a focus on delivering working functionality early and often.
* Examples:
  + Developing a large e-commerce website by first implementing basic product browsing and purchasing, then adding features like user reviews, wishlists, and payment gateway integrations in subsequent increments.
  + Building a compiler by first implementing the parsing stage, then the semantic analysis, code generation, and optimization stages as separate increments.
  + Developing a software suite where different modules are built and released *independently.*

***TASK 6. Advantages and Disadvantages of SDLC models:***

|  |  |  |
| --- | --- | --- |
| Model | Advantage | Disadvantage |
| Waterfall Model | Simple to understand and implement due to its linear and sequential nature. | Inflexible to changes once a phase is complete. |
| Agile Model | Highly flexible and adaptable to changing requirements. | Can lack predictability in terms of timelines and budget. |
| Spiral Model | High emphasis on risk analysis, suitable for complex, high-risk projects. | Can be costly and requires significant risk assessment expertise. |
| V-Model | Strong emphasis on verification and validation, early defect detection. | Rigid and less flexible to changes after the requirements phase. |
| Iterative Model | Allows for early implementation and incorporates feedback in iterations. | Requires disciplined planning and management to control scope creep. |
| RAD Model | Focuses on speed and rapid prototyping, leading to quicker cycles. | Requires strong user involvement and may compromise on functionality or quality. |
| Prototype Model | Helps clarify user requirements and design early through working models | May lead to poorly defined solutions if the prototype isn't managed well |
| Incremental Model | Delivers working functionality in increments, early value delivery. | Requires careful planning and design for effective increment integration. |

**Task 7. what do you understand by SCRUM:**

Scrum is like breaking down a big project into smaller, manageable tasks that take 2-4 weeks to complete. The team meets every day to discuss progress and any problems they're facing. There are three main people involved: someone who manages the process (Scrum Master), someone who decides what needs to be built (Client), and the team that does the actual work (Development Team).

**Task 8. What is meant by sprint?**

sprint is a set period of time during which specific work has to be completed and made ready for review. Incorporating sprints into the software development lifecycle encourages iterative development and continuous improvement, ultimately helping organizations to produce quality software faster and often at lower cost.

Task 8. What are do’s and don’ts within a sprint?

Do:

* A Frequent and transparent communication is must in sprint, always inform your team about the progress and any blockers and any help needed. This helps everyone stay aligned and main issues are addressed quickly.
* Always Prioritize tasks aligned with the Sprint Goal. It helps to focus efforts on completing the task: This helps the team deliver value and stay on track.
* Provide regular progress updates: Keep your assigned tasks updated on the team's tracking board (e.g., Jira, Trello) so everyone has a clear view of the sprint's overall progress and potential bottlenecks.

Don't:

* Introduce new, unplanned work into the sprint: Avoid adding tasks or features that weren't agreed upon during Sprint Planning. This can disrupt the team's focus and jeopardize the Sprint Goal.
* Work in silos without interaction: Don't isolate yourself from the team. Regularly engage in discussions, offer help, and seek input to foster collaboration and shared understanding.
* Hesitate to raise impediments or challenges: Don't wait if you're blocked or foresee a problem. Bring it to the team's attention immediately so that the Scrum Master and the team can work to find a solution.

Task 9: Backlog and stories in sprint?

**Backlogs in Sprint:**

The **Product Backlog** is the prioritized master list of all desired product features and improvements. During **Sprint Planning**, the team pulls high-priority **user stories** from it to form the **Sprint Backlog**. This sprint-specific backlog details the work committed for the current sprint, including the stories and the actionable tasks required to deliver them. It's the team's roadmap for the sprint.

This is the set of Product Backlog items selected for a specific sprint, along with the plan for delivering the Increment and achieving the Sprint Goal. It's a highly visible, real-time plan by and for the Developers, outlining the "what" and the "how" of the sprint.

**Stories in Sprint:**

Within a sprint, **user stories** from the Product Backlog provide a user-centric understanding of the functionality to be built. They are broken down into smaller, manageable **tasks** in the **Sprint Backlog**. The Development Team works to complete these tasks to deliver the value described in the user stories by the end of the sprint, aiming to meet the defined acceptance criteria for each story.

A user story is a short, simple description of a feature told from the perspective of the person who will benefit from the new capability, usually following the format: "As a [type of user], I want [some goal] so that [some reason]." It helps the team understand the value for the end-user.

**Task 10: what are scrum artifacts?**

Product Backlog: The prioritized, single source of all work for the product.

Sprint Backlog: The selected Product Backlog items and the plan for the current sprint.

Increment: The sum of all completed Product Backlog items, creating a usable version of the product.

**Burn-Down Chart** is like a graph that shows how many tasks are left on that list each day.

* The line goes down as you finish tasks.
* If the line goes to zero by the end of the week, you finished everything!
* If the line stays high, you might not finish on time.

It's a simple way to see if you're on track to complete your work for the week.

**TASK 11: PORTS AND PROTOCOLS**

Protocols are sets of rules that dictate how data is formatted, transmitted, and received over a network, ensuring devices can communicate despite different systems. Think of them as the language computers use to talk.

Ports are virtual doorways on a computer that allow different applications or services to use the network connection without interfering with each other. Each port is identified by a number, acting like an address for a specific service running on a device.

**TASK 12: TYPE OF PORTS**

1. LAN (Local Area Network)
2. WAN (Wide Area Network)
3. MAN (Metropolitan Area Network)
4. WLAN (Wireless Local Area Network)
5. VPN (Virtual Private Network)
6. VLAN (Virtual Local Area Network

**TASK 13: WHAT ARE TYPES OF SERVER?**

1. Web Servers - Serve web pages and web applications (HTTP/HTTPS) - Examples: Apache, Nginx, Microsoft IIS

2. Application Servers - Host and run applications/software - Examples: Tomcat, JBoss, WebSphere

3. Database Servers - Store, manage, and serve database files - Examples: MySQL, PostgreSQL, Oracle, Microsoft SQL Server

4. File Servers - Dedicated to storing and sharing files - Examples: NFS servers, Windows File Servers, FTP servers

5. Mail Servers - Handle email sending, receiving, and storage - Examples: Microsoft Exchange, Postfix, Send mail.

6. DNS Servers - Resolve domain names to IP addresses - Examples: BIND, Microsoft DNS

7. Proxy Servers - Intermediate between users and the internet - Types: Forward proxy, reverse proxy, caching proxy

8. DHCP Servers - Automatically assign IP addresses to devices on a network

9. API Servers - Provide application programming interfaces

**Task 14: What do you know about DNS? Domain Name System**

DNS is the internet's directory system that converts human-readable website names (like google.com) into numerical IP addresses (like 172.217.168.238) that computers use to identify each other. It works through a global network of servers that maintain these name-to-address mappings. Without DNS, we would need to remember number sequences instead of names to access websites. DNS also handles email routing, service locations, and other internet addressing functions. It operates invisibly in the background but is essential for nearly all internet activity - when you type a website address, DNS is what finds the actual server for your browser to connect to.

**TASK 15: VPN (Virtual Private Network)**

A VPN creates an encrypted tunnel for your internet traffic, protecting your data and privacy online while masking your IP address. It allows secure access to resources across public networks as if directly connected to a private network.

Types of VPNs

1. **Remote Access VPN**: Connects individual users to a private network from remote locations (used for remote work)
2. **Site-to-Site VPN**: Links entire networks together (connects branch offices to headquarters)
3. **Protocol-based VPNs**:
   * **IPsec**: High security, widely used for business VPNs
   * **SSL/TLS**: Browser-based, doesn't always require special software
   * **OpenVPN**: Popular open-source solution with strong security
   * **WireGuard**: Newer, faster protocol with simplified code
   * **L2TP/IPsec**: Combined protocols for better security
4. **Consumer VPNs**: Commercial services focused on privacy and accessing geo-restricted content
5. **Corporate VPNs**: Focus on secure business communications and resource access

**TASK 16: WHAT IS TOPOLOGY? DIFFERENT TYPES OF TOPOLOGIES AND ITS DIAGRAM**

Network topology refers to the physical or logical layout of devices and connections in a network. It defines how computers, switches, routers, and cables are arranged and connected to each other.

Types of Network Topologies

1. Point-to-Point Topology

- Direct connection between two nodes

- Simple but limited to two devices

2. Bus Topology

- All devices connect to a single central cable

- Simple, inexpensive, but vulnerable to cable failures

3. Ring Topology

- Each device connects to exactly two other devices, forming a circle

- Data travels in one direction around the ring

4. Star Topology

- All devices connect to a central hub or switch

- Most common in modern networks

- Easy to troubleshoot but depends on central device

5. Tree/Hierarchical Topology

- Branches out like a tree with layers of connections

- Good for large networks with distinct sections

6. Mesh Topology

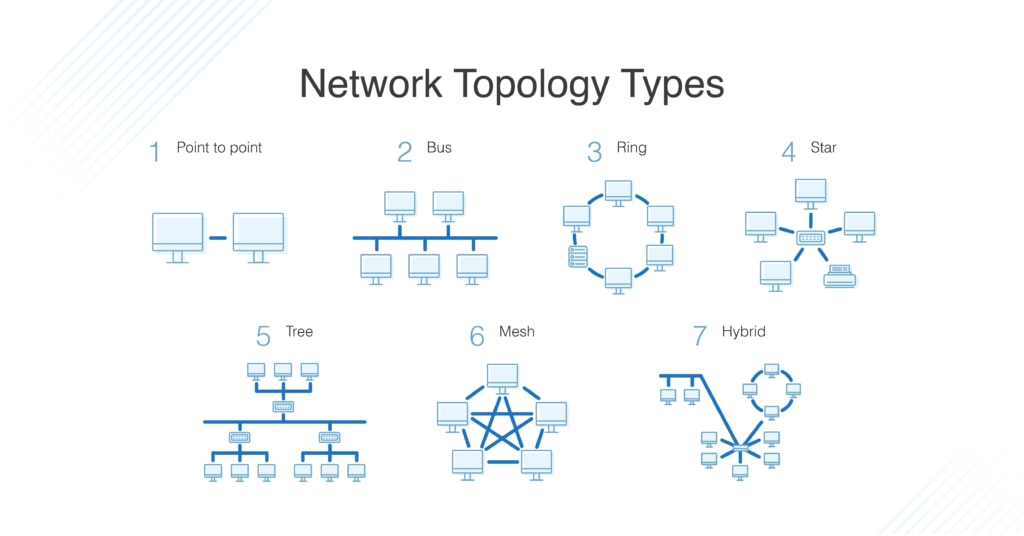
- Devices interconnect with multiple redundant connections

- Highly reliable but expensive and complex

7. Hybrid Topology

- Combines two or more different topologies

- Tailored to specific organizational needs



**TASK 17: What is OSI Model? What are different layers in osi model?**

The OSI (Open Systems Interconnection) Model is a conceptual framework that standardizes the functions of a telecommunication or computing system into seven distinct layers. Here's a detailed explanation of each layer, from bottom to top:

- Provides network services to applications

1. Physical Layer

- Raw bit transmission

- Hardware specifications

- Physical medium characteristics

- Cables, switches, network adapters

2. Data Link Layer

- Physical addressing (MAC addresses)

- Error detection and correction

- Frame formatting and flow control

- Divided into LLC and MAC sublayers

3. Network Layer

- Routing and logical addressing

- IP addressing and packet forwarding

- Determines best path for data delivery

4. Transport Layer

- End-to-end communication

- Protocols: TCP (reliable) and UDP (unreliable)

- Handles flow control and error correction

5. Session Layer

- Manages sessions between applications

- Establishes, maintains, and terminates connections

- Handles authentication and authorization

6. Presentation Layer

- Data translation and encryption

- Data formatting and encryption

- Handles data compression and conversion

7. Application Layer

- Direct interaction with end-user applications

- Examples: HTTP, FTP, SMTP, DNS

