**SDLC:**

SDLC or Software Development Life Cycle is a process used for developing and maintaining a software applications efficiently, with quality and in expected time. It outlines a series of steps that guide the software development team from the initial idea to the final deployment and maintenance of the application. It includes  planning, designing, developing, testing, deploying, and maintaining the software in an efficient way.

**Why is SDLC**

It is important because it gives a clear, organized way to approach software development.

* helps you understand how a software project progresses step by step
* estimate resources, costs, and time more accurately.
* less prone to bugs.
* improving teamwork

1. Steps in SDLC
   1. Requirement Gathering & Analysis
   2. Planning
   3. Design
   4. Implementation
   5. Testing
   6. Deployment

1. SDLC Models
   1. Waterfall Model
      1. A linear and sequential approach. Each phase must be completed before the next begins.
      2. **Phases:** Requirements → Design → Implementation → Testing → Deployment → Maintenance
      3. Simple and easy to manage
      4. Well-defined stages
   2. V-Model (Validation and Verification)
      1. An extension of the Waterfall model with corresponding testing phases for each development stage.
      2. **Key Idea:** For every development activity, there's a corresponding test activity.
      3. Early detection of defects
      4. Better test planning
2. Incremental Model
   1. The system is developed and delivered in smaller parts or increments.
   2. **Key Idea:** Each increment adds functionality.
   3. Easier to test and debug
   4. Delivers usable product early
3. Iterative Model
   1. Starts with a small part of the system, then iteratively enhances it through repeated cycles.
   2. Flexibility to change requirements
   3. Continuous feedback
4. Spiral Model
   1. Combines iterative development with risk analysis. Software is developed in loops (spirals).
   2. **Phases in each loop:** Planning → Risk Analysis → Development → Evaluation
   3. Focus on risk management
   4. Flexible and iterative
5. Agile Model
   1. An adaptive model with rapid, iterative development cycles (sprints), continuous feedback, and collaboration.
   2. Highly flexible
   3. Fast delivery and customer feedback
6. DevOps Model
   1. Combines development and operations for continuous integration and continuous deployment (CI/CD).
   2. Fast and automated deployment
   3. Improved collaboration

**Network Types**

* Personal Area Network (PAN)
* Local Area Network (LAN)
* Wireless Local Area Network (WLAN)
* Metropolitan Area Network (MAN)
* Wide Area Network (WAN)
* Storage Area Network (SAN)
* Virtual Private Network (VPN)

**Types of servers**

* Web Server
  + Function: Hosts websites and serves web pages to users over the internet or intranet
  + Example Software: Apache, Nginx, Microsoft IIS.
* File Server
  + Function: Stores and manages files so users can access and share them over a network.
  + Use Case: Central file storage in offices or schools.
* Database Server
  + Function: Hosts and manages databases, handles data requests from clients or applications.
  + Example Software: MySQL, Microsoft SQL Server, Oracle DB
* Mail Server
  + Function: Sends, receives, stores, and manages email.
  + Example Software: Microsoft Exchange, Postfix, Sendmail.
* Application Server
  + Function: Hosts and runs specific applications or software used by clients.
  + Use Case: Middleware between database and end user (e.g., business apps)
* DNS Server (Domain Name System)
  + Function: Translates domain names into IP addresses.
  + Use Case: Allows users to access websites using names instead of IPs.
* Proxy Server
  + Function: Acts as an intermediary between a client and other servers to enhance security, privacy, or caching.
  + Use Case: Corporate internet filtering, content caching.
* Virtual Server
  + Function: A software-based emulation of a physical server.
  + Use Case: Hosting multiple virtual environments on a single physical machine (via virtualization).
* DHCP Server
  + Function: Automatically assigns IP addresses to devices on a network.
  + Use Cas**e:** Simplifies network configuration.

**OSI Model**

The OSI Model is a framework that explains how data moves from one device to another over a network. It divides this process into 7 layers.

* application layer - The layer closest to the user - Browsers (Chrome)
* Presentation - Prepares data so the app understands it.
* Session - Manages and controls the dialog between two computers.
* Transport - Ensures reliable data delivery.
* Network - Finds the best path for data to travel.
* data link - Transfers data between devices on the same network.
* Physical - Transmits raw bits

We use a VPN (Virtual Private Network) for several key reasons:

* Privacy Protection: A VPN hides your IP address and encrypts your internet traffic, making it harder for websites, advertisers, or hackers to track your online activity.
* Secure Public Wi-Fi Use: When using public Wi-Fi (like in coffee shops or airports), a VPN protects your data from potential hackers on the same network.
* Bypass Geo-Restrictions: VPNs can help access content or websites that are restricted based on your geographic location (e.g., streaming services or censored websites).
* Remote Access to Networks: VPNs allow employees to securely access a company's internal network from remote locations.
* Avoid Bandwidth Throttling: Some ISPs slow down your connection based on usage; a VPN can help avoid this by hiding your activity.

**Access VPN**

* Use: For individual users to securely connect to a private network (e.g., employees accessing a company network from home).
* How it works: Connects the user to a central VPN server, encrypting the connection.
* Example protocols: OpenVPN, L2TP/IPsec, IKEv2.

**Site-to-Site VPN**

* Use: To connect entire networks (e.g., branch offices connecting to a company HQ network).
* How it works: VPN gateways on each side create a secure tunnel between networks.
* Common in: Enterprises with multiple office locations.

Types of Network Topologies

* Bus
  + All devices connected to a single central cable
  + Easy to install
  + low cost.
  + Slow with many devices
* Star
  + devices are connected to a central hub or switch.
  + Easy to manage
  + failure in one node doesn’t affect others.
  + Hub failure takes down the whole network.
* Tree
  + A extended bus topologies
  + Scalable, structured layout
  + (looks like tree data structure)
* Ring
  + Devices are connected in a closed loop
  + Performance is Predictable
  + One failure can affect the entire loop
* Mesh
  + Every device is connected to every other device.
  + Expensive and complex to set up

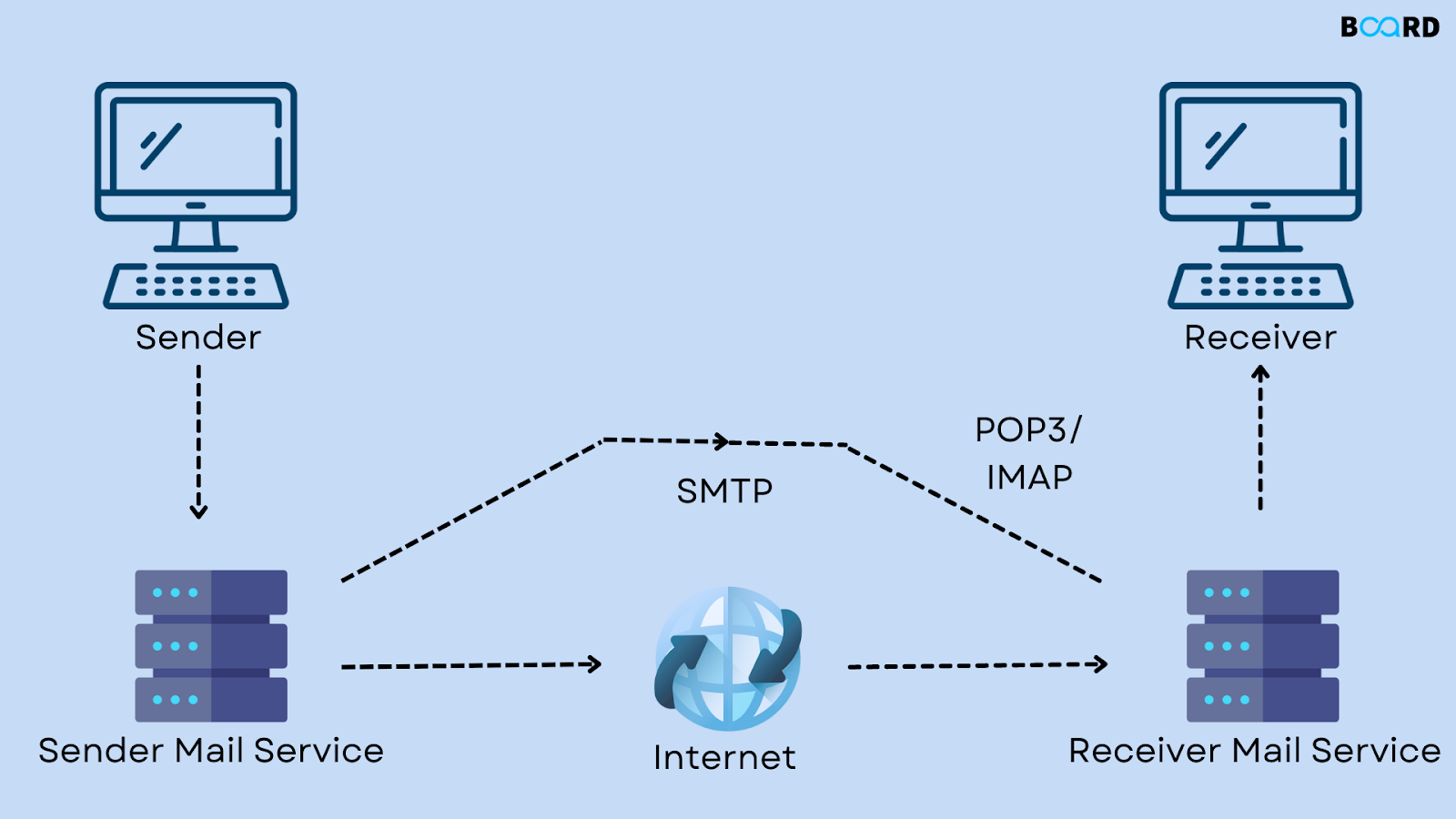
Router:

* Connects multiple similar networks
* Routes data packets between networks
* Ex: my laptop and phone are connected to same network via wifi (wireless network) and pc via ethernet cable (LAN cable)

Gateway

* Connects different networks
* Translates data between different networks
* Ex: Router connects the internal network to the internet (here the router is the gateway)

SMTP - Simple Mail Transfer Protocol



OSI and TCP/IP difference

OSI

* 7 layers - Physical, Data Link, Network, Transport, Session, Presentation, Application
* Developed by ISO
* Its a model to understand and design a network system
* Theoretical (not used anywhere)

TCP/IP

* 4 layers - Network Interface, Internet, transport and appliation
* Developed by US DOD
* model for real-world networking
* Practical (real world networking)

**Software Development Life Cycle (SDLC)**, **High-Level Design (HLD)** and **Low-Level Design (LLD)  
  
High-Level Design (HLD)**

* Gives an overview of the system's architecture and components.
* Data flow diagrams
* High-level database design/diagrams
* Interfaces between systems or modules

Describing that the system will have a frontend web app, backend services, and a database, and how they interact.

**Low-Level Design (LLD)**

* Provides detailed logic and design of each module/component.
* Class diagrams
* Method-level logic
* Pseudocode or algorithms
* API specifications
* Error handling
* validation rules

Detailing how a login module works, including method names, input validation, error messages, and database queries.

**SRS (Software Requirements Specification)**

* It is a formal document that outlines exactly what a software system should do or how it should performer and the constraints under which it must operate
* Structure of an SRS Document
  + Introduction
    - Purpose
    - Scope
    - Definitions, acronyms, and abbreviations
    - References
    - Overview of the document
  + Overall Description
    - Product perspective
    - Product functions
    - User characteristics
    - Constraints
    - Assumptions and dependencies
  + Specific Requirements
    - Functional requirements (what the system should do)
    - Non-functional requirements (performance, security, usability, etc.)
    - Interface requirements (user, hardware, software, communication interfaces)