1. What is a web server? Explain its primary function.

A **web server** is a software application or hardware device that serves web content to users over the internet or an intranet. It processes requests from clients (usually web browsers) and responds with the requested web pages, files, or other resources.

Primary Function of a Web Server

The primary function of a web server is to **deliver web content** to users. This involves:

1. **Handling HTTP Requests** – Receiving requests from clients (browsers) via the **Hypertext Transfer Protocol (HTTP/HTTPS)**.
2. **Processing Requests** – Locating and retrieving the requested files (HTML, CSS, JavaScript, images, etc.) or forwarding dynamic requests to application servers.
3. **Sending Responses** – Returning the requested resources or appropriate error messages (e.g., 404 Not Found) to the client.
4. **Executing Server-side Scripts** – Running backend scripts (PHP, Python, Node.js, etc.) if needed for dynamic content.
5. **Security and Access Control** – Implementing authentication, encryption (SSL/TLS), and firewall protections to secure data transmission.
6. **Logging and Monitoring** – Keeping records of requests, errors, and server performance for analysis.

2. What are the key components of a web server?

Key Components of a Web Server

1. Hardware Components

These are the physical components that host the web server software and handle network connections.

- **CPU & Memory (RAM):** Process HTTP requests and manage multiple users simultaneously.
- **Storage (HDD/SSD):** Stores website files, databases, and logs.
- **Network Interface:** Facilitates data transmission over the internet via TCP/IP protocols.

---

2. Software Components

These include the core functionalities that enable a web server to operate efficiently.

a) HTTP Server Software

- Handles **HTTP/HTTPS requests** and responses.
- Examples: **Apache, NGINX, Microsoft IIS, LiteSpeed.**

b) Operating System (OS)

- The underlying system that manages hardware resources.
- Common OS choices for web servers:
    - **Linux** (Ubuntu, CentOS, Debian)
    - **Windows Server**
    - **Unix-based systems (FreeBSD, macOS Server)**

c) Web Server Engine

- Responsible for processing and serving static and dynamic content.
- Works with **modules or extensions** to support scripting languages like PHP, Python, Ruby, etc.

d) Database Server

- Stores and manages website data dynamically.
- Examples: **MySQL, PostgreSQL, MongoDB, MariaDB.**

e) Server-side Scripting Languages

- Allows execution of **dynamic web applications**.
- Examples: **PHP, Python, Ruby, Node.js, ASP.NET.**

f) Security & Encryption Components

- **SSL/TLS Certificates:** Encrypt data for secure communication (HTTPS).
- **Firewalls & Access Control:** Protect against unauthorized access (e.g., ModSecurity, fail2ban).
- **Authentication & Authorization Mechanisms:** Secure login systems (OAuth, JWT, LDAP).

g) Caching Mechanisms

- Improves website speed by storing frequently accessed content.
- Examples: **Varnish, Redis, Memcached, Cloudflare (CDN caching).**

h) Logging & Monitoring Tools

- Track server activity, errors, and performance.
- Examples: **Logrotate, ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, Grafana.**

3. **Discuss the difference between IP-based and name-based virtual hosting.**

Feature	IP-Based Virtual Hosting	Name-Based Virtual Hosting
Number of IPs	Requires multiple IPs	Uses a single IP address
Host Differentiation	Based on IP address	Based on domain name (Host header)
SSL Support	Works without SNI	Requires SNI for HTTPS
Resource Usage	Higher (more IPs needed)	Lower (shared IPs)
Cost	More expensive	More cost-effective
Scalability	Limited (IP availability)	High (many sites per IP)