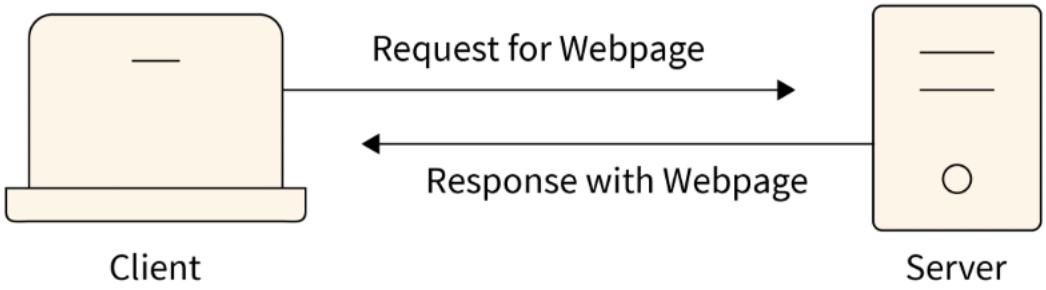
Question1: How does client server architecture works?

Answer:

Client-server architecture is a model where a **client** requests services or resources, and a **server** responds over a network.



How it works:

- **Client Requests:**
  A client (like a web browser or app) sends a request to the server.
  Example: User types www.example.com.
- **Server Receives:**
  The server listens and receives the request.
  Example: It sees the HTTP request for the homepage.
- **Processing:**
  The server processes the request, like fetching data or performing tasks.
  Example: Gets homepage content from the database.
- **Server Responds:**
  The server sends back a response (HTML, JSON, etc.).
  Example: Sends homepage HTML.
- **Client Displays:**
  The client displays or uses the data.
  Example: Browser shows the website to the user.

Question 2: What are the factors to be considered when designing a software?

Answer:

When designing software, several factors must be considered to ensure it is functional, efficient, and user-friendly.

Here are key factors :

1. Requirements Gathering:

    * Understand user needs and business objectives.

    * Identify functional and non-functional requirements.

    * Specify system constraints and performance requirements.

2. Architecture and Design Patterns:

    * Choose an appropriate architecture (e.g., monolithic, microservices, client-server).

    * Consider common design patterns (e.g., MVC, Singleton, Observer) to structure the system effectively.

3. Scalability:

    * Plan for future growth, ensuring the software can handle increased users, data, or transactions.

    * Design for horizontal or vertical scaling depending on the needs.

4. Performance:

    * Optimize code and system resources for speed and efficiency.

    * Address factors like load times, response times, and throughput.

    * Consider caching, database optimization, and algorithmic efficiency.

5. Security:

    * Incorporate security measures like encryption, authentication, and authorization.

    * Prevent common vulnerabilities (e.g., SQL injection, cross-site scripting).

    * Use secure coding practices and ensure data privacy.

Question3: Why do we really need network programming tools and platform? Explain some of them.

Answer: Network programming tools and platforms are vital for developing, testing, and managing network-based applications. They simplify the design, implementation, and monitoring of communication protocols and services, making complex network systems easier and faster to build and optimize.

Tools:

* Wireshark: A widely-used tool for capturing and analyzing network packets. It helps developers inspect network traffic and debug communication issues.

* NetFlow Analyzer: Used to monitor network traffic patterns and identify anomalies, helping optimize the network performance.

* Iperf: A tool to measure network performance, including bandwidth, latency, and jitter.