1. How does client server architecture works?

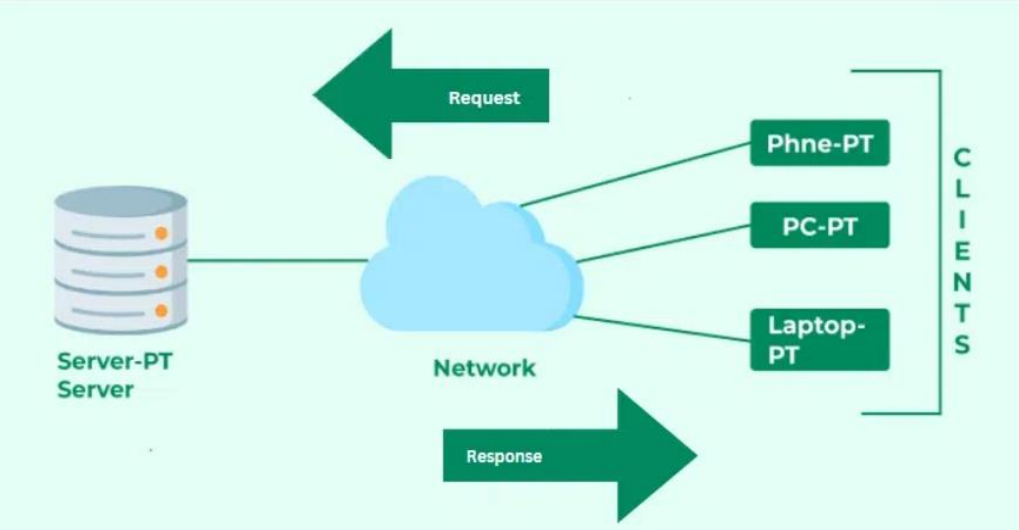
Client-server architecture is a computing model where multiple devices interact with a centralized server to request and receive services, data or resources. It follows a request-response model.

a. Client Initiates Request
The client sends a request to the server using a protocol such as HTTP, HTTPs, FTP or TCP/IP. Eg. A user enters a website URL, and the browser sends a request to the web browser.

b. Server Processes the Request
The server receives the request, processes it, and fetches data from a database or performs necessary computation. Eg. A webserver retrieves a webpage from a databases and prepares an HTML response.

c. Server sends a Response
After processing, server sends back the response to the client. Eg. The webserver returns an HTML page, image file to the browser.

d. Client renders the response
The client processes the received data and displays it to the user. Eg. A browser renders the received HTML into a webpages.



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

These are the factors:

a. User Requirements
    i. Functional Requirements: What specific tasks or functions should the software perform?
    ii. Non-Functional Requirements: These include performance, security, usability and scalability requirements.
b. User Experience(UX)
    i. Usability: Ensure the software is easy to use and intuitive.
    ii. Accessibility: Make the software accessible to people with disabilities.
    iii. Design: Create an appealing and consistent user interface.
c. Architecture and Design Patterns
    i. Scalability: Ensure the software can handle growth in users and data.
    ii. Maintainability: Design for ease of maintenance and updates.
    iii. Modularity: Break down the software into manageable, independent modules.
d. Technology Stack:
    i. Programming Language: Choose the right language based on projects requirement.
    ii. Framework and Libraries: Utilize framework and libraries to speed up development.
    iii. Platforms: Decide whether the software will be web-based, mobile or desktop.
e. Performance and Efficiency
    i. Speed: Optimize the software for fast response times.
    ii. Resource Utilization: Ensure efficient use of CPU, memory and other resources.
f. Security
    i. Data Protection: Implement measures to protect user data from breaches.
    ii. Authentication and Authorization: Ensure proper user authentication and access control.
    iii. Encryption: Use encryption to protect data in transit and at rest.
g. Testing and Quality Assurance
    i. Automated Testing: Use automated tests to catch bugs early.
    ii. Manual Testing: Perform manual testing for usability and edge cases.
    iii. Continuous Integration: Implement continuous integration to streamline the development process.
h. Documentation
    i. Code Documentation: Comment code and provide clear explanation of complex logic.
    ii. User Documentation: Provide user manuals and help guides.

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

They are important for below reasons:

i. Easy Communications: Help computers, server and applications talk to each other.
ii. Security: Protects data from hackers and unauthorized access.
iii. Debugging and Troubleshooting: Helps find and fix network problems.
iv. Faster Performance: Make sure data is sent and received quickly.
v. Cloud and Web Applications: Allow app to work online, such as e-commerce website and cloud storage.

Some popular tools and platforms are:

i. Wireshark - Network Traffic Monitor
   - Helps track and analyze network activity.
   - Used for fixing slow internet, checking security threats.
ii. Postman - API Testing Tool
   - Used to send and receive data from web services.
   - Helps developers test websites and apps.
iii. PuTTY – Remote Server Access
   - Allow users to connect to another computer remotely.
   - Used by IT admins to manage websites and server.
iv. Node.js with socket.io – Real Time Communication
   - Helps create real-time applications like chat apps and online games.
v. OpenSSL - Security Encryption
   - Used to encrypt data for secure communication.
   - Helps website use HTTPs for safe browsing.