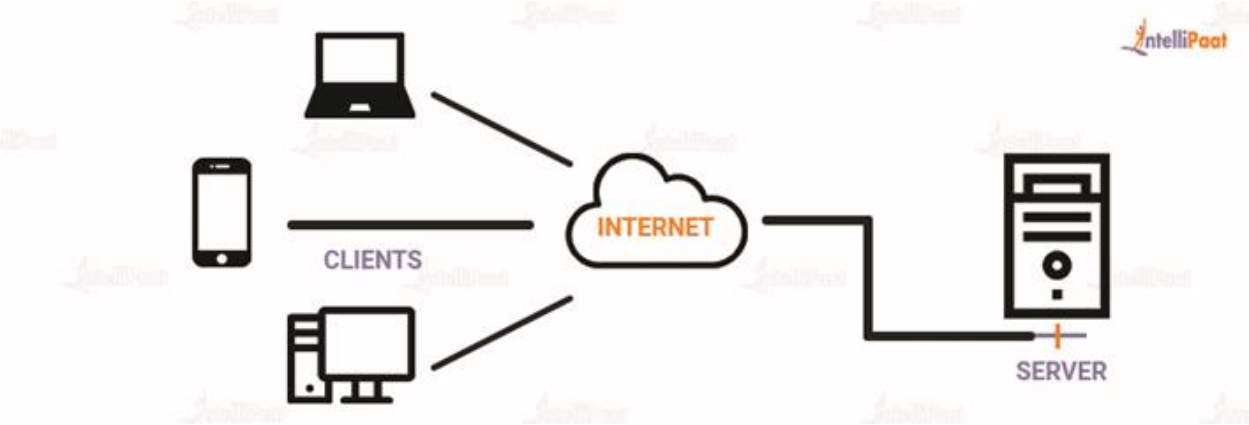
Server Side Programming Concepts

Client-Server Architecture:

a. What is client-server architecture, and how does it work?

- Client-Server architecture is a **centralized network** model in which the server processes the request from the client and sends response and provides services to the client.
- In this architecture, it mainly consists of two factors
  - **Client** - Sends **requests** to the server, the client must be connected to internet for accessing the server for services.
  - **Server** - Receives requests and sends **response** to the client



Working:

- First, the client sends the request to the server.
- The server receives the request and process the request.
- Finally, the server sends the response to the client.

b. Explain the roles of the client and server in this architecture.

Client:

It does not share any of its resources, but it **sends requests** (for contents or services) to the server.

Server:

It receives request from client and processes the request, and then sends back the processed content or **services** to the client.

c. What are the benefits of using a client-server model for building applications?

- This model is **more efficient** in delivering resources to the client.
- It requires **low cost** maintenance.
- In this model, the data is only stored in the server, not in all the devices of the clients, which **reduces the data replication** for the application.
- **Scales Vertically** - We can move to more powerful machines, to take advantage of the larger system's performance.
- **Scales Horizontally** - Each servers having capabilities, processing power and can be added to the existing servers to distribute processing load.

d. Can you give an example of a real-world application that follows a client-server architecture?

Real world applications that follows a client-server architecture are

- Email
- Network printing
- World Wide Web

RESTful API (Representational State Transfer):

a. What is a RESTful API, and what are its key principles?

- An API is a set of definitions and protocols which is used as an **interface** between two software for exchanging information securely.
- RESTful API is a specific type of API which have a specific **REST architecture style**.
- When a client requests a resource using a RESTful API, the server transfers back the **current state** of the resource in a standard representation.

Principles:

- Statelessness
- Uniform interface
- Resource-based architecture
- Self-descriptive messages
- Separation of client and server

b. What are the main components of a RESTful API request?

- RESTful API request is made up of four things:
  - Endpoint
  - Method
  - Headers
  - Data (or Body)
- **Endpoint** - It is the URL you request for.

  It follows the structure:

  root-endpoint/?
- **Method** - It indicates the **type** of your request
- **Headers** - Used to provide information to both client and server.
- **Data(or Body)** - It contains the information you want to send to the server.

c. Describe the common HTTP methods used in RESTful APIs (GET, POST, PUT, DELETE) and their purposes.

- **GET** - This method is used to **retrieve** a record or a collection of record from the server.
- **POST** - This method is used to send a data to **create** a **new record** on the server.
- **PUT** - This method is used to **update** an existing data on the server.
- **DELETE** – This method is used for **deleting** a record on the server.

d. What is the significance of status codes (e.g., 200, 404, 500) in RESTful API responses?

- **Code 200** indicates that the request is **succeeded**.
- **Code 303** indicates that you are being **redirected** to another resource.
- **Code 400** indicates that the server was unable to process the request due to invalid information send by the client **(client side problem)**.
- **Code 404** indicates the server was **unable to find resource** that was requested.
- **Code 500** indicates that the processing of the request on the **server failed** unexpectedly.

e. How is statelessness achieved in RESTful APIs, and why is it important?

- Statelessness refers to a communication method in which the server completes every client request independently of all previous requests.
- Each request contains all of the data necessary to complete itself successfully.

Importance of statelessness:

- In stateless API, the server does not need to manage any session, deploying the services to any number of servers is possible, and so scalability will never be a problem.
- Seamless implementation with HTTP protocols is possible as HTTP is itself a stateless protocol

f. Explain the concept of resource identification in RESTful APIs and how it's reflected in URL design.

- RESTful APIs use URLs and Global IDs to identify a resource.
- By using URLs and Global IDs the REST client can easily access the resources from the RESTful API servers.
- The reflection of identifying resource using RESTful API is expressed by giving you the format of an URL identifying a resource:
  http(s)://{Domain name (:Port number)}/{A value indicating REST API}/{API version}/{path for identifying a resource}

Client-Server Communication:

a. How do clients and servers communicate in a client-server architecture?

Client:

The client sends **request** to the server for a service or an information.

Server:

The server receives the request from the client and processes the request then, it sends the **response** to the client as a service or an information.

b. What are some common communication protocols used for client-server interactions?

The most common communication protocol used for client-server interaction is **TCP/IP** protocol suite.

c. Describe the steps involved in making a typical request from a client to a server and receiving a response.

Steps involved in client-server communication:

1. Client sends an HTTP **request** to the server.
2. Server **receives** the request.
3. Server **process** the request.
4. Server returns an HTTP **response** to the Client.
5. Client **receives** the response.

d. What is the role of HTTP headers in client-server communication?

- HTTP headers are used to **pass additional information** between the clients and the server through the request and response header.
- HTTP headers consists of its **case-insensitive** name followed by **key-value pairs** which are separated by colon.
- Whitespace before the value is ignored.

Web Services:

a. What are web services, and how do they relate to RESTful APIs?

- Web service is a collection of standards or protocols for exchanging information between two devices or application.
- It is a method of communication between two devices over the network.
- REST API is a standardized architecture style for creating a Web Service API.
- One of the requirements to be a REST API is the utilization of HTTP methods to make a request over a network.

b. What differentiates SOAP (Simple Object Access Protocol) web services from RESTful web services?

- **SOAP** is a protocol, it follows a strict standard to allow communication between the client and the server.
- **RESTful API** is an architectural style that doesn't follow any strict standards.
- **SOAP** transports data in standard XML format.
- **RESTful API** transports data in JSON.
- **SOAP** works with WSDL.
- **RESTful API** works with GET, POST, PUT, DELETE.

c. What are the advantages of using RESTful web services over other types of web services?

Advantages of RESTful web services:

- Scalable
- Uniform Interface
- Cacheable
- Independence and Modularity
- Uses standard HTTP methods
- Flexibility and Compatible
- Efficiency

Data Formats:

a. **What are the commonly used data formats for exchanging data in RESTful APIs?**
   - The two most common data exchange formats are **JSON and XML**.
   - Many RESTful web services can use both formats interchangeably.

b. **Compare and contrast JSON and XML as data interchange formats in the context of RESTful APIs.**

   JSON
   - It stands for JavaScript Object Notation.
   - It is based on JavaScript language.
   - It is a way of representing objects.
   - It supports array.

   XML
   - It is Extensible markup language
   - It is derived from SGML.
   - It is a markup language and uses tag structure to represent data items.
   - It doesn't supports array.

c. **When would you choose JSON over XML or vice versa?**

   JSON over XML:
   - JSON requires less tags than XML.
   - JSON is easier to read than XML.
   - JSON is transportation-independent.
   - You can get JSON data from anywhere.

   XML over JSON:
   - XML supports complex datatypes.
   - Used to separate data from presentation.
   - Structure of the data is embedded within the data itself.

Security Considerations:

a. What security mechanisms can be employed to secure a RESTful API?

Security mechanisms employed to secure a RESTful API are:

- HTTP/TLS
- authentication and authorization
- SSL/TLS encryption
- Restrict access to sensitive data.

b. How does authentication work in a RESTful API? What are some common authentication methods?

- Web API provides a built-in authorization filter, AuthorizeAttribute.
- This filter checks whether the user is authenticated or not.
- If not, it returns HTTP status code 401 (Unauthorized),without invoking the action.

Common authentication methods:

- Username & Password Authentication.
- JWT Authentication.
- OAuth2 Authentication.
- Token Authentication.

c. Explain the concept of CORS (Cross-Origin Resource Sharing) and how it affects API security.

CORS:

- Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism
- It allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources.

How it affects API security:

- The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers.
- Modern browsers use CORS in APIs such as XMLHttpRequest or Fetch to mitigate the risks of cross-origin HTTP requests.