1. What is a Microservice?

* Microservices are an architectural approach to developing software applications as a collection of small, independent services that communicate with each other over a network.
* Instead of building a monolithic application where all the functionality is tightly integrated into a single codebase, microservices break down the application into smaller, loosely coupled services.

-------------------------------------------------------------------------------------------------------------------------

1. What is Web Service?

* **A web service is a software module that helps different applications communicate over a network (like the internet).**
* **It performs specific tasks and sends or receives data between a client and a server.**
* **Web services use standard protocols so that apps built in different languages or platforms can share data easily.**

**-----------------------------------------------------------------------------------------------------------**

1. What are Cookies?

* A **cookie** is a small message sent from a **web server to a user's browser** when visiting a website.
* It is a **small text file** created or updated by the website and **stored in the user's browser**.
* Cookies store **information like user sessions, preferences, and website data**.
* They help websites **remember users** and **track their activities**.
* This allows websites to offer a more **personalized experience** for each user.

---------------------------------------------------------------------------------------------------------

1. What is Server?

* A **server** is a **hardware device or software** that **processes requests** from other devices over a network.
* It **responds to requests** sent by a client (like a browser or app).
* On the Internet, a server **stores and sends files or data** to clients upon request.
* A server provides **services** like data sharing, resource access, or running applications.
* This setup is part of the **client-server model**, where the server serves, and the client consumes.
* **One server** can handle **multiple clients** at the same time.

**----------------------------------------------------------------------------------------------------------**

1. What is Localhost?

* **Localhost** is a **hostname** that refers to **your own computer** — the one you're currently using.
* It uses the **IP address 127.0.0.1**, also known as the **loopback address**.
* It allows you to **connect to services running on your own machine** without needing an internet connection.
* Commonly used for **testing and development**, like running web applications locally before deploying them online.
* It helps developers check and debug their applications **safely and privately**.

---------------------------------------------------------------------------------------------------------

1. What is Domain?

* A **domain** is a network group of computers, servers, and devices that **share resources and information**.
* A **Domain Controller** manages **user authentication**, **security**, and **access control** within the domain.
* Domains help organize networks by assigning **different access levels** to users and can include **subdomains** for better management.

-----------------------------------------------------------------------------------------------------

1. What is Endpoint in URL?

The **Endpoint URL** is the complete and specific address used to access a particular resource or function within an API. It combines the **Base URL** with a defined path, similar to navigating to a specific section.

-----------------------------------------------------------------------------------------------------------

1. What is the difference HTTP AND HTTPS?

**HTTP (HyperText Transfer Protocol)**

1. URL starts with **http://**.
2. Uses **port 80** for communication.
3. **No encryption** – data is transferred as **plain text**, making it **insecure**.
4. Works at the **Application Layer** of the OSI model.
5. Generally **faster** than HTTPS due to **no encryption overhead**.

**HTTPS (HyperText Transfer Protocol Secure)**

1. URL starts with **https://**.
2. Uses **port 443** for secure communication.
3. **Encrypts data** using **SSL/TLS**, ensuring **privacy and security**.
4. Works at the **Transport Layer**, handling encryption and decryption.
5. Slightly **slower** than HTTP due to **encryption processing**

**----------------------------------------------------------------------------------------------------------------**

1. What is CRUD? and which methods are used for this and give some details about those methods.

**CRUD** stands for the four basic operations you can perform on data in a database or through an API:

* **C** – Create
* **R** – Read
* **U** – Update
* **D** – Delete

These operations are commonly used in web development, databases, and RESTful APIs to manage data effectively.

1. **POST** (Create):
   * Sends data to the server to **create a new resource**.
   * Example: Creating a new user account.
2. **GET** (Read):
   * Requests data from a specified resource **without making any changes**.
   * Example: Viewing a list of products.
3. **PUT** / **PATCH** (Update):
   * **PUT**: Replaces **entire resource** with new data.
   * **PATCH**: Updates **partial fields** of a resource.
   * Example: Changing a user's email address.
4. **DELETE** (Delete):
   * Removes a specific resource from the server.
   * Example: Deleting a blog post.
5. What is Payload, Header, Status code?

**Payload**

* The **payload** is the **actual data** sent in an **HTTP request or response**.
* In a **request**, it's the **body** you send (e.g., JSON data in a POST request).
* In a **response**, it’s the **data returned** by the server.
* Example: In a login API, your username and password sent to the server are the **payload**

**Header**

* **Headers** carry **metadata** about the request or response.
* They describe **content type**, **authentication tokens**, **language**, etc.
* Headers are sent in both requests and responses.
* Example:
  + Content-Type: application/json
  + Authorization: Bearer <token>

**Status Code**

* The **status code** is a **3-digit number** returned by the server.
* It shows the **result** of the request (success, error, etc.).
* Common status codes:
  + 200 OK – Successful request
  + 201 Created – Resource created
  + 400 Bad Request – Client-side error
  + 401 Unauthorized – Authentication required
  + 500 Internal Server Error – Server-side issue

-----------------------------------------------------------------------------------------------------------------------

1. What is Load Balancer?

* A **load balancer** is a device or software that **distributes incoming network traffic** across multiple servers to ensure **high availability and performance**.
* It acts like a **traffic controller**, routing client requests so that **no single server is overloaded**.
* Commonly used in **cloud computing, data centers, and large-scale web apps** to manage traffic efficiently.

1. What is Client and Server?

**Client**

* A **client** is a **device or program** (like a web browser or mobile app) that **sends requests** to a server.
* It **asks for services or data** (like a web page, file, or information).
* Example: When you open a website in Chrome, your browser is the **client**.

**Server**

* A **server** is a **computer or program** that **receives and processes requests** from clients.
* It **provides services or data** back to the client (like delivering a web page or database result).
* Example: A **web server** sends HTML pages when requested by a browser.

1. How API Works
2. **Client Sends Request**
   * Contains **endpoint**, **method** (GET, POST), **headers**, and **data** (payload).
3. **Server Processes Request**
   * Validates input, fetches data, or performs operations.
4. **Server Sends Response**
   * Returns **status code** (200, 404), **headers**, and **response body** (usually JSON).

----------------------------------------------------------------------------------------------------------------

1. How many Protocols are there for API/Network.

| **Protocol Name** | **Used For** | **Example Use** |
| --- | --- | --- |
| **HTTP / HTTPS** | Most common for web APIs | Calling REST APIs like weather info, login |
| **SOAP** | XML-based APIs | Used in banking or government apps |
| **WebSocket** | Live updates in real time | Chat apps, online games |
| **FTP / SFTP** | Sending/receiving files | Uploading a document to a server |
| **SMTP / IMAP / POP3** | Sending/receiving emails | Gmail, Outlook, etc. |
| **TCP / IP** | Basic internet communication | All online apps use this underneath |

1. What is the full form of API?

**Application Programming Interface**

----------------------------------------------------------------------------------------------------------

1. Full form of URL?

Uniform Resource Identifier

1. What is the max length of the URL?

* **2,048 characters** (2 KB) is the **maximum URL length** supported by most browsers (like Google Chrome, Internet Explorer, etc.).
* This includes the full URL: protocol (https), domain name, path, query parameters, etc.

| **Browser** | **Max URL Length** |
| --- | --- |
| Internet Explorer | 2,083 characters |
| Google Chrome | ~2,048 characters |
| Firefox | >65,000 characters (but not recommended) |
| Safari | ~80,000 characters |

1. How or when the IP address update of our mobile or laptop device

 **Changing Networks**: Your IP changes when you switch between Wi-Fi, mobile data, or different networks (e.g., home to office).

 **Router Reboot or IP Lease Renewal**: Most devices get dynamic IPs from the router or ISP, which can change if the router restarts or the lease expires.

 **Using VPNs or Proxies**: When connected to a VPN, your IP address changes to the VPN server's IP, masking your original address.

1. What happen when you type google.com in browser and how you see the google page and every time you time gogole.com in the browser then always same process happen?

Need to do

-----------------------------------------------------------------------------------------------------

1. **What is JSON? Explain its purpose.**

* **JSON** stands for **JavaScript Object Notation**.
* It is a **lightweight data format** used for storing and exchanging data.
* It is written in a **key-value pair** format, similar to how dictionaries or maps work.

**Purpose of JSON:**

1. **Data Exchange:**  
   JSON is commonly used to send data between a **client (browser/app)** and a **server** in web applications.
2. **Readable Format:**  
   It is **easy for humans to read** and **easy for machines to parse** and generate.
3. **Language Independent:**  
   JSON works with many programming languages like Java, Python, JavaScript, etc., making it ideal for APIs and cross-platform communication.
4. **Which data types are supported in JSON? List all.**

[JSON](https://www.geeksforgeeks.org/javascript/what-is-json/) supports mainly 6 data types:  String, Number, Boolean, Null, Object, Array

1. What is the correct JSON format? Give an example.

**JSON (JavaScript Object Notation)** is a lightweight format for storing and exchanging data.  
It is easy for humans to read and write and easy for machines to parse and generate.

Rules :

* Data is in key-value pairs
* Keys and string values must be enclosed in double quotes
* Data is separated by commas
* Curly braces {} represent JSON objects
* Square brackets [] represent JSON arrays

{

"name": "Aananya",

"age": 25,

"email": [aananya@example.com](mailto:aananya@example.com)

}

1. **What is the difference between a JSON object and a JSON array? Give examples.**

| **Feature** | **JSON Object** | **JSON Array** |
| --- | --- | --- |
| **Structure** | Unordered collection of key-value pairs | Ordered collection of values |
| **Enclosed with** | Curly braces {} | Square brackets [] |
| **Accessed by** | Keys | Indexes (starting from 0) |
| **Use Case** | Represents real-world entities (e.g. a person) | Represents lists (e.g. list of names) |

**JSON Object Example:**

{

"name": "John",

"age": 30,

"isStudent": false

}

**JSON Array Example:**

[

"Java",

"Python",

"JavaScript"

]

1. How do we represent null in JSON? Give an example.

In JSON, null is used to represent **no value** or **a missing value**. It is written as the **literal null** (without quotes).

{

"name": "Arun",

"age": null,

"email": null

}

1. **Does JSON allow comments? Explain why.**

* **JSON does not allow comments** because it’s designed to be a lightweight, fast, and consistent data format.
* Allowing comments would break compatibility with strict parsers and add complexity.
* JSON is meant for machines, not for documentation or configuration, so comments are intentionally excluded.

Example : we are using special key ‘\_’

{

"\_comment": "This is a note",

"name": "Alice",

"age": 25

}

1. **What is the default character encoding for JSON?**

The **default character encoding for JSON** is **UTF-8**.

**Key Points:**

* JSON text must be encoded in **Unicode**, and **UTF-8 is the most commonly used and default encoding**.
* Most JSON parsers expect data to be in UTF-8 unless specified otherwise.
* UTF-8 supports all characters in the Unicode standard, making it ideal for international data exchange.

1. Convert this data into JSON format:

Name: Priya

Age: 25

Skills: Java, API

**Answer :**

{

“Name” = “Priya”,

“Age” = “25”,

“Skills” = [“Java”,”API”]

}