**Learn about what is a microservice.**

Ans. A **microservice** is a small, self-contained part of an application that does one specific job and can run on its own.

**🔹 Key Points:**

* **Small & Focused**: Each microservice handles one task (e.g., login, payment).
* **Independent**: Can be built, deployed, and updated separately.
* **Flexible**: Different services can use different tools or languages.
* **Scalable**: You can scale only the parts that need more power.
* **Communicates**: Microservices talk to each other via APIs or messages.

1. **About what is a web service.**

Ans. A **web service** is a way for different computers or applications to communicate with each other over the internet (or a network) using standard web protocols.

**Simple explanation:**

* It’s like a **software helper** that lives on the web.
* It **accepts requests** (usually over HTTP) and **sends back responses**.
* It allows different programs—often written in different languages or running on different systems—to work together.

**Key points:**

* Uses standard protocols like **HTTP**, **SOAP**, or **REST**.
* Sends data usually in formats like **XML** or **JSON**.
* Enables interoperability between applications.

**Example:**

If your app wants to get weather info, it can call a weather web service, which returns the data in a format your app understands.

**What are Cookies?**

Ans. **Cookies** are small pieces of data that websites store on your computer or device when you visit them.

**What do cookies do?**

* Remember who you are (like staying logged in).
* Save your preferences (like language or theme).
* Track your activity to personalize your experience or show ads.

**How do cookies work?**

1. You visit a website.
2. The website sends a cookie to your browser.
3. Your browser saves the cookie.
4. Next time you visit, your browser sends the cookie back to the website.
5. The website uses the cookie to recognize you.

**What is Server?**

Ans. A **server** is a computer or system that provides services, resources, or data to other computers, called clients, over a network.

**Simple explanation:**

* Think of a server as a **helper computer** that listens for requests.
* When a client (like your laptop or phone) asks for something (like a webpage, file, or email), the server sends it back.

**Common types of servers:**

* **Web server**: Delivers websites.
* **File server**: Shares files.
* **Mail server**: Manages emails.
* **Database server**: Stores and manages data.

**How it works:**

1. Client sends a request (e.g., open a webpage).
2. Server processes the request.
3. Server sends the response (e.g., webpage data).

**What is LocalHost?**

Ans. **Localhost** is a special name that means **“this computer”** or **your own device** when talking about networks.

**Simple explanation:**

* When you type **localhost** in your browser or use it in a program, you’re telling it to connect to a server running on **your own computer**.
* It usually points to the IP address **127.0.0.1**, which is the loopback address.

**Why use localhost?**

* Developers use localhost to **test websites or apps** on their own machines before making them live.
* It helps run servers locally without needing the internet.

**Example:**

If you run a web server on your laptop, typing [**http://localhost**](http://localhost) in a browser will show the website hosted on your own machine.

**What is Domain?**

Ans. A **domain** is the human-friendly address you type into a web browser to visit a website.

**Simple explanation:**

* Instead of remembering a long IP address like 192.168.1.1, you use a **domain name** like example.com.
* Domains make it easy to find websites on the internet.

**How it works:**

* When you enter a domain in your browser, the **Domain Name System (DNS)** translates it into the website’s IP address.
* Your browser then connects to that IP to load the website.

**Example:**

* google.com
* facebook.com
* openai.com

**What is Endpoint in the URL?**

Ans. An **endpoint** in a URL is the specific address where a web service or API can be accessed.

**Simple explanation:**

* It’s like the **exact door or location** on a server where you send a request.
* The endpoint tells the server **what resource or action** you want.

**Example URL:**

https://api.example.com/users/123

* https://api.example.com = server address (base URL)
* /users/123 = **endpoint** — here, it might mean “get info about user with ID 123”

**What is the Difference between HTTP and HTTPS?**

Ans.

| **Feature** | **HTTP** | **HTTPS** |
| --- | --- | --- |
| **Full form** | HyperText Transfer Protocol | HyperText Transfer Protocol Secure |
| **Security** | Not secure — data is sent **plain** (unencrypted) | Secure — data is **encrypted** using SSL/TLS |
| **Data Privacy** | Data can be intercepted or stolen | Data is protected from eavesdropping |
| **Port used** | 80 | 443 |
| **Use case** | Websites without sensitive info | Websites with sensitive info (login, payments) |
| **URL prefix** | http:// | https:// |
| **Trust indicator** | No padlock icon in browser | Padlock icon in browser address bar |

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

Ans. **CRUD** stands for the four basic operations you can do on data in an application:

* **C**reate — Add new data
* **R**ead — Retrieve or view data
* **U**pdate — Modify existing data
* **D**elete — Remove data

**HTTP Methods Used for CRUD:**

| **CRUD Operation** | **HTTP Method** | **Description** |
| --- | --- | --- |
| **Create** | **POST** | Sends new data to the server to create a resource. Example: Adding a new user. |
| **Read** | **GET** | Requests data from the server. Example: Fetching a list of users. |
| **Update** | **PUT / PATCH** | Changes existing data on the server. - PUT: Replaces the entire resource.- PATCH: Modifies part of the resource. Example: Changing a user’s email. |
| **Delete** | **DELETE** | Removes data from the server. Example: Deleting a user account. |

**What is Payload, Header, Status code?**

Ans. **1. Payload**

* The **payload** is the actual data sent in an HTTP request or response.
* For example, when you submit a form, the form data is the payload.
* In responses, payload often contains the requested information (like JSON data).

**2. Header**

* **Headers** are extra information sent along with HTTP requests or responses.
* They include details like content type, authorization, cookies, and more.
* Headers help the server and client understand how to process the data.

**3. Status Code**

* A **status code** is a 3-digit number sent by the server in response to a client’s request.
* It tells if the request was successful, failed, or something else.

**Common examples:**

* 200 — OK (success)
* 404 — Not Found (resource missing)
* 500 — Server Error

**What is Load Balancer? (Research about this)**

Ans. A **load balancer** is a device or software that **distributes incoming network traffic** across multiple servers.

**Why use a Load Balancer?**

* To **spread the workload** evenly so no single server gets overwhelmed.
* To improve **performance** and **speed**.
* To increase **reliability** by redirecting traffic if a server fails.

**How it works:**

1. Users send requests to a single entry point (the load balancer).
2. The load balancer forwards each request to one of several backend servers.
3. Servers process the requests and send responses back through the load balancer.

**What is Client and Server?**

Ans. **Client:**

* The **client** is the device or program that **requests services or resources** from another computer.
* Examples: your web browser, mobile app, or any device you use to access websites or apps.

**Server:**

* The **server** is the computer or program that **provides services or resources** to clients.
* It waits for requests from clients, processes them, and sends back responses.
* Examples: web servers hosting websites, email servers managing emails.

**How they work together:**

* The client sends a request (like asking for a webpage).
* The server receives the request, processes it, and sends back the data.
* This communication happens over a network (like the internet).