Q1. What is an API? Give an example, where an API is used in real life.

An **API (Application Programming Interface)** is a set of **rules and protocols** that allows different software applications to communicate with each other. It defines how requests and responses should be structured, enabling seamless interaction between systems.

APIs can be used to retrieve or send data, integrate services, or automate tasks between different applications.

One common example of an API in real life is the **Google Maps API**:

* Many apps (such as Uber, Zomato, or travel websites) integrate Google Maps API to show locations, calculate distances, and provide navigation.
* The app **sends a request** to the Google Maps API, which then **returns data** like maps, routes, and traffic conditions.

Q2. Give advantages and disadvantages of using API.

1.**Automation**

* APIs enable systems to interact without human intervention, improving efficiency.
* Example: Automating data transfers between apps (e.g., syncing Google Calendar with a scheduling app).

2.**Integration**

* APIs allow different applications to connect and work together.
* Example: Using a **Payment API** (like PayPal or Stripe) in an e-commerce website.

3.**Scalability**

* APIs support scaling by allowing multiple applications to interact without needing major code changes.
* Example: Cloud-based services (AWS, Google Cloud) provide APIs for storage, computing, and networking.

4.**Security**

* APIs allow controlled access to data by using authentication methods (e.g., API keys, OAuth).
* Example: Google’s OAuth API ensures secure login across multiple websites.

5.**Speed & Efficiency**

* APIs eliminate the need for manual processes by allowing apps to exchange data quickly.
* Example: **Google Maps API** loads maps in apps instantly rather than requiring manual input.

6.**Encapsulation & Modularity**

* APIs allow developers to use services without needing to understand their internal logic.
* Example: Using a **weather API** without building a weather forecasting system.

Q3. What is a Web API? Differentiate between API and Web API.

A **Web API** is an **API that is accessed using HTTP or HTTPS protocols** over the internet. It allows different applications to communicate over the web by sending requests and receiving responses in formats like **JSON or XML**.

Web APIs are commonly used to fetch or send data between a client (like a web browser or mobile app) and a server.

**Example of a Web API:**

* **Weather API**: Fetch real-time weather data from a server.
* **Google Maps API**: Get location details and maps using HTTP requests.
* **Difference Between API and Web API**

| **Feature** | **API (General)** | **Web API** |
| --- | --- | --- |
| **Definition** | A set of rules allowing software components to interact. | A type of API that is accessed via HTTP/HTTPS over the internet. |
| **Communication** | Can be local (within a system) or remote. | Works over the web using HTTP requests. |
| **Data Format** | Can use various data formats (JSON, XML, CSV, etc.). | Mostly uses **JSON** or **XML**. |
| **Access** | Can work offline or within an application. | Requires internet connectivity. |
| **Examples** | OS API, Database API, Library API. | RESTful APIs, SOAP APIs, Google Maps API. |

Q4. Explain REST and SOAP Architecture. Mention shortcomings of SOAP.

REST (**Representational State Transfer**) and SOAP (**Simple Object Access Protocol**) are two architectural styles for building web APIs.

**REST Architecture**

**Definition:**  
REST is an architectural style that uses **HTTP methods** (GET, POST, PUT, DELETE) for communication between clients and servers. It follows a **stateless** client-server model.

**Key Features:**

* Uses JSON or XML for data exchange (mostly JSON).
* Works over HTTP/HTTPS.
* Stateless – Each request is independent, and no client session is stored on the server.
* Lightweight and fast – Ideal for web and mobile applications.
* Scalable and flexible – Works well with cloud services.

**Common Use Cases:**

* **Web Services** (Google Maps API, Twitter API, OpenWeather API).
* **Cloud applications and microservices.**

**SOAP Architecture**

**Definition:**  
SOAP is a **protocol** that allows structured messaging between applications over a network. It uses **XML** and operates over various protocols like HTTP, SMTP, and TCP.

**Key Features:**

* Uses **strict XML format** for message structure.
* Supports **stateful and stateless** operations.
* Built-in **security (WS-Security)** and **error handling**.
* Works over multiple protocols (**HTTP, SMTP, TCP**).

**Common Use Cases:**

* **Banking and financial services** (high security and reliability required).
* **Enterprise applications** (e.g., government and healthcare systems).

**Shortcomings of SOAP**

* **Heavy and Complex –** Uses XML, making messages larger and slower.
* **Slower Performance –** More bandwidth usage due to large XML payloads.
* **Less Flexible –** Requires strict contracts (WSDL).
* **Limited Browser Support –** Cannot be easily used with JavaScript-based applications.

Q5. Differentiate between REST and SOAP.

| **Feature** | **REST** | **SOAP** |
| --- | --- | --- |
| **Architecture** | Lightweight, flexible | Strict protocol-based |
| **Data Format** | JSON, XML | Only XML |
| **Speed** | Faster | Slower (heavy XML) |
| **Security** | Uses HTTPS, OAuth, JWT | Built-in WS-Security |
| **Use Case** | Web, mobile apps, microservices | Banking, enterprise apps |