Q1. What is an API? Give an example of real-life use.

API (Application Programming Interface):
An API is a **set of rules and protocols** that allows **one software application to communicate with another**.

- APIs define **how requests and responses are structured**.
- They enable integration between different systems, platforms, or applications.

Real-life example:

- **Weather apps:** Your weather app uses a weather API to get data from a server. The app sends a request like `GET /weather?city=London` and receives the temperature and forecast.

Other examples: Google Maps API, Twitter API, PayPal API.

Q2. Advantages and Disadvantages of Using API

Advantages:

1. **Integration**: Enables apps to communicate and share data easily.
2. **Automation**: Reduces manual data handling.
3. **Scalability**: Backend services can evolve without breaking the client.
4. **Reusability**: The same API can be used across multiple platforms.

Disadvantages:

1. **Dependency**: Your app relies on a third-party API being available.
2. **Security risks**: APIs can be exploited if not secured properly.
3. **Rate limits**: Many APIs restrict the number of requests.

4. **Maintenance overhead**: Updates to APIs may require changes in your code.

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

Web API:

- A **Web API** is an API that is accessible over the **internet using HTTP/HTTPS**.
- Typically returns data in **JSON or XML format**.

Difference Table

Feature	API	Web API
Communication	Can be local or remote	Always over the web (HTTP/HTTPS)
Protocol	Any protocol (function calls, libraries)	HTTP/HTTPS
Data format	Any (binary, objects)	Usually JSON or XML
Example	Python `math` module API	Twitter API, Google Maps API

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

REST (Representational State Transfer)

- Architecture style for **Web APIs**.
- Uses standard **HTTP methods**:
    - GET → Retrieve data
    - POST → Create data

    - PUT → Update data
    - DELETE → Delete data
- Data format: **JSON (most common), XML**.
- Stateless: Each request contains all necessary info.

SOAP (Simple Object Access Protocol)

- Protocol for exchanging **structured XML messages** over HTTP, SMTP, or others.
- Strict standards: WSDL defines service structure.
- Supports built-in **security and transaction protocols**.

Shortcomings of SOAP:

1. Complex and heavy due to XML formatting.
2. Slower performance compared to REST.
3. Harder to integrate with web/mobile apps.
4. Requires strict contracts (WSDL), making it less flexible.

Q5. Difference between REST and SOAP

Feature	REST	SOAP
Protocol	Architectural style	Protocol
Data format	JSON, XML, YAML (flexible)	XML only
Performance	Lightweight, faster	Heavy, slower
State	Stateless	Can be stateful

Security	Relies on HTTPS	Built-in (WS-Security)
Standards	Less strict	Strict, WSDL required
Usage	Web apps, mobile apps	Enterprise apps requiring high security
Ease of use	Simple and easy	Complex