**1. API (Application Programming Interface)**

**An API is an interface that allows two software applications to communicate with each other. It defines the rules and conventions for interaction. APIs can be private (used internally within a company) or public (accessible by third-party developers).**

**Key Features:**

* **Communication Bridge: Allows software systems to share data and functionality.**
* **Abstraction: Simplifies complex operations, exposing only necessary details to developers.**
* **Examples:**
  + **Google Maps API: Allows developers to embed Google Maps into their applications.**
  + **Twitter API: Lets developers interact with Twitter data (e.g., tweets, followers).**

**Types of APIs:**

1. **REST API (Representational State Transfer):**
   * **Follows the HTTP protocol.**
   * **Stateless: Each request is independent and contains all the necessary information.**
   * **Uses standard HTTP methods (GET, POST, etc.).**
2. **SOAP API (Simple Object Access Protocol):**
   * **More rigid and protocol-heavy than REST.**
   * **XML-based format for sending requests and responses.**
3. **GraphQL API:**
   * **Allow clients to request exactly the data they need.**
   * **Developed by Facebook to overcome limitations of REST APIs.**
4. **Webhooks:**
   * **APIs that send data to other systems in real time when specific events occur.**

**2. Web Protocol**

**Web protocols are standardized rules for communication on the web.**

**Key Web Protocols:**

1. **HTTP (HyperText Transfer Protocol):**
   * **Foundation of the web.**
   * **Used to fetch resources like HTML pages.**
   * **Stateless: No memory of previous interactions.**
2. **HTTPS:**
   * **Secure version of HTTP.**
   * **Encrypts data using SSL/TLS to protect from eavesdropping.**
3. **WebSocket:**
   * **For real-time, full-duplex communication.**
   * **Commonly used in chat applications and live notifications.**
4. **FTP (File Transfer Protocol):**
   * **Transfers files between a client and server.**
5. **SMTP (Simple Mail Transfer Protocol):**
   * **Protocol for sending email messages.**

**3. JSON File**

**JSON (JavaScript Object Notation) is widely used in web APIs for data exchange because of its simplicity and readability. It represents data as key-value pairs.**

**Why JSON?**

* **Lightweight: Easy to parse and generate.**
* **Language-Independent: Supported by many programming languages.**
* **Readable: Humans and machines can understand it easily.**

**JSON Example for an API Response:**

**{**

**"user": {**

**"id": 101,**

**"name": "Alice",**

**"email": "alice@example.com",**

**"isAdmin": false**

**},**

**"status": "success",**

**"message": "User details retrieved successfully."**

**}**

**4. HTTP Status Codes**

**HTTP status codes indicate the result of the client’s request to the server.**

**Common Status Codes:**

**1xx Informational:**

* **100 Continue: Indicates the request is in progress.**

**2xx Success:**

* **200 OK: Request succeeded.**
* **201 Created: Resource created successfully.**

**3xx Redirection:**

* **301 Moved Permanently: Resource has been moved to a new URL.**
* **304 Not Modified: Cached version of the resource can be used.**

**4xx Client Errors:**

* **400 Bad Request: Request contains invalid syntax.**
* **401 Unauthorized: Authentication required.**
* **403 Forbidden: Access is not allowed.**
* **404 Not Found: Resource not found.**

**5xx Server Errors:**

* **500 Internal Server Error: Server encountered an error.**
* **503 Service Unavailable: Server is temporarily unable to handle the request.**

**5. HTTP Methods**

**HTTP methods define the type of operation you want to perform on a resource.**

**HTTP Methods in Detail:**

1. **GET:**
   * **Fetch data from a server.**
   * **Does not alter data on the server.**
   * **Example: Retrieving a user's details.**

**GET /users/101 HTTP/1.1**

**Host: api.example.com**

1. **POST:**
   * **Send data to the server to create a new resource.**
   * **Example: Creating a new user.**

**POST /users HTTP/1.1**

**Host: api.example.com**

**Content-Type: application/json**

**{**

**"name": "Alice",**

**"email": "alice@example.com"**

**}**

1. **PUT:**
   * **Update an existing resource or create it if it doesn’t exist.**
   * **Example: Updating user details.**

**PUT /users/101 HTTP/1.1**

**Content-Type: application/json**

**{**

**"name": "Alice Updated",**

**"email": "alice.updated@example.com"**

**}**

1. **DELETE:**
   * **Remove a resource from the server.**
   * **Example: Deleting a user.**

**DELETE /users/101 HTTP/1.1**

1. **PATCH:**
   * **Partially update a resource.**
   * **Example: Changing only the email address of a user.**

**Example of How These Concepts Work Together**

**Imagine you want to develop a weather app. Here's how these concepts come together:**

* **API: Use a weather API like OpenWeatherMap.**
* **Web Protocol: Communicate over HTTPS.**
* **HTTP Methods: Use GET to fetch weather data.**
* **JSON File: The server responds with weather data in JSON format.**
* **HTTP Status Codes: The server returns 200 OK if the data is retrieved successfully.**

**Example Interaction**

**Request:**

**GET /weather?q=London&appid=your\_api\_key HTTP/1.1**

**Host: api.openweathermap.org**

**Response:**

**{**

**"city": "London",**

**"temperature": "15°C",**

**"description": "Partly cloudy",**

**"humidity": 67,**

**"status": "success"**

**}**