**REST (Representational State Transfer)** is an architectural style for designing networked applications. A **REST API** (Application Programming Interface) allows different systems to communicate over the web using standard HTTP methods. REST APIs are widely used to expose the functionality of a backend system to frontend applications, mobile apps, or other external systems.

**Key Principles of REST:**

1. **Stateless**
   * Each request from a client to a server must contain all the necessary information to process the request.
   * The server does not store any client context between requests.
2. **Client-Server Architecture**
   * The client (frontend) and server (backend) are separate entities. The client sends requests, and the server responds with the requested data or performs an operation.
3. **Resource-Based**
   * Everything in a REST API is considered a **resource** (e.g., users, orders, products).
   * Resources are identified using **URIs (Uniform Resource Identifiers)**.
4. **Uniform Interface**
   * REST uses standard HTTP methods and response codes to maintain a consistent communication interface.
5. **Stateless Cacheable**
   * Responses can be cached by clients to improve performance.
6. **Layered System**
   * REST APIs can be layered, allowing intermediaries (e.g., load balancers or proxy servers) to handle additional tasks like security or caching.