**2. Basics of REST APIs**

**1. What is REST (Representational State Transfer)?**

• **Theory**

1. **Overview of REST principles: statelessness, resource-based URLs, use of HTTP methods (GET, POST, PUT, DELETE), and status codes:-**

1. **Statelessness**

* Each request from a client to the server **must contain all the information** needed to process the request.
* The server **does not store any client session data** between requests.

2. **Resource-Based URLs**

* RESTful APIs use **unique URLs to represent resources** (data objects).
* Example:
  + GET /users/1 → Fetch user with ID 1
  + DELETE /products/5 → Delete product with ID 5

3. **Use of HTTP Methods**

* **GET** → Retrieve data (Read)
* **POST** → Create a new resource
* **PUT** → Update an existing resource
* **DELETE** → Remove a resource

4. **Status Codes**

* **200 OK** → Request was successful
* **201 Created** → New resource was successfully created
* **400 Bad Request** → Invalid request from the client
* **401 Unauthorized** → Authentication required
* **404 Not Found** → Resource does not exist
* **500 Internal Server Error** → Server-side issue

**2.Key REST concepts: ♣ Resources: Everything is treated as a resource. ♣ URI: Uniform Resource Identifiers for identifying resources. ♣ Stateless Communication: Each request from a client to the server must contain all the information needed to understand and process the request:-**

1. Resources

* In REST, everything is treated as a resource, such as users, orders, and products.
* Resources are represented using JSON or XML format.

2. URI (Uniform Resource Identifier)

* Each resource is identified by a unique URI (Uniform Resource Identifier).
* Example:
  + /users/1 → Refers to user with ID 1
  + /products/5 → Refers to product with ID 5

3. Stateless Communication

* Every request from the client must contain all the necessary information for the server to process it.
* The server does not store client session data between requests.
* **This improves scalability and reduces dependency on the server.**