Lesson 3 -> Rest API

REST stands for Representational State Transfer.

A REST API is a way for **systems to communicate over HTTP** by exposing **resources** and allowing clients to **perform operations on those resources** using standard HTTP methods.

Key idea:
You don’t call functions.
You **operate on resources**.

Example resource:

- User
- Order
- Product
- Recommendation

Each resource is identified by a **URL**.

How to think about REST (simple mental model):

- URL → *What* you are operating on
- HTTP method → *What action* you want to perform
- Body → *Data* you send (if any)
- Response → *Representation* of the resource

Example:

If you see:
`/users/123`

That means:
“User with ID 123”

What you do to it depends on the HTTP method.

Now, **core principles of REST (important but simple)**

1. Stateless
   Each request contains all the information needed.
   Server does not remember client state between requests.
2. Resource-based
   APIs are designed around **nouns**, not verbs.

Good:
`/orders/456`

Bad:
`/getOrderDetails`

3. Uniform interface
   Same HTTP methods behave consistently across resources.

Now, **HTTP methods used in REST APIs**

These methods define **intent**.

---

1. GET – Read data

Purpose:

- Fetch a resource
- No data modification

Example:

- Get user details
- Get product list
- Get recommendations

Request:
GET /users/123

Properties:

- Safe (no side effects)
- Idempotent (calling it multiple times gives same result)

Used heavily in:

- Dashboards
- Recommendation fetch

- Search APIs

---

2. POST – Create something or trigger processing

Purpose:

- Create a new resource
- Trigger a computation

Example:

- Create a new order
- Submit training data
- Trigger batch inference

Request:
POST /orders

Properties:

- Not idempotent
- Often changes system state

In ML systems:
POST /predict
POST /train
POST /features

---

3. PUT – Replace an entire resource

Purpose:

- Update or replace a resource completely

Example:

- Update user profile fully

Request:
PUT /users/123

Important:
Client sends the **full updated representation**, not partial.

Properties:

- Idempotent

---

4. PATCH – Partial update

Purpose:

- Update only specific fields

Example:

- Update user address
- Update model threshold

Request:
PATCH /users/123

Properties:

- Not necessarily idempotent
- More efficient than PUT

Very common in modern APIs.

---

5. DELETE – Remove a resource

Purpose:

- Delete a resource

Example:

- Delete user
- Remove experiment

Request:
DELETE /users/123

Properties:

- Idempotent
- No body usually

---

6. HEAD – Metadata only

Purpose:

- Same as GET but without response body

Used for:

- Health checks
- Cache validation

---

7. OPTIONS – Discover capabilities

Purpose:

- Ask server which methods are allowed

Used for:

- CORS preflight checks

Common in browsers, rarely called manually.

Summary table (conceptual):

GET → Read
POST → Create / Trigger
PUT → Replace
PATCH → Partial update
DELETE → Remove
HEAD → Metadata
OPTIONS → Capabilities

REST APIs in microservices and ML systems:

In a real ML system:

- API Gateway receives request
- Routes to:
  - User service
  - Product service
  - ML inference service

Example flow:
Client → GET /recommendations?user_id=123
→ Recommendation service returns ranked items

Key REST benefits here:

- Loose coupling
- Language-agnostic
- Easy to scale
- Easy to version

Common REST mistakes (important):

- Using verbs in URLs
  `/getUserData` ❌
- Using GET to modify state
  GET /updateScore ❌
- Overloading POST for everything
  Common but not ideal