Introduction: What is a RESTful API?

- **Definition:** A **RESTful API** is a way for different apps and websites to talk to each other using the internet.
- **Example:** When you open a weather app, it asks a RESTful API for weather data and shows it to you.

A. Why Are RESTful APIs Important?

- They let apps share information (e.g., YouTube shows TikTok videos).
- They make websites and apps work faster and better.
- Many apps use RESTful APIs to get information (like Google Maps or Facebook login).

B. Important Terms

Word	What It Means (Simple Explanation)
API	A way for apps to talk to each other.
REST	A set of rules that helps APIs work smoothly.
Client	The device or app asking for information (your phone, browser).
Server	The place that stores and gives back the requested information.
Endpoint	A specific address (URL) where the API sends or gets data.
Request	The action the client sends (like asking for a webpage).
Response	The answer the server gives (like showing a webpage).
Stateless	Each request is handled separately, no memory of past requests.

What is HTTP and HTTPS?

A. What is HTTP?

- **HTTP (Hypertext Transfer Protocol)** is the basic system used to send and receive information on the web.
- It allows you to open websites, send messages, and load pictures.

B. What is HTTPS? (The "S" Means Secure)

- **HTTPS (Secure HTTP)** is the safer version of HTTP.
- It **encrypts** (hides) your data so hackers can't see it.
- **Used for:** Banking apps, online shopping, logging into accounts.

C. Comparing HTTP vs. HTTPS

Feature	HTTP	HTTPS
Security	Not secure	Encrypted and safe
Use Case	Simple websites	Login pages, payments
Lock Icon	No	Yes, in browser bar

How Does a RESTful API Work?

A. The Request-Response Process

1. The **Client (You)** asks for information.
2. The **API (Middleman)** sends the request to a **Server**.
3. The **Server (Computer)** finds the information and sends it back.

Example:

- You open YouTube → YouTube's API asks the server for videos → You see the videos.

B. What is an API Request?

- An API request is like asking a waiter for food.
- You tell the API what you want, and it delivers the correct data.

Example of an API request (simplified):

```
GET /weather?city=NewYork
```

Explanation: "Give me the weather for New York!"

Common HTTP Methods in RESTful APIs

APIs use different "verbs" (actions) to do things, like GET and POST.

Method	What It Does	Example
GET	Get information	"Show me the weather in my city."
POST	Send new data	"Add my new TikTok video."
PUT	Change existing data	"Update my username to 'CoolGamer'."
DELETE	Remove something	"Delete my old photo from Instagram."

API Responses: What Comes Back?

When you send a request, the API gives you a **response**.

Example API Response (in simple JSON format):

json

```
{
  "temperature": "25°C",
  "condition": "Sunny"
}
```

- The **temperature** is **25°C**.
- The **condition** is **Sunny**.

Common HTTP Status Codes

APIs send status codes to tell if a request was successful or not.

Code	What It Means	Example
200 OK	Everything is fine.	The API found the weather data.
201 Created	Something new was added.	You uploaded a new profile picture.
400 Bad Request	Your request was wrong.	You typed something incorrectly.
404 Not Found	The resource doesn't exist.	Trying to load a deleted post.
500 Server Error	The API is broken.	The app crashes due to a bug.

Task 01. Consume data from an API using command line tools (curl)

Introduction: What is `curl`?

- **Definition:** `curl` (Client URL) is a **command-line tool** that helps computers talk to each other using the internet.
- **Why is it useful?**
  - Allows you to **get** or **send** information from websites and APIs.
  - Helps with **testing and debugging APIs**.
  - Works with many internet protocols, like **HTTP, HTTPS, and FTP**.

Example: When you search for the weather, an API sends data back. `curl` can do the same thing from the command line!

What You Need to Get Started

- A computer with **Linux, Mac, or Windows**.
- A working **internet connection**.
- The **`curl` tool installed** (usually already installed on Mac/Linux).

A. How to Install `curl` (if not installed)

Mac/Linux:

```
sudo apt install curl  # Ubuntu/Debian
sudo yum install curl  # CentOS/RHEL
```

- **Windows:**
  - Use **Windows Subsystem for Linux (WSL)** or download `curl.exe` from the official site.

B. Check If `curl` is Installed

Run the following command:

```
curl --version
```

Expected Output: Details about the `curl` version installed.

Fetching Data from a Website

Now, let's use `curl` to **download a webpage**!

```
curl http://example.com
```

Expected Output: You will see the **HTML code** of the website.

Fetching Data from an API

We will use a **public API called JSONPlaceholder**, which provides **fake data** for testing.

A. Get a List of Posts

Run this command:

```
curl https://jsonplaceholder.typicode.com/posts
```

Expected Output: A list of posts in **JSON format**, like this:

```
[
  {
```

```
    "userId": 1,
    "id": 1,
    "title": "Example Title",
    "body": "This is a test post."
  },
  ...
]
```

What's happening?

- The API is returning a list of **posts** with a `userId`, `id`, `title`, and `body`.

Fetching Only Headers

Sometimes, we don't need all the data, just some **information about the request** (status code, content type).

Run this command:

```
curl -I https://jsonplaceholder.typicode.com/posts
```

Expected Output:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Cache-Control: max-age=43200
```

What does this mean?

- `200 OK`: The request was successful.
- `Content-Type`: The data is in JSON format.
- `Cache-Control`: Tells browsers how long to store data before updating.

Sending Data to an API (POST Request)

Now, let's **send new data** to the API!

Run this command:

```
curl -X POST -d "title=foo&body=bar&userId=1"
https://jsonplaceholder.typicode.com/posts
```

Expected Output:

json

```
{
  "title": "foo",
  "body": "bar",
  "userId": 1,
  "id": 101
}
```

What's happening?

- `-X POST`: This tells `curl` to **send** data (instead of just getting it).
- `-d "title=foo&body=bar&userId=1"`: This is the **data** we are sending.
- The API responds with a **new post** (but it's only a test, the data isn't saved!).

Understanding API Responses

APIs use **status codes** to tell us if something went wrong.

Code	Meaning	Example
200 OK	The request was successful.	Getting posts from an API.
201 Created	A new resource was created.	Creating a new user.
400 Bad Request	The request is incorrect.	Sending an empty form.
404 Not Found	The resource doesn't exist.	Looking for a deleted post.
500 Server Error	The API server has a problem.	The website crashes.