DAY 4:

REST / REST API Notes

What is REST?

REST (Representational State Transfer) is an architectural style used to design networked applications.

It defines a set of rules to build web services that allow communication between client and server using HTTP.

In simple words:

REST is a standard way to build APIs so that frontend and backend can communicate easily.

What is a REST API?

A **REST API** is an API that follows REST principles and uses:

- HTTP Methods
- URLs (Endpoints)
- JSON Data

Example Endpoints:

```
GET    /notes

POST   /notes

PATCH  /notes/1

DELETE /notes/1
```

HTTP Methods in REST API

🔹 1. GET

Used to fetch data.

Example:

GET /notes

➡ Returns all notes.

🔹 2. POST

Used to create new data.

Example:

POST /notes

Request Body:

```
{
  "title": "REST API",
  "description": "Learning backend development"
}
```

🔹 3. PUT

Used to update entire resource.

Example:

PUT /notes/1

🔹 4. PATCH

Used to partially update a resource.

Example:

PATCH /notes/1

Difference:

- PUT → Replace full data
- PATCH → Update specific fields only

🔹 5. DELETE

Used to delete a resource.

Example:

DELETE /notes/1

REST API Structure

A REST API contains:

- Base URL → http://localhost:3000
- Resource → /notes
- Parameter → /notes/:id

Example:

http://localhost:3000/notes/1

Here:

- notes → Resource
- 1 → Parameter (ID)

REST Principles

1️⃣ Stateless

Each request is independent.
Server does not store client session.

2️⃣ Client-Server Architecture

Frontend and backend are separate.

3️⃣ Uniform Interface

Uses standard HTTP methods.

4️⃣ Data Format

Mostly uses JSON.

Status Codes in REST

Status Code	Meaning
200	OK (Success)
201	Created
400	Bad Request
404	Not Found
500	Internal Server Error

Example:

```
res.status(201).json({
  message: "Note created successfully"
});
```

Express Example (Notes API)

```
app.post('/notes', (req, res) => {
  notes.push(req.body);
  res.status(201).json({ message: "Note created" });
});

app.get('/notes', (req, res) => {
  res.status(200).json(notes);
});

app.delete('/notes/:id', (req, res) => {
  const id = parseInt(req.params.id);
  notes.splice(id, 1);
  res.status(200).json({ message: "Note deleted" });
});
```

Real-Life Analogy

Think of REST API like a restaurant:

- GET → View menu
- POST → Place order
- PATCH → Modify order
- DELETE → Cancel order

Conclusion

REST API is a standard and widely used way to build backend services.
It allows client and server to communicate using HTTP methods and JSON data.
