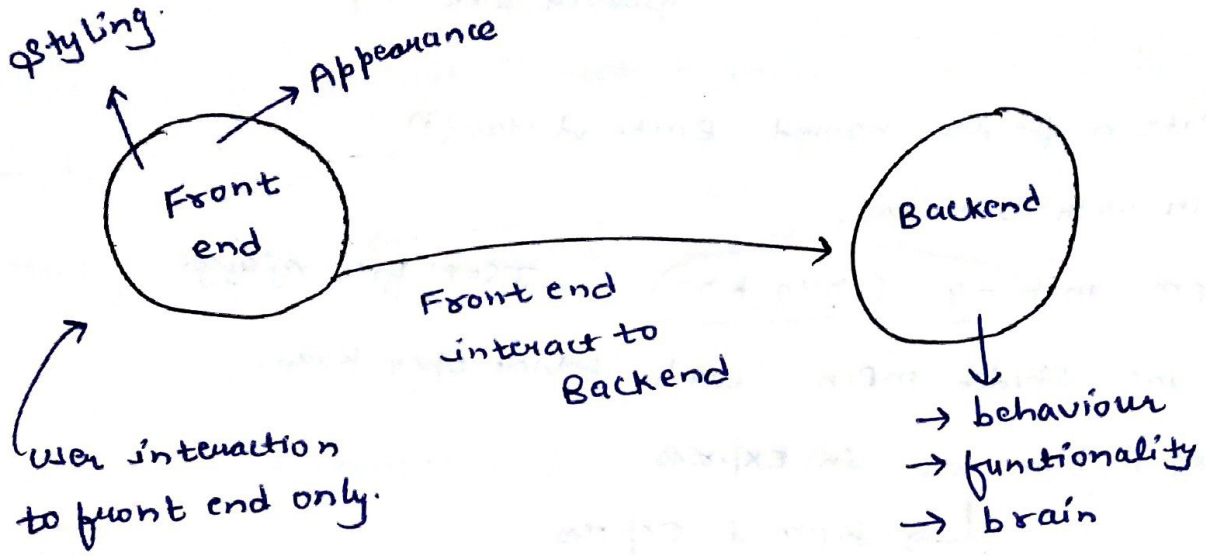
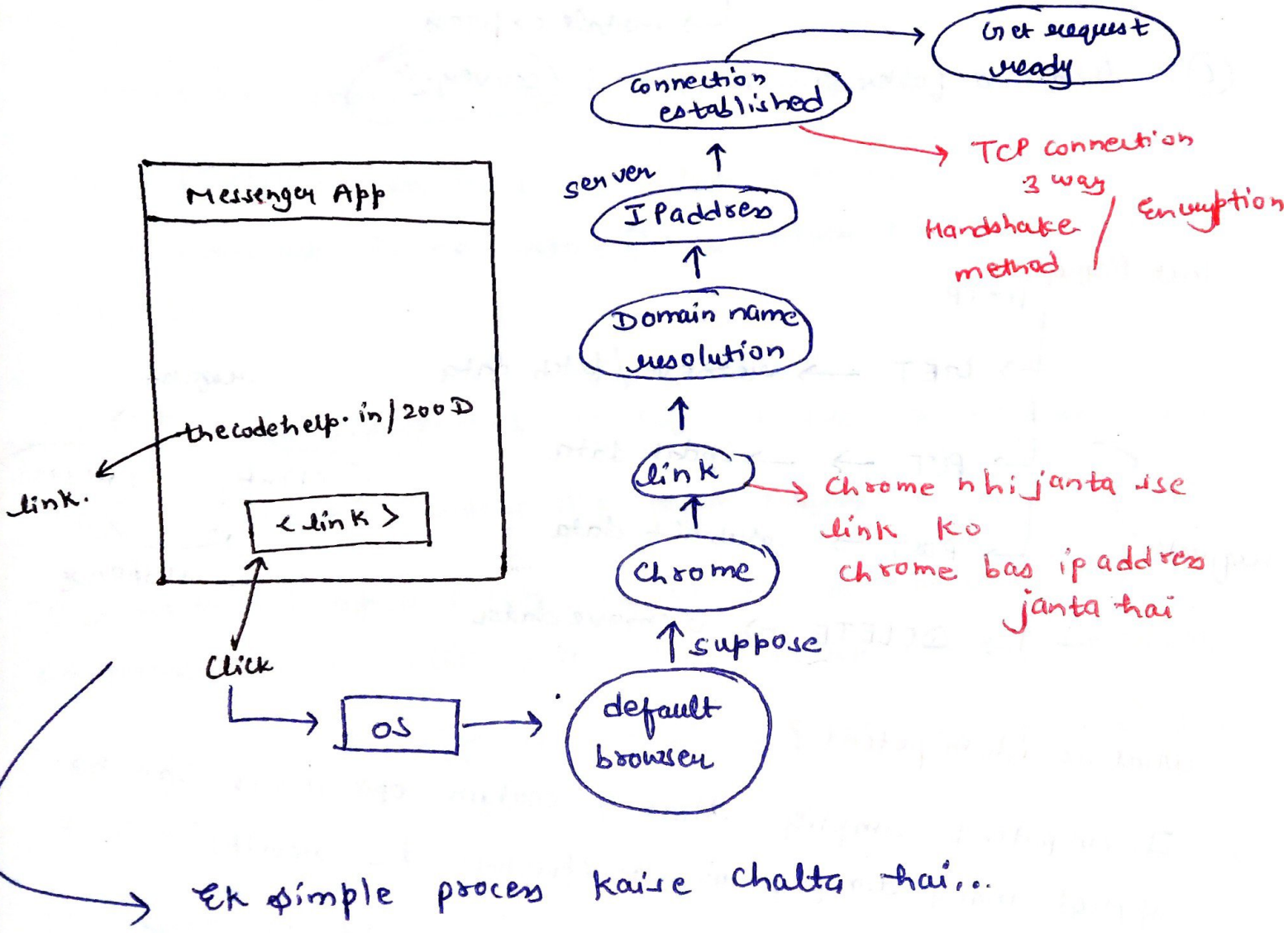
+ Backend by • Batch. 1
by Love Babbar.

Video → 1.





→ Ek simple process kaise chalta hai...

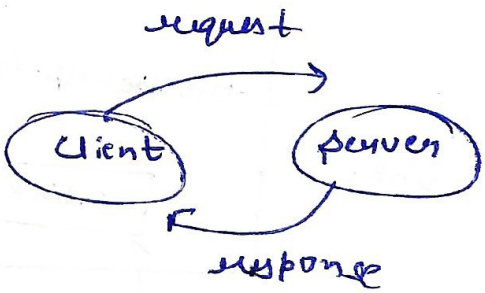
Next target → Server create
↳ using Express JS
↳ ek framework hai used in Server side Application

Steps

① Create a folder named Backend class ①
② Open with terminal.
③ npm init -y (run karo) → JSON file aj jayegi
④ Visual Studio mein wah folder open karo.
⑤ Open terminal in Express
↳ npm i express.
↳ install express
⑥ Create a folder in vs named (server.js)

Get Request

HTTP requests
↳ GET → retrieve / fetch data
↳ PUT → → update data
↳ POST → submit data
↳ DELETE → remove data



What is idem potent?

⇒ Idempotent implies that a certain operations can be applied many times, without changing the result.

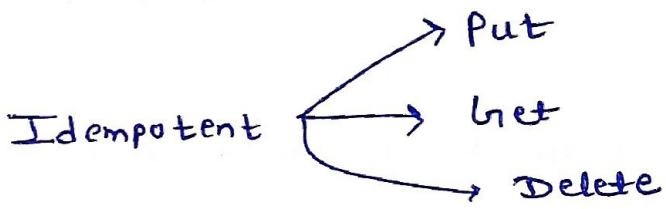
Essentially, it is like multiplying a number by zero.

No matter how many times you multiply a number,

the result will always be 0.

In order to change the result, you need to change the operation.

For exp → if you call "create payment" multiple times, you should still get one payment. That is the type of behaviour that idempotent operation support.



Put

↳ is an idempotent method because it updates a record.

If put/payment endpoint is called with an identical request, it will result in no state change other than the first request.

Delete

↳ is an idempotent method because consecutive similar request wouldn't change the delete request.

The first call of a Delete may return a 200 (OK) but additional Delete calls will likely return a 404 (Not Found).

The response is different after the first request but there is no response change of state.

Get

↳ it is an idempotent method because it is only a read operation and doesn't cause any state change in the Backend

If GET/payment endpoint is called through multiple identical requests, you would get the same response every time just as you got the first time.

POST

Post is not an idempotent method since calling it multiple times may result in incorrect updates.

usually, POST API's create new resources on the server.

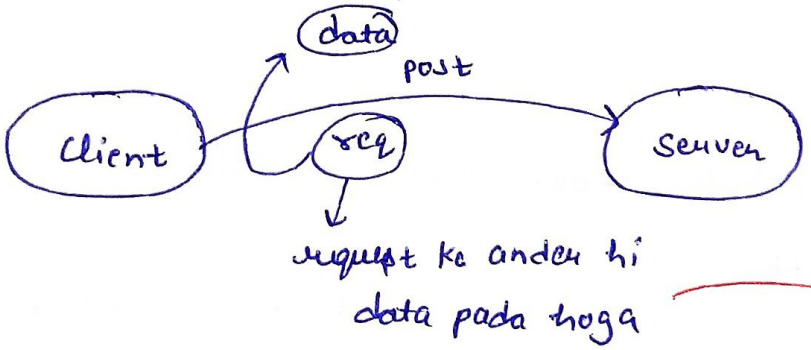
If POST/payment endpoint is called with an identical request body, you will create multiple records. To avoid this, you must have your own custom logic preventing duplicate records.

```
const express = require('express');
const app = express();

app.listen(3000, () => {
console.log("server started at port no 3000");
});

// first route create karte hai

app.get('/', (request, response) => {
response.send("hello jee, kaise ho saare");
})
```

// By doing this it shows on Browser at Localhost:3000

hello jee, kaise ho saare

// ab kuch post krna chahte hai.



request ke ander hi data pada hoga

This will give an error not appropriate way

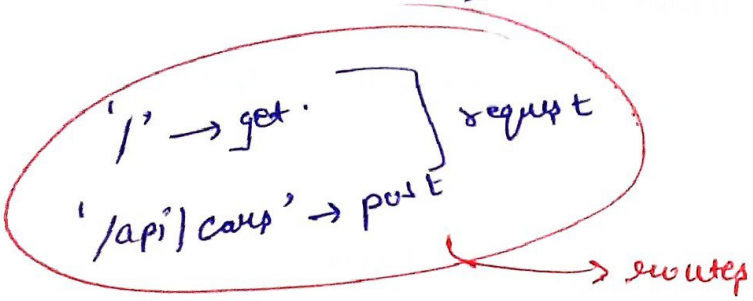
```
app.post('/api/cars', (request, response) => {
const {name, brand} = request.body;
console.log(name);
console.log(brand);
respond.send("car submitted successfully")
})
```

Server Live → yhi target hai
↳ yeh krna hai

① Create your own server
↳ create expres = require ("expres");
const app = expres();

② app.listen (3000, () => {
console.log ("server started");

3)



piche post use kra par wah chala nhi kyunki yhi tarika nhi thai wah isliye thas POSTMAN use krenge

POSTMAN.

Postman is a popular tool that developers use to test
↳ software Application
APIs during the development of web Applications.

When working with Express (a web framework for Node.js) POSTMAN becomes a valuable tool to simulate and test how the server handle HTTP. request.

In simple terms:

- A tool is to send HTTP requests (GET, POST, PUT, DELETE) to your server and view responses.

→ provides a user friendly interface for configuring requests details like URL, method, headers, and body.

Why use POSTMAN with Express

① Send Requests → simulate different http requests to test your Express server.

② Test Endpoints → Ensure API routes work as expected before deployment.

③ Inspect Responses → View status codes and data returned from the server to verify correctness.

Testing Express with POSTMAN:

① Get Request:

- URL = 'http://localhost:3000/
- Method = GET
- Response : { "message": "Hello jee kaise ho" }

② Post Request:

- URL = "http://localhost:3000/api/cars/
- Method = POST
- Response :- "Car submitted successfully"

Body : Raw JSON

```
{ "name": "Model"
  "brand": "Tesla".
}
```

How Postman working

1. Open postman
2. Click on New in file
3. Click on HTTP request
4. Select Post and paste URL
5. Click on Body → raw → Choose json file
6. write data {

   =

   }
7. Send the request.

Benefit of Postman

1. Ease of use
2. Quick Feedback
3. Debugging.

Summary :- Postman is crucial for testing and debugging Express APIs, offering a straightforward way to send requests and analyze responses..

parsing means → fetch, retrieve, get

ham kaise access ka paa rhe thai postman pe by using

URL - "http://localhost:3000/"

Ab baat krte thai.. ek database ki

↳ MongoDB

↳ No-SQL DB.

↳ Store in the form of key-value pair data

↳ document] forms to store data
↳ graphs
↳ etc

Not same as RDMS.

known for scalability and better performance

MongoDB

↳ setup → Mongodb

↳ Mongoshell (Mongosh)

CRUD operations
- Read
- Delete
- Update
- Create

MongoDb compass
↳ a GUI for CRUD operations

H:W → Do Cheatsheets
↳ express & MongoDB.

CRUD operation in compass (MongoDb)

① Create :-

Add Data ▾

Import JSON

Insert document ← click on that → then add data and Insert

By this you can Create Database ←

```
[
  {"name": "audi",
   "brand": "Abhishek"
  },
  {
   "name": "DLICR",
   "brand": "Kaaan"
  }
]
```

② Read :-

By type query above in search box
↓
Key-value Pair

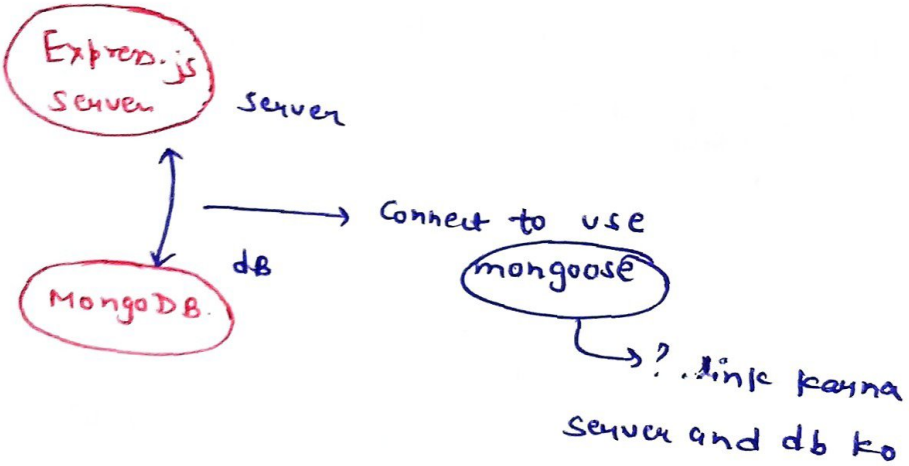
{"name": "Audi"}

↓
click.

③ update :-

↳ By simply click on edit and update the data and save it

④ Delete :- By simply click on delete button to delete the data.



Mongoose :-

→ Mongoose is an object data modeling (ODM) library for MongoDB and Node.js.

→ It provides a straightforward, schema based solution to model your application data.

→ Mongoose is used to define the structure of the documents within your mongoDB collection and provides various methods to interact with the data in a more intuitive way.

ODM → is used for NoSQL database like MongoDB
↳ programming technique used to map and manage the relationship between objects in code and data in a NoSQL.

*

Nodejs → (in dono ke bich ki mapping ko sambhalna use facilitate krna, yeh ek ODM Library hai) → MongoDB

Nodejs ↓ data treat as object form

MongoDB ↓ data treat as Document form

object Data modeling.

Other works of mongoose

- ↳ create schemas
- ↳ create models
- ↳ CRUD operation

Mongoose
- ↳ setup.

setup code

```
Const mongoose = require ('mongoose');

mongouse.connect ('mongodb://localhost:27017/myDatabase Sample', {
        useNewURLParser : true,
        useUnified Topology : true
    })
  .then(() => {
    Console.log ("Connection Successful");
    })
  .then (() => {
    Console.log ("Connection Successful");
})
  .catch ((error)) => {
    Console.log("Received an error", error);
  });
```