**✅ Most Commonly Used HTTP Status Codes (with Hinglish Explanation)**

**2xx – Success (Sab kuch sahi hai)**

* **200 OK – Request sahi thi, server ne response bhej diya. (Normal success)**
* **201 Created – Naya resource ban gaya, jaise new user create. (POST ke baad)**
* **204 No Content – Sab kuch sahi hua, par server ke paas kuch return karne ko nahi tha.**

**3xx – Redirection (Dusri jagah bhejna pad raha hai)**

* **301 Moved Permanently – Resource ka address badal gaya permanently, ab naye URL pe chala jao.**
* **302 Found – Temporary redirect, abhi ke liye dusre URL pe jao.**
* **304 Not Modified – Server keh raha hai: "Tere paas jo cache hai, woh hi sahi hai, naye data ki zarurat nahi."**

**4xx – Client Errors (Galti client ki taraf se hai)**

* **400 Bad Request – Request galat hai ya format thik nahi hai.**
* **401 Unauthorized – Tu login nahi hai, pehle authenticate ho ja.**
* **403 Forbidden – Tere paas yeh resource access karne ka permission nahi hai.**
* **404 Not Found – Jo cheez tu dhoond raha hai, woh server pe nahi mili.**
* **405 Method Not Allowed – Iss URL pe tu jis method se request bhej raha hai (jaise POST/GET), woh allowed nahi hai.**
* **409 Conflict – Request server ke current data se clash kar rahi hai. (Jaise duplicate entry)**
* **429 Too Many Requests – Tu baar-baar request bhej raha hai, thoda ruk ja!**

**5xx – Server Errors (Galti server ki taraf se hai)**

* **500 Internal Server Error – Server me kuch galat ho gaya, samajh nahi aaya kya.**
* **502 Bad Gateway – Server ko dusre server se galat response mila.**
* **503 Service Unavailable – Server busy hai ya maintenance me hai, abhi ke liye available nahi hai.**
* **504 Gateway Timeout – Server ne kisi aur server se response manga, par time pe reply nahi aaya.**

**Ye sab HTTP status codes har developer ko aane chahiye — API ka response samajhne me madad milti hai! 💡**

**Node.js** ek runtime environment hai jo JavaScript ko server pe run karne deta hai.

**Node.js allows you to run JavaScript outside the browser.**

**✅ "type": "commonjs"**

**👉 Isse Node.js tumhare .js files ko CommonJS syntax (require, module.exports) ke roop mein samjhta hai.**

**✅ "type": "module"**

**👉 Isse Node.js tumhare .js files ko ES Module syntax (import, export) ke roop mein samjhta hai.**

**🔷 1) "type": "commonjs"**

**📌 Matlab:**

**Node.js tumhare .js files ko CommonJS module system ke according padhega.**

**✅ CommonJS mein hum use karte hain:**

**const fs = require('fs'); // Module ko import karna**

**module.exports = myFunction; // Function ya object ko export karna**

**🧠 CommonJS kab use hota hai?**

* **Sabse purana aur default system Node.js ka.**
* **Agar tum require() aur module.exports use kar rahe ho.**
* **Express.js, MongoDB, JWT jaise modules by default CommonJS compatible hote hain.**

**🔷 2) "type": "module"**

**📌 Matlab:**

**Node.js tumhare .js files ko ECMAScript Module (ESM) ke roop mein treat karega.**

**✅ ESM mein hum likhte hain:**

**import fs from 'fs'; // Module ko import karna**

**export default myFunction; // Export karna**

**🧠 ESM kab use hota hai?**

* **Jab tum import/export syntax use karna chahte ho (modern JS jaise React ya TypeScript mein hota hai).**
* **Jab tum Next.js ya TypeScript project kar rahe ho.**
* **Jab tum .mjs file extension use karte ho.**

**Express.js ek lightweight aur flexible framework hai jo Node.js ke upar kaam karta hai, aur APIs aur web servers banana easy bana deta hai.**  
Iska use routing, middleware, aur HTTP requests handle karne ke liye hota hai.

**Routing ka matlab hota hai URL ke according request ko handle karna.**  
Jaise user kisi specific path pe request bhejta hai (/home, /about), toh server us route ke liye proper response bhejta hai.

**Dynamic routing ka matlab hai route ke path me variable ya dynamic value ka use karna.**  
Jaise ek hi route multiple users/products/posts ke liye kaam kare — bas URL ka part change hota hai.

**Dynamic routing wo hoti hai jisme URL ka kuch part dynamic hota hai, yani change ho sakta hai.**  
Jaise user/:id — yahan :id ek variable hai jo har request ke liye alag ho sakta hai.

**Middleware ek function hota hai jo request aur response ke beech me chalta hai.**  
Ye request ko process karta hai, modify karta hai, ya next function ko call karta hai.

Express.js me middleware ka use hota hai **authentication, logging, error handling, ya JSON parsing** jaise kaam ke liye.  
Har middleware req, res ko access karta hai, aur next() ke through aage badhta hai.

Mongoose is a JavaScript object-oriented programming library that creates a connection between MongoDB and the Node.js JavaScript runtime environment. It provides a straightforward, schema-based solution to model application data.

**Schema ek blueprint hota hai jo batata hai ki database me kaunsa data aayega, uska type kya hoga, aur wo kaise behave karega.**  
Mongoose me hum schema use karke define karte hain ki ek document me kaunse fields honge aur unka structure kya hoga.

**Model ek Mongoose object hota hai jo schema ke base par banta hai, aur isse hum MongoDB me data create, read, update, delete (CRUD) karte hain.**  
Yaani schema sirf structure batata hai, lekin model se actual database operations hote hain.

Schema ek template hai, aur model us template ka practical use — jaise **blueprint se ghar banana**.

Authentication ka matlab hota hai **user ki identity verify karna**—jaise login system mein check karna ki user ka email/username aur password sahi hai ya nahi. Agar sahi hua, toh user ko access milta hai protected routes ya features ka.

Authentication is the process of verifying a user's identity, usually by checking credentials like a username and password. It ensures that only authorized users can access the system or specific resources."

**Authorization** ka matlab hota hai:  
 *"User ko kya access milna chahiye aur kya nahi, ye decide karna."*  
Ye **authentication ke baad** hota hai.

Example: Login ke baad sirf admin ko hi "delete" ka option dikhana.

**One-liner trick yaad rakhne ke liye:**  
 *“Authentication bataata hai* ***tum ho kaun****, Authorization bataata hai* ***tum kar kya sakte ho***.”

**bcrypt** ek hashing library hai jo passwords ko **securely encrypt** karne ke kaam aati hai.  
Yeh plain-text password ko hash banakar store karta hai, taaki agar database leak ho jaye to actual password kisi ko na mile.

**How it works:**

* **Hashing:** bcrypt.hash(password, saltRounds)
* **Comparing:** bcrypt.compare(enteredPassword, hashedPassword)
* **JWT** ek token-based authentication method hai jo backend aur frontend ke beech **secure data transfer** ke liye use hota hai — bina session store kiye.

**Kaise kaam karta hai:**

1. User login karta hai → server JWT token banata hai.
2. Token user ko milta hai → har request ke saath bhejta hai.
3. Server token verify karta hai → agar valid hai to access deta hai

**🔰 Types of APIs (Interview ke hisaab se):**

**1. Public APIs (Open APIs)**

**🔹 Definition:**

Ye APIs **publicly available** hoti hain. Koi bhi developer bina special permission ke inhe use kar sakta hai.

**🔸 Example:**

* OpenWeatherMap API (weather info)
* GitHub Public APIs
* REST Countries API (<https://restcountries.com/>)

**📌 Use-case:**

Agar tu chaahta hai ki duniya ke developers teri API use karein — documentation ke saath.

**2. Private APIs**

**🔹 Definition:**

Ye APIs **internal use only** hoti hain. Sirf ek specific organization ya app ke andar ka code hi inhe access kar sakta hai.

**🔸 Example:**

* Zomato ke kitchen tools backend APIs
* Company ki payroll service ke APIs (HR software)

**📌 Use-case:**

Jab tu chaahta hai ki API sirf teri company ya teri app use kare, aur bahar koi access na kar sake.

**3. Partner APIs**

**🔹 Definition:**

Ye APIs select **partners ke liye shared** hoti hain — not completely public, but not fully private either. Inhe access karne ke liye **API key ya token** chahiye.

**🔸 Example:**

* Razorpay API (Business partners ke liye)
* Paytm Developer API (Merchant ke liye)

**📌 Use-case:**

Tu kisi partner ke saath kaam kar raha hai aur use specific access dena hai.

**4. Composite APIs**

**🔹 Definition:**

Ek single API jo **multiple internal APIs ko call** karti hai — ek hi request mein multiple kaam ho jaate hain.

**🔸 Example:**

E-commerce app mein ek API call:

* Order details
* Shipping info
* Payment info —all at once.

**📌 Use-case:**

Performance improve karne ke liye — kam requests mein zyada data.

**✅ Bonus (Interview Point of View):**

| **Type** | **Access Level** | **Use-case** |
| --- | --- | --- |
| Public | Open for all | Data sharing, public dev tools |
| Private | Internal only | App-specific backend logic |
| Partner | Limited access | Business collaboration (via tokens) |
| Composite | Internal/External | Aggregated info in single response |

**REST API (Representational State Transfer) – Full Focus**

**📌 1. Stateless**

* Har request **independent** hoti hai.
* Server **client ki pehli request yaad** nahi rakhta.

🗣️ *"Server har baar fresh request handle karta hai."*

**📌 2. Client-Server Architecture**

* Client (React frontend) & Server (Express backend) alag hote hain.
* Dono independent develop ho sakte hain.

🗣️ *"Frontend backend se alag hota hai, dono apna kaam alag karte hain."*

**📌 3. HTTP Methods (Very Important)**

| **Method** | **Kaam kya karta hai** | **Example** |
| --- | --- | --- |
| GET | Data **read** karta hai | /todos |
| POST | Data **create** karta hai | /todos |
| PUT | Data **update** karta hai | /todos/:id |
| DELETE | Data **delete** karta hai | /todos/:id |

🗣️ *"GET read karta hai, POST create karta hai, PUT update karta hai, DELETE remove karta hai."*

**📌 4. Resources via URL**

* URL se hi batate hain kya chahiye.

🧠 Example:

bash

CopyEdit

GET /todos => saare todos

POST /todos => naya todo banao

PUT /todos/:id => ek todo update karo

DELETE /todos/:id => ek todo delete karo

**📌 5. JSON format**

* REST APIs ka data mostly **JSON** format me hota hai.

json

CopyEdit

{

"title": "Learn REST",

"description": "REST API in MERN stack"

}

**📌 6. HTTP Status Codes (Must Know)**

| **Code** | **Meaning** |
| --- | --- |
| 200 | OK |
| 201 | Created |
| 400 | Bad Request |
| 404 | Not Found |
| 500 | Internal Server Error |

🗣️ *"200 ka matlab success, 404 ka matlab data nahi mila."*

**📌 7. Best Practices (Short Notes)**

* Use **plural nouns** in URLs → /todos, /users
* Use **proper status codes**
* Keep it **stateless**
* Use **try-catch** and error handling
* Use **middleware** (like for auth)
* **API Definition:**
* **API** ek interface hai jo do software applications ko ek dusre se interact karne ka tareeka deta hai. Isme defined rules hote hain jo ek program ko doosre program se data ya services ko request karne aur receive karne ka tarika batate hain.
* API ek software component ko doosre software component ke saath connect karne ka kaam karta hai.
* **API** ek set of rules hai jisse do software systems ek dusre ke saath **data exchange** kar sakte hain.