**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.**

**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