

# NODE JS

- **Modules**

- Pehle to har usme **npm init -y** krke package.json file bana lo
- Module like import waala. To 2 tareekhe ke hote hai. Inko 2 tareeko se use karte hai.

- **Old way - using require :**

```
utils.js      X
1Module > utils.js > [?] <unknown>
1   const sum = (a,b) => a+b;
2   const mul = (a,b) => a*b;
3   const div = (a,b) => a/b;
4   const sub = (a,b) => a-b;
5
6   module.exports = {sum , mul , div , sub}
```

Ye export krne ka hai

```
1modules.js X
1Module > 1modules.js > ...
1   const {sum , sub , mul , div} = require('./utils')
2
```

Or ye import krne ka

- **New way - using import**

- Iske liye pehle package.json mai kuch change krna hoga

```
"type": "module"
```

- Then

utils.js

module.js

2Module > utils.js > ...

```
1 export const sum = (a,b) => a+b;
2 export const mul = (a,b) => a*b;
3 export const div = (a,b) => a/b;
4 export const sub = (a,b) => a-b;
5
```

Isko direct export kr liya

utils.js

module.js

2Module > module.js

```
1 import {sum , sub , mul , div} from './utils'
```

Ese import kr skta hoon

## ● File handling in Node js

- Like how to read file
- Node jo predefined functions deta hai wo bhi dekhenge

index.js

3FileHandling > index.js

```
1 import {} from 'fs/promises'
```

Here fs means file system and promises is js waala.

Ham yaha pe async await waali cheez use karne waale hai

Is line se file module waali sabhi cheeze use kr skte hai

index.js

sample.txt

3FileHandling > sample.txt

```
1 hello this is sample file
```

Ek sample file bana li

The screenshot shows a VS Code interface. In the top left, there are two tabs: 'index.js' and 'sample.txt'. Below them is a code editor window containing the following JavaScript code:

```
3FileHandling > index.js > [read_file]
1 import {readFile} from 'fs/promises'
2
3 const read_file = async (fileName) => {
4     const data = await readFile(fileName, 'utf-8');
5     console.log(data);
6 }
7 read_file('sample.txt')
```

Below the code editor is a tab bar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined), and 'PORTS'. Under the 'TERMINAL' tab, there is a terminal window showing the following output:

- PS D:\Programming\Node Js Backend\3FileHandling> node index.js
- hello this is sample file
- PS D:\Programming\Node Js Backend\3FileHandling>

Ek file milegi parameter mai, us file ko read krlo, us file ka type utf-8 hoga, and uske andar ka data print karwa do and last mai function call krke argument mai value pass kardi.

Now lets see how we create a new file.

- **Write file**

- Is baar new file bana re to koi new variable ya log krne ki jarrat nahi hai
- Parameter mai 2 cheez bhejo, file ka naam or kya content bhejna hai

The screenshot shows a VS Code interface. In the top left, there are two tabs: 'write.js' and 'ai.py'. Below them is a code editor window containing the following JavaScript code:

```
3FileHandling > write.js > ...
1 import { writeFile } from "fs/promises";
2
3 const write_file = async (fileName, content) => {
4     await writeFile(fileName, content);
5     console.log("File create successfully");
6 }
7 write_file('ai.py', 'this is a python file');
```

- **Update file**

- Already file hai, usme new cheeze append karenge.

- 2 cheeze lega ye bhi, konsi file mai content update krna hai + kya content new daalan ya change krna hai

The screenshot shows the VS Code interface. The top bar has tabs for 'append.js' and 'sample.txt'. Below the tabs, the code editor contains the following JavaScript code:

```

3FileHandling > JS append.js > ...
1 import { appendFile } from "fs/promises";
2
3 const append_file = async (fileName , content) => {
4     await appendFile(fileName,content);
5     console.log("Updated successfully");
6 }
7 append_file("sample.txt" , "new line of content added")

```

Below the code editor is the terminal window. It shows the command 'node append.js' being run and the output 'Updated successfully'.

The screenshot shows the VS Code interface. The top bar has tabs for 'append.js' and 'sample.txt'. The code editor shows the contents of 'sample.txt':

```

JS append.js X sample.txt X
3FileHandling > sample.txt
1 hello this is sample filenew line of content added

```

- Path module
- Joining 2 or more file

The screenshot shows the VS Code interface. The top bar has tabs for 'one.js' and 'sample.txt'. Below the tabs, the code editor contains the following JavaScript code:

```

4PathModule > JS one.js > ...
1 import path from 'path'
2
3 const fullPath = path.join('/path' , 'index.py' , 'file.java')
4 console.log(fullPath);
5

```

Below the code editor is the terminal window. It shows the command 'node one.js' being run and the output '\path\index.py\file.java'.

- Kisi file ka absoulte/full path dekhna hoga to

A screenshot of the Visual Studio Code interface. On the left is a code editor window titled "one.js" containing the following Node.js code:

```
JS one.js    X
4PathModule > JS onejs > ...
1 import path from 'path'
2
3 // const fullPath = path.join('/path' , 'index.py' , 'file.java')
4 // console.log(fullPath);
5
6 const absolutePath = path.resolve()
7 console.log("Currently we are working on - " , absolutePath);
8
```

The code editor has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the following terminal output:

```
PS D:\Programming\Node Js Backend\4PathModule> node one.js
Currently we are working on - D:\Programming\Node Js Backend\4PathModule
PS D:\Programming\Node Js Backend\4PathModule>
```

- Get extension name

A screenshot of the Visual Studio Code interface. On the left is a code editor window containing the following Node.js code:

```
9 const extensionName = path.extname("image.pdf")
10 console.log(extensionName);
11
```

The code editor has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the following terminal output:

```
● PS D:\Programming\Node Js Backend\4PathModule> node one.js
.pdf
○ PS D:\Programming\Node Js Backend\4PathModule>
```

A radio button is highlighted with a blue circle, indicating it is selected.

- Conditional on basis of extension name

```

  9  const extensionName = path.extname("image.pdf")
10  console.log(extensionName);
11  if (extensionName == ".pdf") {
12      console.log("Ok");
13  }
14  else{
15      console.log("Not ok");
16  }
17
18 }
19

```

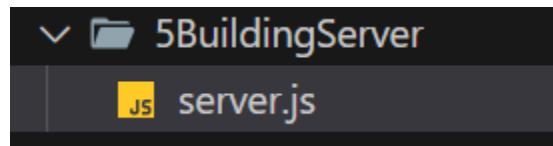
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```

● PS D:\Programming\Node Js Backend\4PathModule> node one.js
.pdf
Ok
○ PS D:\Programming\Node Js Backend\4PathModule> []

```

- **Building server (HTTP modules)**



- File ka name always ye rakho

- Package.json bhi add krlo, then "type": "module" krlo.

```

server.js  X
5BuildingServer > server.js
1 import http from 'http'

```

- Is module ka use karte hai

```

import http from 'http'

const server = http.createServer();

```

- Is function ko call krna hai, ab hame ek port ki jarrof bhi padti hai

A screenshot of the Visual Studio Code interface. The left pane shows a code editor with a file named 'server.js'. The code is a simple Node.js script that creates a server on port 3000 and logs a message to the console. The right pane shows a terminal window with the output of running the script using 'nodemon'. The terminal output shows the nodemon version, the path being watched, extensions being checked, and the server starting on port 3000.

```
JS server.js  X
5BuildingServer > JS server.js > ...
1 import http from 'http'
2
3 const server = http.createServer();
4 const port = 3000;
5 server.listen(port, () => console.log(`Server is running on port no ${port}`));
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  PORTS + 1: node (5BuildingServer)

○ PS D:\Programming\Node Js Backend\5BuildingServer> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port no 3000
```

- Ab isme ek req and response jesi cheez bhi hoti hai jaha pe user req karega and ham usko response denge.

A screenshot of the Visual Studio Code interface. The code editor shows a modified 'server.js' file. The script now includes a handler function that sends a response with the message 'Your requested for something' whenever it receives a request. The right pane shows a browser window displaying this response.

```
JS server.js  X
5BuildingServer > JS server.js > ...
1 import http from 'http'
2
3 const server = http.createServer((req, res) => {
4   res.end("Your requested for something");
5 });
6 const port = 3000;
7 server.listen(port, () => console.log(`Server is running on port no ${port}`));
8

← → ⌂ ⌂ localhost:3000
Texture Wallpaper H... Pattern Background

Your requested for something
```

Ab browser pe apna port number daalo, or output dekho.  
res.end() means user ko response deke uski req ko poora kr re.

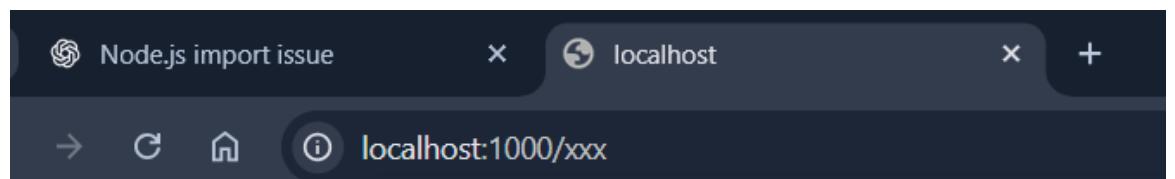
## ● Routing

- Suppose user likhta hai apple, to apple ki image aaye, ese hi user jo likhta hai uski image aani chahiye.
- Pehle server bana lo

```
JS server.js X
6Routing > JS server.js > ...
1 import http from 'http'
2
3 const server = http.createServer((req,res) => {
4     res.end('<h1>Your requested has been accepted</h1>')
5 })
6
7 const port = 1000
8 server.listen(port, () => console.log(`Server is running on port no ${port}`));
9
```

```
JS server.js X
6Routing > JS server.js > ...
1 import http from 'http'
2
3 const server = http.createServer((req,res) => {
4     // res.end('<h1>Your requested has been accepted</h1>')
5
6     console.log(req); // kis browser se req aari ye batayega
7     console.log(req.url); // kya url ban raha hai ye batayega
8 })
9
10 const port = 1000
11 server.listen(port, () => console.log(`Server is running on port no ${port}`));
12
```

I wrote this

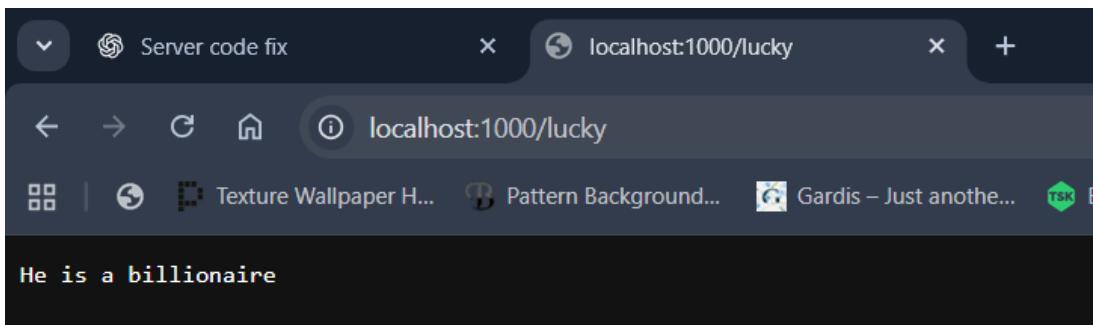
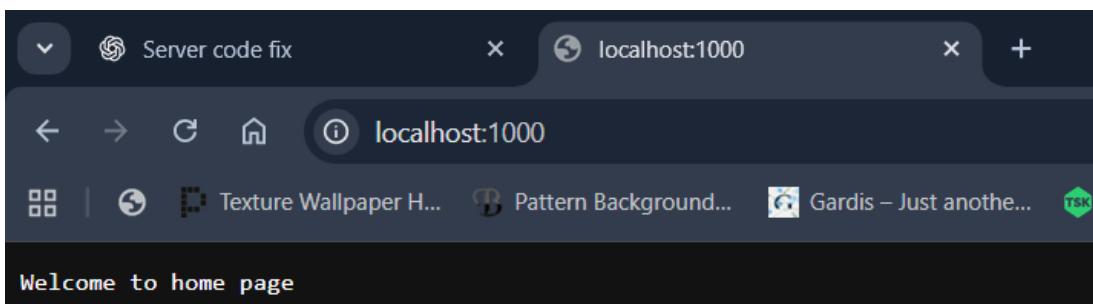


```
[Symbol(kTrailers)]: null,
[Symbol(kTrailersCount)]: 0
}
/xxx
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Server is running on port no 1000
```

output mai url dekh skte,

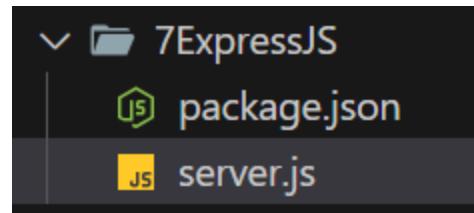
console pe.

```
JS serverjs X
6Routing > JS server.js > [o] server > http.createServer() callback
1 import http from 'http'
2
3 const server = http.createServer((req,res) => {
4     // res.end('<h1>Your requested has been accepted</h1>')
5
6     // console.log(req); // kis browser se req aari ye batayega
7     // console.log(req.url); // kya url ban raha hai ye batayega
8
9     if(req.url === '/'){
10         res.end("Welcome to home page")
11     }
12     else if (req.url === '/apple') {
13         res.end("This is apples image")
14     }
15     else if(req.url === '/lucky'){
16         res.end("He is a billionaire")
17     }
18     else {
19         res.end('404 Not Found');
20     }
21
22 })
23
24 const port = 1000
25 server.listen(port, () => console.log(`Server is running on port no ${port}`));
26
```



## ● Express js

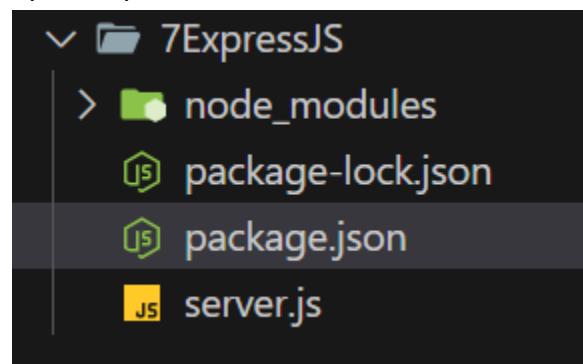
- API banane ke kaam aata hai ye



Type module krlo

```
7ExpressJS > package.json > type
1  {
2    "name": "7expressjs",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "type": "module"
13 }
```

- Lets install express now and then package file mai kuch new dependencies add ho jayengi.
- Npm i express



Some new files and folder added.

Node modules wo engine hota hai jisse express ka server banta hai

- Create server & routing in ExpressJs

- Ab yaha pe http ki jarrof nahi padhti, express ki jarrat padti hai

A screenshot of a code editor showing a file named `server.js`. The code is as follows:

```
JS server.js X
7ExpressJS > JS server.js
1 import express from 'express'
```

A screenshot of a code editor showing the completed `server.js` file and its execution output in the terminal.

The code in `server.js` is:

```
JS server.js X
7ExpressJS > JS server.js > ...
1 import express from 'express'
2
3 const server = express();
4 const port = 2000;
5 server.listen(port , () => console.log(`Server is running on port ${port}`));
6
```

The terminal output shows the server starting at port 2000:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + 1: node (7)
○ PS D:\Programming\Node Js Backend\7ExpressJS> nodemon .\server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node .\server.js`
Server is running on port 2000
```

Ye krke server create ho gaya.

- Ab routing krte hai

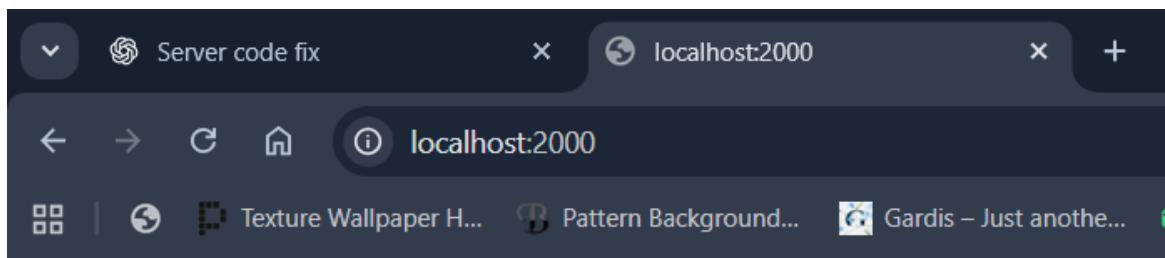
A screenshot of a code editor showing the `server.js` file with a basic routing rule added.

The code in `server.js` is:

```
JS server.js X
7ExpressJS > JS server.js > ...
1 import express from 'express'
2
3 const server = express();
4
5 server.get('/', (req , res) => {
6   res.send("You are requested for home route");
7 }
8
9 const port = 2000;
10 server.listen(port , () => console.log(`Server is running on port ${port}`));
11
```

Dekh skte ho code chhota ho gaya express mai. Ye hi same kaam jab node se krre the to bada tha code.

Pehle itna chala ke dekho.



You are requested for home route

```
server.js  X
7ExpressJS > server.js > ...
1 import express from 'express'
2
3 const server = express();
4
5 server.get('' , (req , res) => {
6   res.send("You are requested for home route");
7 }
8
9 server.get('/lucky' , (req , res) => {
10   res.send("You are a lucky person");
11 }
12
13 const port = 2000;
14 server.listen(port , () => console.log(`Server is running on port ${port}`));
15
```

A screenshot of a code editor showing a file named "server.js". The code is written in JavaScript and uses the Express.js framework. It defines a server, sets up two routes: a home route that sends the message "You are requested for home route", and a "/lucky" route that sends "You are a lucky person". It also specifies that the server should listen on port 2000 and logs a message to the console when it starts.

Ek or route hamne create kiya.

- Send JSON, HTML, HTML File In Response
- sending response in JSON

```
server.js  X
8SendFileInResponse > server.js > [o] fruits
  1 import express from 'express'
  2
  3 const app = express()
  4
  5 const fruits = [
  6   {
  7     name : "Apple",
  8     price : 100
  9   },
 10   {
 11     name : "Banana",
 12     price : 50
 13   },
 14   {
 15     name : "Mango",
 16     price : 20
 17   }
 18 ]
20 // sending response in JSON
21 app.get('/', (req , res) => {
22   res.json({
23     msg : "Successfully fetched all fruits",
24     fruitsDetails : fruits,
25     success : true
26   });
27 }
28
29 const port = 2000;
30
31 app.listen(port , () => console.log(`Server is running on port ${port}`))
```

```

{
  "msg": "Successfully fetched all fruits",
  "fruitsDetails": [
    {
      "name": "Apple",
      "price": 100
    },
    {
      "name": "Banana",
      "price": 50
    },
    {
      "name": "Mango",
      "price": 20
    }
  ],
  "success": true
}

```

Output :

```

// sending response in HTML
app.get('/', (req, res) => {
  res.send("<h1>Hello</h1>")
})

```

- **Sending html file**

- Iske liye absoulte path dena hota hai, jo upper sheekha hamne

```

// sending response in HTML file
app.get("/", (req, res) => {

  const dir = path.resolve(); // yaha se path mil jayega

  const url = path.join(dir, './index.html'); // path ko file ke sath join krdo

  console.log("Full path = " , url);

  res.sendFile(url)
})

```

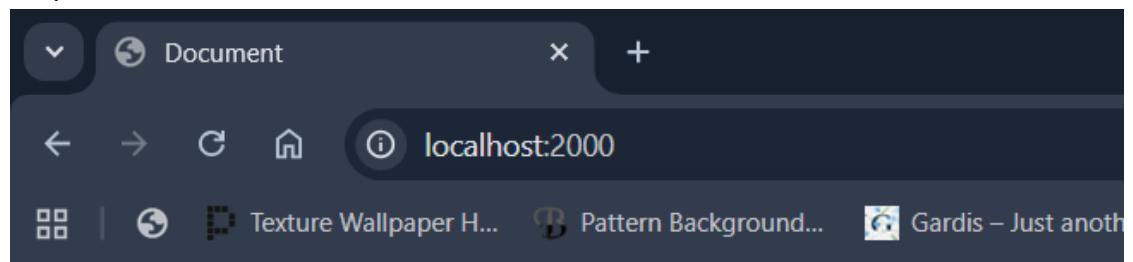
- Log mai path esa dikhra hai :

```

[nodemon] restarting due to changes...
[nodemon] starting `node .\server.js`
Server is running on port 2000
Full path = D:\Programming\Node Js Backend\8SendFileInResponse\index.html

```

- Output :



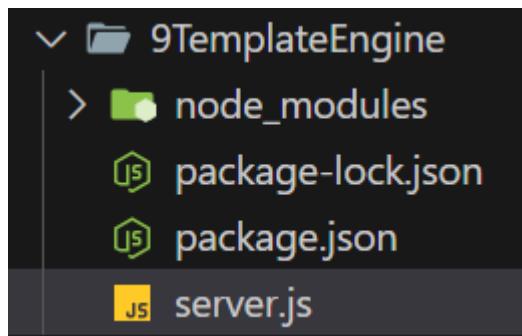
## This data is from index . html file

- **Template Engine (ejs) || SSR ( Server Side Rendering )**

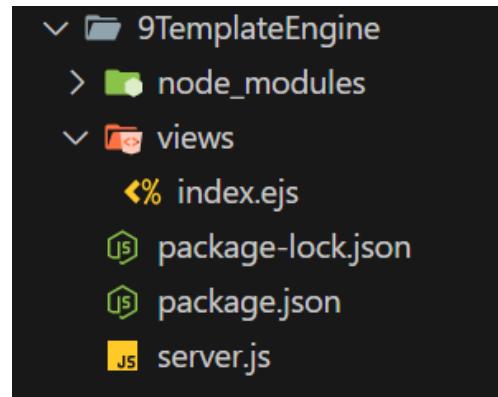
```
const name = "Lalit";
res.sendFile(url)
```

- Suppose esa hota upper waale code mai, mujhe agr dynamic naam dikhana hota apne html mein, to mai nahi dikha pata.  
Wo dikhane ke liye template engine ki jarrorat hoti hai.

- Express mai kaafi saare template engine hote hai, unme se ham ejs ko use karne wale hai, which means embedded java script. Iski help se html ke code mai hi js ham likh sakte hai.



- pehle to hamne itna kaam kar liye, ab hame ejs ko bhi install krna hoga
- Npm i ejs
- Ejs pehel use hota tha jab react nahi aaya tha, ab iska use nahi hota.
- Iske liye ek views naam ka folder banana hota hai. Uske andr index.ejs file.

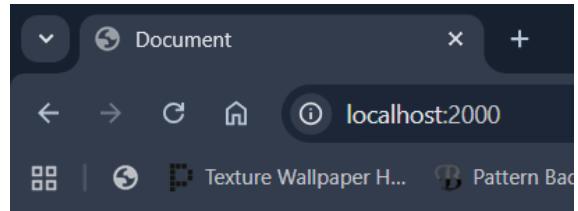


```
❷% index.ejs  ✘
9TemplateEngine > views > ❷% index.ejs > 📁 html
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>i am learning node js</h1>
10    </body>
11  </html>
```

Is html file ko mujhe render krke dikhna hai.

```
❷% index.ejs  📁 server.js  ✘
9TemplateEngine > 📁 serverjs > ...
1   import express from 'express'
2
3   const app = express()
4
5   app.get('/', (req , res) => {
6     res.render('index.ejs')
7   })
8
9   const port = 2000
10
11  app.listen(port , () => console.log(`Server is running at port ${port}`))
```

Abhi itna hamne kiya hai,



## i am learning node js

Ye output aa raha hai.

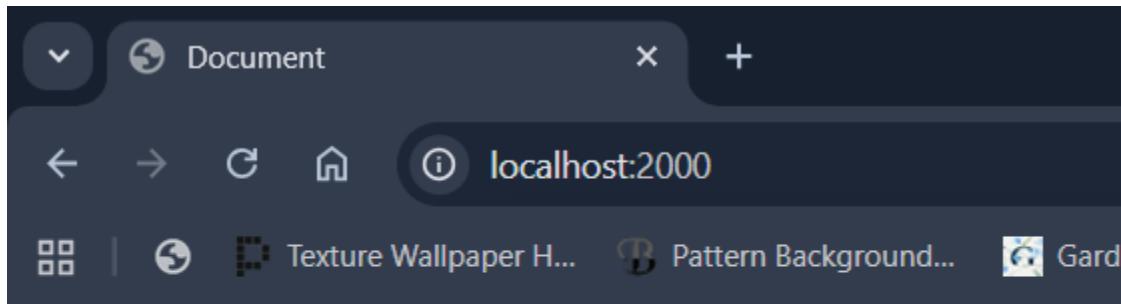
- Ab wo hi name waala kaam jo ham krne waale the wo karna hai.

```
app.get('/', (req, res) => {
  let name = 'lalit'
  res.render('index.ejs', {name})
})
```

Yaha maine is trh pass kiya name

```
<body>
  <h1>i am learning node js <%= name %></h1>
</body>
```

Yaha maine is trh access kiya.



## i am learning node js lalit

Is trh se dynamically yaha pe aa gaya.

- Ab ek baar list pass krke dekhte hai

```
const fruits = [
  {name : "Apple",price : 100},
  {name : "Banana",price : 65},
  {name : "Litchi",price : 50}
]

app.get('/', (req , res) => {
  let name = 'lalit'
  res.render('index.ejs' , {name , fruits})
})
```

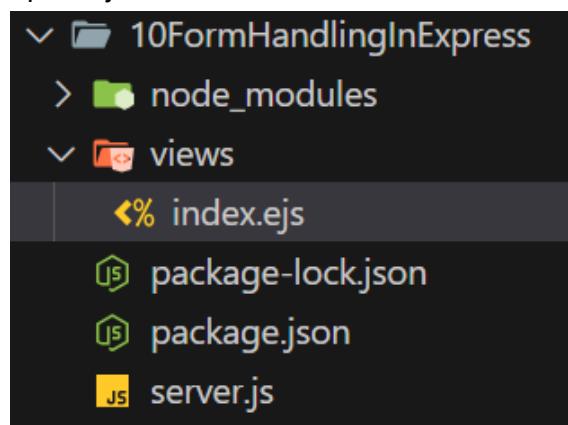
```
<ul>
  <li><%=fruits[0].name%></li>
  <li><%=fruits[0].price%></li>

  <li><%=fruits[1].name%></li>
  <li><%=fruits[1].price%></li>

  <li><%=fruits[2].name%></li>
  <li><%=fruits[2].price%></li>
</ul>
```

## ● Form Handling In Express.JS

- Npm init -y
- Npm i express
- Npm i ejs

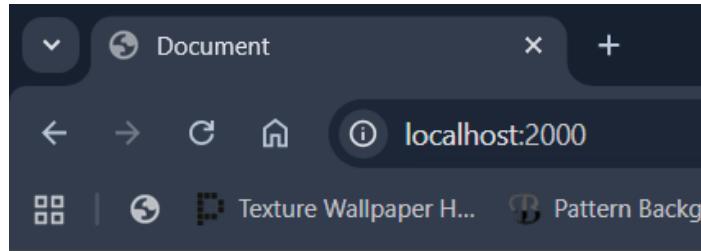


```
server.js    ↵% index.ejs  X
10FormHandlingInExpress > views > ↵% index.ejs > ⚡ html > ⚡ body > ⚡ form > ⚡ br
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1>Form Handling</h1>
10
11      <form action="/form-submit" method="post">
12          name <input type="text" name="name" placeholder="Enter your name"/>
13          <br/>
14          <br/>
15          email <input type="email" name="email" placeholder="Enter your email"/>
16          <br/>
17          <br/>
18          <input type="submit" value="Save">
19      </form>
20   </body>
21   </html>
```

### Index.ejs

```
server.js  X  ↵% index.ejs
10FormHandlingInExpress > server.js > ...
1   import express from 'express'
2
3   const app = express();
4
5   app.get('/', (req , res) => {
6       res.render('index.ejs')
7   })
8
9   const port = 2000;
10  app.listen(port , () => console.log(`Server is running on port ${port}`));|
```

### Server code



## Form Handling

name

email

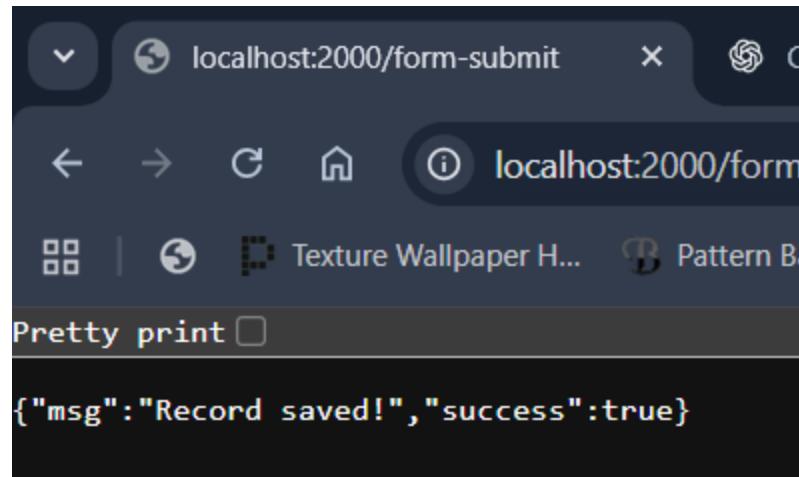
- Output :
- Ab is form ko handle krna hai. Form handle krne ke liye express mai hamko ek middleware use karna hota hai

```
app.get('/', (req , res) => {
  res.render('index.ejs')
}

// -> /form-submit : ye route hota hai, form submit hone ke baad yaha jayega. ye form-submit html page mai
// form ke action tag mai pass karenge. like .net mai action method ka naam pass krte the
app.post('/form-submit' , (req , res) => {

  // yaha DB mai save krne ka code hota hai, pr abhi ke liye db mai save nahi kar re
  res.json({
    msg : "Record saved!",
    success : true
  })
})
```

- Post se data save hoga and us data ko home page pe render karwayega get method.



```
{"msg": "Record saved!", "success": true}
```

form submit krne

pe data save ho gaya.

- Suppose form mai data to post kr diya, pr ab tumhe suppose express se data lena hai, to req.body ka use kerte hai.

```
app.post('/form-submit' , (req , res) => {  
  
    console.log(req.body)  
    // yaha DB mai save krne ka code hota h.  
    res.json({  
        msg : "Record saved!",  
        success : true  
    })  
})
```

log karwaaya hai data

```
[nodemon] Starting `node ./server.js`  
Server is running on port 2000  
{ name: 'Lalit Kumar', email: 'kumar.lalit.tsx@gmail.com' }  
{ name: 'Lalit Kumar', email: 'kumar.lalit.tsx@gmail.com' }
```

Is tareeke se express se data lete hai

## ● MongoDB Atlas Setup

- Login kiya 0101 wale se.
- Projects pe jake new projects pe click kiya

The screenshot shows the MongoDB Atlas interface for creating a new project. On the left, there's a sidebar with sections for Identity & Access (All Projects, Users, Applications, Teams, Federation), Billing (Overview, Invoices, Cost Explorer), and Configurations (Resource Policies, Activity Feed). At the bottom of the sidebar is an Organization Settings link. The main area is titled "Create a Project" and has two tabs: "Name Your Project" (which is active) and "Add Members". Under "Name Your Project", it says "Project names have to be unique within the organization (and other restrictions)." A text input field contains "NodeJs Learning Course". Below that is a section for "Add Tags (Optional)" with a note about using tags to label projects. A table allows adding tags with columns for Key, Value, and Actions. A "Next" button is at the bottom right.

Proj ka naam deke next karo.

## Create a Project

✓ Name Your Project > Add Members

### Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

luckyarya0101@gmail.com  
(you)

Project Owner

Back

Cancel

Create Project

Then neeche create proj pe click krdo

- Ab create cluster pe click kardo
- Cluster banate wqt user name and password aaya  
Username : luckyarya0101\_db\_user  
Password : ZJcto5ONGZHidEiL

## Connect to Cluster0



You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

### 1. Add a connection IP address

Your current IP address (49.42.35.35) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access](#).

### 2. Create a database user

This first user will have [atlasAdmin](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

i You'll need your database user's credentials in the next step. Copy the database user password.

Username

luckyarya0101\_db\_user

Password

ZJcto5ONGZHidEiL

HIDE

Copy

[Create Database User](#)

[Close](#)

[Choose a connection method](#)

- Ek mongodb compass tool download krna hota hai jo ki GUI based hota hai. Use download kr liya hai

## Connecting with MongoDB Compass

[I don't have MongoDB Compass installed](#)

[I have MongoDB Compass installed](#)

### 1. Choose your version of Compass

1.38 or later

See your Compass version in "About Compass"

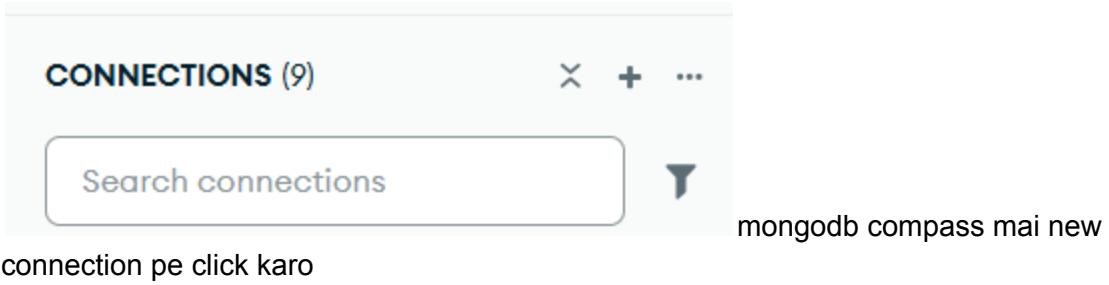
### 2. Copy the connection string, then open MongoDB Compass

Use this connection string in your application

`mongodb+srv://<db_username>:<db_password>@cluster0.ww01jq6.mongodb.net/`



Yaha maine choose kiya hai I have mongodb compass installed and neeche link mila hai  
[mongodb+srv://<db\\_username>:<db\\_password>@cluster0.ww01jq6.mongodb.net/](mongodb+srv://<db_username>:<db_password>@cluster0.ww01jq6.mongodb.net/)  
[mongodb+srv://luckyarya0101\\_db\\_user:<db\\_password>@cluster0.ww01jq6.mongodb.net/](mongodb+srv://luckyarya0101_db_user:<db_password>@cluster0.ww01jq6.mongodb.net/)



## New Connection

Manage your connection settings

URI [Edit Connection String](#)

```
mongodb+srv://%3Cluckyarya0101_db_user%3E:*****@cluster0.ww01jq6.mongodb.net/
```

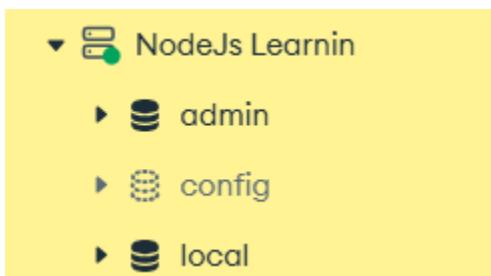
Name  Color  [How do I find my connection string in Atlas?](#)

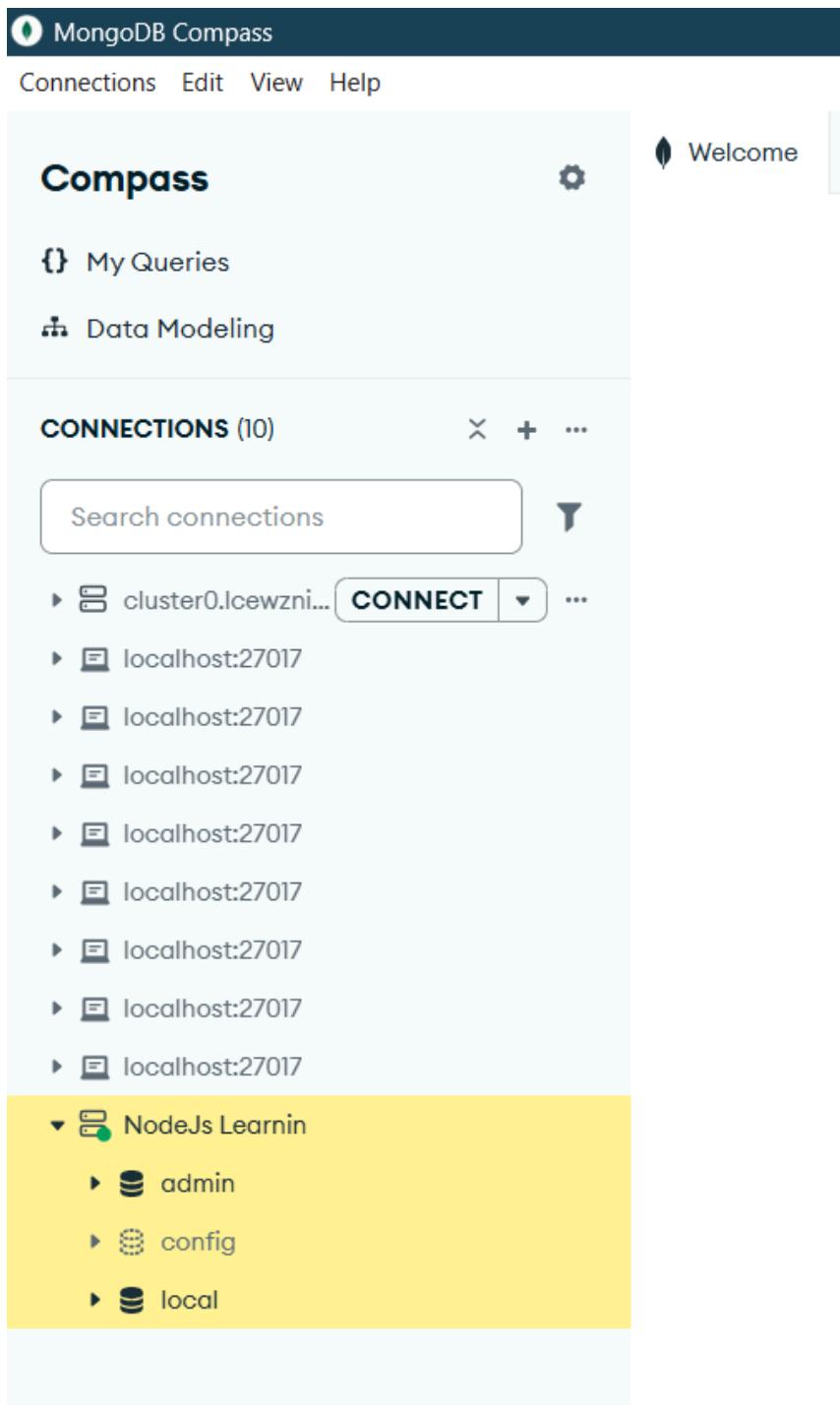
Favorite this connection  
Favoriting a connection will pin it to the top of your list of connections

[Advanced Connection Options](#)

[Cancel](#) [Save](#) [Connect](#) [Save & Connect](#)

Neeche name and color bhi select krlo  
And save and connect



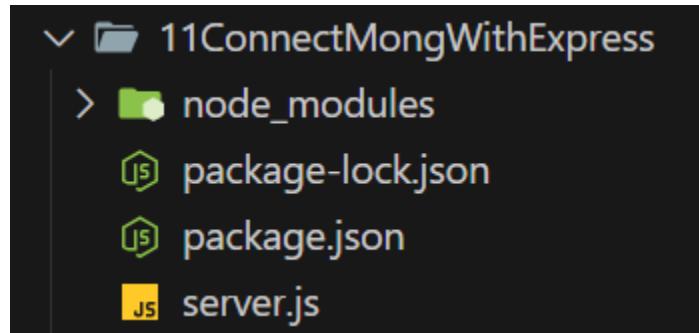


Connect hoke kuch is trh dikhega compass

- Connecting mongodb with atlas

- Connect krne ke liye ek package ki jarurat padegi, which is called mongoose.
- Is library ke through mongodb ko express ke sath connect krte hai.

- Npm i mongoose
- Init and expres bhi npm kr lena.



Ab server banane ka code likhlo

```

11ConnectMongWithExpress > server.js > ...
1 import express from 'express'
2
3 const app = express()
4
5 const port = 2000
6
7 app.listen(port, () => console.log(`Server is running at port ${port}`))


```

- Ab mongoose se connect karenge

```

import express from 'express'
import mongoose from 'mongoose'

```

Pehele import krlo

- Ab mongoose mein method hote hai kuch

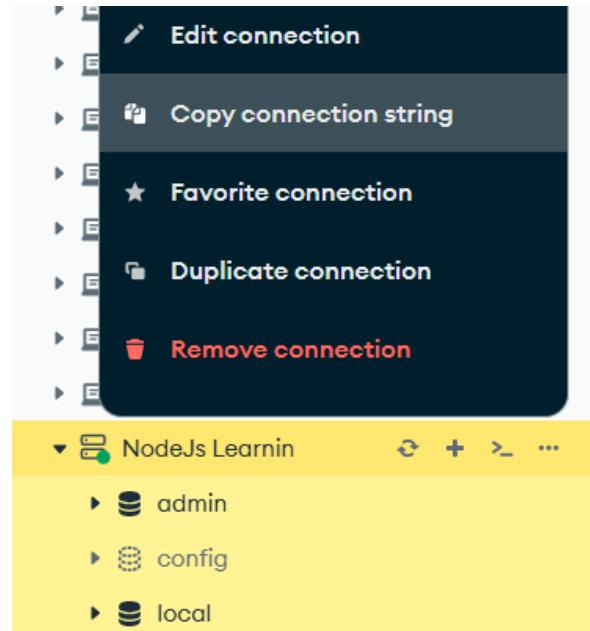
```

import express from 'express'
import mongoose from 'mongoose'

const app = express()
mongoose.connect() // Opens Mongoose's default connection to MongoDB, see connections docs

```

Is method mai ye uri maang ra. Ye uri or kuch nahi connecting string hai, ye compass se mil jayegi.



```
mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/")
```

Last mein hame db ka naam dena hota hai kuch is trh

```
mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {  
  dbName:"Nodejs Learning"  
})
```

Then ye ek promise return karta hai, to then cathc se handle karna hoga

```
6  mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {  
7    dbName:"Nodejs Learning"  
8  })  
9  .then(() => console.log("MongoDb Connected"))  
10 .catch((err) => console.log(err))  
11  
12 const port = 2000  
13  
14 app.listen(port , () => console.log(`Server is running at port ${port}`))
```

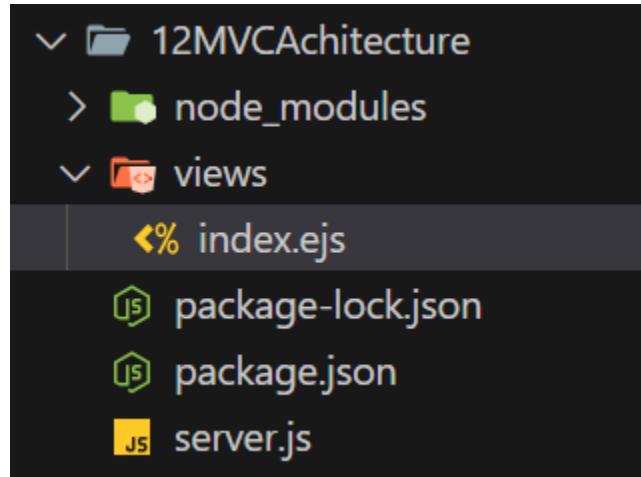
TERMINAL

```
MongoDb Connected  
[nodemon] restarting due to changes...  
[nodemon] starting `node ./server.js`  
Server is running at port 2000  
MongoDb Connected  
[nodemon] restarting due to changes...  
[nodemon] starting `node ./server.js`  
Server is running at port 2000  
[nodemon] restarting due to changes...  
[nodemon] starting `node ./server.js`  
Server is running at port 2000  
MongoDb Connected
```

Console pe dekh skte, mongodb connected likha aa gaya.

## ● MVC - Models Views & Controllers In Express.JS

- Project setup krlo
- Folder banaya, usme npm init -y , npm i express ejs mongoose  
Ye sb install krlo
- Package.json mai "main": "index.js", ki jgh "main": "server.js", likh diya.



```

<% index.ejs > x
12MVCArchitecture > views > <% index.ejs > <html>
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>Model Views and Controllers</h1>
10    </body>
11  </html>

```

```

<% index.ejs > server.js > x
12MVCArchitecture > <js> server.js > ...
1 import express from 'express'
2 import mongoose from 'mongoose'
3
4 const app = express();
5
6 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
7   dbName:"Nodejs Learning"
8 })
9 .then(() => console.log("Connected"))
10 .catch((err) => console.log(err))
11
12 const port = 2000
13
14 app.listen(port , ()=> console.log(`Server is running at port ${port}`))

```

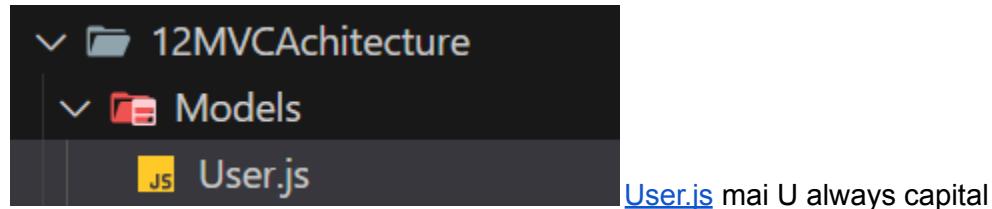
Itna to ye poorana wala kaam kar liya

- Model : database ke col ka naam

Views : jaha ejs ki coding, html likhre

Controller :

- Lets create model



```
1 import mongoose from 'mongoose'
2
3 // userSchema means table ka kaam - userTable bhi likh skte
4 const userSchema = new mongoose.Schema({
5
6     // ab yaha col define kardo
7     name:{type:String},
8     email:{type:String},
9     password:{type:String},
10    age:{type:Number},
11    createdAt:{
12        type:Date,
13        default:Date.now
14    },
15 }
16
17 export const User = mongoose.model("user" , userSchema)
```

So userSchema/userTable bana li and usme kya kya col aane waale hai kis kis type ke wo wo define kar diye.

Createat automatically aayeaga

Last line smjhte hai,

- **export const User** - isme User kaa naam ka U always capital rakhna, ye User.js hi hai.
  - **("user" , userSchema)** - user jo likha hai , mongoose jb usko mongodb compass mai col banayega to automatically plural kar dega, like users kr dega. Ye users hamari model ka naam hai, hamne likha bhi hai mongoose.model and userSchema to hamari table ka naam hai.
- 
- Ab in fields ko yaha se notnull/ required bhi kar skte

```
12MVCArchitecture > Models > User.js > User
  1 import mongoose from 'mongoose'
  2
  3 // userSchema means table ka kaam - userTable bhi likh skte
  4 const userSchema = new mongoose.Schema({
  5
  6   // ab yaha col define kardo
  7   name:{type:String , required:true},
  8   email:{type:String, required:true},
  9   password:{type:String, required:true},
 10   age:{type:Number},
 11   phoneNumber:{type:Number},
 12   createdAt:{
 13     type:Date,
 14     default:Date.now
 15   },
 16 })
 17
 18 export const User = mongoose.model("user" , userSchema)
```

- Ab views mai aake form create kr lenge.

```
12MVCArchitecture > views > index.ejs > html > body > form > input
  1 <!DOCTYPE html>
  2 <html lang="en">
  3 <head>
  4   <meta charset="UTF-8">
  5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
  6   <title>Document</title>
  7 </head>
  8 <body>
  9   <h1>Model Views and Controllers</h1>
 10
 11   <form action="">
 12     Name : <input type="text" name="name" placeholder="Enter name">
 13     Email : <input type="email" name="email" placeholder="Enter email">
 14     Age : <input type="number" name="age" placeholder="Enter age">
 15     Phone Number : <input type="number" name="phonenumer" placeholder="Enter phone number">
 16     Password : <input type="password" name="password" placeholder="Enter password">
 17     <input type="submit" value="Save">
 18   </form>
 19 </body>
 20 </html>
```

Ab ye form bana liya, pr abhi ye chalega nahi, isko chalne ke liye route banana padega, server mai jake banayenge.

```

import express from 'express'
import mongoose from 'mongoose'

const app = express();
app.use(express.urlencoded({
  extended:true
}))

mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
  dbName:"Nodejs Learning"
})
.then(() => console.log("Connected"))
.catch((err) => console.log(err))

app.get('/', (req,res) =>{
  res.render('index.ejs')
})

const port = 2000
app.listen(port , ()=> console.log(`Server is running at port ${port}`))

```

- app.use(express.urlencoded({

extended:true

}))

So express se jo data aayega usko encoded kr denge and ek get route banaya, taki form page pe dikhne lage.

- Ab post route banate hai taki form ka data submit hoke db mai save ho

- Hame pata hai hamara data req.body se aata hai

To pehle dekh lete hai data aa kese raha hai, then db mai save karenge

```

// post route : to save data
app.post('/form-submit' , (req,res) => {
  console.log(res.body);
  res.json({
    msg : "Your form has been submitted",
    success : true
  })
})

```

Ye code likha, then ab form submit karo

/form-submit jo likha hai, isko html ke form ke action mai daal do

```
Server is running at port 2000
Connected
{
  name: 'Lalit Kumar',
  email: 'kumar.lalit.tsx@gmail.com',
  age: '20',
  phonenumer: '07900678792',
  password: '112'
}
```

Is tareeke se data aa raha hai

- Ab is data ko mongodb mai save karna hai.
- To pehle apne model([User.js](#)) ko import krlo, server.js mai, kyuki model ke through hi database mai save kar payenge.

```
import express from 'express'
import mongoose from 'mongoose'
import { User } from './Models/User.js';
```

- Ab apne post route ko thhoda change krenge. Try catch or async await ka use kareng

```

// post route : to save data
app.post('/form-submit' , async (req,res) => {
    console.log(req.body);

    try
    {
        let user = await User.create(req.body);
        res.json({
            msg : "User created successfully",
            NewUser : user,
            success : true,
        });
    }
    catch (error)
    {
        res.json({
            msg : error.message
        });
    }
});

```

**let user = await User.create(req.body)** : ye create() mongoose ka method hai, is create() method ki through jo bhi aap save karoge wo db mai save ho jayega and hame pata hai data req.body se aata to wo hamen pass kar diya.  
**User.create()** mai User hamare model ka naam hai ([User.js](#)) jo hamne upper imprt kiya hai wo

Ye krne ke baad denge ham response, jo pehle dete aare the.

- Or form submit hote hi compass mai dekh skte ho

The screenshot shows the MongoDB Compass interface. On the left, the connection tree is visible, with a yellow highlight on the 'NodeJs Learnin' node under 'Connections'. The main panel shows the 'users' collection under 'NodeJs Learnin > Nodejs\_Learning > users'. A single document is listed:

```

_id: ObjectId('69135892fba83b04f835f5f6')
name : "Lalit Kumar"
email : "kumar.lalit.tsx@gmail.com"
password : "123"
age : 20
createdAt : 2025-11-11T15:38:58.520+00:00
__v : 0

```

- Data submit hone ke baad is trh ka response bhi mila

The screenshot shows a browser window with the URL `localhost:2000/form-submit`. The page displays a JSON response with 'Pretty print' checked:

```

{
  "msg": "User created successfully",
  "NewUser": {
    "name": "Lalit Kumar",
    "email": "kumar.lalit.tsx@gmail.com",
    "password": "123",
    "age": 20,
    "_id": "69135892fba83b04f835f5f6",
    "createdAt": "2025-11-11T15:38:58.520Z",
    "__v": 0
  },
  "success": true
}

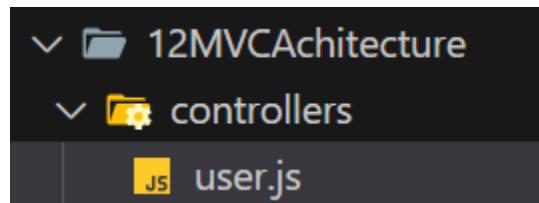
```

Ek id bhi mili hai, or createdAt bhi dekh skte.

Thhoda data or daal liya

- **Controller :**

- Ek controller naam se folder bana lo.



Name chhota file and folder dono ka.

- So jo code hamne [server.js](#) mai post method ke andar likha wo code waha nahi controller ke andar likha jata hai.

```

12MVCArchitecture > controllers > js user.js > [e] userRegister
  1   import { User } from "../Models/User.js";
  2
  3   export const userRegister = async (req,res) => {
  4     console.log(req.body);
  5
  6     try
  7     {
  8       let user = await User.create(req.body);
  9       res.json([
 10         msg : "User created successfully",
 11         NewUser : user,
 12         success : true,
 13       ]);
 14     }
 15     catch (error)
 16     {
 17       res.json({
 18         msg : error.message
 19       });
 20     }
 21   }

```

Ab is controller ko use karnege [server.js](#) mai

- [server.js](#) se ye last line hatao

```

import express from 'express'
import mongoose from 'mongoose'
import { User } from './Models/User.js';

```

And controller ko import karo

```

import express from 'express'
import mongoose from 'mongoose'
import { userRegister } from './controllers/user';

```

- Ab bas us userRegister ko pass kardo post route mai.

```

// post route : to save data
app.post('/form-submit' , userRegister);

```

## Project 1 - URL Shortener

- Sabse pehle project setup karlo

```

server.js U X
13UrlShortner > server.js > ...
1   import express from 'express'
2   import mongoose from 'mongoose';
3
4   const app = express();
5
6   mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
7     dbName:"Nodejs_Learning"
8   }).then(() => console.log("Connected")).catch((err) => console.log(err))
9
10
11  app.get('/', (req,res) =>{
12
13})
14
15  const port = 2000;
16
17
18
19  app.listen(port , () => console.log(`Server is running at port ${port}`));

```

Server or mongoose se connect kr liya ab ui bana lo ejs file mai

```

<h1>URL Shortner Project</h1>
<form action="">
  <label for="longUrl">Paste your URL here</label>
  <input type="text" name="longUrl" id="longUrl" required>
  <button type="submit">Shortner</button>
</form>

<!-- shortUrl ek variable hai, to if shortUrl present hoga to hi neeche shortUrl show hoga, wrna hide ho jayega -->

<% if(shortUrl) {%
  <p>Short Url : <a href="<%= shortUrl%>" target="_blank"></a>
  <%= shortUrl%>
  </p>
<% }%>

```

Itna kaam kiya sirf hamne apne html page mai.

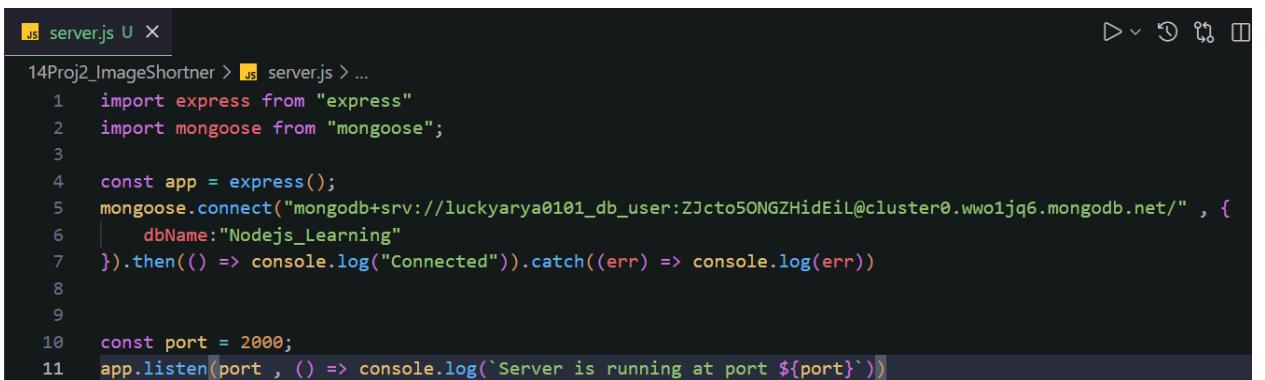
- View ban gaya ab models and controller bana lo like this



- Jese body mai se date lene ke liye likhte hai req.body, wese hi parameter mai se bhi date le skte like <http://localhost:2000/lalit> - yaha pe lalit parameter hai, issse date le skte like this - req.params.id/name

## Project 2 - Image Uploader - Multer & Cloudinary

- Image upload krne ke liye 2 extra library install krni hoti hai
  1. Multer : is package ke through is saara image wagera upload krne ka kaam hota hai. Iske through file upload hoga or cloudinary mai save hoga.
  2. Cloudinary
  3. Npm i multer cloudianary



```

js server.js U X
14Proj2_ImageShortner > js server.js > ...
1  import express from "express"
2  import mongoose from "mongoose";
3
4  const app = express();
5  mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
6    |   dbName:"Nodejs_Learning"
7  }).then(() => console.log("Connected")).catch((err) => console.log(err))
8
9
10 const port = 2000;
11 app.listen(port , () => console.log(`Server is running at port ${port}`))

```

Itna kaam kar liye,

Ab file upload ka karna hai to route banana padega.

```

// rendering ejs file
app.get('/', (req,res) =>{
  |   res.render('index.ejs' , {url:null});
});

```

Url ham pass karenge html part mai uska variable banaya hai, filhal null kr diya.

```

<form action="" method="post" enctype="multipart/form-data">
  |   <div class="form-group">
  |     |   <input type="file" name="file" id="file-input">
  |   </div>

```

Jese ye name="file" likha hai, ye file req.body se aata hai.

```

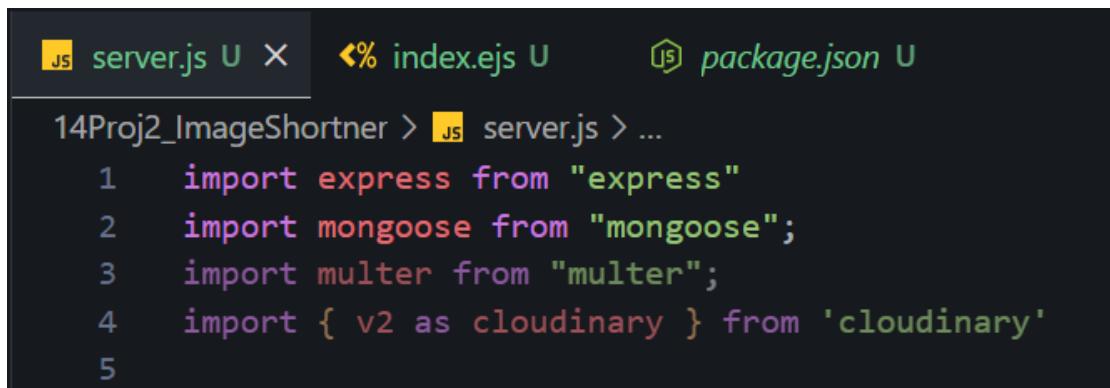
<div class="container">
  <form action="" method="post" enctype="multipart/form-data">
    <div class="form-group">
      <input type="file" name="file" id="file-input">
    </div>
    <div class="form-group">
      <input type="submit" value="Upload">
    </div>
  </form>
</div>

<div id="uploaded-image-container">
  <% if(url) {>
    <h2>File Uploaded Successfully</h2>
    
  <%}>
</div>

```

Neeche logic likha hai ki if image ka url aa raha hai, to hi waha image show krna.

- Ab logic likhna hai ki jo file upload karenge wo cloudinary pe upload honi chahiye na ki mongodb pe. Ham cloudinary pe daalke image ka url le lenge



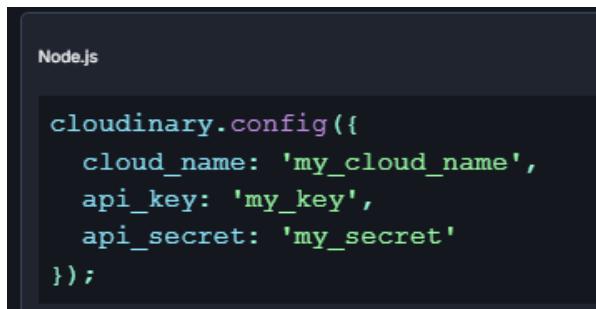
```

server.js U X index.ejs U package.json U
14Proj2_ImageShortner > server.js > ...
1 import express from "express"
2 import mongoose from "mongoose";
3 import multer from "multer";
4 import { v2 as cloudinary } from 'cloudinary'
5

```

Is trh import krna hai. Cloudinary ko import krne ka syntax uski website pe milega.

- Import ke baad configuration add karni hoti hai



```

Node.js

cloudinary.config({
  cloud_name: 'my_cloud_name',
  api_key: 'my_key',
  api_secret: 'my_secret'
});

```

Isme ye cloud name, api key, api secret key dhoondni hai,

API Keys (Cloud name: dr3yyh78j)

Manage API key and secret pairs for your product environment. To build the environment variable for each pair, copy the provided format and substitute your actual values for the placeholders. Make sure to store your secrets securely.

+ Generate New API Key

API environment variable  
CLOUDINARY\_URL=cloudinary://<your\_api\_key>:<your\_api\_secret>@dr3yyh78j

Key Name	Date Created	API Key	API Secret	Status
Root	Nov 12, 2025	451433846639675	*****	Active

Page 1 of 1 < >

Is page se milegi wo.

Dr3yyh78j cloud name hai, jo ki cloudinary url ke end mai @ ke baad likha hai,

```

14Proj2_ImageShortner > server.js > ...
  1 import express from "express"
  2 import mongoose from "mongoose";
  3 import multer from "multer";
  4 import { v2 as cloudinary } from 'cloudinary'
  5
  6
  7 const app = express();
  8
  9 cloudinary.config({
 10   |   cloud_name: 'dr3yyh78j',
 11   |   api_key: '451433846639675',
 12   |   api_secret: 'zMA-pP8AmEt-90bPQ0vSs02zJ_A'
 13 });
 14
 15
 16 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
 17   |   dbName:"Nodejs_Learning"
 18 }).then(() => console.log("Connected")).catch((err) => console.log(err))
 19
 20 // rendering ejs file
 21 app.get('/', (req,res) =>{
 22   |   res.render('index.ejs' , {url:null});
 23 });

```

- Ab npm multer pe chalo, waha file upload krne ka method milega.

```

app.post('/profile', upload.single('avatar'), function (req, res, next) {
  // req.file is the `avatar` file
  // req.body will hold the text fields, if there were any
})

```

Is part ko copy krke past karo home route neeche.

```

mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto50NGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
  dbName:"Nodejs_Learning"
}).then(() => console.log("Connected")).catch((err) => console.log(err))

// rendering ejs file
app.get('/', (req,res) =>{
  res.render('index.ejs' , {url:null});
});

app.post('/upload', upload.single('avatar'), async (req, res) => {

})

const port = 2000;
app.listen(port , () => console.log(`Server is running at port ${port}`))

```

Thhoda modification kiya hai code mai.

- Ab neeche scroll kro multer mai disk storage krke milega kuch

### DiskStorage

The disk storage engine gives you full control on storing files to disk.

```

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, '/tmp/my-uploads')
  },
  filename: function (req, file, cb) {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9)
    cb(null, file.fieldname + '-' + uniqueSuffix)
  }
}

const upload = multer({ storage: storage })

```

Is poore ko copy krke get route se neeche paste krdo.

- Ham isme se unnecessary cheeze hatayenge,  
**destination: './public/uploads'**, ye ek line change ki
- Abhi mere ko extension get krna hai image ka to path module ko import kr lete hai.
- Ye bana final disk waala code :

```

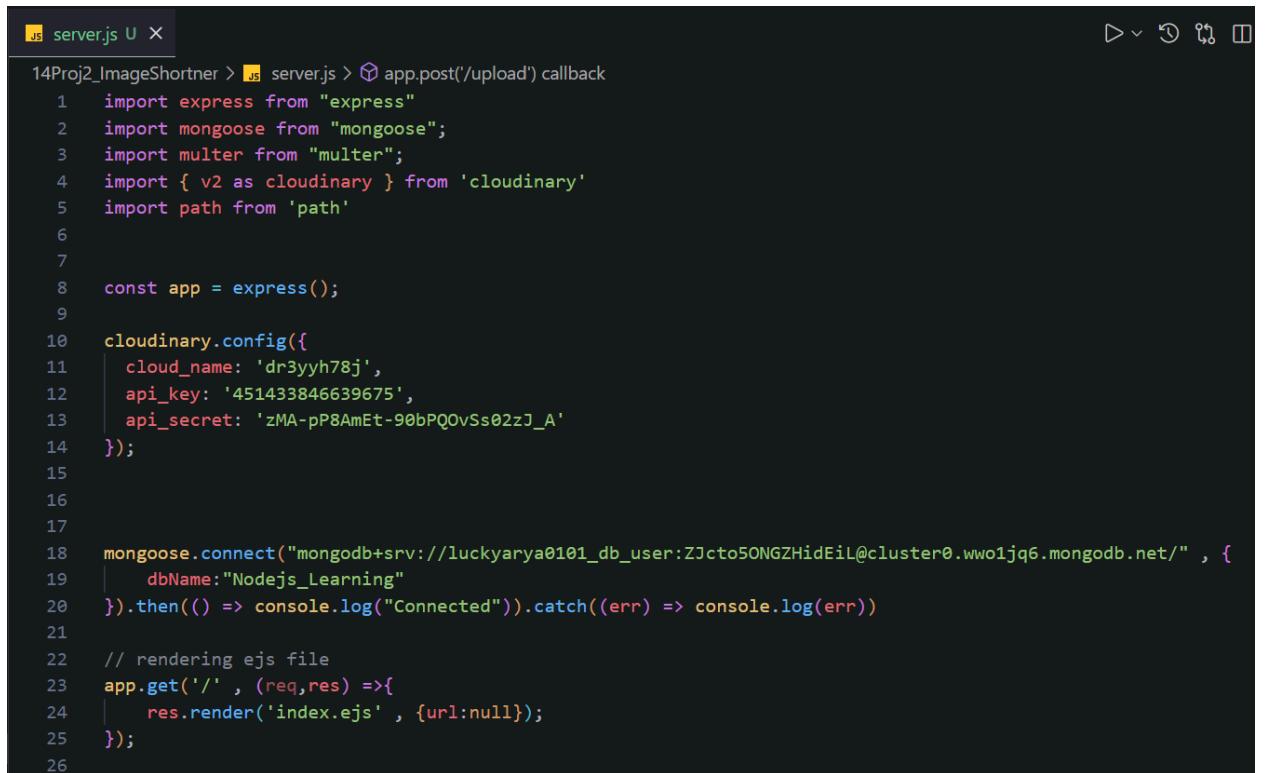
const storage = multer.diskStorage({
  destination: './public/uploads',
  filename: function (req, file, cb) {
    const uniqueSuffix = Date.now() + path.basename(file.originalname); // yaha extension nikaal liya
    cb(null, file.fieldname + "-" + uniqueSuffix)
  }
})

```

```
app.post('/upload', upload.single('file'), async (req, res) => {
  const file = req.file.path; // is line se file ke naam ka complete path aa jayega
  res.json({
    message: 'file uploaded'
  })
})
```

Itna krke chalaya to file upload ho rahi hai.

- Server js ka poora code dekho ab :



The screenshot shows a code editor window with the file 'server.js' open. The code is written in JavaScript and performs several tasks:

- Imports necessary modules: express, mongoose, multer, and cloudinary.
- Configures cloudinary with specific cloud name, API key, and secret.
- Connects to a MongoDB database using mongoose.
- Defines a route for '/upload' using the multer single file middleware.
- Handles the '/upload' route by saving the uploaded file to disk and returning a JSON response indicating success.
- Defines a route for '/' that renders an EJS file named 'index.ejs'.

```
14Proj2_ImageShortner > server.js > app.post('/upload') callback
1 import express from "express"
2 import mongoose from "mongoose";
3 import multer from "multer";
4 import { v2 as cloudinary } from 'cloudinary'
5 import path from 'path'
6
7
8 const app = express();
9
10 cloudinary.config({
11   cloud_name: 'dr3yyh78j',
12   api_key: '451433846639675',
13   api_secret: 'zMA-pP8AmEt-90bPQ0vSs02zJ_A'
14 });
15
16
17
18 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJctoSONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
19   dbName:"Nodejs_Learning"
20 }).then(() => console.log("Connected")).catch((err) => console.log(err))
21
22 // rendering ejs file
23 app.get('/', (req,res) =>{
24   res.render('index.ejs' , {url:null});
25 });
26
```

```

27  const storage = multer.diskStorage({
28    destination: './public/uploads',
29    filename: function (req, file, cb) {
30      const uniqueSuffix = Date.now() + path.basename(file.originalname); // yaha extension nikaal liya
31      cb(null, file.fieldname + "-" + uniqueSuffix)
32    }
33  })
34
35  const upload = multer({ storage: storage })
36
37
38  app.post('/upload', upload.single('file'), async (req, res) => {
39    const file = req.file.path; // is line se file ke naam ka complete path aa jayega
40    res.json({
41      message:'file uploaded'
42    })
43  })
44
45  const port = 2000;
46  app.listen(port , () => console.log(`Server is running at port ${port}`))

```

- Ab mujhe pehle ye code ko cloud pe save krna hai then image ke url ko mongodb pe.  
To pehle cloudinary pe upload krte hai then mongodb pe.  
Iske liye uploader function milta hai  
Cloudinary pe mil jayega wo function. Pr usko yaad krna hi better option hai.

```

app.post('/upload', upload.single('file'), async (req, res) => {
  const file = req.file.path; // is line se file ke naam ka complete path aa jayega

  const cloudinaryResponse = await cloudinary.uploader.upload(file,{
    folder:"NodeJs_Learning"
  })

  res.json({
    message:'file uploaded',
    cloudinaryResponse
  })
})

```

- Iske baad file upload krke dekhna, ye response mila

```

{
  "message": "file uploaded",
  "cloudinaryResponse": {
    "asset_id": "8271c1b2dd9417168bf3bfa27ce24603",
    "public_id": "NodeJs_Learning/fiavxatri3sd28tvno43",
    "version": 1762954340,
    "version_id": "9e0bc132cdb074a22789c52831a4f507",
    "signature": "3805d780285cdad43e43cc01f806b245dd21bce4",
    "width": 1600,
    "height": 900,
    "format": "png",
    "resource_type": "image",
    "created_at": "2025-11-12T13:32:00Z",
    "tags": [],
    "bytes": 710420,
    "type": "upload",
    "etag": "d12b7b24a237ad1d567941bcf6fb915f",
    "placeholder": false,
    "url": "http://res.cloudinary.com/dr3yyh78j/image/upload/v1762954340/NodeJs_Learning/fiavxatri3sd28tvno43.png",
    "secure_url": "https://res.cloudinary.com/dr3yyh78j/image/upload/v1762954340/NodeJs_Learning/fiavxatri3sd28tvno43.png",
    "asset_folder": "NodeJs_Learning",
    "display_name": "fiavxatri3sd28tvno43",
    "original_filename": "file-1762954322080",
    "api_key": "451433846639675"
  }
}

```

Isme jo public\_id hai, uss id se baad mai image ko update ya delete bhi kr skte hai.

Like primary key hai ye.

Url jo likha hai, us link ko open krne pe uploaded image dikh jayegi

And iske neeche jo secure url hai isko rakhna hai database mai mongodb ke.

- To schema bana lete hai, or schema is baar isi server js mai hi bana re.  
Db connectivity ke neeche banana bas kahi bhi.

```

const imageSchema = new mongoose.Schema({
  filename : String,
  public_id : String,
  imgUrl : String
  // ye 2 wo hi hai jo cloudnary ke url mai dikha tha.
})
const File = mongoose.model("cloudinary" , imageSchema);

```

Ye hamara schema ready.

- Full server cde :

```
JS server.js U X
14Proj2_ImageShortner > JS server.js > ...
1   import express from "express"
2   import mongoose from "mongoose";
3   import multer from "multer";
4   import { v2 as cloudinary } from 'cloudinary'
5   import path from 'path'
6
7
8   const app = express();
9
10  cloudinary.config({
11    cloud_name: 'dr3yyh78j',
12    api_key: '451433846639675',
13    api_secret: 'zMA-pP8AmEt-90bPQ0vSs02zJ_A'
14  });
15
16
17
18  mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
19    dbName:"Nodejs_Learning"
20  }).then(() => console.log("Connected")).catch((err) => console.log(err))
21
22 // rendering ejs file
23 app.get('/', (req,res) =>{
24   res.render('index.ejs' , {url:null});
25 });
26
27 const storage = multer.diskStorage({
28   destination: './public/uploads',
29   filename: function (req, file, cb) {
30     const uniqueSuffix = Date.now() + path.extname(file.originalname); // yaha extension nikaal liya
31     cb(null, file.fieldname + "-" + uniqueSuffix)
32   }
33 })
34
35 const upload = multer({ storage: storage })
36
37 const imageSchema = new mongoose.Schema({
38   filename : String,
39   public_id : String,
40   imgUrl : String
41   // ye 2 wo hi hai jo cloudinary ke url mai dikha tha.
42 })
43 const File = mongoose.model("cloudinary" , imageSchema);
44
```

```

46 app.post('/upload', upload.single('file'), async (req, res) => {
47   const file = req.file.path; // is line se file ke naam ka complete path aa jayega
48
49   const cloudinaryResponse = await cloudinary.uploader.upload(file,{
50     folder:"NodeJs_Learning"
51   })
52
53   // ab database mai save karlo
54   const db = await File.create({
55     filename : file.originalname,
56     public_id : cloudinaryResponse.public_id,
57     imgUrl : cloudinaryResponse.secure_url
58   })
59
60   res.render('index.ejs',{
61     url:cloudinaryResponse.secure_url
62   })
63
64   // res.json({
65   //   message:'file uploaded',
66   //   cloudinaryResponse
67   // })
68 })
69
70 const port = 2000;
71 app.listen(port , () => console.log(`Server is running at port ${port}`))

```

## Project 3 - Full Stack Authentication With File Upload

1. Npm init -y
2. Npm express ejs mongoose multer cloudinary
  - Ye sab install karlo
  - Project 2 ke server ka code as it is copy past kar diya project 3 ki server file mai.
  - Jayada kuch khaas nahi tha
  - Ye schema bananya :

```

const userSchema = new mongoose.Schema({
  name:String,
  email:String,
  password : String,
  filename : String,
  public_id : String,
  imgUrl : String
})
const User = mongoose.model("userNew" , userSchema);

```

- Register krke login page pe redirect kr diya

```

app.post('/register', upload.single('file'), async (req, res) => {
  const file = req.file.path; // is line se file ke naam ka complete path aa jayega

  // const name = req.body.name;
  // const email = req.body.email;
  // const password = req.body.password;
  // ye teeno variable ko 1 line mein bhi bana skta hoon
  const {name , email , password} = req.body

  const cloudinaryResponse = await cloudinary.uploader.upload(file,{
    folder:"NodeJs_Learning"
  })

  // ab database mai save karlo
  const db = await User.create([
    name,
    email,
    password,
    filename : file.originalname,
    public_id : cloudinaryResponse.public_id,
    imgUrl : cloudinaryResponse.secure_url
  ])

  res.redirect('/login')
})

```

```

// login logic
app.post('/login' , async (req,res) => {
  const {email,password} = req.body;

  // check karo ki user exist bhi karta h ya nahi
  let user = await User.findOne({email})
  if(!user){
    res.render('login.ejs')
  }
  else if(user.password != password){
    res.render('login.ejs')
  }
  else{
    res.render('profile.ejs', {user})
  }
})

```

- Ye raha login ka logic

# API

- API : application programming interface
  - So hamare pass 3 cheeze hoti hai frontend , backend, and API  
To client and server ke beech mai communication karwane ka kaam karta hai.  
So api ek chhota sa part, backend mai se kuch leke aana hai ya kuch leke jana hai use krne ka kaam karta hai.

## ● Project

- Ab npm init -y
- Isme ejs ka koi kaam nahi aane waala.

**Ham isme ejs ko render nahi karenge, isme response denge json ke format mai, then us API ko aap kisi bhi framwork mai use kr skte like react, angular, jquery etc.**

**Pr agr apne ejs ko render karwa diya to ham koi doosre framework mai usko use nahi kar sakte, cuz waha pe ejs already frontend ki trh kaam kar raha hai.**

- Npm i express mongoose
- Itna karne ke baar server bana lo + mongoose connect krlo

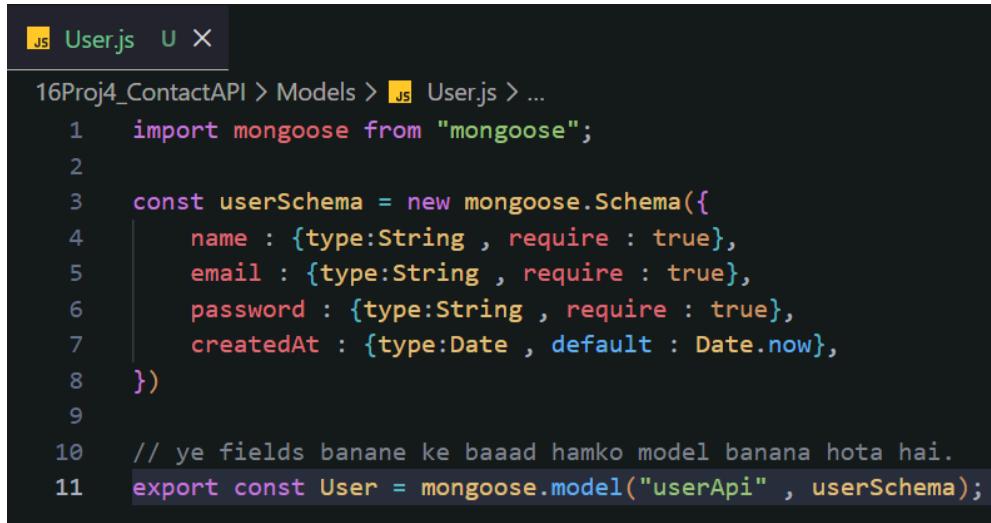


A screenshot of a code editor showing the `server.js` file. The code is written in Node.js and uses Express and Mongoose. It connects to a MongoDB database and starts a server on port 2000, logging the port number.

```
package.json M server.js U X
16Proj4_ContactAPI > server.js > ...
1 import express from 'express'
2 import mongoose from 'mongoose'
3
4 const app = express();
5
6 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
7   dbName: "Nodejs_Learning"
8 })
9 .then(() => console.log("Connected"))
10 .catch((err) => console.log(err))
11
12 const port = 2000;
13 app.listen(port , () => console.log(`Server is running on port ${port}`))
```

- Ab mai banaunga user route. Jaha user ake login wagera karega
- MVC ka use karunga pr Views to dikhana nahi API mai to views folder nahi banaunga

- To model pehle bana liya



```

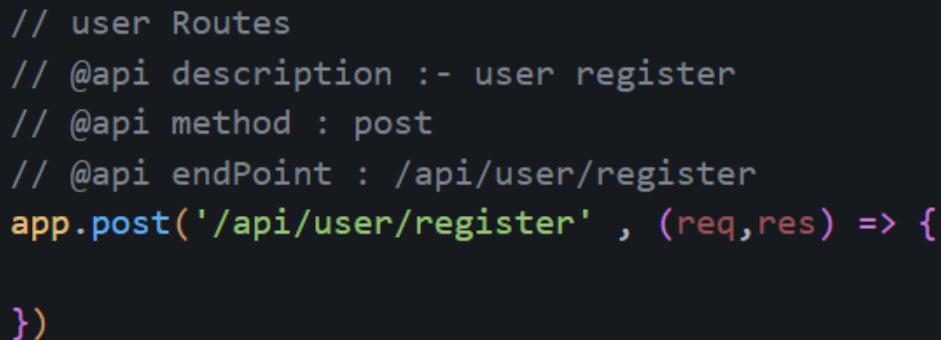
User.js  U X

16Proj4_ContactAPI > Models > User.js > ...
1   import mongoose from "mongoose";
2
3   const userSchema = new mongoose.Schema({
4       name : {type:String , require : true},
5       email : {type:String , require : true},
6       password : {type:String , require : true},
7       createdAt : {type:Date , default : Date.now},
8   })
9
10 // ye fields banane ke baaad hamko model banana hota hai.
11 export const User = mongoose.model("userApi" , userSchema);

```

Ye hi industy ki best practise hai, ise follow karo model banate hue.

- Itna karne ke baad ab hame routes banane hai, server ki file hai.  
So user ko register krne ka route banata hoon, post request ke liye.



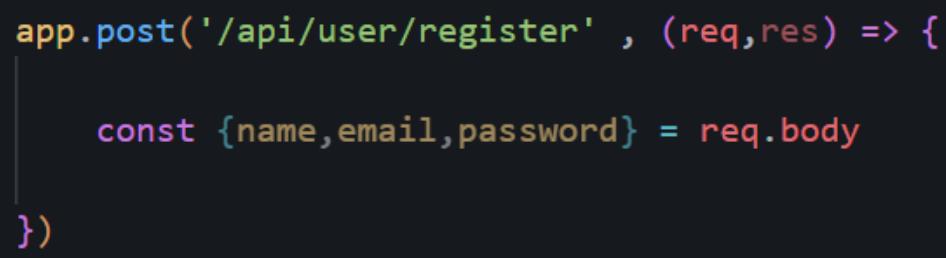
```

// user Routes
// @api description :- user register
// @api method : post
// @api endPoint : /api/user/register
app.post('/api/user/register' , (req,res) => {
    })

```

Is tareeke se comment likhte hai.

- Ab data we know hamara req.body se aata hai.



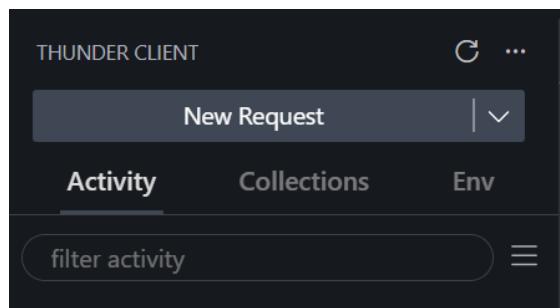
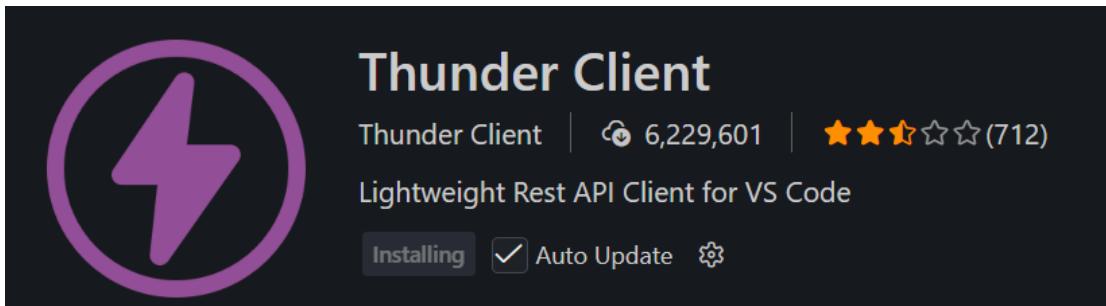
```

app.post('/api/user/register' , (req,res) => {
    const {name,email,password} = req.body
})

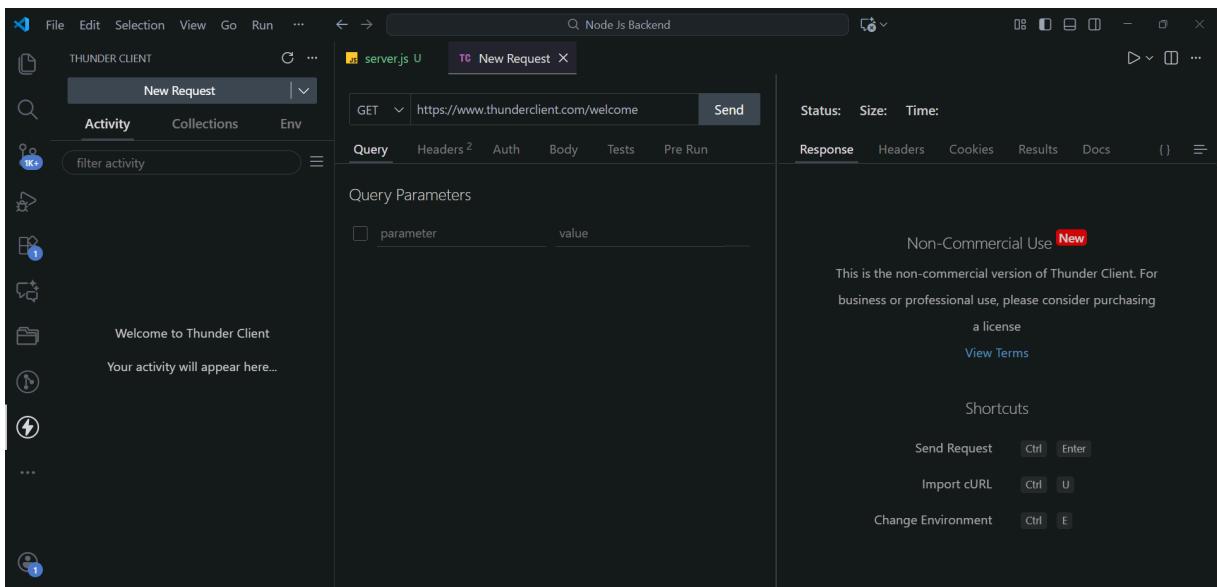
```

Abhi tk ham ejs frontend ka use krke ye data leke aare the, pr ab frontend to use kar nahi re to ye data hamara post man ke through aayega, waha se testing hogi.  
Postman hame bina frontend ke frontend ki poori facility deta hai.  
So bina form create kiya ham data daal payenge.

- `/api/user/register` : ye hi hamari api hoti hai, iski testing hame karni hai.
- To testing ke liye post man ko vs code mai install kr skte, ya fir ek or tool hai thunder client, uska use ham karne waale hai. Ye hi extension hai



- Open extension:
- Click on new req.



Waha url daalna, the right mai response milega tumhe

- To ek testing ki api bana lete hai home route ki pehle test krne ke liye ki thunder sahi chal raha h ya nahi.

```
// home api
app.post('/', (req,res) => {
  res.json({
    message : "Every thing is fine"
  });
});
```

The screenshot shows a Postman interface. On the left, there's a file tree with 'server.js U' and a tab bar with 'New Request X'. A search bar at the top right contains 'Status: 200 OK Size: 33 Bytes Time: 7 ms'. Below it, a toolbar has 'Send' and other options like Headers, Cookies, Results, Docs, and a copy icon. The main area has tabs for 'Query', 'Headers 2', 'Auth', 'Body', 'Tests', and 'Pre Run'. Under 'Query', there's a table for 'Query Parameters' with columns for 'parameter' and 'value'. The 'Body' tab is selected, showing a JSON response:

```
1  {
2    "message": "Every thing is fine"
3  }
```

To sahi kaam kar raha hai.

- Ab us post api ko complete karte hai.

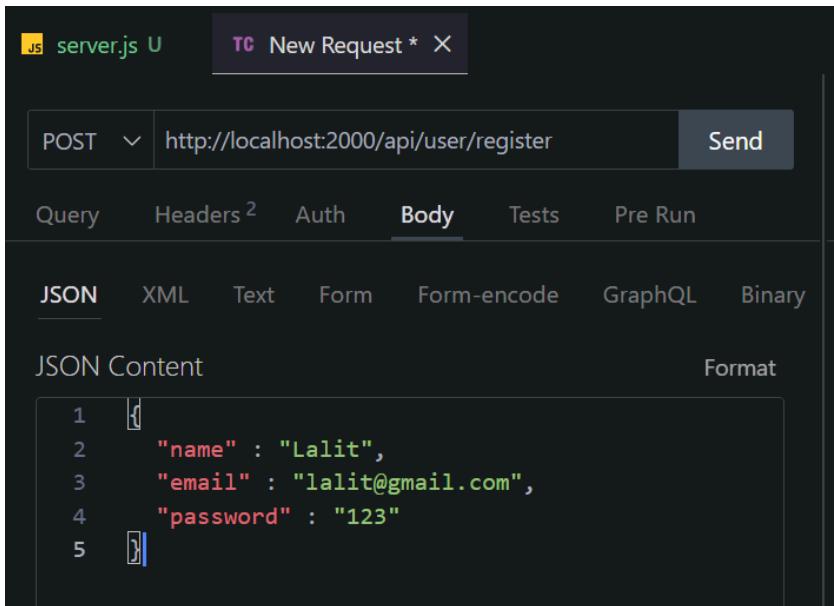
```
// user Routes
// @api description :- user register
// @api method : post
// @api endPoint : /api/user/register
app.post('/api/user/register' , (req,res) => {

  // const {name,email,password} = req.body
  console.log(req.body);

})
```

Ek baar log mai print karwak dekhte hai sbse pehle.

Ab is : **/api/user/register** isko postman mai daalo.



Is trh, method ko POST kara, neeche JSON form mai data diya.  
Abhi send karoge to koi data nahi dikhega, cuz

```

app.post('/api/user/register' , (req,res) => {

  // const {name,email,password} = req.body
  console.log(req.body);

})

```

Yaha kuc response diya nhi hai, or log mai bhi abhi kuch print nahi hoga.

- So testing tool se to hamne data bhej diya, pr express ko thhodi pata hai ki ham data bhejne waale hai, to log pe print hoga hi nahi, cuz express use parse hi nahi kar paa raha hai. Parse krne ke liye body parser ka use karte hai. Ye express hi deta hai.  
Ise import krlo pehle

```

import mongoose from 'mongoose'
import bodyParser from 'express'

```

Ab ise use karna hai.

```

app.use(bodyParser.json())

```

Is line ka mtlb hai, body mai se parse krke json mai jo data aara hai uske lelo, cuz thunder mai ham json se data dere body ke andar.

```

Server is running on port 2000
Connected
{ name: 'Lalit', email: 'lalit@gmail.com', password: '123' }

```

So console pe wo data hame mil gaya.

```

// user Routes
// @api description :- user register
// @api method : post
// @api endPoint : /api/user/register
app.post('/api/user/register' , (req,res) => {

    // const {name,email,password} = req.body
    console.log(req.body);
    res.json({
        message:'getting data from req.body',
        data : req.body
    })
})

```

Ab hamne response bhi bhej diya, tool pe jake ab dobara data bhejo

The screenshot shows the Postman interface with the following details:

- Request URL:** POST http://localhost:2000/api/user/register
- Body Content:**

```

{
  "name": "Lalit",
  "email": "lalit@gmail.com",
  "password": "123"
}

```
- Response Headers:** Status: 200 OK, Size: 107 Bytes, Time: 48 ms
- Response Body:**

```

1  []
2   "message": "getting data from req.body",
3   "data": {
4     "name": "Lalit",
5     "email": "lalit@gmail.com",
6     "password": "123"
7   }
8 []

```
- Terminal Output:**

```

at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
at async ModuleJob._link (node:internal/modules/esm/module_job:162:19)

Node.js v22.18.0
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting 'node ./server.js'
Server is running on port 2000
Connected
{ name: 'Lalit', email: 'lalit@gmail.com', password: '123' }

```

Ab data dono jgh dikhega, console pe bhi or tool pe bhi

- Ab karenge user ko register. Usi api mai se console waala hata do.

Hame model ki jarort padegi, to model ko import karlo

```
import express from 'express'
import mongoose from 'mongoose'
import bodyParser from 'express'
import User from './Models/User.js'
```

```
// user Routes
// @api description :- user register
// @api method : post
// @api endPoint : /api/user/register
app.post('/api/user/register' , async (req,res) => {

    // console.log(req.body);

    const {name,email,password} = req.body;
    if(name == "" || email == "" || password == ""){
        return res.json({ // .json likha hai to json ki form mai msg do
            message : "All fields are required"
        });
    }

    // Adding the user into DB
    let user = await User.create({name,email,password});

    res.json({
        message:'User created successfully',
        success : true,
        user : user // jisse pata chal jaye ki konsa user register hua hai
    })
})
```

Ye poora create ka api likh liya validation ke sath, ab ise test karo tool pe.

The screenshot shows the Postman interface with the following details:

- Request Method: POST
- URL: http://localhost:2000/api/user/register
- Response Status: 200 OK
- Response Size: 201 Bytes
- Response Time: 92 ms
- Response Body (JSON):

```
1  {
2      "message": "User created successfully",
3      "success": true,
4      "user": {
5          "name": "Lalit",
6          "email": "lalit@gmail.com",
7          "password": "123",
8          "_id": "6915a8bd018efcd38c8c1287",
9          "createdAt": "2025-11-13T09:45:33.519Z",
10         "__v": 0
11     }
12 }
```

Yaha hame mongodb ka id bhi dekhne ko mil gaya.

The screenshot shows the MongoDB Compass interface. In the top left, there's a tree view with 'userapis' selected. Below it, a search bar shows 'NodeJs Learnin > NodeJs\_Learning > userapis'. The main area is titled 'Documents' with a count of 1. A query input field says 'Type a query: { field: 'value' } or Generate query'. Below the input are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. To the right are navigation buttons for 'Find' and 'Options'. At the bottom, a table shows the single document with the following data:

	_id	name	email	password	createdAt	__v
	ObjectId('6915a8bd018efcd38c8c1287')	"Lalit"	"Lalit@gmail.com"	"123"	2025-11-13T09:45:33.519+00:00	0

Database mai bhi save ho gaya.

- Ab ek check or lagate hai, same email se koi doosri baar register nahi kar skta. Check karna hoga ki user kahi already exists to nahi karta db mai. To save krne se pehle check karlo

```
app.post('/api/user/register' , async (req,res) => {
    // console.log(req.body);

    const {name,email,password} = req.body;
    if(name == "" || email == "" || password == ""){
        return res.json({ // .json likha hai to json ki form mai msg do
            message : "All fields are required"
        });
    }

    let user = await User.findOne({email}) // email ke basis pe find krr
    if(!user){
        // Adding the user into DB
        user = await User.create({name,email,password});
    }
    else{
        res.json({
            message : "User already exist",
            success : false
        })
    }

    res.json({
        message:'User created successfully',
        success : true,
        user : user // jisse pata chal jaye ki konsa user register hua hai
    })
})
```

To ye kar liya hamne

## ● Bcrypt js

- Ab jese hamne ye password ko plain mai save kr diya, pr ham password ko kabhi plain mai save nahi karte, ham is library ka use karte hai.  
Ise kehte hai password ko hash karna.
- Type npm i bcryptjs and then import it : import bcrypt from 'bcryptjs'
- DB mai add krne se pehle hamko hash karna hai  
Hame ek hash() method milta hai. Iske and 2 cheeze pass krte,
  1. Password variable
  2. Kitni lenght ka encryption chahiye

```
let user = await User.findOne({email}) // email ke basis pe find krre
if(!user){

    // DB mai add krne se pehle hamko hash karna hai
    const hashPassword = await bcrypt.hash(password,10);

    // Adding the user into DB
    user = await User.create({name,email,password : hashPassword});
}

else{
    res.json({
        message : "User already exist",
        success : false
    })
}
```

Is hashPassword variable ko pass kr dena fir save krate time.

- Ab ek baar chala ke dekhte hai.

The screenshot shows a Postman interface with a successful API call. The left pane displays a POST request to `http://localhost:2000/api/user/register`. The right pane shows the response details: Status: 200 OK, Size: 258 Bytes, Time: 336 ms. The response body is a JSON object:

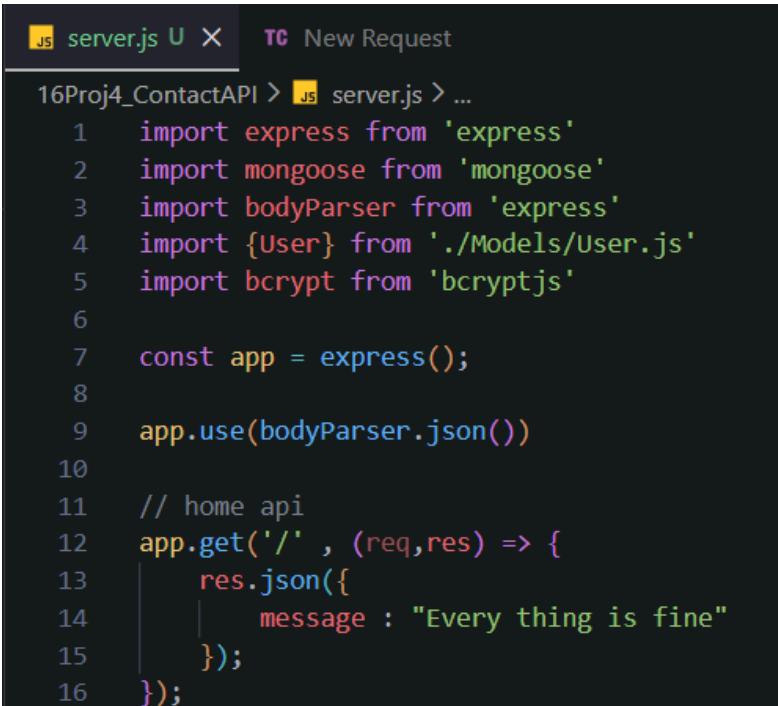
```
1  {
2      "message": "User created successfully",
3      "success": true,
4      "user": {
5          "name": "Lucky",
6          "email": "lucky@gmail.com",
7          "password": "$2b$10$K1aCfm0TUJowaVq.9Scevu9NUqpARe
               .LJ9qYpozvfQ7dSvslvx3bu",
8          "_id": "6915adc9ea7d31145b1638dd",
9          "createdAt": "2025-11-13T10:07:05.466Z",
10         "__v": 0
11     }
12 }
```

Left mai dekho kya data diya and right mai dekho kya password bana.

ADD DATA	EXPORT DATA	UPDATE	DELETE
<pre>_id: ObjectId('6915adc9ea7d31145b1638dd') name : "Lucky" email : "lucky@gmail.com" password : "\$2b\$10\$K1aCfm0TUJowaVq.9Scevu9NUqpARe.LJ9qYpozvfQ7dSvslvx3bu" createdAt : 2025-11-13T10:07:05.466+00:00 __v : 0</pre>			
<pre>_id: ObjectId('6915cdf5ea7d31145b1638e3') name : "Ricky" email : "ricky@gmail.com" password : "\$2b\$10\$k6d4GQ0iQ4eOCvX5.D5Y5el5Tf1yAgD6NEfkuoJbUBrXm7AEy24hG" createdAt : 2025-11-13T12:24:21.231+00:00 __v : 0</pre>			

Table mai bhi data hash mai save hora hai.

- [server.js](#) ka poora code dera hoon



```
16Proj4_ContactAPI > js server.js < New Request

1   import express from 'express'
2   import mongoose from 'mongoose'
3   import bodyParser from 'express'
4   import {User} from './Models/User.js'
5   import bcrypt from 'bcryptjs'
6
7   const app = express();
8
9   app.use(bodyParser.json())
10
11  // home api
12  app.get('/', (req, res) => {
13    |  res.json({
14    |    |  message : "Every thing is fine"
15    |  });
16});
```

```

server.js U X TC New Request
16Proj4_ContactAPI > server.js > ...
19 // user Routes
20 // @api description :- user register
21 // @api method : post
22 // @api endPoint : /api/user/register
23 app.post('/api/user/register' , async (req,res) => {
24
25     // console.log(req.body);
26
27     const {name,email,password} = req.body;
28     if(name == "" || email == "" || password == ""){
29         return res.json({ // .json likha hai to json ki form mai msg do
30             message : "All fields are required"
31         });
32     }
33
34     let user = await User.findOne({email}) // email ke basis pe find krre
35     if(!user){
36
37         // DB mai add krne se pehle hamko hash karna hai
38         const hashPassword = await bcrypt.hash(password,10);
39
40         // Adding the user into DB
41         user = await User.create({name,email,password : hashPassword});
42     }
43     else{
44         res.json({
45             message : "User already exist",
46             success : false
47         })
48     }
49
50
51
52     res.json({
53         message:'User created successfully',
54         success : true,
55         user : user // jisse pata chal jaye ki konsa user register hua hai
56     })
57 }
58
59
60
61 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto50NGZHidEiL@cluster0.ww01jq6.mongodb.net/" , {
62     dbName:"Nodejs_Learning"
63 })
64 .then(() => console.log("Connected"))
65 .catch((err) => console.log(err))
66
67 const port = 2000;
68 app.listen(port , () => console.log(`server is running on port ${port}`))

```

- Abhi is file ko thhoda clean karte hai. Ye post waala poora code controller mai file bana ke karenge.
- [server.js code now :](#)

```
server.js U X user.js U New Request
16Proj4_ContactAPI > server.js > ...
1   import express from 'express'
2   import mongoose from 'mongoose'
3   import bodyParser from 'express'
4   import { register } from './Controllers/user.js';
5
6   const app = express();
7
8   app.use(bodyParser.json())
9
10 // home api
11 app.get('/', (req, res) => {
12   res.json({
13     message : "Every thing is fine"
14   });
15 });
16
17
18 // user Routes
19 // @api description :- user register
20 // @api method : post
21 // @api endPoint : /api/user/register
22 app.post('/api/user/register' , register);
23
24
25
26 mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
27   dbName:"Nodejs_Learning"
28 }
29 .then(() => console.log("Connected"))
30 .catch((err) => console.log(err))
31
32 const port = 2000;
33 app.listen(port , () => console.log(`Server is running on port ${port}`))
```

- Controller code:

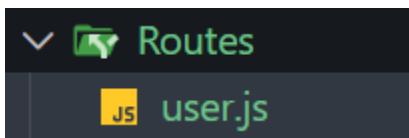
```

16Proj4_ContactAPI > Controllers > user.js > [register]
  1 import {User} from '../Models/User.js'
  2 import bcrypt from 'bcryptjs'
  3
  4 export const register = async (req,res) => {
  5
    // console.log(req.body);
    const {name,email,password} = req.body;
    if(name == "" || email == "" || password == ""){
      return res.json({ // .json likha hai to json ki form mai msg do
        message : "All fields are required"
      });
    }
  13
  14 let user = await User.findOne({email}) // email ke basis pe find krra
  15 if(!user){
  16
    // DB mai add krne se pehle hamko hash karna hai
    const hashPassword = await bcrypt.hash(password,10);
  19
  20     // Adding the user into DB
  21     user = await User.create({name,email,password : hashPassword});
  22   }
  23   else{
  24     res.json({
  25       message : "User already exist",
  26       success : false
  27     })
  28   }
  29
  30   res.json({
  31     message:'User created successfully',
  32     success : true,
  33     user : user // jisse pata chal jaye ki konsa user register hua hai
  34   })
  35 }

```

import express from 'express'  
 import mongoose from 'mongoose'  
 Ye line bhi add krna top pe

- Abhi ek or cheez saafi karni hai.  
`app.post('/api/user/register' , register);` : jese maine ye register ka route banaya, agr mujhe login ka route banana hota to? Ya fir 10 or route banane hote to [server.js](#) mai hi nahi karta, mai route ka ek alg folder banaunga, or usme karunga ye kaam.



- Ab is file ke andar router banana hoga.

```
JS user.js U X TC New Request
16Proj4_ContactAPI > Routes > JS user.js > [?] default
1   import express from 'express'
2
3   const router = express.Router();
4
5   export default router;
```

- Ab isme denge wo api waala, pr isme short mai dena hota hai.

```
JS user.js U X JS server.js U TC New Request
16Proj4_ContactAPI > Routes > JS user.js > [?] default
1   import express from 'express'
2   import { register } from './Controllers/user.js';
3
4   const router = express.Router();
5
6   // user registration
7   // @api description :- user register
8   // @api method : post
9   // @api endPoint : /api/user/register
10  router.post('/register' , register)
11
12  export default router;
```

Jese server mai [app.post](#) kte the, yaha pe [router.post](#) kr re, or /api/user/regiter ki jgh sirf /register dena hai is file mai. Yaha se maine export kiya hai, waha jake ise import krna hoga

Ab chalo [server.js](#) mai , waha bhi kuch alg dena hota hai.

```
import userRouter from './Routes/user.js'
```

```
// user Routes
app.post('/api/user' , userRouter);
```

- wrong

[server.js](#) mai ab [app.post](#) nahi hoga, app.use hoga

```
// user Routes
app.use('/api/user' , userRouter);
```

app.use() middleware ki trh kaam karega, as we know from .net ki trh hai ye.

- Hoga kya ab.  
Maine server waale mai route diya hai : /api/user  
Maine route file mai likha hai : /register  
To express in dono ko automatically combine/merge kar dega : /api/user/register  
/register : ise kehte hai end point. Mujhe sirf ye change krna hota hai, like log in ka endpoint will be /login
- Server waala code to complete hai, like ye line `app.use('/api/user' , userRouter);`, ye complete hai cuz itna part `/api/user` to command rahega, then `userRouter` se agr register ki req aayegi to url banega `/api/user/register` and if `userRouter` se login ki request aayi to route banega `/api/user/login`.
- To hame bas route file mai new route create krne hai bas.
- Server code ko merge krti hai, saara code server se link hona jaroori hai.

## ● Login API

- To route file mein hamko bas itna karna hai

```
// user login
// @api description :- user login
// @api method : post
// @api endPoint : /api/user/login
router.post('/login' , login)
```

Ab controller mai jake ek login function banana hoga, use route mai import krke jaha maine login pass kar hi diya hai '/login' ke bagal mai jo login hai wo controller se aaya hai, ise upper import krlo route mai like this

```
user.js  U X  server.js U  New Request
16Proj4_ContactAPI > Routes > user.js > ...
1 import express from 'express'
2 import { register , login } from '../Controllers/user.js';
```

```
export const register = async (req,res) => { ...  
}  
  
export const login = async (req,res) => {  
}
```

Ab iska code likhte hai.

To req.body se lenge email and password uske basis pe check karenge.

```
export const login = async (req,res) => {  
    const {email,password} = req.body  
}  
}
```

Is kaam ko ese bhi kr skte,

Const email = req.body.email

Const password = req.body.password

Pr ye long way hai

```
export const login = async (req,res) => {
    const {email,password} = req.body

    if(User.email == "" || User.password == ""){
        return res.json({
            message : "All fields are required"
        });
    }

    const user = await User.findOne({email})
    if (!user) {
        res.json({
            message:"User does not exist",
            success:false
        })
    }
}
```

Ab jese maine last mai check kr liya user us email se exists nhi krta to msg dikha do.  
Ab password bhi to verify krna hoga na , hame make sure krna hoga user sahi password daalke hi enter kare, and password hamne encrypt kiya hai, use decrypt bhi karna hoga.

```
// dcrypting password
const validPassword = await bcrypt.compare(password , user.password)
```

Is line mein hamne basically compare karwa ki user textbox pe password dera hai kya wo equal hai jo password database mai save hai.

- Ab chala ke dekhte hai.

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:2000/api/user/login
- Status:** 200 OK
- Size:** 40 Bytes
- Time:** 145 ms
- Body Content:**

```

1  {
2      "email" : "gaj@gmail.com",
3      "password" : "123"
4  }
    
```
- Response Content:**

```

1  {
2      "message": "Welcome Gaj",
3      "success": true
4  }
    
```

Waha POST rakha hai, neeche credentials daale right mai result dekh skte ho.

## ● Cookies/Token

- Jese ham instagram app mai login krte hai, fir kitni baar bhi app ban kare to wapas se login krne ki jaroor nahi hoti, to ye kaam hota kese hota hai ki like ek baar hamne login kr diya, to us information ko ham ek token mai save kr dete hai. Then jab ham frontend banate hai to un token ko cookies ya session mai store klete hai.
- To login krne ke baad user ki info ko ek token mai daalna hoga or usko bhejna hoga jab frontend karenge
- User ko info ko token mai daalne ke liye ek solid secure token hona chahiye, or uske liye use karte hai JWT token.
- JWT : Json Web Token
- npm i jsonwebtoken kro
- Ab controller mai kaam hai saara, import karlo jwt ko

```
import jwt from 'jsonwebtoken'
```

```

// decrypting password
const validPassword = await bcrypt.compare(password , user.password)
if (!validPassword) {
  return res.json({
    message:"Invalid password",
    success:false
  })
}
else{
  res.json({
    message:`Welcome ${user.name}`,
    success:true
  })
}
  
```

Yaha use karo jwt ko successfull login krne ke baad, else part mai.

- Ab mujhe jwt mai ek sign function milte hai. Ye 3 cheeze leta hai
  1. userId or user
  2. Secret key, means wo information jiske through aap isko verify karoge baad mai, ye ham apne hisaab se kuch bhi bana sakte hai
  3. Expiresin , kitne din ami expire hoga

```
// decrypting password
const validPassword = await bcrypt.compare(password , user.password)
if (!validPassword) {
    return res.json({
        message:"Invalid password",
        success:false
    })
}
else{
    // jwt
    const token = jwt.sign({user : user} , "mysecretkey@123" ,{ expiresIn: "1h" })

    res.json({
        message:`Welcome ${user.name}`,
        token : token,
        success:true
    })
}
```

Token ko response mai dena jaroori hai.

Output

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:2000/api/user/login
- Body:** JSON (selected)
- JSON Content:**

```
1  {
2      "email" : "lucky@gmail.com",
3      "password" : "luckyarya"
4 }
```
- Status:** 200 OK
- Size:** 453 Bytes
- Time:** 176 ms
- Response:**

```
1  {
2      "message": "Welcome Lucky",
3      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
4          .eyJ1c2VYIjp7I19pZCI6IjY5MTVhZGM5ZWE3ZDMxMTQ1Y
5          jE2MzhkZCisIm5hbWUiOjI3MdwNreSIsImVtYWIlsIjoibHV
6          ja3IAZ21haiWwuY29tiIwicGFzc3dvcmQiOiIkMmIkMTAkS
7          zFhQ2ZtMFRVm93YVZxLj1TY2V2dT1OVFwQVJ1LkxKOXF
8          ZcG96dmZRN2RTdnNsngzYnUiLCJjcmVhdGVkQXQiOiiyM
9          DI1LTExLTEzVDEwOja3OjA1LjQ2NloilCJfx3Yiojb9LCJ
10         pYXQiojE3NjMwNDI5NjgsImV4cCI6MTc2MzA0NjU2OHO
11         .pnvPn321sdDXpLbMf9GKQ5yQTK4Cg90b5i8Yn9CPm1Y",
12         "success": true
13     }
```

- Is token ko agr tum jwt ki webiste mai paste karoge to is token ki sabhi information aa jaayegi.

The screenshot shows the JWT Debugger interface. On the left, under "JSON WEB TOKEN (JWT)", there is an "Invalid Signature" message. The token itself is displayed as a long string of characters. On the right, under "DECODED PAYLOAD", the token is parsed into a JSON object:

```
{
  "alg": "HS256",
  "typ": "JWT"
}

{
  "user": {
    "_id": "6915adc9ea7d31145b1638dd",
    "name": "Lucky",
    "email": "lucky@gmail.com",
    "password": "$2b$10$K1aCfm0TUJowaVq.9Scevu9NuqpARe.LJ9qYpozvfQ7dSvs1vx3bu",
    "createdAt": "2025-11-13T10:07:05.466Z",
    "_v": 0
  },
  "iat": 1763042968,
  "exp": 1763046568
}
```

Pr mujhe itni details nahi dikhani or na hi chahiye mujhe sirf id chahiye, `_id`

- To

```
// decrypting password
const validPassword = await bcrypt.compare(password, user.password)
if (!validPassword) {
  return res.json({
    message: "Invalid password",
    success: false
  })
} else{
  // jwt
  const token = jwt.sign({userId: user._id}, "mysecretkey@123", { expiresIn: "1h" })

  res.json([
    message: `Welcome ${user.name}`,
    token: token,
    success: true
  ])
}
```

Maine poora user pass nahi kiya, sirf user.\_id pass ki.

- Ab chaloge to chhota token bange

The screenshot shows the JWT Debugger interface. In the 'Encoded Value' section, a JSON Web Token (JWT) is pasted:

```
eyJhbGciOiJIUzI1NiIsInR5cCIkXVCJ9.eyJlc2VyswQiOiI2OTE1YWRjOWVhN2QzMTE0NWIXNjM4ZGQ1LcJpYXQ1oje3NjMwNDMyNTgsImV4cCI6MTc2Mza0Njg1OH0.BWRDPvQB5m-uSwI3bXjWk-ve9t2RQ5jomZYUHwgSb5c|
```

The 'Decoded Header' section shows:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

The 'Decoded Payload' section shows:

```
{
  "userId": "6915adc9ea7d31145b1638dd",
  "iat": 1763043258,
  "exp": 1763046858
}
```

## ● Create Contact API

- Sbse pehle [Contact.js](#) naam se model bana lo.

```
JS Contact.js U X
16Proj4_ContactAPI > Models > JS Contact.js > [?] Contact
1 import mongoose from "mongoose"
2
3 const contactSchema = new mongoose.Schema({
4   name:{type:String , require : true},
5   email:{type:String , require : true},
6   phone:{type:String , require : true},
7   type:{type:String , require : true},
8   createdAt:{type:Date , default : Date.now},
9   user : {type:mongoose.Schema.Types.ObjectId} // ye FK se aayega
10 });
11
12 export const Contact = mongoose.model("contact" , contactSchema)
```

Yaha dekh sakte ho foreign key ese banai hai.

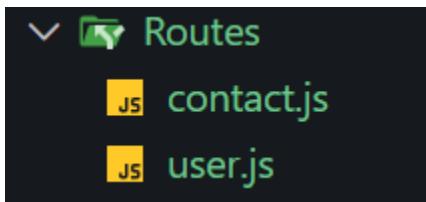


- Ab banao [contact.js](#) controller.

```
JS contact.js U X
16Proj4_ContactAPI > Controllers > JS contact.js > [Θ] newContact
1   import {contact} from '../Models/Contact.js'
2
3   // create new contact
4   export const newContact = async (req , res) => {
5
6 }
```

Abhi itna bana ke route bana lo

- Ab pehle route banao lo.



```
JS contact.js U X
16Proj4_ContactAPI > Routes > JS contact.js > ...
1   import express from 'express'
2   import {newContact} from '../Controllers/contact.js'
3
4   const router = express.Router();
5
6
7   // new contact
8   // @api description :- creating contact
9   // @api method : post
10  // @api endPoint : /api/contact/new
11
12  router.post('/new' , newContact)
13
14  export default router;
```

Bas itna hi kaam hona hai router mai.

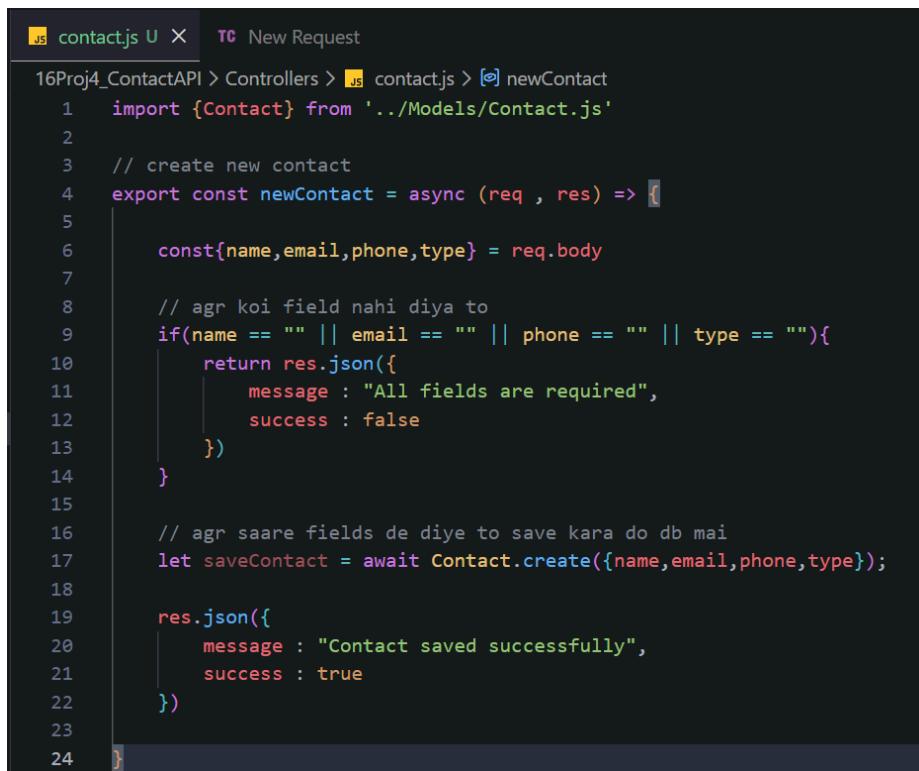
- Ab is route ko server mai link krna hoga. Sabhi file server se link honi chahiye.

```
import contactRouter from './Routes/contact.js'
```

```
// contact router
app.use('/api/contact' , contactRouter)
```

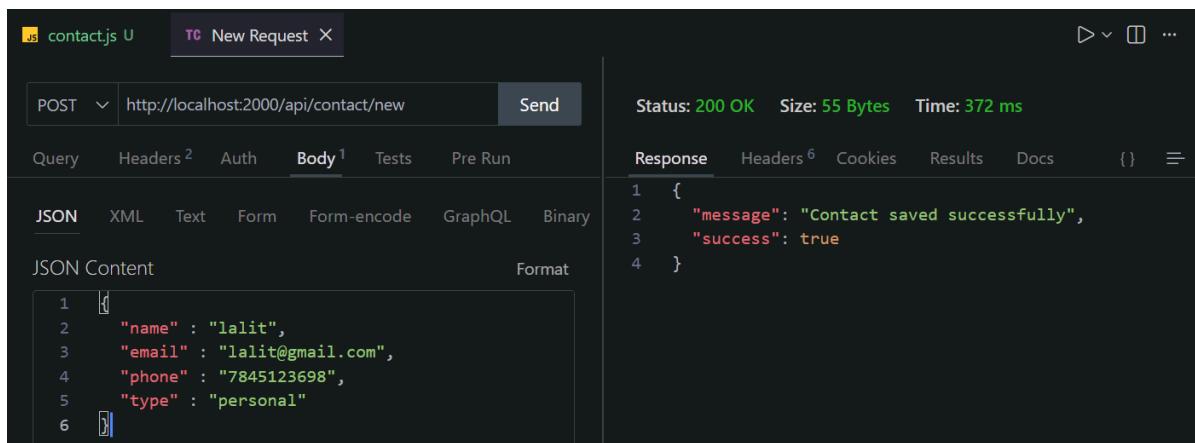
server.js mein ye kaam kar diya.

- Ab controller ka code poora karte hai.



```
1 import {Contact} from '../Models/Contact.js'
2
3 // create new contact
4 export const newContact = async (req , res) => {
5
6   const{name,email,phone,type} = req.body
7
8   // agr koi field nahi diya to
9   if(name == "" || email == "" || phone == "" || type == ""){
10     return res.json({
11       message : "All fields are required",
12       success : false
13     })
14   }
15
16   // agr saare fields de diye to save kara do db mai
17   let saveContact = await Contact.create({name,email,phone,type});
18
19   res.json({
20     message : "Contact saved successfully",
21     success : true
22   })
23
24 }
```

abhi user ko nahi daal re, ye user specific nahi bana re, baad mai karenge wo.  
Ab isko test karo.



Body		Response										
POST	http://localhost:2000/api/contact/new											
Query	Headers <sup>2</sup>	Auth	Body <sup>1</sup>	Tests	Pre Run	Response	Headers <sup>6</sup>	Cookies	Results	Docs	{}	≡
JSON	XML	Text	Form	Form-encode	GraphQL	Binary						
JSON Content						Format						
<pre>1 { 2   "name" : "lalit", 3   "email" : "lalit@gmail.com", 4   "phone" : "7845123698", 5   "type" : "personal" 6 }</pre>						<pre>1 { 2   "message": "Contact saved successfully", 3   "success": true 4 }</pre>						

Suppose save hone ke baad yhi tool pe mai wo data dekhna bhi chahta, to response mai bhej do bas,

```

// agr saare fields de diye to save kara do db mai
let saveContact = await Contact.create({name,email,phone,type});

res.json({
  message : "Contact saved successfully",
  saveContact : saveContact,
  success : true
})

```

Response mai saveContact bhej diya.

- Ab new record daala

Body <sup>1</sup>		Send	Status: 200 OK	Size: 237 Bytes	Time: 133 ms	
Query	Headers <sup>2</sup>	Auth	Headers <sup>6</sup>	Cookies	Results	
JSON	XML	Text	Form	Form-encode	GraphQL	
JSON Content		Format				
<pre> 1  [ 2    { 3      "name" : "lalit kumar", 4      "email" : "lalit@gmail.com", 5      "phone" : "7845123698", 6      "type" : "personal" 7    } 8  ] </pre>						
						Response
						Headers <sup>6</sup>
						Cookies
						Results
						Docs
						{}
						≡

```

1  {
2    "message": "Contact saved successfully",
3    "saveContact": {
4      "name": "lalit kumar",
5      "email": "lalit@gmail.com",
6      "phone": "7845123698",
7      "type": "personal",
8      "_id": "69172673f49b2ab09c4f6f39",
9      "createdAt": "2025-11-14T12:54:11.189Z",
10     "__v": 0
11   },
12   "success": true
13 }

```

## ● Get All Contact - API

- Ab contact create to kar liya, ab get krte hai contact, to controller mai likhenge get ka code.

```
contact.js U X TC New Request
16Proj4_ContactAPI > Controllers > contact.js > [?] getAllContact > ✎ success
1   import {Contact} from '../Models/Contact.js'
2
3   // get all contact
4   export const getAllContact = async (req,res) => {
5       const userContact = await Contact.find();
6
7       // agr data nahi mila to
8       if(!userContact){
9           return res.json({
10               message : "No contact exists",
11               success : false
12           })
13       }
14
15       // agr data mil jaye to
16       res.json({
17           message : "All contact fetched",
18           userContact : userContact,
19           success : true
20       });
21 }
```

Get ka poora controller ka code.

Ab iska route banana hoga, us route pe jake hi to check hoga na? Us route ko hit karenge to ye gata get waala show hoga.

```
import {getAllContact, newContact} from '../Controllers/contact.js'
```

Router mai ye import krlo pehle

```
// get contact
// @api description :- fetching contact
// @api method : get
// @api endPoint : /api/contact/
router.get('/', getAllContact)
```

Then ye maine bana diya route.

Ab ise hit karo. Test karo

```

Status: 200 OK  Size: 394 Bytes  Time: 109 ms

Response Headers 6 Cookies Results Docs { } ≡

1  {
2   "message": "All contact fetched",
3   "userContact": [
4     {
5       "_id": "691725b3f7193dd871e9fce8",
6       "name": "lalit",
7       "email": "lalit@gmail.com",
8       "phone": "7845123698",
9       "type": "personal",
10      "createdAt": "2025-11-14T12:50:59.947Z",
11      "__v": 0
12    },
13    {
14      "_id": "69172673f49b2ab09c4f6f39",
15      "name": "lalit kumar",
16      "email": "lalit@gmail.com",
17      "phone": "7845123698",
18      "type": "personal",
19      "createdAt": "2025-11-14T12:54:11.189Z",
20      "__v": 0
21    }
22  ],
23  "success": true
24 }

```

2 record the, dono aa gaye.

## ● Get by id Contact - API

- Chalo pehel controller pe.

```

// get by id
export const getContactById = async (req , res) => {
  const id = req.params.id
  const userId = await Contact.findById(id);
  if(!userId){
    return res.json({
      message : "No contact exists with this id",
      success : false
    })
  }

  // agr data mil jaye to
  res.json({
    message : "Contact with given id is fetched",
    userId : userId,
    success : true
  });
}

```

pehle id nikaal li parameter mai se,cuz parameter hi to pass hota hai id waala

- **Note :** maine req.params.id, likha hai, yaha jo id likha hai, wo important hai, ye hi naam se ab route banega, to route ke end pont ka nam bhi ab id hona chhaiye.

- `req.params.id` mai id === route ke end point ki id. Both name should be same.
- Ab route file ka code

```
// get contact by id
// @api description :- fetching contact by id
// @api method : get
// @api endPoint : /api/contact/id
router.get('/:id' , getContactById)
```

: dena jarori tha and id naam same hona chahie.

Now test karo.

Response	Headers	Cookies	Results	Docs
<pre> 1  { 2      "message": "Contact with given id is fetched", 3      "userId": { 4          "_id": "691725b3f7193dd871e9fce8", 5          "name": "lalit", 6          "email": "lalit@gmail.com", 7          "phone": "7845123698", 8          "type": "personal", 9          "createdAt": "2025-11-14T12:50:59.947Z", 10         "__v": 0 11     }, 12     "success": true 13 }</pre>				

- Id upper search box mai hi di hai.

## ● Update Contact - API

- Controller mai sabse pehle  
Isme bhi id ke basis pe kaam hota hai, to params mai se id nikaal lenge
- Update krne ke liye ek method hota hai : `findByIdAndUpdate()`  
`findByIdAndUpdate` 3 cheeze leta hai,
  1. Id
  2. data jo ham update krna chhahte hai.
  3. `new:true`, means new field agr aaye to use bhi set kardo

```

// update contact
export const updateContactById = async (req,res) => {

    const id = req.params.id; // id nikaal li

    // ab id ke basis pe kya kya data change karna hai wo bhi nikaalna hogा
    const{name,email,phone,type} = req.body

    let updatedContact = await Contact.findByIdAndUpdate(id , {name,email,phone,type},{new:true})
    if(!updatedContact){
        return res.json({
            message : "No contact exists with this id",
            success : false
        })
    }

    // agr data mil jaye to
    res.json({
        message : "Contact with given id is fetched",
        updatedContact : updatedContact,
        success : true
    });
}

```

- Ab route bana lo

```

// Update contact
// @api description :- Update contact by id
// @api method : PUT
// @api endPoint : /api/contact/id
router.put('/:id' , updateContactById)

```

- Test karlo

The screenshot shows a Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:2000/api/contact/69172673f49b2ab09c4f6f39
- Headers:** Content-Type: application/json
- Body (JSON):**

```

1 {
2     "name": "Lucky Arya",
3     "email": "lucky.arya@gmail.com",
4     "phone": "7845123698",
5     "type": "personal"
6 }

```
- Status:** 200 OK
- Size:** 245 Bytes
- Time:** 94 ms
- Response:**

```

1 {
2     "message": "Contact with given id is fetched",
3     "updatedContact": {
4         "_id": "69172673f49b2ab09c4f6f39",
5         "name": "Lucky Arya",
6         "email": "lucky.arya@gmail.com",
7         "phone": "7845123698",
8         "type": "personal",
9         "createdAt": "2025-11-14T12:54:11.189Z",
10        "__v": 0
11    },
12    "success": true
13 }

```

## ● Delete Contact - API

- Pehle controller

```
// delete contact
export const deleteContactById = async (req,res) => {

    const id = req.params.id; // id nikaal li

    let deleteContact = await Contact.findByIdAndDelete(id)
    if(!deleteContact){
        return res.json({
            message : "No contact exists with this id",
            success : false
        })
    }

    // agr data mil jaye to
    res.json({
        message : "Contact with given id is deleted",
        deleteContact : deleteContact,
        success : true
    });
}
```

Hame yaha req.body se kuch lena nahi hai, and findbyidanddelete method ka use karna hai bas. Baaki poora update waale ki trh hi hai.

```
// Delete contact
// @api description :- Delete contact by id
// @api method : Delete
// @api endPoint : /api/contact/id
router.delete('/:id' , deleteContactById)
```

- testing

```

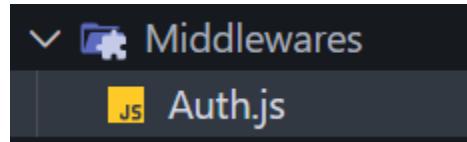
Status: 200 OK   Size: 244 Bytes   Time: 214 ms

Response Headers Cookies Results Docs { } =
1  {
2    "message": "Contact with given id is deleted",
3    "deleteContact": {
4      "_id": "69172673f49b2ab09c4f6f39",
5      "name": "Lucky arya",
6      "email": "lalit@gmail.com",
7      "phone": "7845123698",
8      "type": "personal",
9      "createdAt": "2025-11-14T12:54:11.189Z",
10     "__v": 0
11   },
12   "success": true
13 }

```

## ● User specific contact

- Iske liye user ka middleware banana padega.
- User jab login hoga to hi ye kam ho payega, isliye middleware bana re hai.
- To ye karlo



Auth capital mai rakhna. Authentication ka middleware h ye.

- Ye check karega ki user authenticated h ya nahi. Login h ya nahi. Bina login ke mai usko CRUD perform krne nahi dunga contact page mai.
- Isme 2 cheeze jati hai : req,res,next.
- So user ne jab login kiya tha to usko ek token mila tha, wo token hamko yaha chahiye, us token ke basis pe hi ham pata laga payenge ki user login h ya nahi hai.
  - So data lene ke 3 tareeke hai. 2 hame pata hai
    1. Method 1 - req.body
    2. Method 2 - req.params
    3. Method 3 - req.header
      - req.header('kuch\_bhi\_naam\_dedo')
- Filhal itna hamne kar diya middleware file mai

```

export const isAuthenticated = async (req,res,next) =>{

    // token lere
    const token = req.header('Auth')
    console.log(token);

}

```

- Ab ise use kaha karenge??
- So isko ham use karenge jab jab ham naya contact add kar rahe honge, to new contact add hone se pehle login check hona chahiye. To routes file mai import karo ise.

```

contact.js ...\\Routes U X contact.js ...\\Controllers U TC New Request
16Proj4_ContactAPI > Routes > contact.js > ...
1   import express from 'express'
2   import {deleteContactById, getAllContact, getContactById, new
3   contact.js'
4
5   import { isAuthenticated } from '../Middlewares/auth.js';

```

- And create contact waali route mai ye karo

```

// new contact
// @api description :- creating contact
// @api method : post
// @api endPoint : /api/contact/new
router.post('/new' , isAuthenticated , newContact)

```

So jab koi is /new route ko hit karega, pehle isAuthenticated waala function chalega and verify karega ki login hua ya nahi, and is function ke poore chalne ke baad newContact waala function chalega.

Agr verification fail ho gaya to newContact fn kabhi call hi nahi hogा

- Ab test karne chalo

POST ▾ http://localhost:2000/api/contact/new Send

Query Headers<sup>3</sup> Auth Body Tests Pre Run

HTTP Headers  Raw

<input checked="" type="checkbox"/> Accept	/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> Auth	LalitTest

Sbse pehle header mai jake maine Auth naam se middleware banaya tha, wo exact naam diya and LalitTest ki jgh kuch bhi de sakte hai.

And mai ek new contact create kr raha hoon to create nahi hora, cuz authenticate nahi hai, and console pe LalitTest print hora hai token.

contact.js ...\\Routes U TC New Request X contact.js ...\\Controllers U Auth.js U D v ⌂ ...

POST ▾ http://localhost:2000/api/contact/new Send

Query Headers<sup>3</sup> Auth Body Tests Pre Run

HTTP Headers  Raw

<input checked="" type="checkbox"/> Accept	/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> Auth	LalitTest
<input type="checkbox"/> header	value

Processing, Please Wait...

Response Headers Cookies Results Docs { } ⌂

Processing...  
Cancel Request

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + 1: node (16Proj4\_Conta ⌂ ⌂ ... | [ ] X

```
PS D:\Programming\Node Js Backend\16Proj4_ContactAPI> nodemon .\server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node .\server.js`
Server is running on port 2000
LalitTest
Connected
```

RHS mai sirf processing aara and console pe print hora.

- To is trh se ham middle ware mai ek header bana lenge, us header mai ham daalenge user ka login token, us token ko fir ham middleware mai le lenge, then us token se user ko ham find karenge because us token se user ki id store kar rakhi hai. Agr user hame mil gaya to ham bol denge user login hai.

- To pehle login karo

The screenshot shows a Thunder Client interface. On the left, there are tabs for contact.js ...\\Routes, New Request (highlighted), contact.js ...\\Controllers, and Auth.js. In the center, a POST request is made to <http://localhost:2000/api/user/login>. The Body tab is selected, containing the following JSON:

```

1 {
2   "email" : "lucky@gmail.com",
3   "password" : "luckyarya"
4 }

```

The Response tab shows the following JSON output:

```

1 {
2   "message": "Welcome Lucky",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJ1c2VySWQiOiI2OTE1YWRjOWVhN2QzMTE0NWIxNjM4ZGQiLCJpYXQiOjE3NjMxOTMzMzsImV4cCI6MTc2MzE5Njk
        zN30.YI1Kr57sr881FFLWEaa_Fn_6mWTnIxdt8YPwAaGFY
        _8",
4   "success": true
5 }

```

Status: 200 OK Size: 229 Bytes Time: 135 ms

Maine login kiya or mera token generate ho gaya.

Is token ko ab aap copy karlo and Auth jaha banaya tha waha paste kardo

The screenshot shows a Thunder Client interface. On the left, there are tabs for contact.js ...\\Routes, New Request \* (highlighted), contact.js ...\\Controllers, and Auth.js. In the center, a POST request is made to <http://localhost:2000/api/contact/new>. The Headers tab is selected, showing the following configuration:

Header	Value
Accept	/*
User-Agent	Thunder Client (https://www.thunderclient.com)
Auth	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2OTE1YWRjOWVhN2QzMTE0NWIxNjM4ZGQiLCJpYXQiOjE3NjMxOTMzMzsImV4cCI6MTc2MzE5NjkzN30.YI1Kr57sr881FFLWEaa_Fn_6mWTnIxdt8YPwAaGFY_8
header	value

The Response tab shows a processing message: "Processing, Please Wait..." with a circular progress indicator.

Console pe token print hua hai, ispe user ka information hai, isme se user ki information ko bahar nikaalna hai.

- Chalo auth file mai. Jo token aaya, usko pehle verify karna hota hai  
**Next() function ka kaam? -> hamne ye kiya hua hai :- router.post('/new' , isAuthenticated , newContact) :** so pehle isAuthenticated jake verify karega, verify agr success raha, to next() chalega and newContact next fn chalega.

```

import jwt from 'jsonwebtoken'
import {User} from '../Models/User.js'

export const isAuthenticated = async (req,res,next) => [
    // token lere
    const token = req.header('Auth')

    console.log(token);

    if (!token) { // agr token nahi mila
        return res.json({
            message : "Login First",
            success : false
        })
    }

    // if token hamare pass aa chuka hai,to usko verify karna hai
    const decoded = jwt.verify(token , "mysecretkey@123");

    // dekh lo decoded mai aa kya raha hai
    console.log("Token data" , decoded);
]

```

```

// dekh lo decoded mai aa kya raha hai
console.log("Token data" , decoded);

const id = decoded.userId;

// db se find karna hai
let user = await User.findById(id);
if(!user){
    return res.json({
        message : "User not found",
        success : false
    })
}

// agr user mil jaye to
// user ko ham apne hisaab se globally save kar sakte hai
req.userGlobal = user; // so jb jb ham isAuth middleware ko use karenge tb tb ham is global variable ko
use kar payenge

// itna karne se verify ho gaya, ab

next();
]

```

Is tareke se verification ho jayegा

- Ab jb contact ka schema banaya tha, us wqt hamne ek field chhod rakha tha user ka, ab wo bhi daalenge.
  - Usko ham daalenge : req.userGlobal se, cuz hame isi mai save kr rakha hai.

```
// agr saare fields de diye to save kara do db mai
let saveContact = await Contact.create({name,email,phone,type , user:req.userGlobal});
```

- Ab authentication mai delete and update krne se pehle bhi check karunga cuz koi bhi random banda delete na kar paye.

```
// Update contact
// @api description :- Update contact by id
// @api method : PUT
// @api endPoint : /api/contact/id
router.put('/:id' , isAuthenticated , updateContactById)

// Delete contact
// @api description :- Delete contact by id      You, 30 minute
// @api method : Delete
// @api endPoint : /api/contact/id
router.delete('/:id' , isAuthenticated , deleteContactById)
```

Contact route mai jake likha ye.

- Ab test krte hai.
  - Jese hi mai contact ka data update karne gaya,

PUT		http://localhost:2000/api/contact/691725b3f7193dd	Send	Status: 200 OK Size: 41 Bytes Time: 10 ms							
Query	Headers <sup>2</sup>	Auth	Body <sup>1</sup>	Tests	Pre Run	Response	Headers <sup>6</sup>	Cookies	Results	Docs	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary					
JSON Content						Format	1 {           2   "message": "Login First",           3   "success": false           4 }				
<pre> 1  { 2   "name":"lalit arya", 3   "email":"lalit@gmail.com", 4   "phone":"7845123698", 5   "type":"office" 6 }</pre>											

Bolra hai login first

- **Ab banega user specific contact**

```

// get contact by user id
export const getUserSpecificData = async (req , res) => {
    const id = req.params.id
    const userId = await Contact.find({user:id});
    if(!userId){
        return res.json({
            message : "No contact exists with this id",
            success : false
        })
    }

    // agr data mil jaye to
    res.json({
        message : "User specific contact is fetched",
        userId : userId,
        success : true
    });
}

```

You, 14 seconds ago • Uncommitted changes

Yaha pe find({user:id}) isliye kiye cuz upper ham jaha pe contact add kr re waha likha hai user:req.userGlobal, isme id definatly hoga.

- Ab iska router banana padega.

```

// get user specific contact
router.get('/userId/:id' , getUserSpecificData)

```

- Testing

Query Parameters		Response				
parameter	value					
		Status: 200 OK    Size: 226 Bytes    Time: 103 ms <pre> 1  { 2      "message": "All contact fetched", 3      "userContact": [ 4          { 5              "_id": "691725b3f7193dd871e9fce8", 6              "name": "lalit", 7              "email": "lalit@gmail.com", 8              "phone": "7845123698", 9              "type": "personal", 10             "createdAt": "2025-11-14T12:50:59.947Z", 11             "__v": 0 12         } 13     ], 14     "success": true 15 }</pre>				

Sbse pehle mere pass ek bande ka contact pada hai.

## ● Deploy API to Render

- Iske liye ek chhota sa setup karna hota hai.
- Sbse pehle .env file banate hai.  
Pehle install karo.  
npm i dotenv  
Hamko isko esa bana re h ki koi bhi isko hamara github se utha ke use kar sakte,  
to  
npm i cors
- .env pe rakhte hai file pehle

```
PORT = 2000

MONGO_URI = "mongodb+srv://luckyarya0101_db_user:ZJcto50NGZHidEiL@cluster0.ww01jq6.mongodb.net/"

JWT = "mysecretkey@123"
```

Ab jaha se remove kiya h waha pe kya rakhna hai?

```
import { config } from 'dotenv'
server.js mai : ...
```

You, 6 seconds ago | 1 author (You)

```
import express from 'express'
import mongoose from 'mongoose'
import bodyParser from 'express'
import userRouter from './Routes/user.js'
import contactRouter from './Routes/contact.js'
import { config } from 'dotenv'

const app = express();

app.use(bodyParser.json())

// .env setup
config({path: '.env'})
```

config({path:'env'})

- Ab neeche

```
mongoose.connect(process.env.MONGO_URI , {
  |   dbName:"Nodejs_Learning"
  |})
.then(() => console.log("Connected"))
.catch((err) => console.log(err))

const port = process.env.PORT;
```

PORT or MONGOOSE pe diya process.env.\_\_\_\_\_

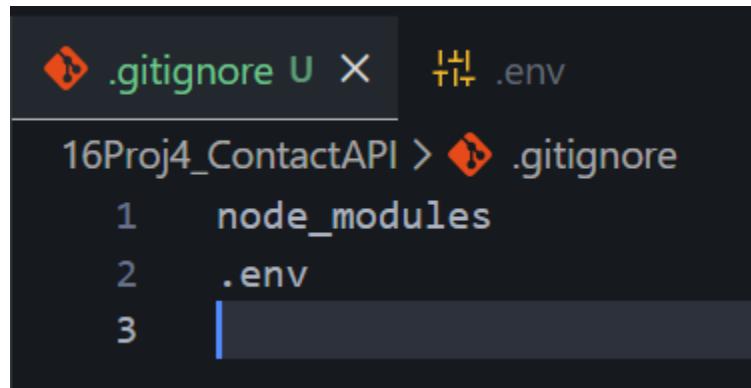
- Ab JWT ki jgh do

```
// jwt
const token = jwt.sign({userId : user._id} , process.env.JWT ,{ expiresIn: "1h" } )
```

- Ye hi same cheez [Auth.js](#) mai bhi

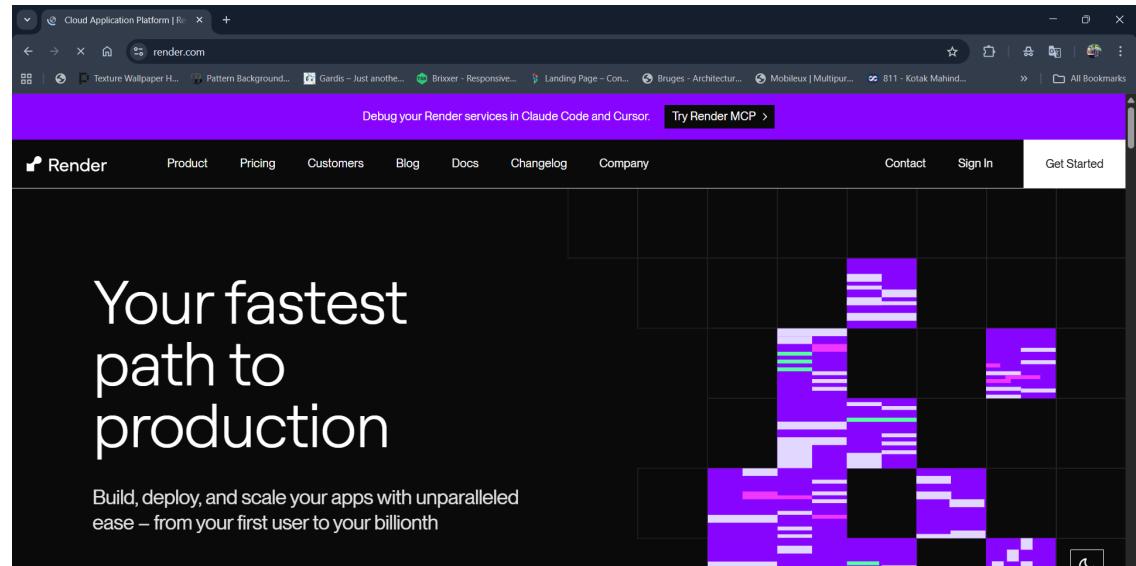
```
// if token hamare pass aa chuka hai,to usko verify karna hai
const decoded = jwt.verify(token , process.env.JWT);
```

- .gitignore file banao, or jo jo cheeze github pe push nahi karni wo wo cheeze likh do isme.



```
.gitignore U X +.env
16Proj4_ContactAPI > .gitignore
1 node_modules
2 .env
3 |
```

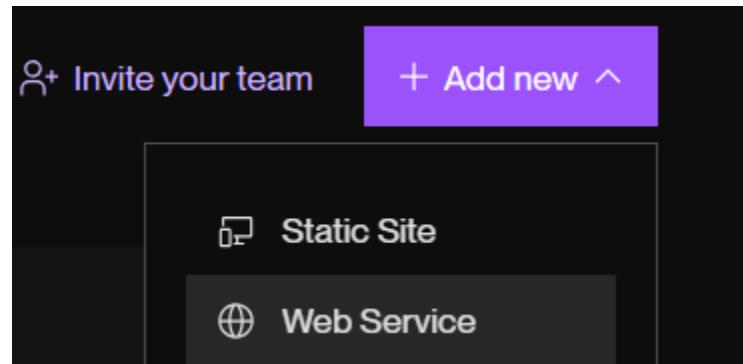
- Saare files ko github pe daal do pehle
- [rendor.com](#) pe jao, signup karo



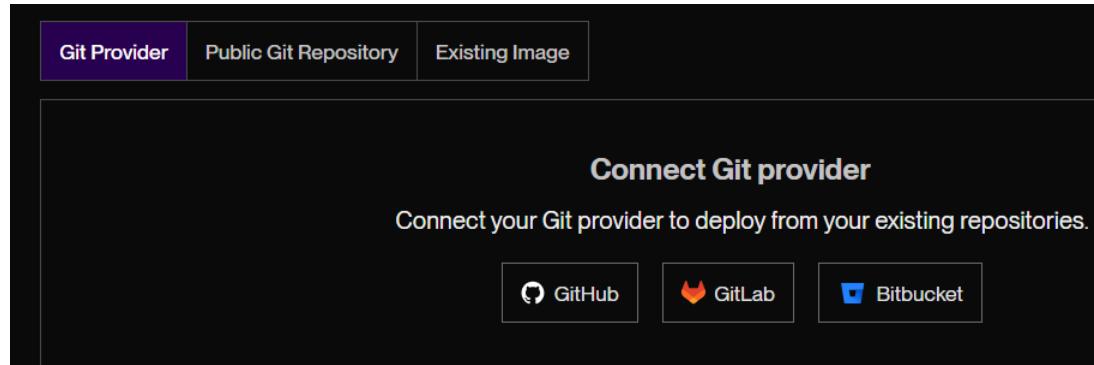
- Dashboard pe aa jao

The screenshot shows the Render.com dashboard for "Lalit's workspace". On the left, there's a sidebar with a tree view of "Projects", "Blueprints", "Environment Groups", "INTEGRATIONS" (Observability, Webhooks, Notifications), "NETWORKING" (Private Links), "WORKSPACE" (Billing), "Changelog", "Invite a friend", "Contact support", and "Render Status". The main area is titled "Overview" and displays a message: "You haven't created any services yet. Deploy in just a few minutes. Spin up web services, static sites, cron jobs, and more." It features four service creation options: "Deploy a Web Service" (Dynamic web app. Ideal for full-stack apps, API servers, and mobile backends.), "Deploy a Static Site" (Static content served over a global CDN. Ideal for frontend, blogs, and content sites.), "Create a Postgres database" (Relational data storage. Supports point-in-time recovery, read replicas, and high availability.), and "Explore all service types". At the top right, there are "Search", "Upgrade", and "Add new" buttons. A "Search" bar at the very top of the page also contains the text "Search K".

Top pe add new



Webservice



Link your github here

This screenshot shows the 'Environment Variables' section of the Heroku settings. It includes a table for adding variables, a link to 'Advanced' settings, and a summary bar at the bottom indicating a free plan (\$0/month).

NAME_OF_VARIABLE	value	Generate	⋮
PORT	2000	Generate	⋮
MONGO_URI	"mongodb+srv://luckyarya0101_db_user:ZJcto50NGZHidEiL@cluster0.wo1jq6.mongodb.net/"	Generate	⋮
JWT	"mysecretkey@123"	Generate	⋮

> Advanced

Deploy Web Service Cancel

Free  
\$0 / month

Poora page as it is rehndo, yaha env variables daalo, Add from .env pe click karo

This screenshot shows the 'Add from .env' modal. It contains a code editor with sample .env content and a file upload input labeled 'Choose a file'. At the bottom are 'Add variables' and 'Cancel' buttons.

```
PORT = 2000
MONGO_URI =
"mongodb+srv://luckyarya0101_db_user:ZJcto50NGZHidEiL@cluster0.wo1jq6.mongodb.net/"
JWT = "mysecretkey@123"
```

Choose a file ⏹

Add variables Cancel

Poori env file copy paste krni

Add variables pe click kardo

- Yaha starting command de dena

#### Start Command

Render runs this command to start your app with each deploy.

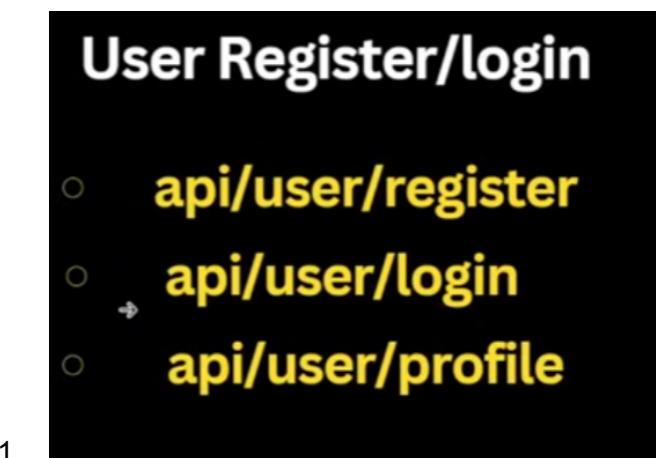
```
$ node server.js
```

Then deploy webservices

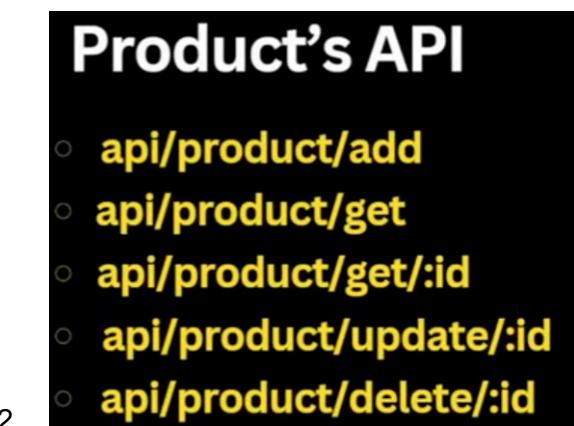
- Itna krne ke baad link mil jayega. Hamara deploys nahi ho skta cuz hamne to poora proj hi github pe daala hai, usme kaafi saari [server.js](#) file hai. Specific proj ko daaloge to chal jayega.

## Ecommerce project

- Flow



1.



2.

## Cart API

- **api/cart/add**
- **api/cart/usercart**
- **api/cart/remove/:id**
- **api/cart/--qty/:id**
- **api/cart/clear**

3.

## Shipping Address

- **api/address/add**
- **api/address/get**

4.

## Payment API

- **api/payment/pay**
- **api/payment/verify**



5.

# order's API

- **api/order/confirm**
- **api/order/confirm/all**

6.

- **Installation**

- npm i express mongoose
- npm i jsonwebtoken dotenv
- npm i bcryptjs

- Ab server.js bana lo, poora copy paste krdo proj 4 waala.

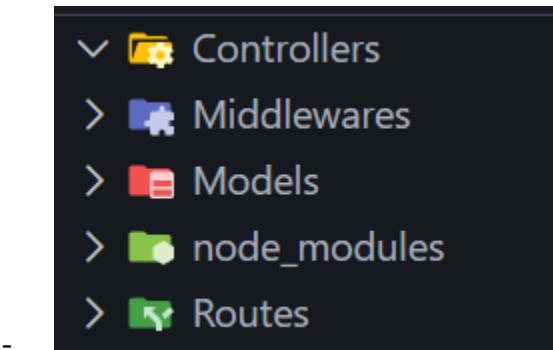
```
server.js U X .env U
17Proj5_EcommerceApp > server.js > ...
1   import express from 'express'
2   import mongoose from 'mongoose'
3   import bodyParser from 'express'
4   import { config } from 'dotenv'
5
6   const app = express();
7
8   app.use(bodyParser.json())
9
10 // .env setup
11 config({path:'./.env'})
12
13 // home route
14 app.get('/', (req,res) => {
15   res.json({
16     message : "Every thing is fine"
17   });
18 });
19
20 mongoose.connect(process.env.MONGO_URI , {
21   dbName:"Nodejs_Learning"
22 })
23 .then(() => console.log("Connected"))
24 .catch((err) => console.log(err))
25
26 const port = process.env.PORT;
27 app.listen(port , () => console.log(`Server is running on port ${port}`))
```

Is tarah hamari file dikh rahi hai and .env ka poora code bhi maine .env waali file mai paste kr diya as it is



```
server.js .env X
17Proj5_EcommerceApp > .env
1 PORT = 2000
2
3 MONGO_URI = "mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.ww01jq6.mongodb.net/"
4
5 JWT = "mysecretkey@123"
```

## ● User MVC

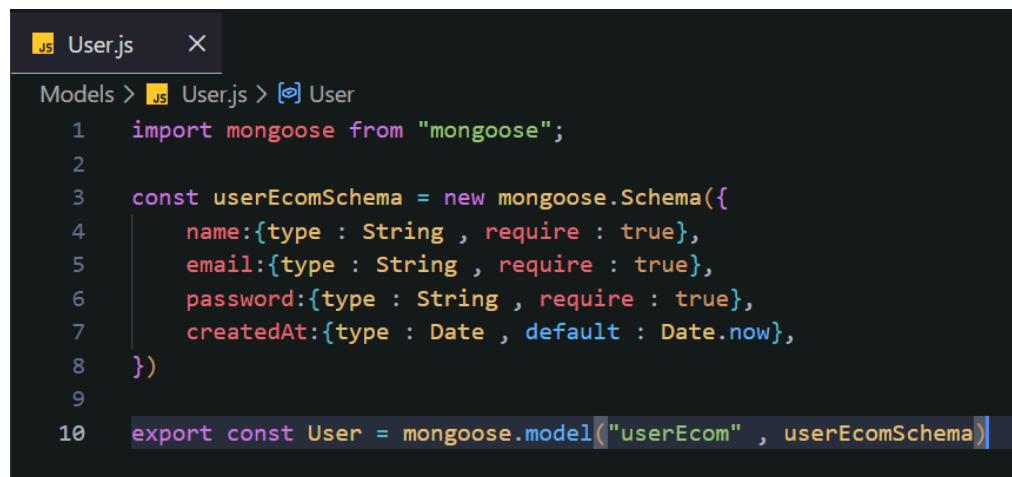


Itne folder bana lo



ye 3 model hamare hone waale hai

- **User schema :**



```
User.js X
Models > User.js > User
1 import mongoose from "mongoose";
2
3 const userEcomSchema = new mongoose.Schema({
4   name:{type : String , require : true},
5   email:{type : String , require : true},
6   password:{type : String , require : true},
7   createdAt:{type : Date , default : Date.now},
8 }
9
10 export const User = mongoose.model("userEcom" , userEcomSchema)
```

- User Controller

```
JS user.js X
Controllers > JS user.js > [o] register
1 import { User } from "../Models/User.js";
2 import bcrypt from "bcryptjs";
3
4 // user register
5 export const register = async (req,res) => [
6     let {name,email,password} = req.body;
7
8     // check if email already exists in DB or no
9     let user = await User.findOne({email})
10    if (user) {
11        res.json({
12            message : "User already exist with this email",
13            success : false
14        })
15    }
16
17    // hash password now
18    const hashPassword = await bcrypt.hash(password,10);
19
20    // create user now
21    user = await User.create({
22        name,email,password:hashPassword
23    });
24    res.json({
25        message : "User Registered successfully",
26        success : true
27    });
28 ]
```

Ye code bana diya register ka, ab route banao

```
JS user.js Controllers X JS user.js Routes X
Routes > JS user.js > [o] default
1 import express from 'express'
2 import { register } from '../Controllers/user.js';
3
4 const router = express.Router();
5
6 // register route
7 router.post('/register' , register)
8
9 export default router;
```

- Ab isko [server.js](#) se link krna hoga

Ye import kiya :

```
import userRouter from './Routes/user.js'
```

Or ye route :

```
// user register route - middleware  
app.use('/api/user', userRouter)
```

- Testing :

The screenshot shows two windows side-by-side. On the left is Postman, displaying a successful POST request to `http://localhost:2000/api/user/register`. The response status is 200 OK, size is 57 Bytes, and time is 441 ms. The response body is a JSON object:

```
1  {  
2      "message": "User Registered successfully",  
3      "success": true  
4  }
```

On the right is the MongoDB Compass interface, showing a collection named `userecoms`. It displays a single document with the following fields:

```
_id: ObjectId('6918746fe7e8c677ebb81395')  
name : "Lalit"  
email : "lalit@gmail.com"  
password : "$2b$10$8WhFWn6cHjZxCWdFdiJeeYGetsyZ0MXxuAPikcAijingyRwu/4qa"  
createdAt : 2025-11-15T12:39:11.471+00:00  
__v : 0
```

Password hash karke store hua hai.

- **Login user :**

- To user ke controller mai hi banega ye.
- Full code :

```

30
31 // user login
32 export const login = async(req,res) =>{
33   const {email,password} = req.body;
34
35   // find user
36   let user = await User.findOne({email})
37   if (!user) {
38     return res.json({
39       message : "User does not exist",
40       success : false
41     })
42
43
44   // check password
45   const validPass = await bcrypt.compare(password, user.password) // usertextbox pe jo password likhra usko
46   compare kar re DB mai jo password save hai
47
48   if (!validPass) {
49     return res.json({
50       message : "Invalid Password",
51       success : false
52     })
53
54   // ab email and pass dono sahi ho to token banayenge.
55   const token = jwt.sign({userId:user._id} , process.env.JWT , {expiresIn:'1d'})
56
57   // token banne ke baad user ko welcome bolo
58   res.json({
59     message : `Welcome ${user.name}`,
60     token : token,
61     success : true
62   })
63 }
```

- Ab router mai jake route banao.

```
// login route
router.post('/login' , login)
```

- Ab test karlo

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:2000/api/user/login
- Body:** JSON (Content: {"email": "lalit@gmail.com", "password": "123"})
- Status:** 200 OK
- Size:** 229 Bytes
- Time:** 268 ms
- Response:**

```

1  {
2    "message": "Welcome Lalit",
3    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
          .eyJlc2VybWQiOiI2OTE4NzQ2ZmU3ZThjNjc3ZWJiODEzO
          TUiLCJpYXQiOjE3NjMyMTEzOTIsImV4cCI6MTc2MzI5Nzc
          5Mn0.-nHVRgKf9FtrktU100Y1zzlfuIPh2hXoGHzJQeonM
          qw",
4    "success": true
5  }

```

- So login/register waala kaam khtm, ab products ka

## Products MVC

- Model :**

- Hame ye schema dynamic banana hai, for example, koi product hai mobile, uske features hone like camera, battery, ram, pixels etc, and suppose koi doosra product hai Tshirt, uske feature hone size, male, female, and suppose koi 3rd products hai food se related, uske alg features honge.
- Is wjh se schema dynamic banana hogा.
- Pr kuch cheeze sabhi products mai common hoti hai like : title, description, price and category.

The screenshot shows a code editor with the following code in a Product.js file:

```

TC New Request JS Product.js X
Models > JS Product.js > ...
1 import mongoose from "mongoose";
2
3 const productSchema = new mongoose.Schema({
4
5 }, {strict:false})
6
7 export const Product = mongoose.model("products" , productSchema);

```

{strict:false} means, dynamic hai, mai req.body mai kuch bhi daalu wo le lega.  
Bas itna hi karna hai

- Controller**

```
TC New Request product.js X
Controllers > product.js > ...
1 import {Product} from '../Models/Product.js'
2
3 // add product - isme ye karunga ki req.body se jo kuch bhi aaye, use aap daal do.
4 export const addProduct = async (req,res) => {
5     try {
6         let product = await Product.create(req.body);
7         res.json({
8             message : "Product added successfully",
9             product : product,
10            success : true
11        })
12    } catch (error) {
13        res.json(error.message)
14    }
15}
16}
```

So hamne direct req.body hi pass kr diya, create mai, that means jo hi cheez aari hai req.body mai, use db mai save karwa do.

- Route

```
TC New Request product.js X server.js
Routes > product.js > [?] default
1 import express from "express"
2 import { addProduct } from "../Controllers/product.js";
3
4 const router = express.Router();
5
6 // add product
7 router.post('/add' , addProduct)
8
9 export default router;
```

- Ab isko [server.js](#) mai register krna hoga

- Server.js

```
import productRouter from './Routes/product.js'

// product route - middleware
app.use('/api/product' , productRouter)
```

- Testing

The screenshot shows the Postman interface. A new request is being made to `http://localhost:2000/api/product/add` using a `POST` method. The request body is set to `JSON` and contains the following data:

```

1  [
2   "title" : "iPhone-17",
3   "description" : "Newly launched",
4   "price" : 150000,
5   "category" : "Mobiles",
6   "ram" : "8GB",
7   "rom" : "128GB"
8 ]

```

The response status is `200 OK`, size is `221 Bytes`, and time is `153 ms`. The response body is:

```

1  {
2   "message": "Product added successfully",
3   "product": {
4     "_id": "69187e29faa21303b62bff37",
5     "title": "iPhone-17",
6     "description": "Newly launched",
7     "price": 150000,
8     "category": "Mobiles",
9     "ram": "8GB",
10    "rom": "128GB",
11    "__v": 0
12  },
13  "success": true
14 }

```

- Get product :

```

// get all product
export const getAllProducts = async (req,res) => {
  try {
    let products = await Product.find()
    if (!products) {
      return res.json({
        message:"No product found!",
        success:false
      })
    }

    // if product found
    res.json({
      message : "Fetched all products",
      products : products,
      success : true
    })
  } catch (error) {
    res.json(error.message)
  }
}

```

- Controller :

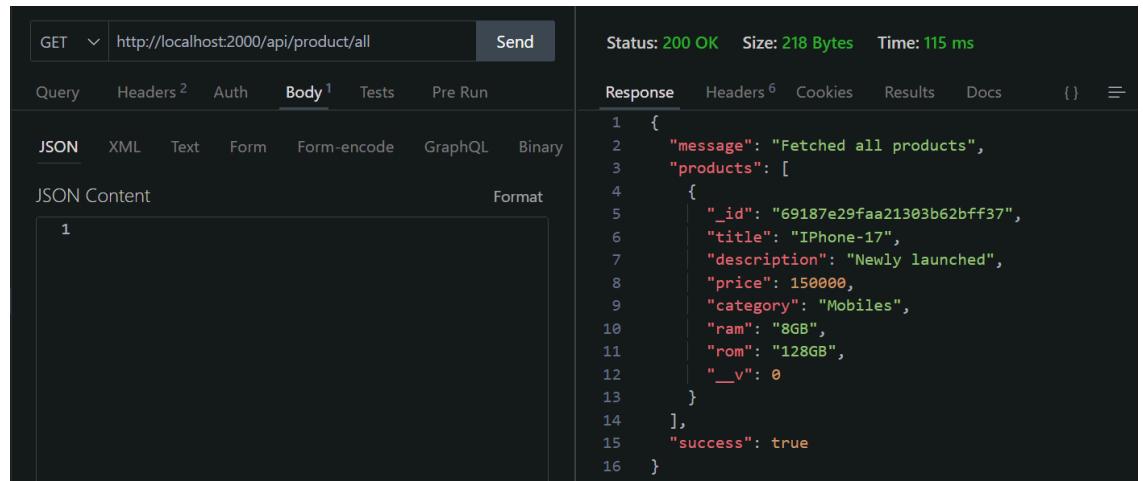
```

// get all product
router.get('/all' , getAllProducts)

```

- Routes :

- **Test :**



The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:2000/api/product/all`. The response details are: Status: 200 OK, Size: 218 Bytes, Time: 115 ms. The response body is a JSON object:

```
1 {  
2   "message": "Fetched all products",  
3   "products": [  
4     {  
5       "_id": "69187e29faa21303b62bff37",  
6       "title": "iPhone-17",  
7       "description": "Newly launched",  
8       "price": 150000,  
9       "category": "Mobiles",  
10      "ram": "8GB",  
11      "rom": "128GB",  
12      "__v": 0  
13    },  
14  ],  
15  "success": true  
16 }
```

● **Get Product by Id**

● **Controller :**

```
// get product by id  
export const getProductById = async (req,res) => {  
  
  const prodId = req.params.id  
  
  try {  
    let product = await Product.findById(id)  
    if (!product) {  
      return res.json({  
        message:"No product found with this id",  
        success:false  
      })  
    }  
    // if product found  
    res.json({  
      message : "Fetched the product with this id",  
      product : product,  
      success : true  
    })  
  } catch (error) {  
    res.json(error.message);  
  }  
}
```

```
// get product by id
router.get('/:id' , getProductId)
```

- **Route :**
  - Ham /:id isliye karte hai cuz, id dynamic aati hai, har product ke liye new id aayegi.
- **Testing :**

The screenshot shows a Postman interface with the following details:

- Request URL:** GET http://localhost:2000/api/product/69187e29faa2130
- Status:** 200 OK
- Size:** 227 Bytes
- Time:** 157 ms
- Response Body:**

```

1   "message": "Fetched the product with this id",
2   "product": {
3     "_id": "69187e29faa21303b62bff37",
4     "title": "iPhone-17",
5     "description": "Newly launched",
6     "price": 150000,
7     "category": "Mobiles",
8     "ram": "8GB",
9     "rom": "128GB",
10    "__v": 0
11  },
12  "success": true
13 }
```

- **Update product**

- **Controller :** `findByIdAndUpdate(productId , req.body , {new:true})` —> ye fun ka use krenge. Ye id lera hai jiske basis pe update hoga, req.body mai jo bhi data mile, sabko update kardo, and new:true means koi new field add karna chahe to wo bhi kar sakte.

```
// update product
export const updateProduct = async (req , res) => {
  const prodId = req.params.id

  try {
    let product = await Product.findByIdAndUpdate(prodId , req.body , {new:true})
    if (!product) {
      return res.json({
        message:"No product found with this id",
        success:false
      })
    }
    // if product found
    res.json({
      message : "Product updated successfully!",
      product : product,
      success : true
    })
  } catch (error) {
    res.json(error.message);
  }
}
```

```
// update product
router.put('/:id' , updateProduct)
```

- Router :

- Testing :

The screenshot shows the Postman interface with a successful API call. The request URL is `http://localhost:2000/api/product/69187e29faa2130`. The response status is `200 OK`, size is `224 Bytes`, and time is `113 ms`. The response body is a JSON object:

```

1 {
2   "message": "Product updated successfully!",
3   "product": {
4     "_id": "69187e29faa21303b62bff37",
5     "title": "IPhone-18",
6     "description": "Newly launched",
7     "price": 150000,
8     "category": "Mobiles",
9     "ram": "8GB",
10    "rom": "128GB",
11    "__v": 0
12  },
13  "success": true
14 }
```

- Suppose mai new field add karna chahu, to wo bhi kar sakta.

The screenshot shows the Postman interface with a successful API call. The request URL is `http://localhost:2000/api/product/69187e29faa2130`. The response status is `200 OK`, size is `240 Bytes`, and time is `85 ms`. The response body is a JSON object:

```

1 {
2   "message": "Product updated successfully!",
3   "product": {
4     "_id": "69187e29faa21303b62bff37",
5     "title": "IPhone-18",
6     "description": "Newly launched",
7     "price": 150000,
8     "category": "Mobiles",
9     "ram": "8GB",
10    "rom": "128GB",
11    "__v": 0,
12    "battery": "50W"
13  },
14  "success": true
15 }
```

- Delete Product

```

// Delete product
export const deleteProduct = async (req,res) => {
    const prodId = req.params.id

    try {
        let product = await Product.findByIdAndDelete(prodId)
        if (!product) {
            return res.json({
                message:"No product found with this id",
                success:false
            })
        }
        // if product found
        res.json({
            message : "Product deleted successfully!",
            product : product,
            success : true
        })
    } catch (error) {
        res.json(error.message);
    }
}

```

- **Controller :**
- **Route :**

```

// delete product
router.delete('/:id' , deleteProduct)

```

- **Testing :**

The screenshot shows a Postman interface with the following details:

- Method:** DELETE
- URL:** http://localhost:2000/api/product/69187e29faa21303b62bff37
- Status:** 200 OK
- Size:** 240 Bytes
- Time:** 133 ms
- Response:**

```

1  {
2      "message": "Product deleted successfully!",
3      "product": {
4          "_id": "69187e29faa21303b62bff37",
5          "title": "iPhone-18",
6          "description": "Newly launched",
7          "price": 150000,
8          "category": "Mobiles",
9          "ram": "8GB",
10         "rom": "128GB",
11         "__v": 0,
12         "battery": "50W"
13     },
14     "success": true
15 }

```

## ● User login :

- Cart functionality banane se pehle user ko login karwaana hoga, agr user login hoga to hi to apne cart ke items ko wo dekh payega, ye karne ke liye middleware folder mai [auth.js](#) file ka code dekho.



- Isme user ko verify karwaana hai token.

```
JS Auth.js X
Middlewares > JS Auth.js > ...
1 import jwt from "jsonwebtoken";
2 import { User } from "../Models/User.js";
3
4 export const isAuthenticated = async (req, res, next) => {
5   const token = req.header("Auth");
6
7   if (!token) {
8     return res.json({
9       message: "Login first",
10      success: false,
11    });
12  }
13
14  // if token found
15  const decoded = jwt.verify(token, process.env.JWT);
16
17  const id = decoded.userId;
18  // id ke basis pe find karunga
19  let user = await User.findById(id);
20  if (!user) {
21    return res.json({
22      message: "User not found",
23    });
24  }
25  req.user = user;
26  next();
27};
28|
```

The image shows a code editor with a dark theme. The title bar says 'Auth.js'. The code itself is a function named 'isAuthenticated' that takes three parameters: req, res, and next. It first checks if there is no token in the header. If there is none, it returns a JSON response with 'message' as 'Login first' and 'success' as false. If there is a token, it uses the 'jsonwebtoken' library to verify it using the environment variable 'JWT'. If the verification fails, it returns a JSON response with 'message' as 'User not found'. If the verification is successful, it sets the 'user' property of the 'req' object to the user found in the database and then calls the 'next()' function to proceed to the next middleware or route handler.

Iske through ham verify kar payenge ki token sahi h ya nahi

- Ab banao cart

## ● Cart MVC

- Schema :

```
Models > JS Cart.js > [?] cartSchema
1   import mongoose from "mongoose";
2
3   const cartItemSchema = new mongoose.Schema({
4
5       productId : {
6           type : mongoose.Schema.Types.ObjectId,
7           ref : "Product",
8           require : true
9       },
10      title : {type : String, require : true},
11      price : {type : Number, require : true},
12      qty : {type : Number, require : true},
13  })
```

Is tareeke se mai foreign key relationship banata hoon isme.

```

Models > Cart.js > ...
1   import mongoose from "mongoose";
2
3   const cartItemSchema = new mongoose.Schema({
4
5       productId : {
6           type : mongoose.Schema.Types.ObjectId,
7           ref : "Product",
8           require : true
9       },
10      title : {type : String, require : true},
11      price : {type : Number, require : true},
12      qty : {type : Number, require : true},
13   })
14
15
16   const cartSchema = new mongoose.Schema({
17       userId : {
18           type : mongoose.Schema.Types.ObjectId,
19           ref : "User",
20           require : true
21       },
22       items : [cartItemSchema]
23   })
24
25   export const Cart = mongoose.model("cart" , cartSchema)

```

2 schema banaye hai, upper waala isliye cuz utna to sabhi product ke liye same rehne waala hai, 2nd waala user specific.

- Controller

```

cart.js  X
Controllers > cart.js > addToCart
1   import {Cart} from "../Models/Cart.js"
2
3   // add to cart
4   export const addToCart = async (req,res) => {
5       const {productId, title, price, qty} = req.body;
6
7       const userId = req.user; // ye global se aari, auth file se
8
9       // now we'll check ki is user id se pehle koi cart available h ya nahi, agr available hai to ham naya
10      cart nahi banayenge, usi mai item ko add kr denge.
11      let cart = await Cart.findOne({userId})
12      if (!cart) {
13          // to naya cart banayenge
14          cart = new Cart({userId , items:[ ]}); // items ko array ki form mai dere cuz schema mai hame array ki
15          // form mai liya hai
16      }
17      else{
18          // ab us product ka index find karna hoga ki wo product already h ya nahi - suppose iphone ko maine
19          add to cart kiya, then dobara use add to cart kiya to uski qty bad jati hai na ki wo dobara add hota
20          hai.
21          const itemIndex = cart.items.array.findIndex((item) => item.productId.toString() == productId);
22      }
23  }

```

```

20      // agr already item hai , to jo cart mai already items hai uski qty ko increase kardo
21      if (itemIndex >= 1) {
22          cart.items[itemIndex].qty += qty
23          cart.items[itemIndex].price += price * qty
24      }
25      else{
26          cart.items.push({productId, title, price, qty})
27      }
28
29      await cart.save();
30      res.json({
31          message : "Item added to cart",
32          cart : cart,
33          success : true
34      })
35  }

```

- **Route**

```

Routes > cart.js > [o] default
1 import express from 'express'
2 import { addToCart } from '../Controllers/cart.js';
3 import {isAuthenticated} from '../Middlewares/Auth.js'
4
5 const router = express.Router();
6
7 // add to cart
8 router.post('/add' ,isAuthenticated ,addToCart)
9
10 export default router;

```

- Ab isko server se link karna hoga

- **Server**

```

import cartRouter from './Routes/cart.js'

// cart route - middleware
app.use('/api/cart' , cartRouter)

```

- **Testing :**

- Sbse pehle maine kuch product add kar liye

POST <http://localhost:2000/api/product/add> Send

Query	Headers <sup>2</sup>	Auth	<b>Body</b> <sup>1</sup>	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary

JSON Content Format

```

1  {
2    "title" : "Iphone 17",
3    "description" : "Better than samsung",
4    "price" : "120000",
5    "category" : "Mobile",
6    "ram" : "32GB",
7    "rom" : "512GB"
8  }

```

Status: 200 OK Size: 228 Bytes Time: 1.17 ms

Response	Headers <sup>6</sup>	Cookies	Results	Docs
1  { 2    "message": "Product added successfully", 3    "product": { 4      "_id": "69199559a86cf4bb774e821a", 5      "title": "Iphone 17", 6      "description": "Better than samsung", 7      "price": "120000", 8      "category": "Mobile", 9      "ram": "32GB", 10     "rom": "512GB", 11     "__v": 0 12   }, 13   "success": true 14 }				

- Ye dekh sakte ho aap :

```

_id: ObjectId('69199559a86cf4bb774e821a')
title: "Iphone 17"
description: "Better than samsung"
price: "120000"
category: "Mobile"
ram: "32GB"
rom: "512GB"
__v: 0

_id: ObjectId('691995a8a86cf4bb774e821c')
title: "Iphone 18"
description: "Better than 17"
price: "150000"
category: "Mobile"
ram: "32GB"
rom: "512GB"
__v: 0

_id: ObjectId('691995b6a86cf4bb774e821e')
title: "Samsung"
description: "Better than IPhone"
price: "150000"
category: "Mobile"
ram: "32GB"
rom: "512GB"
__v: 0

```

- Yaha se product id uth lo uski need hai.
- Ab login token utha lo

POST <http://localhost:2000/api/user/login> Send

Query	Headers <sup>3</sup>	Auth	<b>Body</b> <sup>1</sup>	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary

JSON Content Format

```

1  {
2    "email": "lalit@gmail.com",
3    "password": "123"
4  }

```

Status: 200 OK Size: 229 Bytes Time: 965 ms

Response	Headers <sup>6</sup>	Cookies	Results	Docs
1  [ 2    "message": "Welcome Lalit", 3    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .eyJ1c2VySWQiOii2OTE4NzQ2ZmU3ZThjNjc3ZWJiODEzO TUiLCJpYXQiOjE3NjMyODQ3MTYsImV4cCI6MTc2MzM3MTE xNn0.47xUFbTX-4t2xEyMvi -0vQwCHOIgB3ludqyG8uAyWFc", 4    "success": true 5  ]				

- Ye token uthao or chalo cart mai
- Ab cart ko chalate hai :

The screenshot shows a Postman interface. The request URL is `http://localhost:2000/api/cart/add`. The request method is `POST`. The response status is `200 OK`, size is `41 Bytes`, and time is `8 ms`. The response body is:

```

1  {
2    "message": "Login first",
3    "success": false
4  }

```

Bina user id diye chalaya maine, to login krne ko keh raha hai.

Iski tokon bhi do, jp upper generte hua

Header mai jake auth : isse aaghe hamara chal nahi raha testing. Pr code jo kiya hai wo sahi hai.

## ● User specific cart

- Controller :



```
cart.js    X
Controllers > cart.js > [o] userCart
1 import {Cart} from "../Models/Cart.js"
2
3 // add to cart
4 > export const addToCart = async (req,res) => {
35   }
36
37 // user specific cart
38 export const userCart = async (req , res) => {
39   const userId = req.user
40
41   let cart = await Cart.findOne({userId})
42   if (!cart) {
43     return res.json({
44       message : "Cart no t found" ,
45       success : false
46     })
47   }
48
49   res.json({
50     message : "User cart",
51     cart : cart
52   })
53 }
54
```

- Route :

```
// get user cart
router.get('/user' , isAuthenticated , userCart)
```

- Remove Product

- Controller :

```
// remove product
export const removeProductFromCart = async (req, res) => {
  const productId = req.params.productId

  // login bhi to hona chahiye
  const userId = req.user;

  // check karo jiska delete kar re wo exist bhi karta h ya nahi
  let cart = await Cart.findOne({userId});
  if (!cart) {
    return res.json({
      message : "Cart not found" ,
      success : false
    })
  }

  // cart ke item mai wo saare item daal do, is productId ko chhad ke baaki saare daal do
  cart.items = cart.items.filter((item) => item.productId.toString() != productId)

  await cart.save();
  res.json({
    message : "Product has been removed from cart" ,
    success : true
  })
}
```

- Route

```
// delete product
router.delete('/remove/:productId' , isAuthenticated , removeProductFromCart)
```