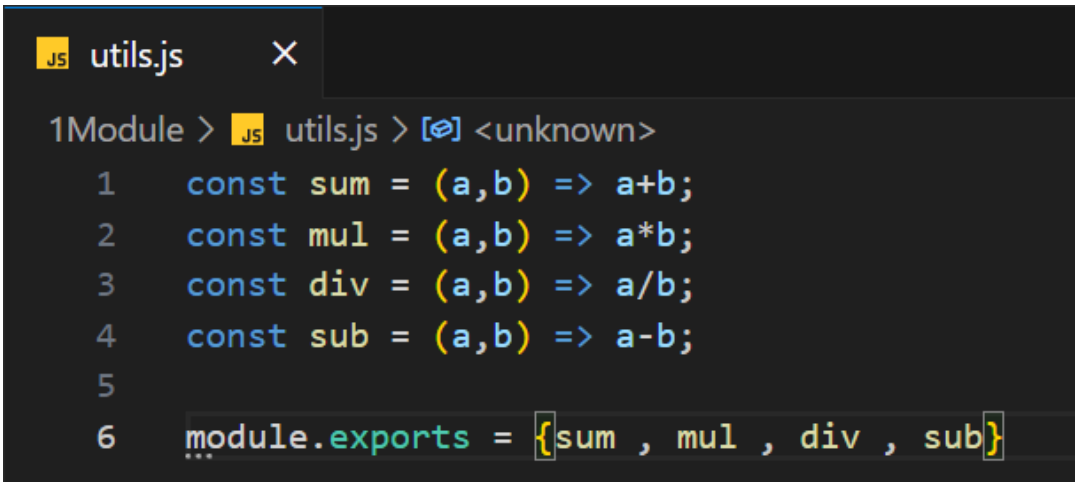
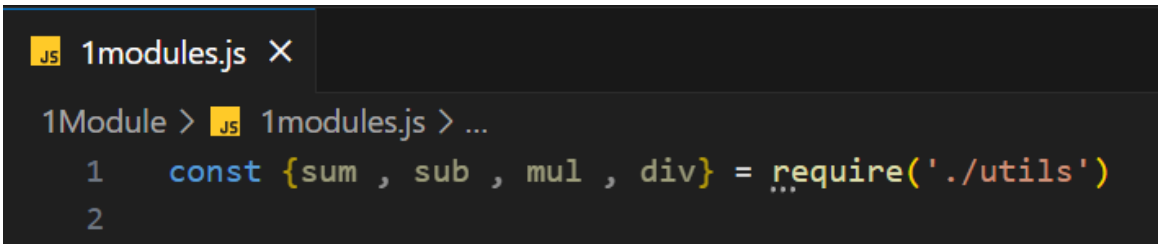
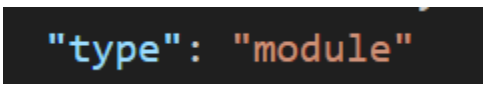
NODE JS

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

- **Old way - using require :**
    - 

    Ye export krne ka hai
    - 

    Or ye import krne ka

- **New way - using import**
    - Iske liye pehle package.json mai kuch change krna hoga
    

    - 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

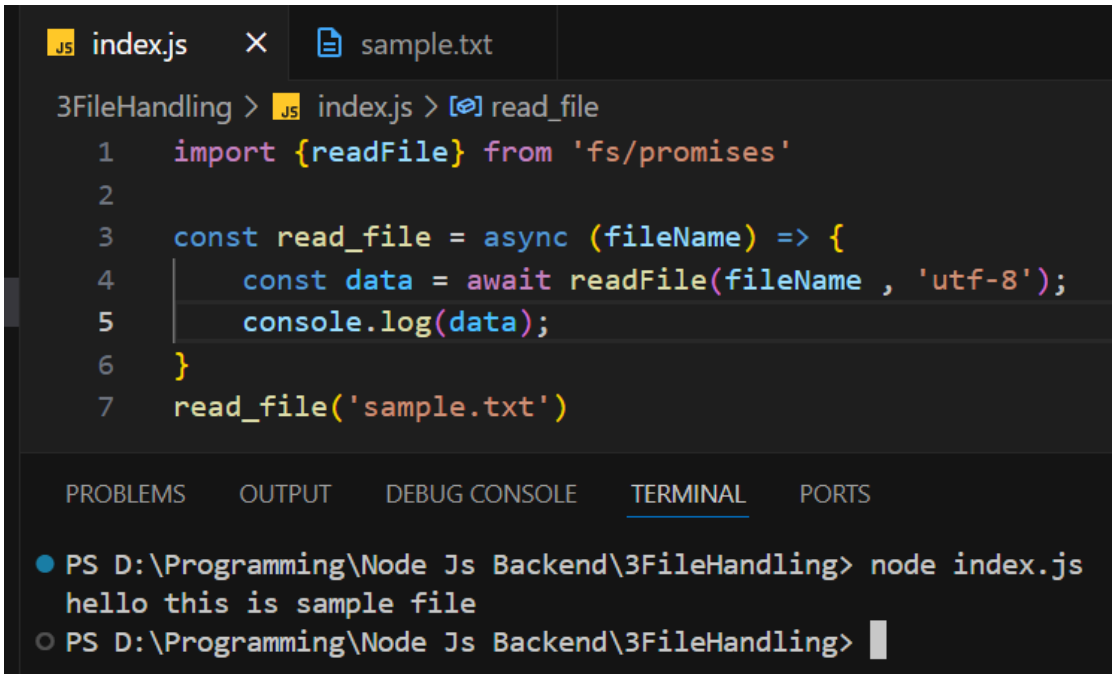
    ```
    1   import {} from 'fs/promises'
    ```

    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

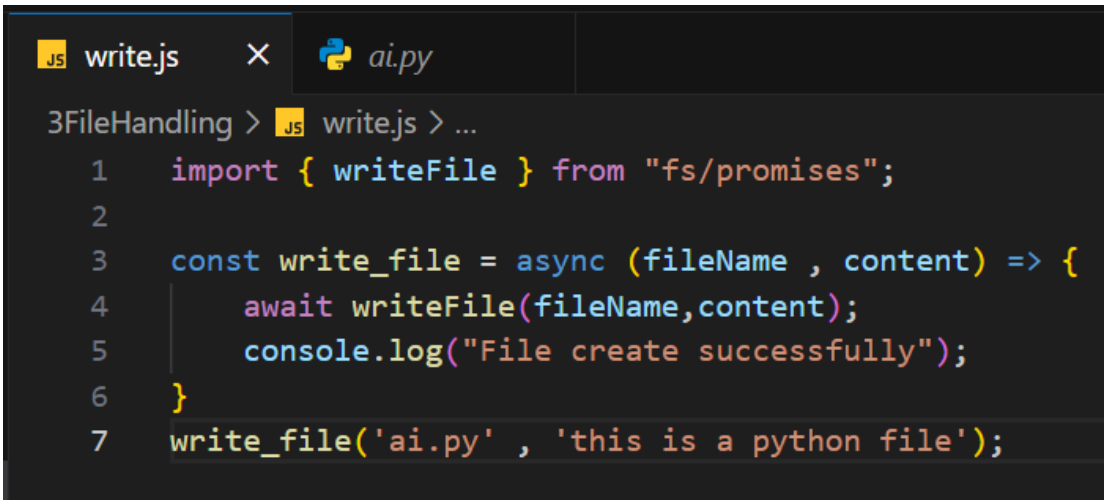
    3FileHandling > sample.txt

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

Ek sample file bana li

- 

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 jarrart nahi hai
  - Parameter mai 2 cheez bhejo, file ka naam or kya content bhejna hai
  - 

- **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

append.js

```
3FileHandling > 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")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Programming\Node Js Backend\3FileHandling> node append.js
Updated successfully
PS D:\Programming\Node Js Backend\3FileHandling>
```

sample.txt

```
3FileHandling > sample.txt
1  hello this is sample filenew line of content added
```

- Path module
- **Joining 2 or more file**

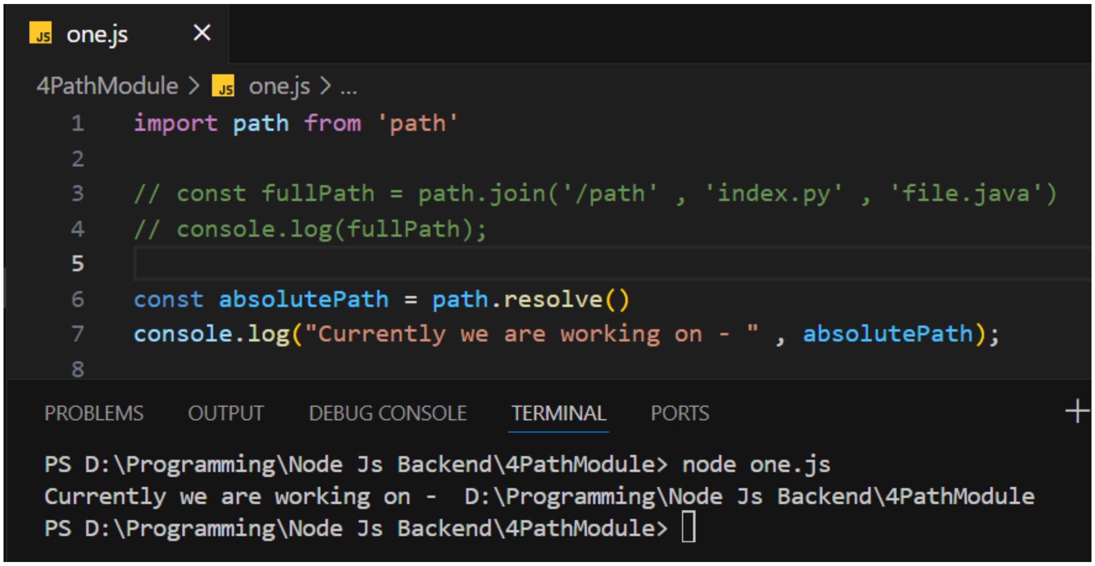
one.js

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

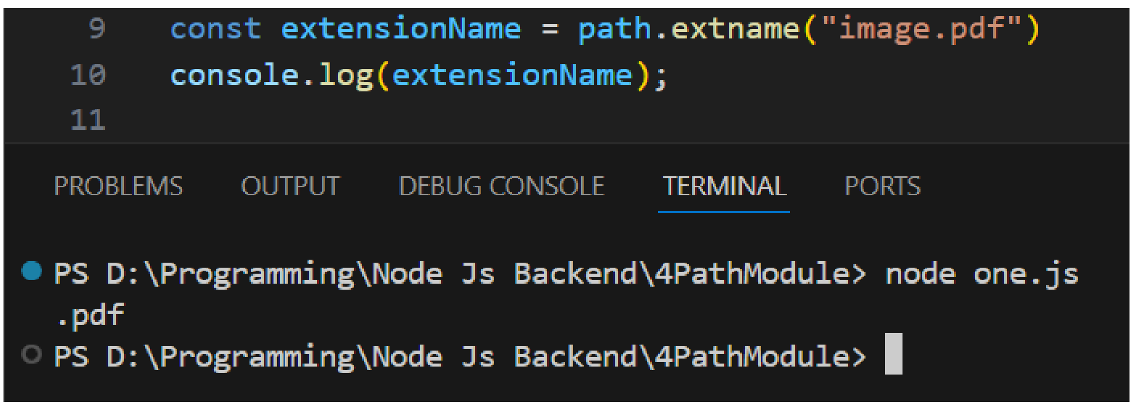
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Programming\Node Js Backend\4PathModule> node one.js
\path\index.py\file.java
PS D:\Programming\Node Js Backend\4PathModule>
```

-

- **Kisi file ka absoulte/full path dekhna hoga to**

- 

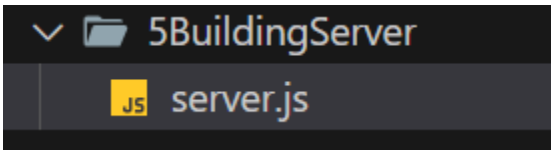
- Get extension name

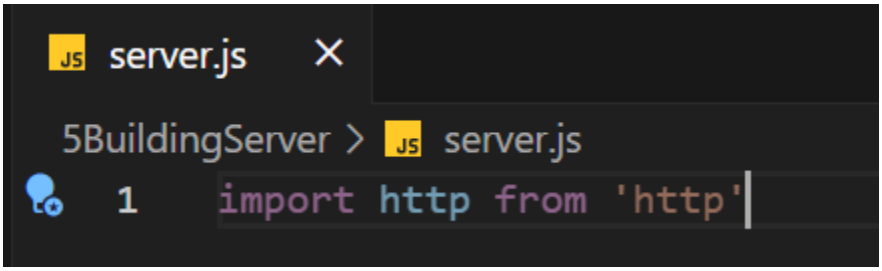
- 

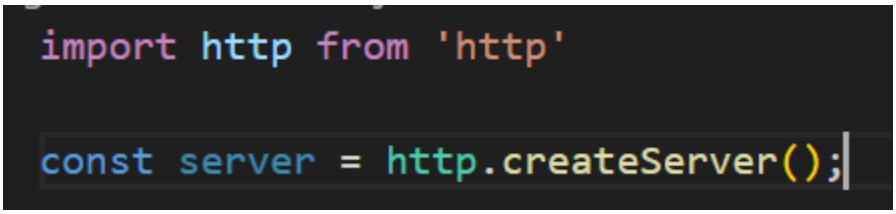
- Conditional on basis of extension name

```
const extensionName = path.extname("image.pdf")
console.log(extensionName);
if (extensionName == ".pdf") {
    console.log("Ok");

}
else{
    console.log("Not ok");

}
```

```
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.
  - 

    Is module ka use karte hai
  - 

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

server.js

5BuildingServer > 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 (5Bui

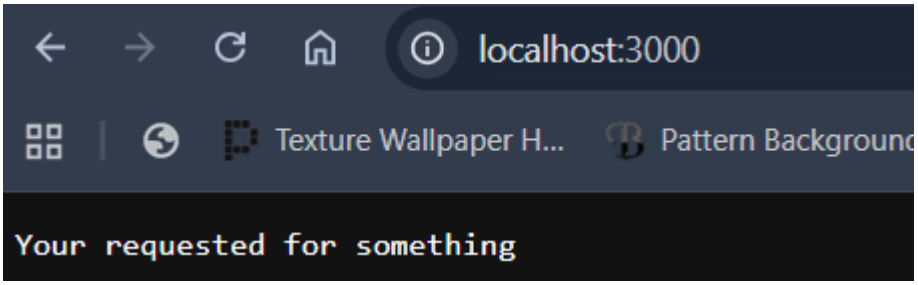
```
PS D:\Programming\Node Js Backend\5BuildingServer> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port no 3000
```

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

server.js

5BuildingServer > 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
```



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

server.js

6Routing > server.js > ...

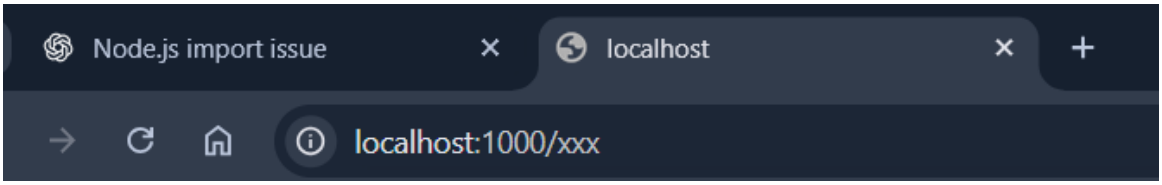
```
1  import http from 'http'
2
3  const server = http.createServer((req,res) => {
4      res.end('<h1>Your requested has been accepted</h1>')
5  })
6
7  const port = 1000
8  server.listen(port, () => console.log(`Server is running on port no ${port}`));
9
```

server.js

6Routing > 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
```

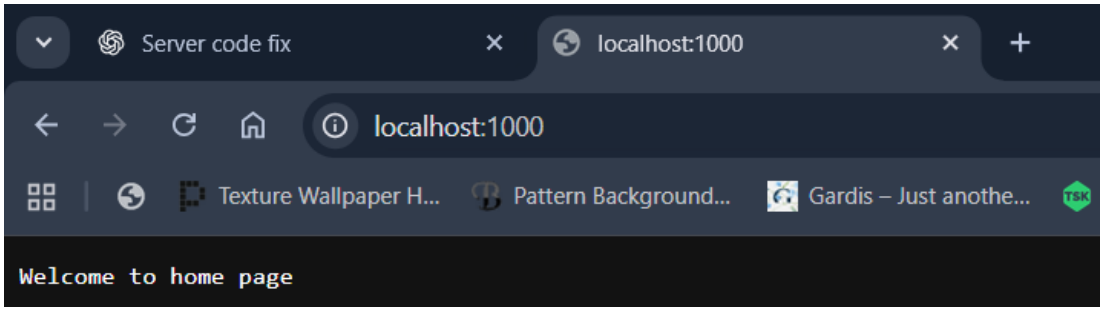
output mai url dekh skte, console pe.

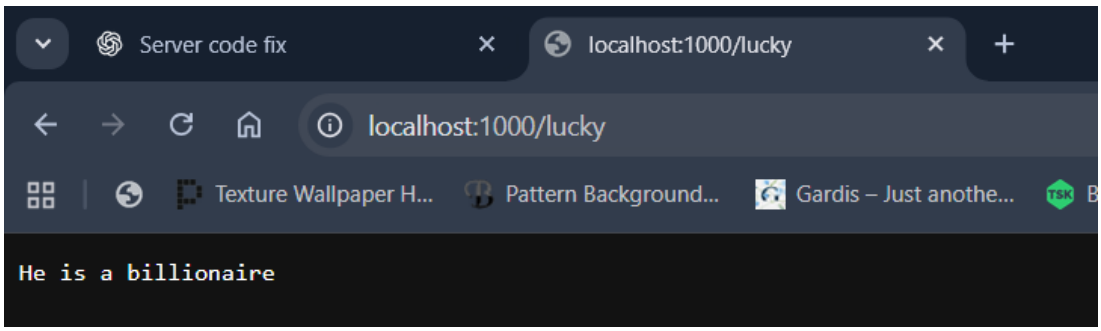
server.js

6Routing > server.js > 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}`));
```

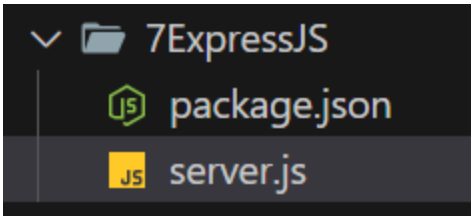
-

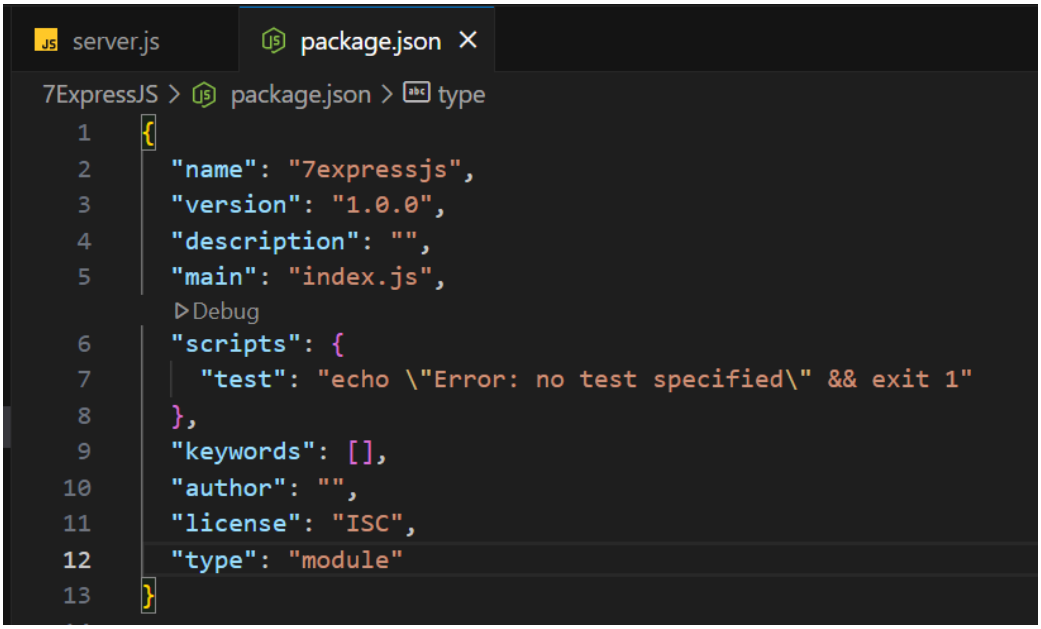




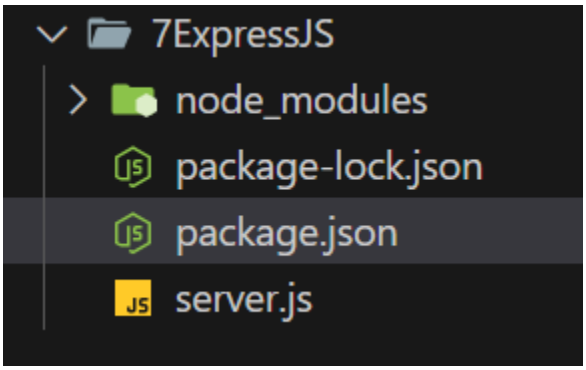
• Express js

- API banane ke kaam aata hai ye

- 

  Type module krlo
- 

- 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 jarrort nahi padhti, express ki jarrart padti hai

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

```
server.js
7ExpressJS > 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

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
```

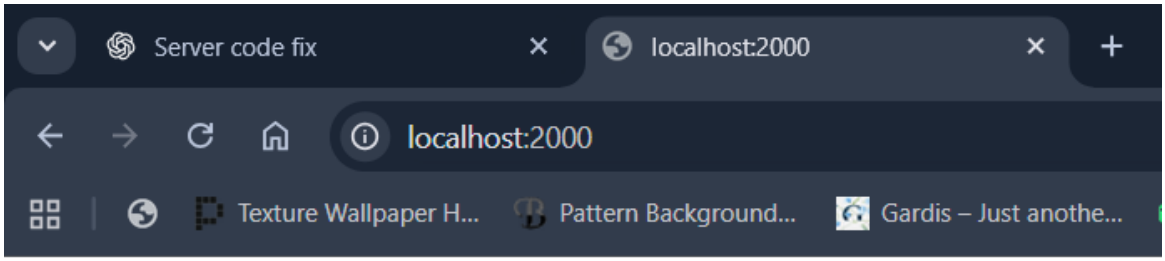
- Ye krke server create ho gaya.

- Ab routing krte hai

```
server.js
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   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

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
```

Ek or route hamne create kiya.

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

- server.js

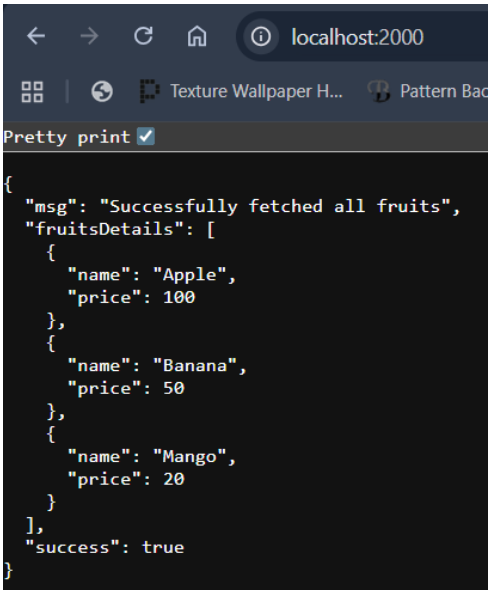
8SendFileInResponse > server.js > fruits

```
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}`))
```

Output :



-

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

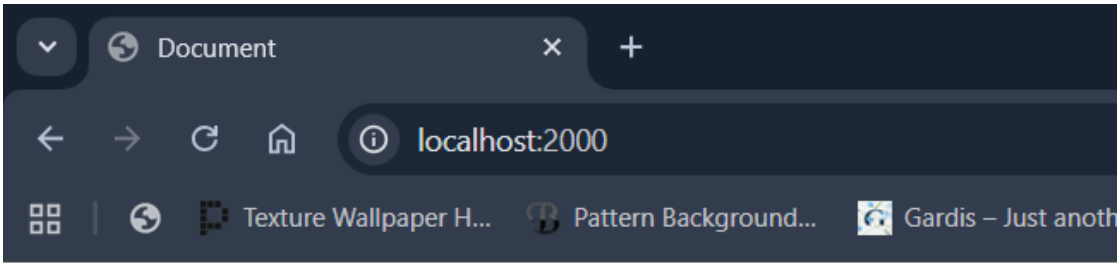
- **Sending html file**
  - Iske liye absoulte path dena hota hai, jo upper sheekha hamne
  -

```
// 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 :

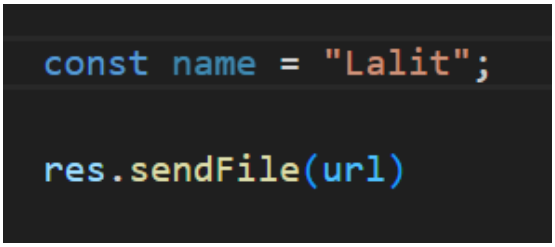
```
[nodemon] restarting due to changes...
[nodemon] starting `node .\server.js`
Server is running on port 2000
Full path =  D:\Programming\Node Js Backend\8SendFileInResponse\index.html
```

- Output :

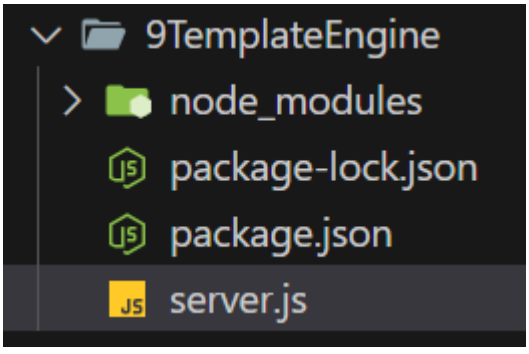


This data is from index . html file

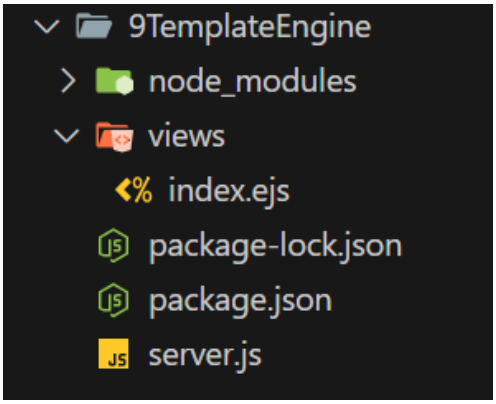
● Template Engine (ejs) || SSR (Server Side Rendering)

- 

  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.



index.ejs

9TemplateEngine > views > index.ejs > html

```ejs
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>i am learning node js</h1>
</body>
</html>
```

Is html file ko mujhe render krke dikhna hai.

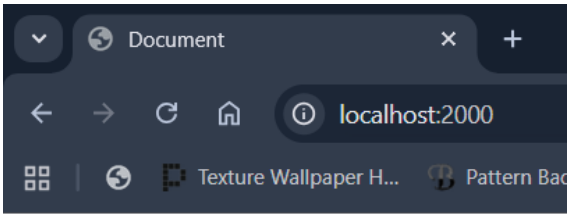
index.ejs | server.js

9TemplateEngine > server.js > ...

```server.js
import express from 'express'

const app = express()

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

const port = 2000

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.

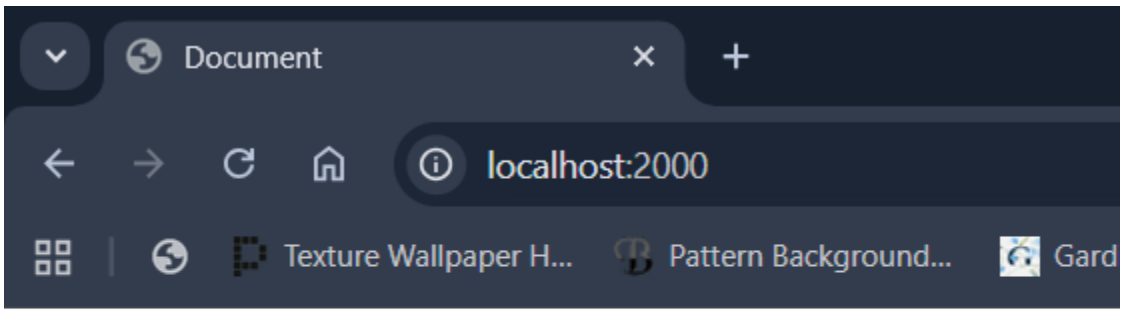
- Ab wo hi name waala kaam jo ham krne waale the wo karna hai.

```
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.



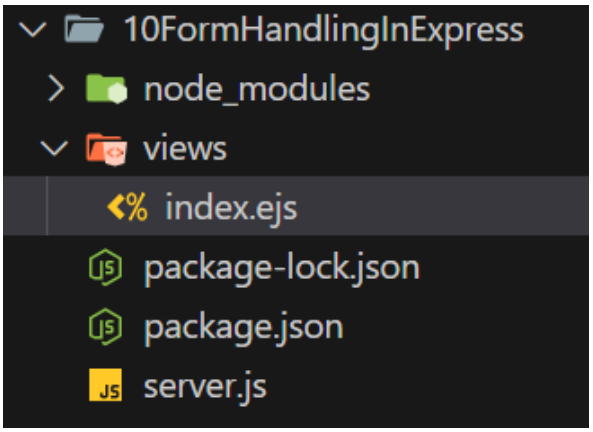
i am learning node js lalit

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

10FormHandlingInExpress > views > index.ejs > html > body > form > br

```ejs
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Form Handling</h1>

    <form action="/form-submit" method="post">
        name <input type="text" name="name" placeholder="Enter your name"/>
        <br/>
        <br/>
        email <input type="email" name="email" placeholder="Enter your email"/>
        <br/>
        <br/>
        <input type="submit" value="Save">
    </form>
</body>
</html>
```

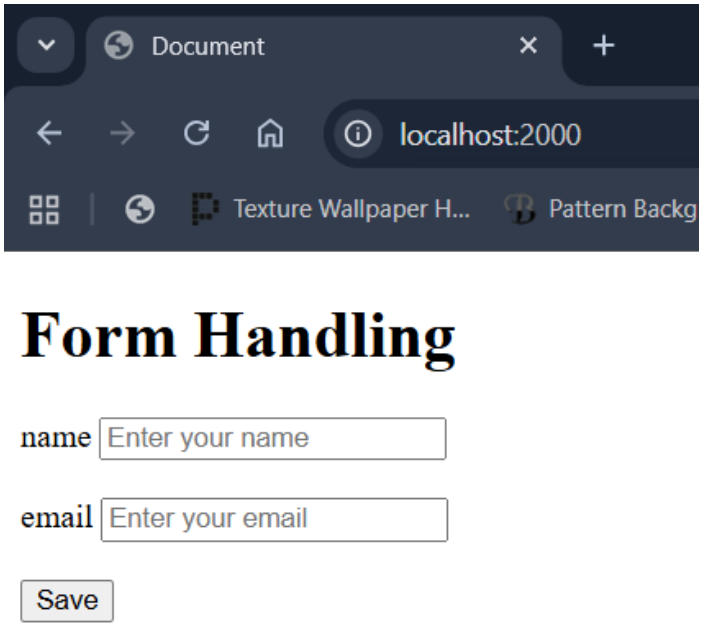
Index.ejs

- server.js | index.ejs

10FormHandlingInExpress > server.js > ...

```server.js
import express from 'express'

const app = express();

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

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

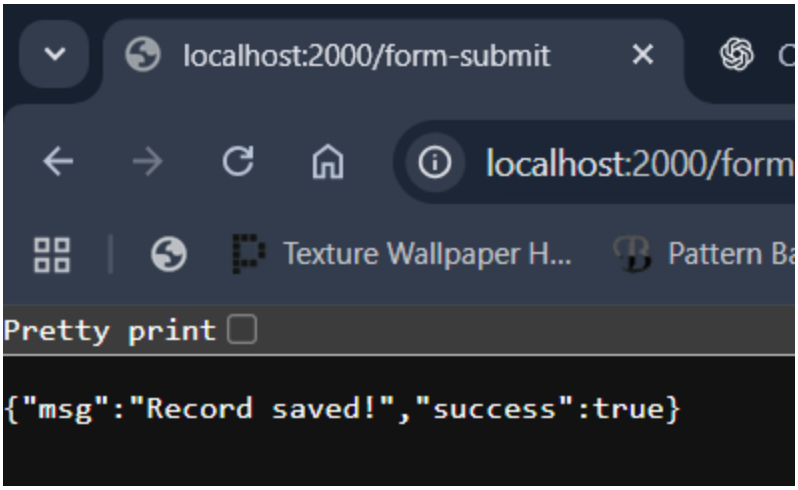
Server code



- 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.



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 krte 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
    })
})
```

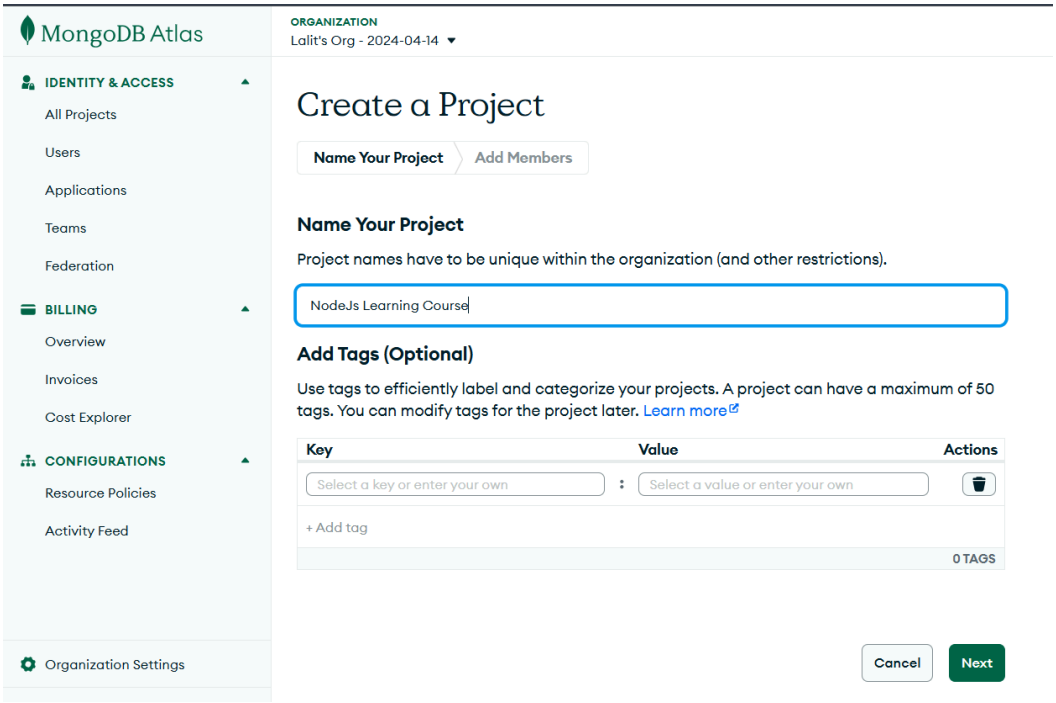
log karwaaya hai data

```
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

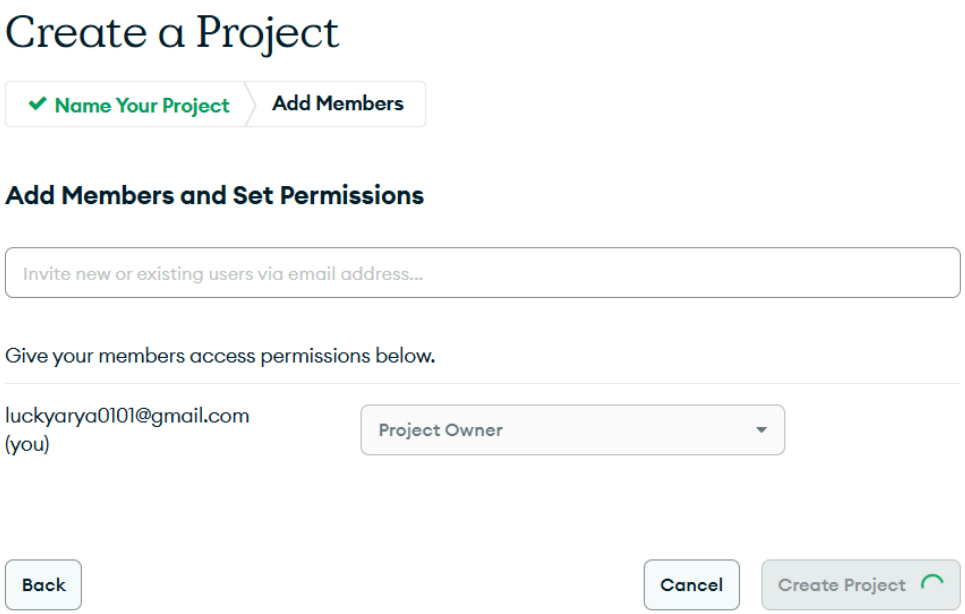
• MongoDB Atlas Setup

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



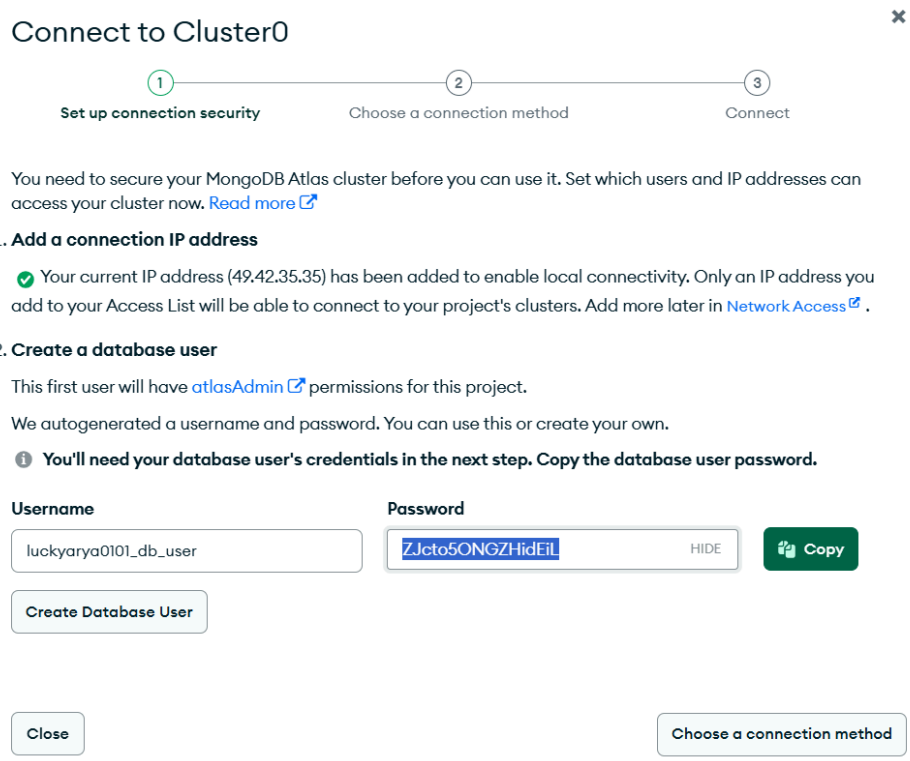
-

Proj ka naam deke next karo.

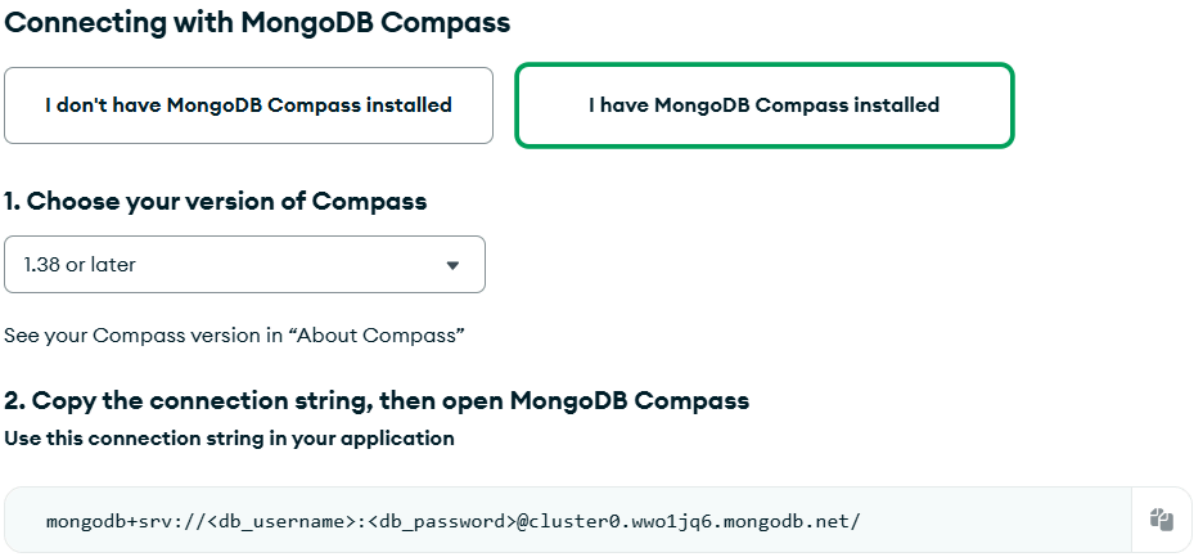


Then neeche create proj pe click krdo

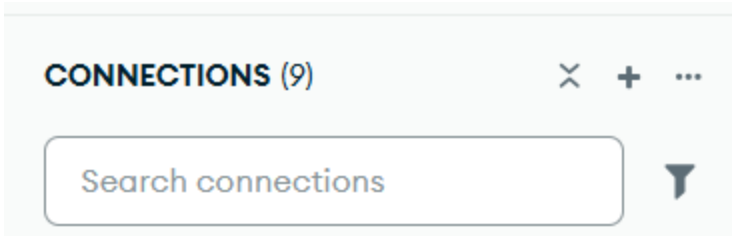
- Ab create cluster pe click kardo
- Cluster banate wqt user name and password aaya
  Username : luckyarya0101_db_user
  Password : ZJcto5ONGZHidEiL



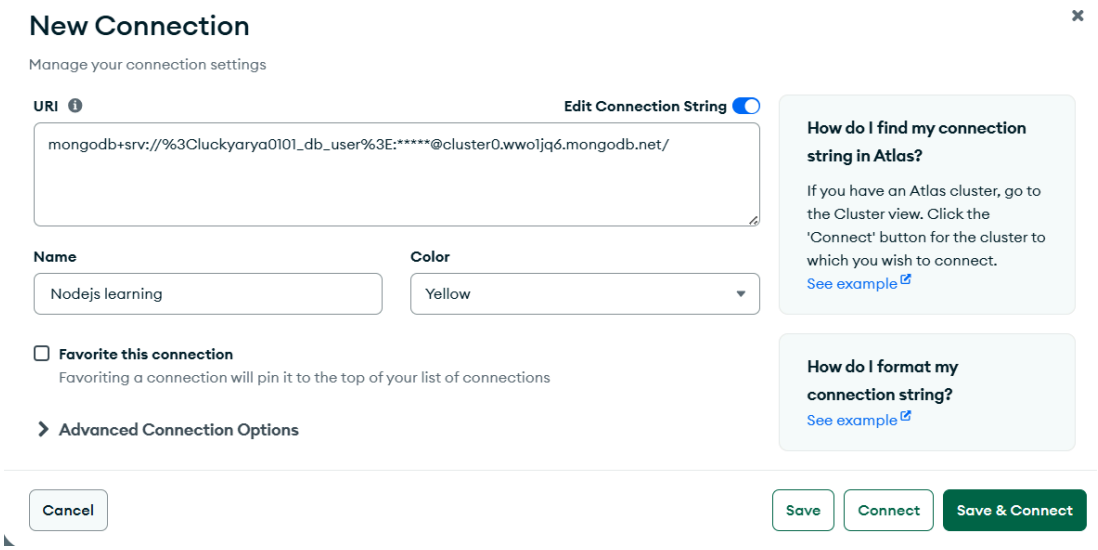
- Ek mongodb compass tool download krna hota hai jo ki GUI based hota hai. Use download kr liya hai



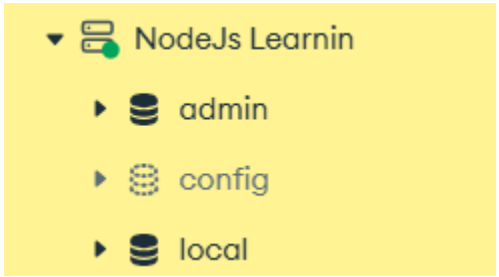
Yaha maine choose kiya hai I have mongodb compass installed and neeche link mila hai **mongodb+srv://<db_username>:<db_password>@cluster0.wwo1jq6.mongodb.net/ mongodb+srv://luckyarya0101_db_user:<db_password>@cluster0.wwo1jq6.mongodb.net/**

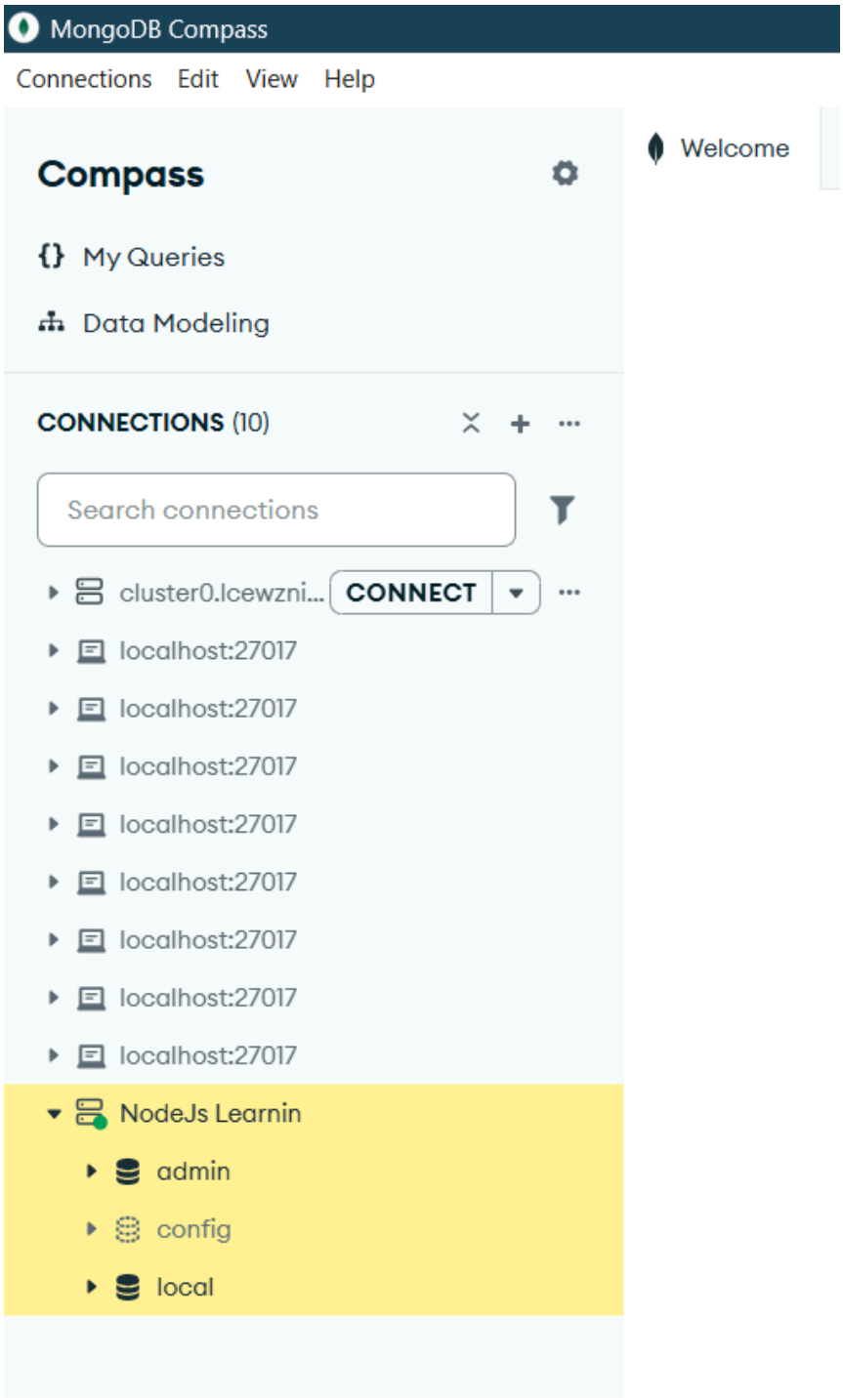
- 

mongodb compass mai new connection pe click karo

- 

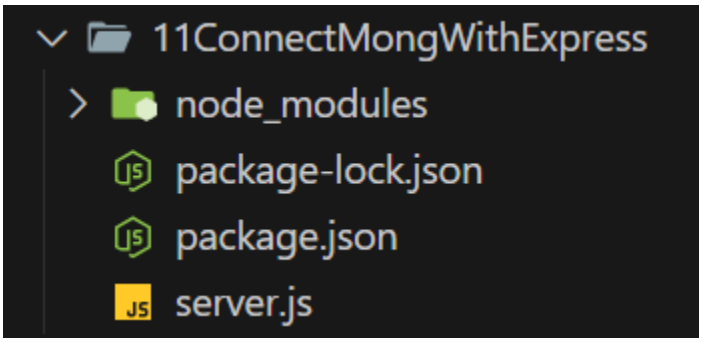
Neeche name and color bhi select krlo
And save and connect

- 



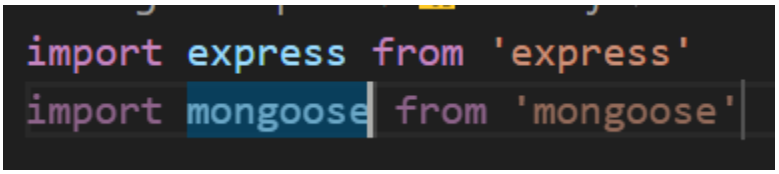
- Connect hoke kuch is trh dikhega compass

- Connecting mongodb with atlas
  - Connect krne ke liye ek package ki jarrart 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.
- 

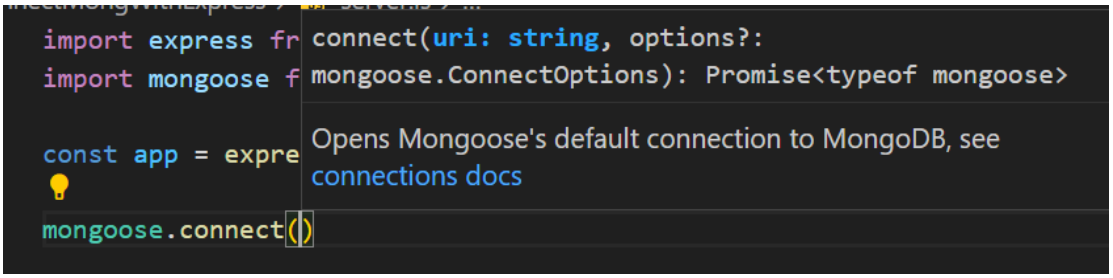
  Ab server banane ka code likhlo

- Ab mongoose se connect karenge

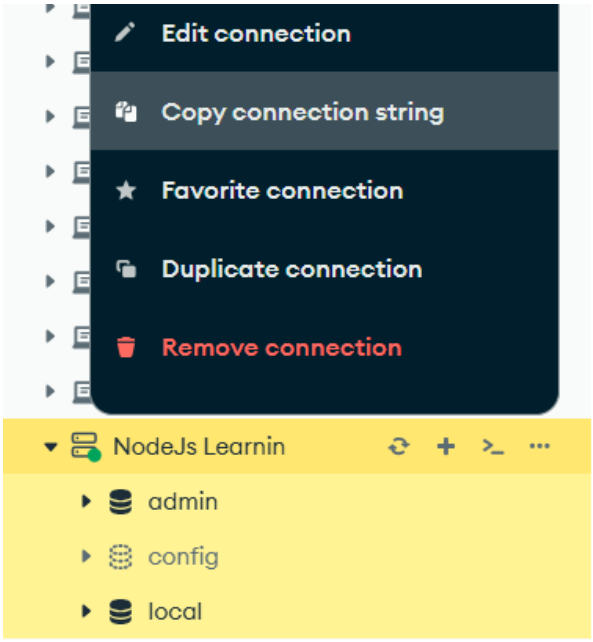


  Pehele import krlo

- Ab mongoose mein method hote hai kuch



  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.wwo1jq6.mongodb.net/")
```

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

-
```
mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
    dbName:"Nodejs Learning"
})
```

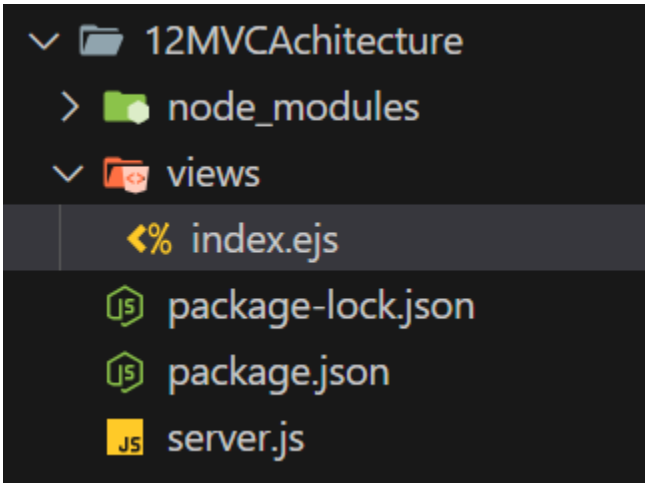
Then ye ek promise return karta hai, to then cathc se handle karna hoga

```
6   mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.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}`))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS   1: node (11ConnectMor

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.
    -

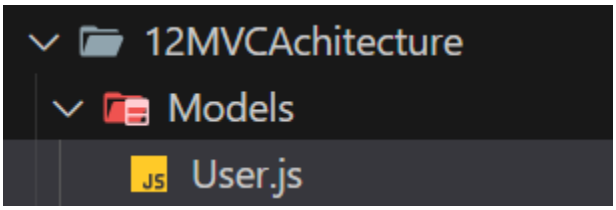


      Itna to ye poorana wala kaam kar liya

    - Model : database ke col ka naam

Views : jaha ejs ki coding, html likhre
Controller :

- Lets create model



User.js mai U always capital

-
```
User.js
12MVCAchitecture > 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},
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

User.js

12MVCAchitecture > Models > User.js > User

```
import mongoose from 'mongoose'

// userSchema means table ka kaam - userTable bhi likh skte
const userSchema = new mongoose.Schema({

    // ab yaha col define kardo
    name:{type:String , required:true},
    email:{type:String, required:true},
    password:{type:String, required:true},
    age:{type:Number},
    phoneNumber:{type:Number},
    createdAt:{
        type:Date,
        default:Date.now
    },
})

export const User = mongoose.model("user" , userSchema)
```

- Ab views mai aake form create kr lenge.

User.js | index.ejs

12MVCAchitecture > views > index.ejs > html > body > form > input

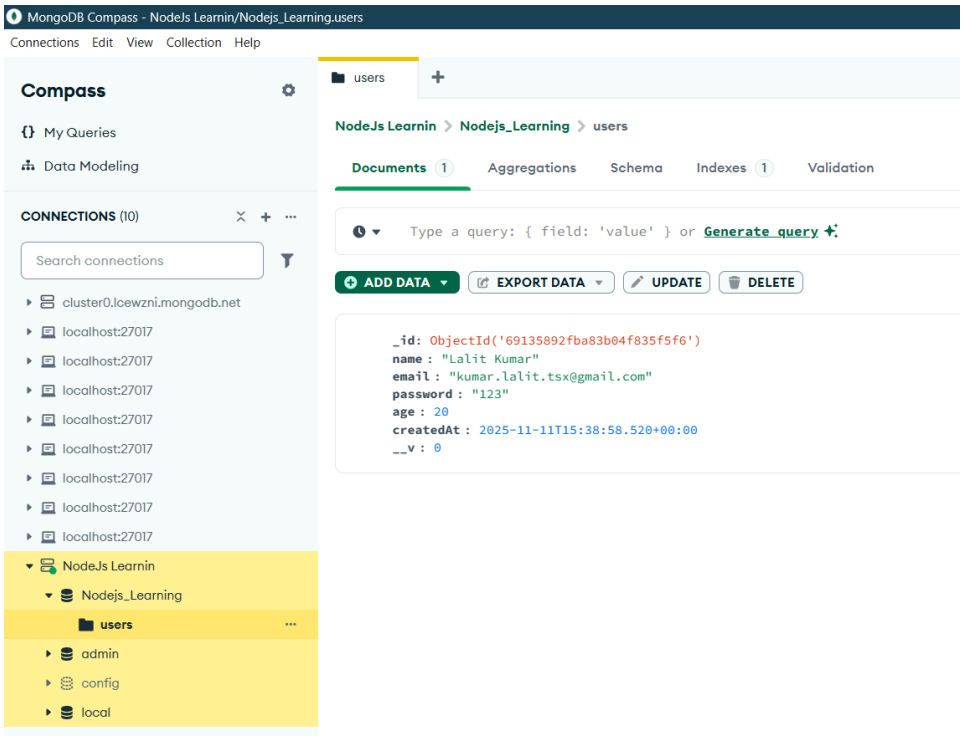
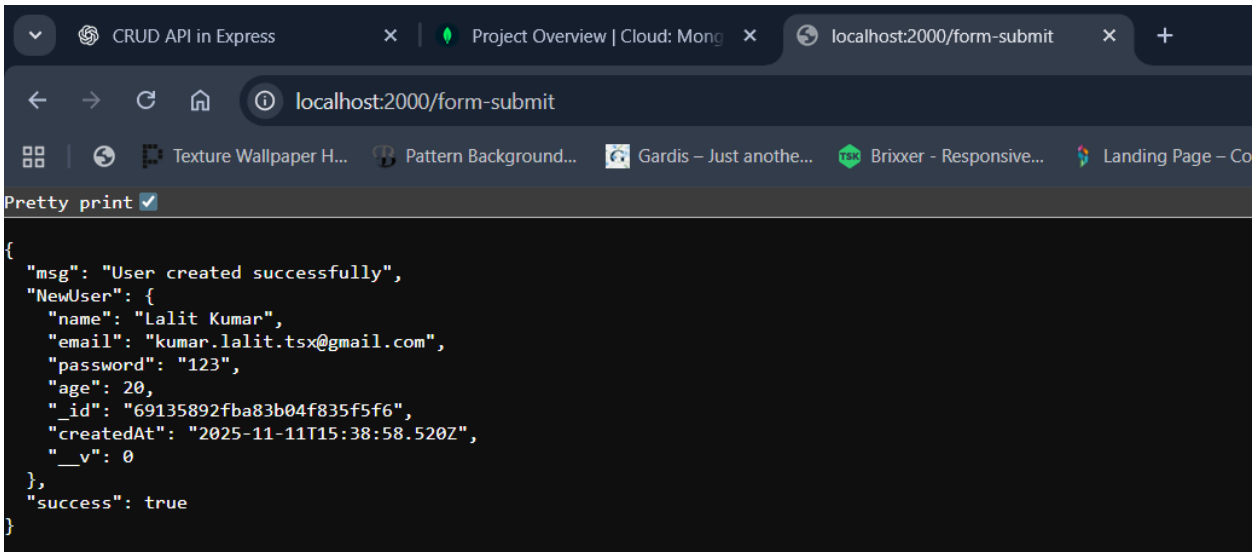
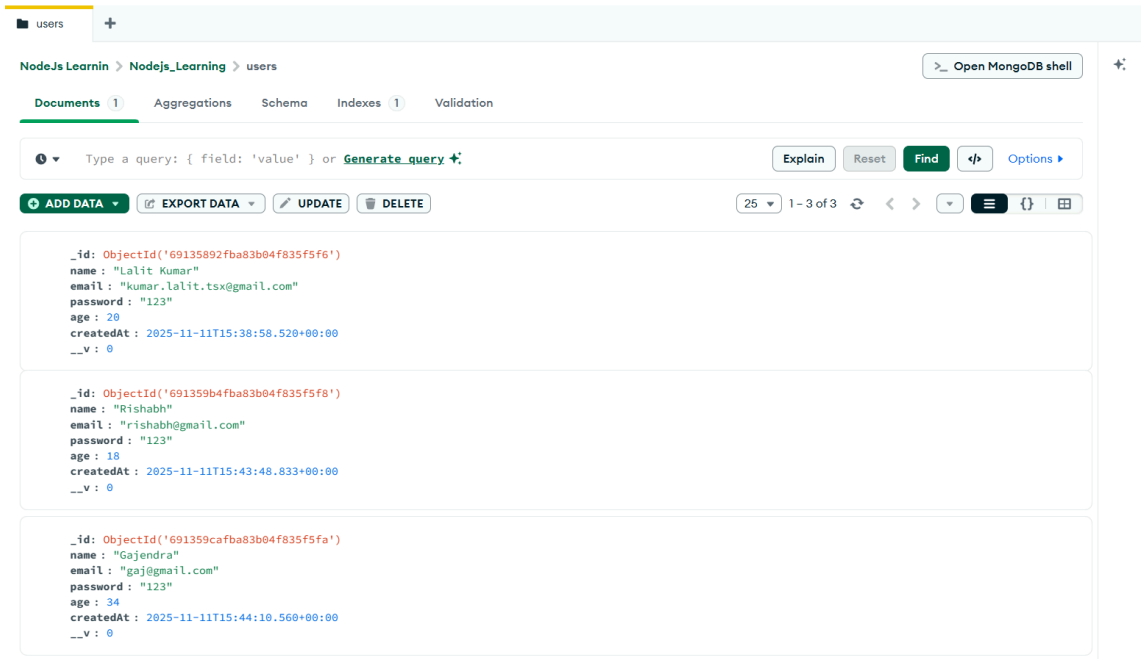
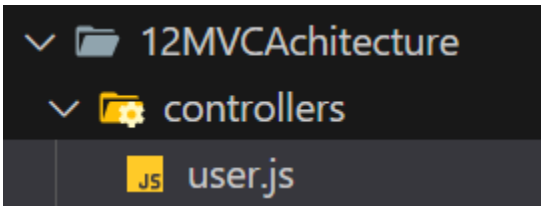
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Model Views and Controllers</h1>

    <form action="">
        Name : <input type="text" name="name" placeholder="Enter name">
        Email : <input type="email" name="email" placeholder="Enter email">
        Age : <input type="number" name="age" placeholder="Enter age">
        Phone Number : <input type="number" name="phonenumber" placeholder="Enter phone number">
        Password : <input type="password" name="password" placeholder="Enter password">
        <input type="submit" value="Save">
    </form>
</body>
</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.wwo1jq6.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',
  phonenumber: '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



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



Ek id bhi mili hai, or createAt 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 controler ke andar likha jata hai.

12MVCAchitecture > controllers > user.js > userRegister

```
import { User } from "../Models/User.js";

export const userRegister = 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
        });
    }
}
```

- 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

13UrlShortner > server.js > ...

```
import express from 'express'
import mongoose from 'mongoose';

const app = express();

mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
    dbName:"Nodejs_Learning"
}).then(() => console.log("Connected")).catch((err) => console.log(err))


app.get('/' , (req,res) =>{

})

const port = 2000;


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
- 

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">

        <input type="file" name="file" id="file-input">

```

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

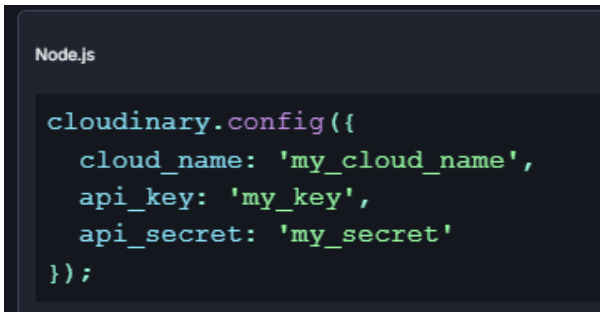
```
<div class="container">
    <form action="" method="post" enctype="multipart/form-data">
        <div class="form-group">
            <input type="file" name="file" id="file-input">
        </div>
        <div class="form-group">
            <input type="submit" value="Upload">
        </div>
  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

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

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

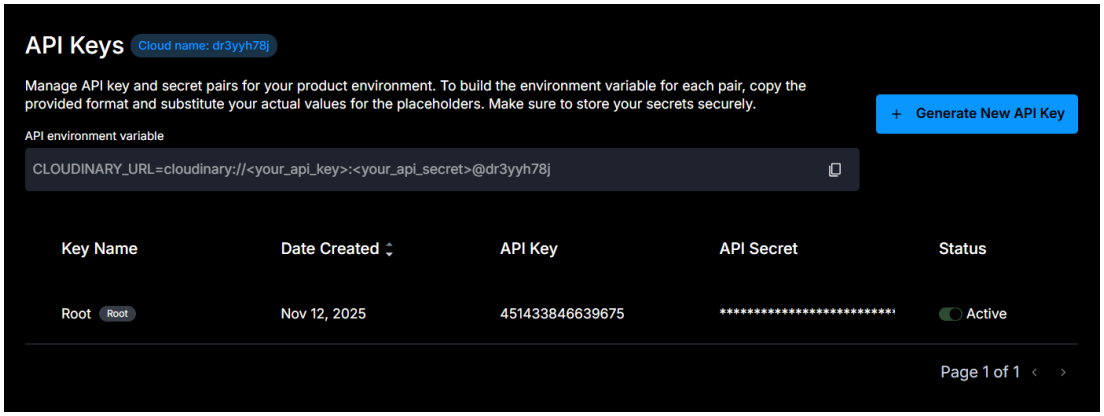
- Ab logic likhna hai ki jo file upload karenge wo cloudinary pe upload honi chahiye na ki mongodb pe. Ham cloudinary pe daalke image ka url le lenge

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

- Import ke baad configuation add karni hoti hai



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



Is page se milegi wo.

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

server.js U | 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
6
7  const app = express();
8
9  cloudinary.config({
10   cloud_name: 'dr3yyh78j',
11   api_key: '451433846639675',
12   api_secret: 'zMA-pP8AmEt-90bPQOvSs02zJ_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 ne neeche.

```
mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.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.extname(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 :

server.js U

14Proj2_ImageShortner > server.js > app.post('/upload') callback

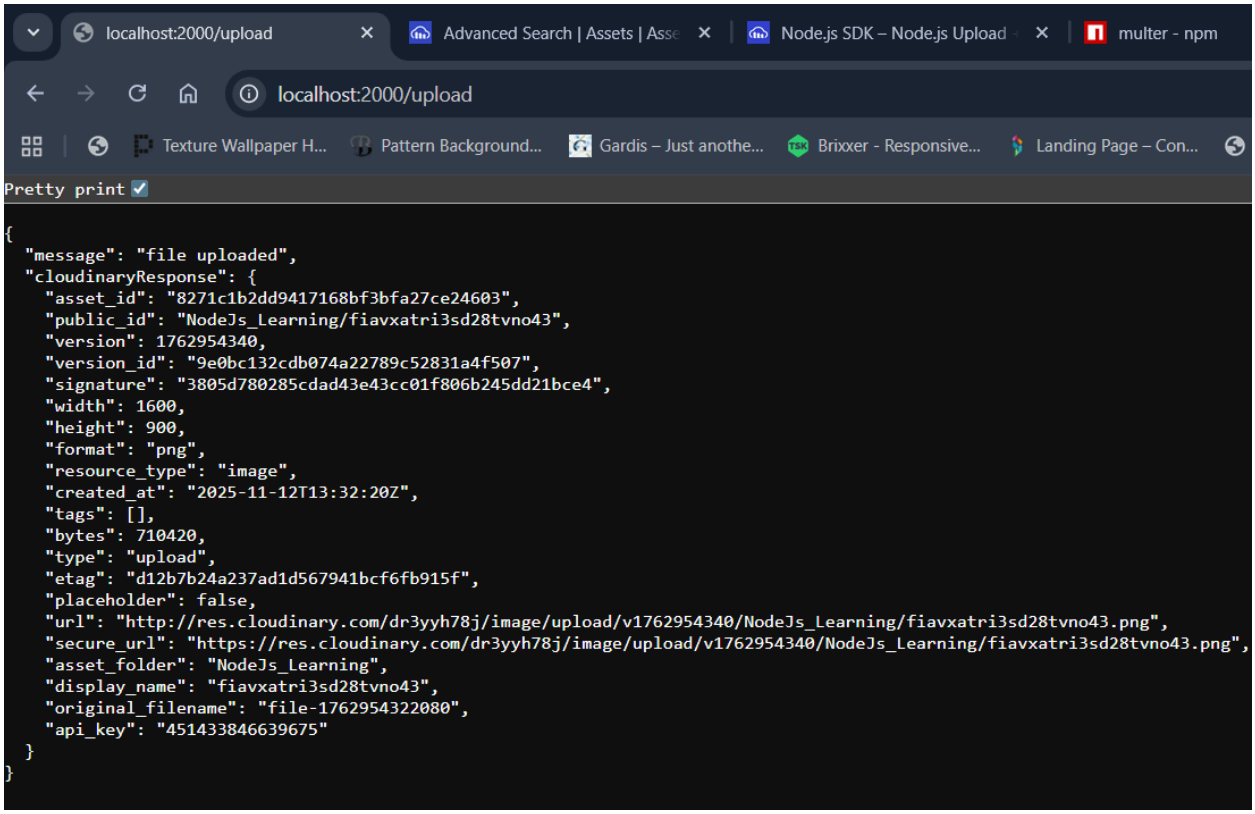
```
import express from "express"
import mongoose from "mongoose";
import multer from "multer";
import { v2 as cloudinary } from 'cloudinary'
import path from 'path'


const app = express();

cloudinary.config({
  cloud_name: 'dr3yyh78j',
  api_key: '451433846639675',
  api_secret: 'zMA-pP8AmEt-90bPQOvSs02zJ_A'
});



mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.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});
});

```

```
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
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 cloudaniay pe upload krte hai then mongodb pe.
  Iske liye uploader funtion 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



Isme jo publich_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 uploded 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 :

server.js U

14Proj2_ImageShortner > server.js > ...

```
import express from "express"
import mongoose from "mongoose";
import multer from "multer";
import { v2 as cloudinary } from 'cloudinary'
import path from 'path'


const app = express();

cloudinary.config({
  cloud_name: 'dr3yyh78j',
  api_key: '451433846639675',
  api_secret: 'zMA-pP8AmEt-90bPQOvSs02zJ_A'
});
```

```
mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.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});
});

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

```
const upload = multer({ storage: storage })

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);
```

```
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"
  })

  // ab database mai save karlo
  const db = await File.create({
    filename : file.originalname,
    public_id : cloudinaryResponse.public_id,
    imgUrl : cloudinaryResponse.secure_url
  })

  res.render('index.ejs',{
    url:cloudinaryResponse.secure_url
  })

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

const port = 2000;
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')
})
```

- Ye raha login ka logic

```
// 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})
    }
})
```

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

- 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

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) => {

})
```

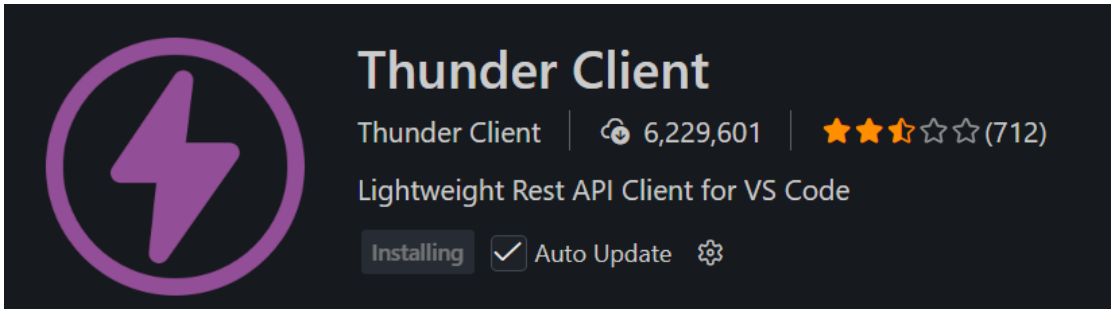
Is tareeke se comment likhte hai.

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

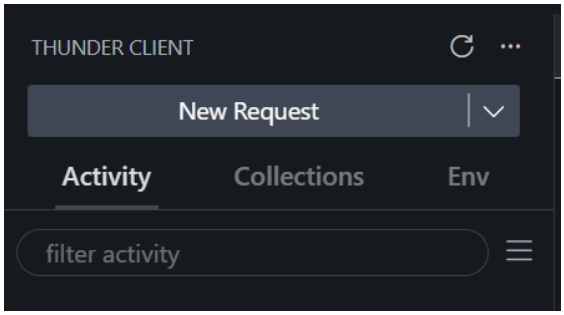
```
app.post('/api/user/register' , (req,res) => {

    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.

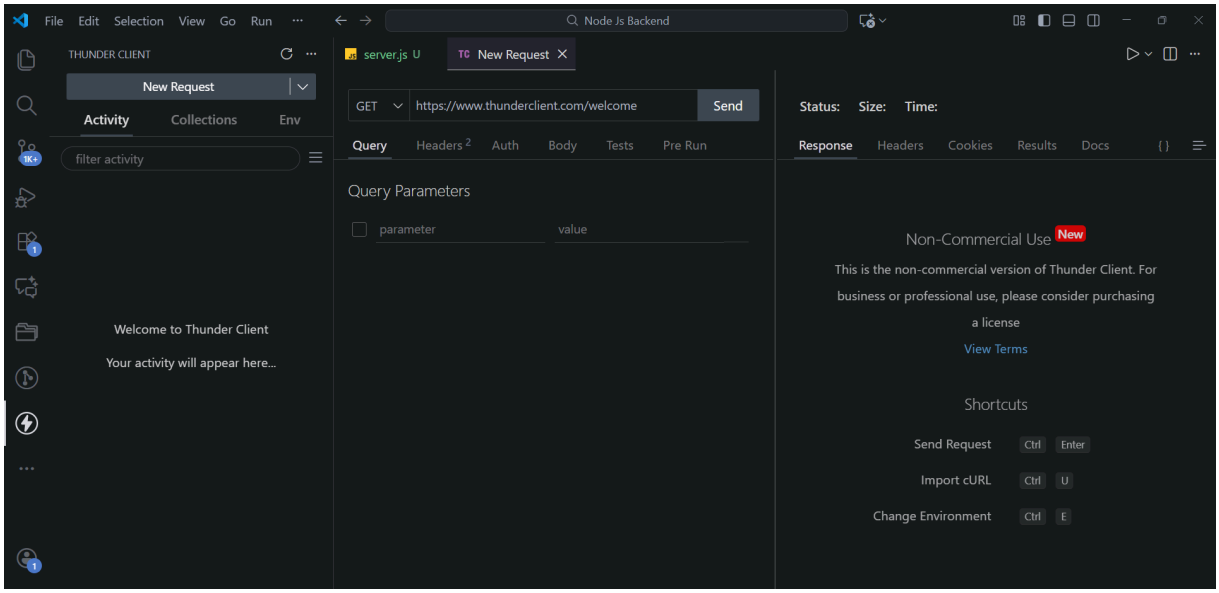
- /api/user/register : ye hi hamari api hoti hai, iski testing hame karni hai.
- To testing ke liye post man ko vs code mai install kr skte, ya fir ek or tool hai thunder client, uska use ham karne waale hai. Ye hi extension hai



- Open extension:



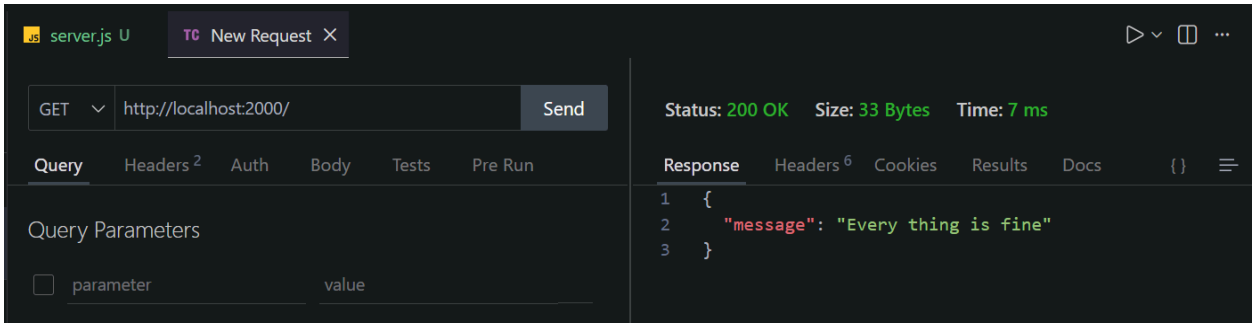
- Click on new req.
-



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"
    });
});
```

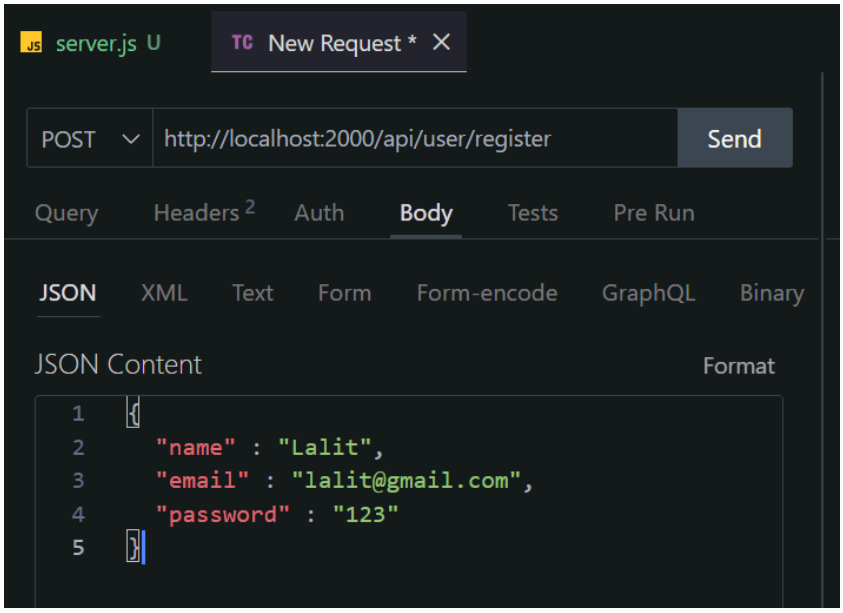


To sahi kaam kar raha hai.

- 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 karwake 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 bodyParser from 'express'
```

  Ab ise use karna hai.

```
app.use(bodyParser.json())
```

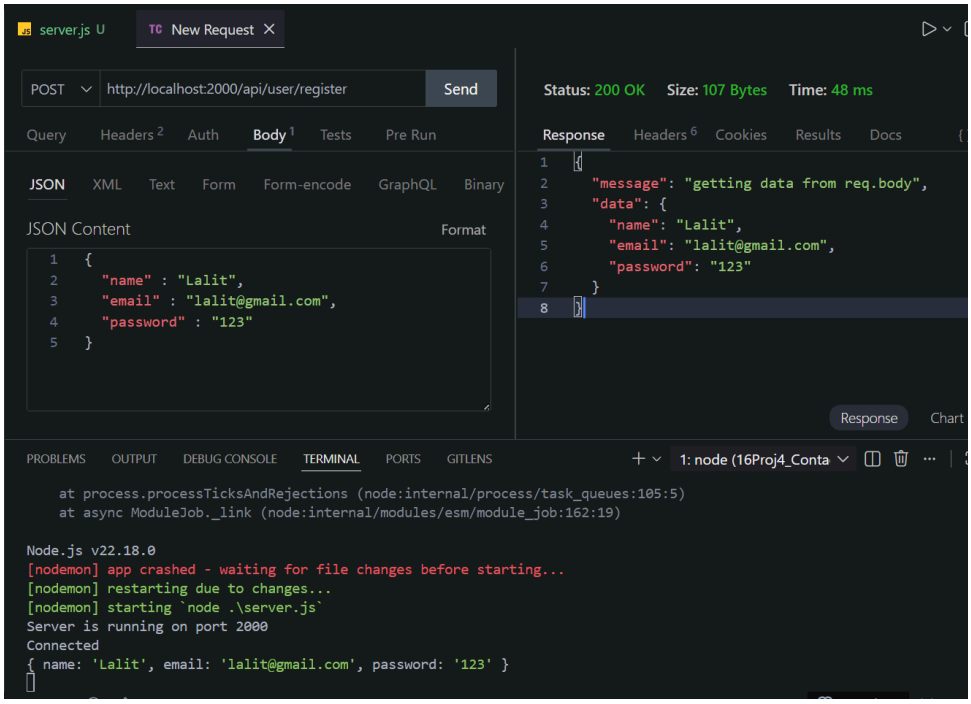
  Is line ka mtlb hai, body mai se parse krke json mai jo data aara hai uske lelo, cuz thunder mai ham json se data dere body ke andar.

```
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



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.

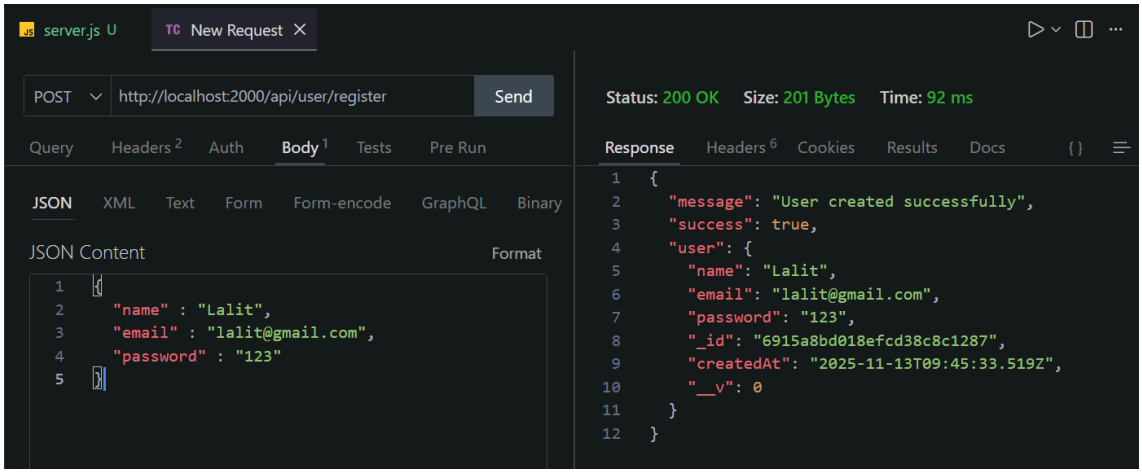
Hame model ki jarrort padegi, to model ko import karlo

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

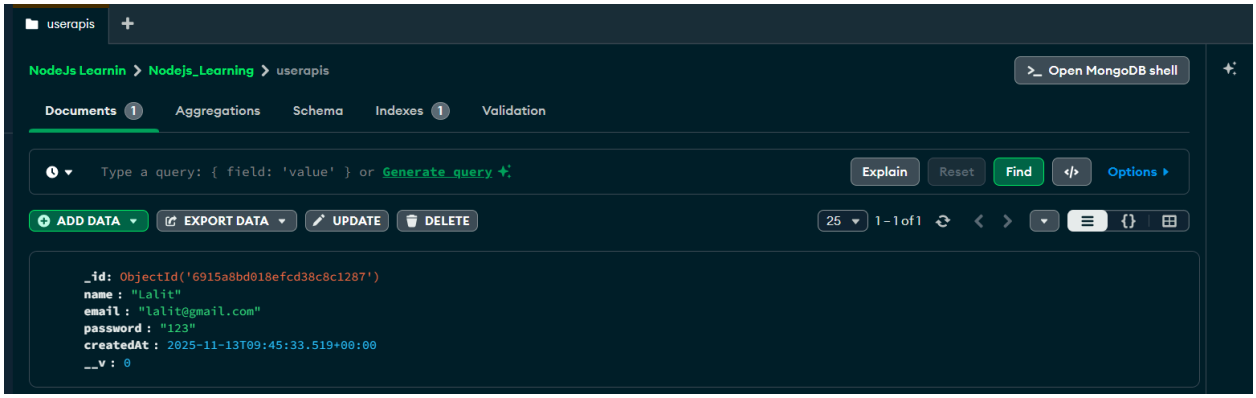
```
// user Routes
// @api description :- user register
// @api method : post
// @api endPoint : /api/user/register
app.post('/api/user/register' , async (req,res) => {

    // 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.



Yaha hame mongodb ka id bhi dekhne ko mil gaya.



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 krre
    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

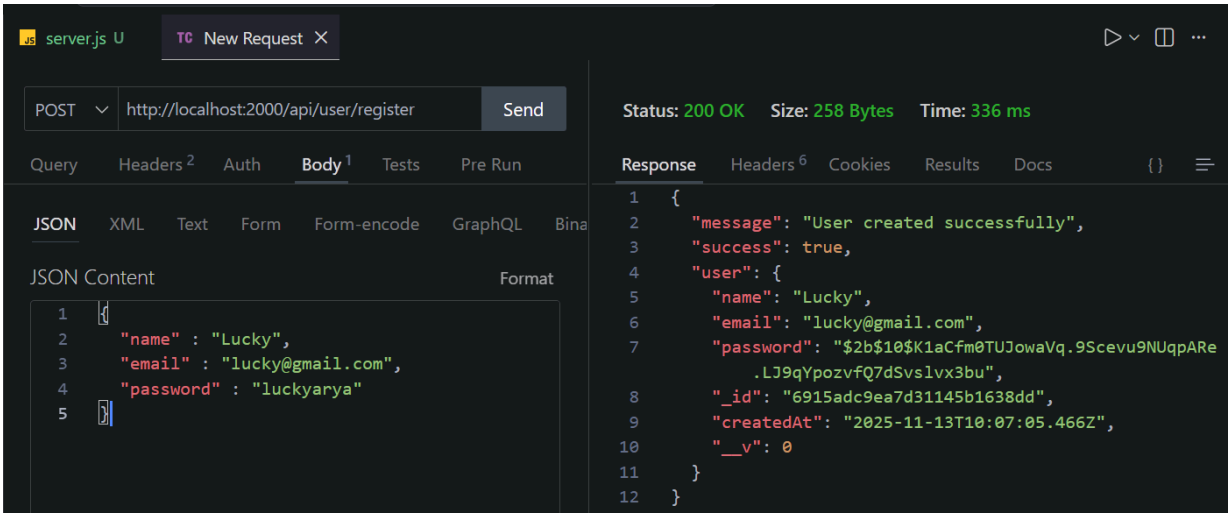
- Bcrypt js
    - Ab jese hamne ye password ko plain mai save kr diya, pr ham password ko kabhi plain mai save nahi karte, ham is library ka use karte hai.
      Ise kehte hai password ko hash karna.

- 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 hashPassoword variable ko pass kr dena fir save krate time.

- Ab ek baar chala ke dekhte hai.



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

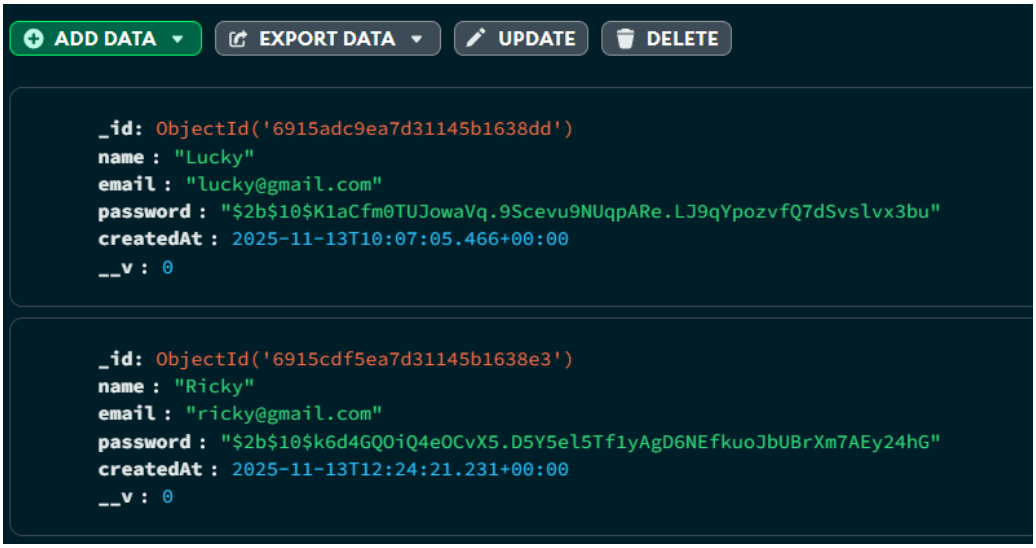


Table mai bhi data hash mai save hora hai.

- server.js ka poora code dera hoon

server.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 {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 × TC New Request

16Proj4_ContactAPI > server.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"
        });
    }

    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
        })
    }



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

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



mongoose.connect("mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/" , {
    dbName:"Nodejs_Learning"
})
.then(() => console.log("Connected"))
.catch((err) => console.log(err))

const port = 2000;
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 | 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);
```

```
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}`))
```

- Controler code:

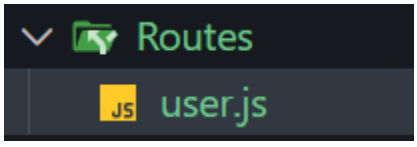
server.js U | user.js U | New Request

16Proj4_ContactAPI > Controllers > user.js > register

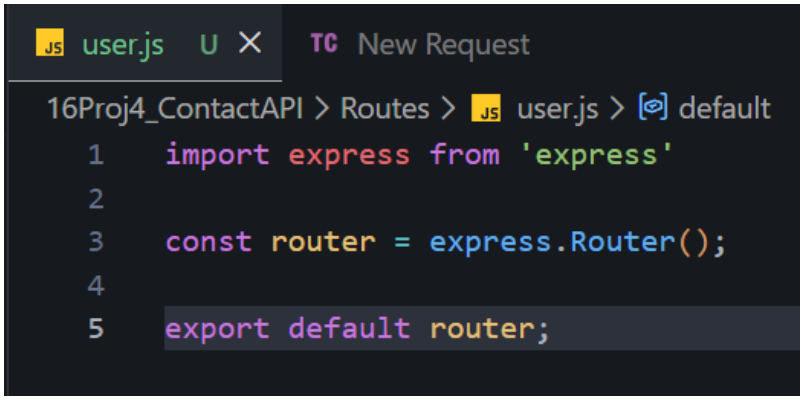
```
1   import {User} from '../Models/User.js'
2   import bcrypt from 'bcryptjs'
3
4   export const register = async (req,res) => {
5
6       // console.log(req.body);
7       const {name,email,password} = req.body;
8       if(name == "" || email == "" || password == ""){
9           return res.json({ // .json likha hai to json ki form mai msg do
10              message : "All fields are required"
11          });
12      }
13
14      let user = await User.findOne({email}) // email ke basis pe find krre
15      if(!user){
16
17          // DB mai add krne se pehle hamko hash karna hai
18          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

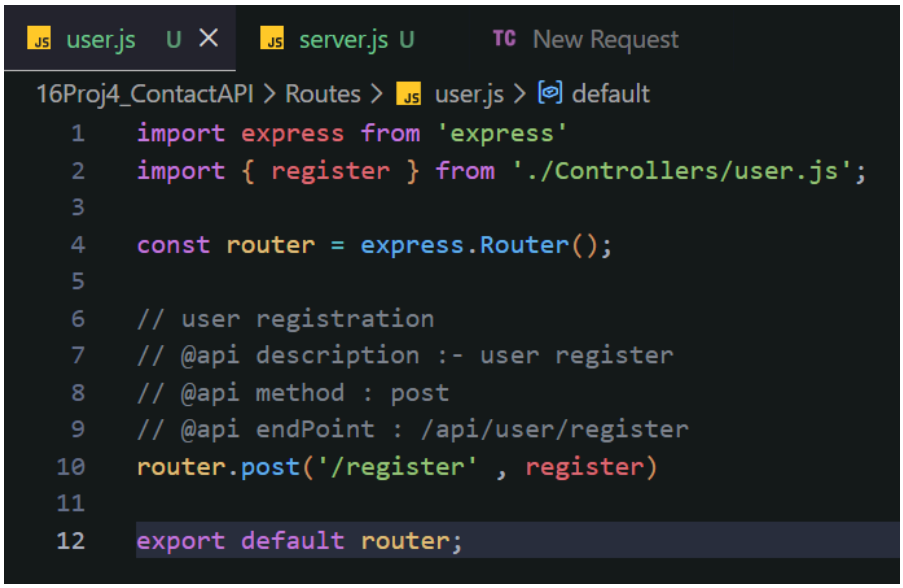
- Abhi ek or cheez saafi karni hai.
  app.post('/api/user/register' , register); : jese maine ye register ka route banaya, agr mujhe login ka route banana hota to? Ya fir 10 or route banane hote to server.js mai hi nahi karta, mai route ka ek alg folder banaunga, or usme karunga ye kaam.



- Ab is file ke andar router banana hoga.



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



Jese server mai app.post krte 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 comman rahega, then **userRouter** se agr register ki req aayegit 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 jarrori 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

-
```
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

}
```

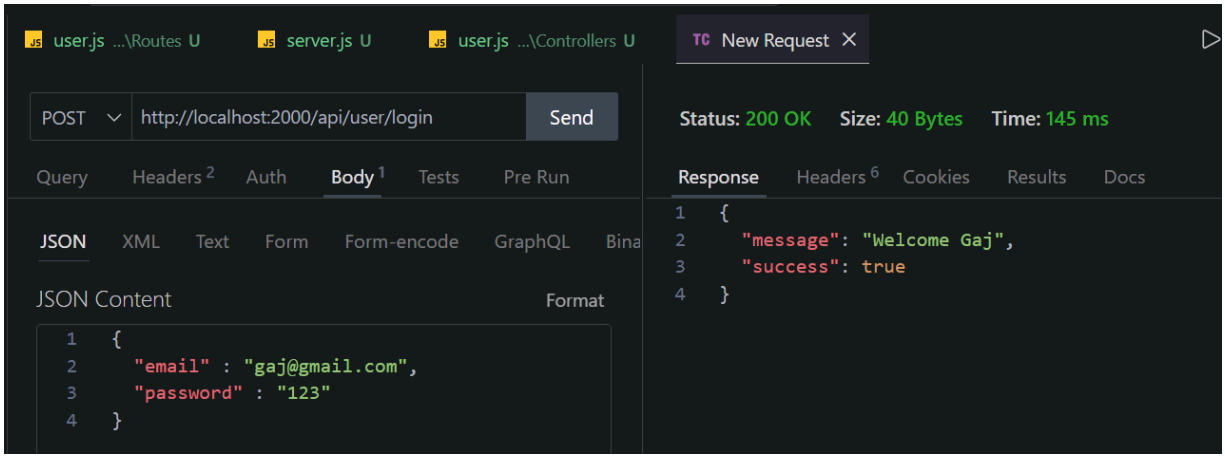
Is kaam ko ese bhi kr skte,
Const email = req.body.email
Const password = req.body.password
Pr ye long way hai

```
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 liya ki user textbox pe password dera hai kya wo equal hai jo password database mai save hai.
- Ab chala ke dekhte hai.



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 jarrort 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 kr lete 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 krdo
- Ab controller mai kaam hai saara, import karlo jwt ko

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

```
// dcrypting 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
    })
}
```

-

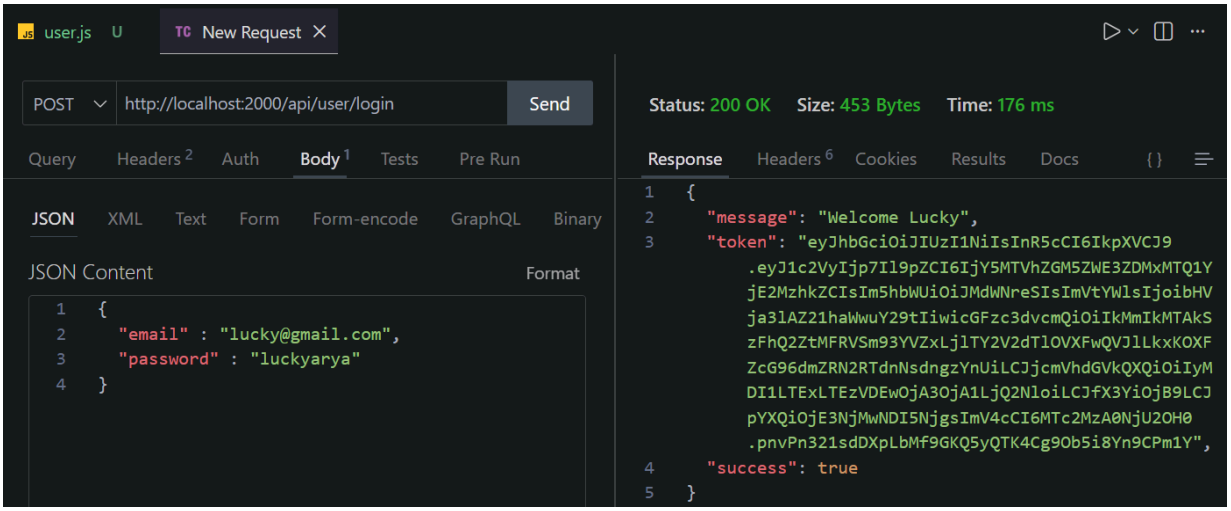
Yaha use karo jwt ko successfull login krne ke baad, else part mai.

- Ab mujhe jwt mai ek sign funtion milte hai. Ye 3 cheeze leta hai
    1. userId or user
    2. Seceret 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

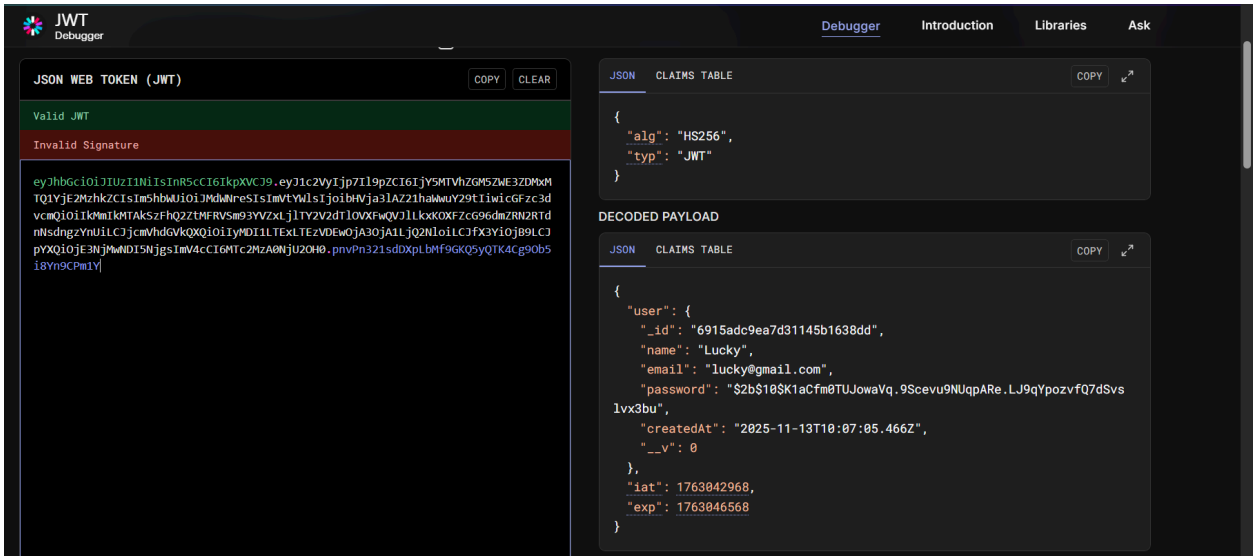
```
// dcrypting password
const validPassword = await bcrypt.compare(password , user.password)
if (!validPassword) {
    return res.json({
        message:"Invalid password",
        success:false
    })
}
else{
    // jwt
    const token = jwt.sign({user : user} , "mysecretkey@123" ,{ expiresIn: "1h"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

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

- Token ko response mai dena jarrori hai.

Output



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



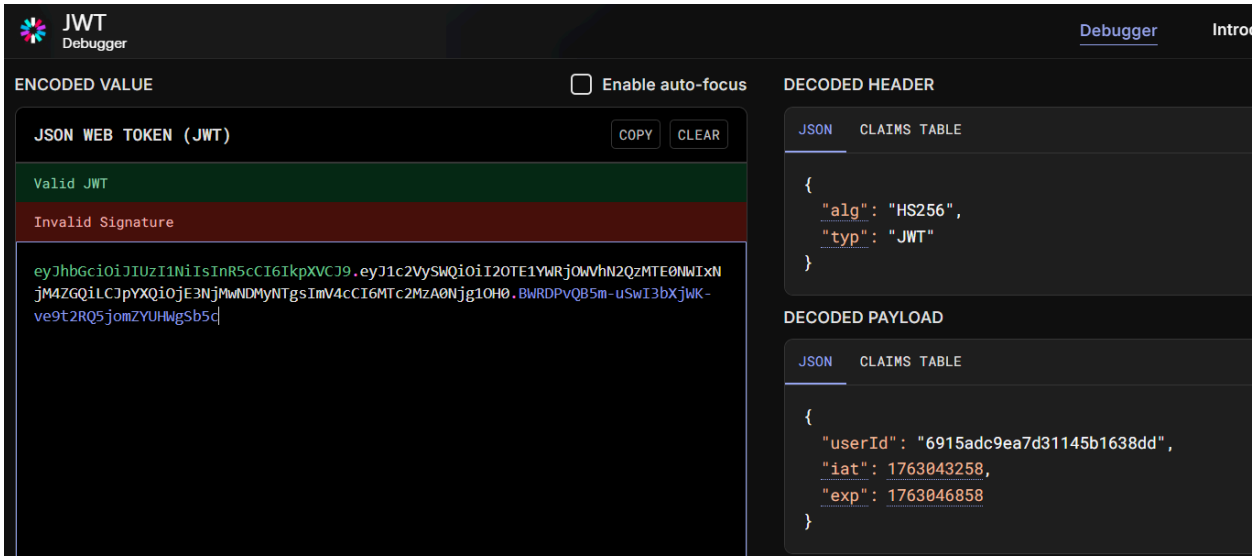
Pr mujhe itni details nahi dikhani or na hi chahiye mujhe sirf id chahiye, _id

- To

```
// dcrypting 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.

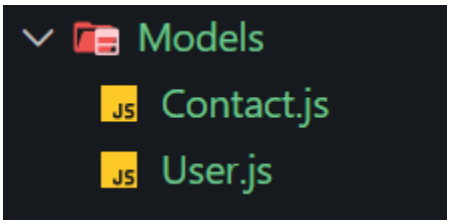
- Ab chaloge to chhota token bange



- **Create Contact API**
  - Sbse pehle Contact.js naam se model bana lo.

```
Contact.js U
16Proj4_ContactAPI > Models > 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.

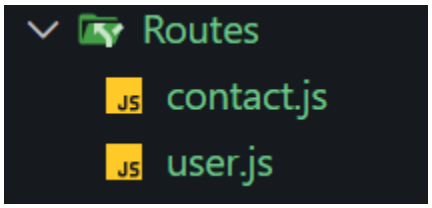


- Ab banao contact.js controller.

```
contact.js U
16Proj4_ContactAPI > Controllers > contact.js > 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.




```
contact.js U
16Proj4_ContactAPI > Routes > 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.

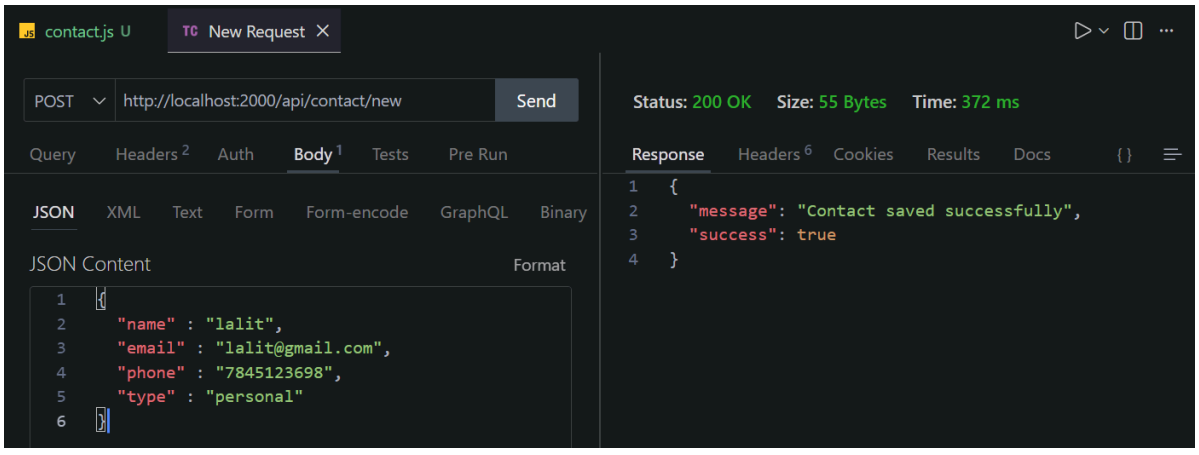
- Ab controller ka code poora karte hai.

contact.js U | New Request

16Proj4_ContactAPI > Controllers > contact.js > newContact

```
import {Contact} from '../Models/Contact.js'

// create new contact
export const newContact = async (req , res) => {

    const{name,email,phone,type} = req.body

    // agr koi field nahi diya to
    if(name == "" || email == "" || phone == "" || type == ""){
        return res.json({
            message : "All fields are required",
            success : false
        })
    }

    // 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",
        success : true
    })

}
```

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

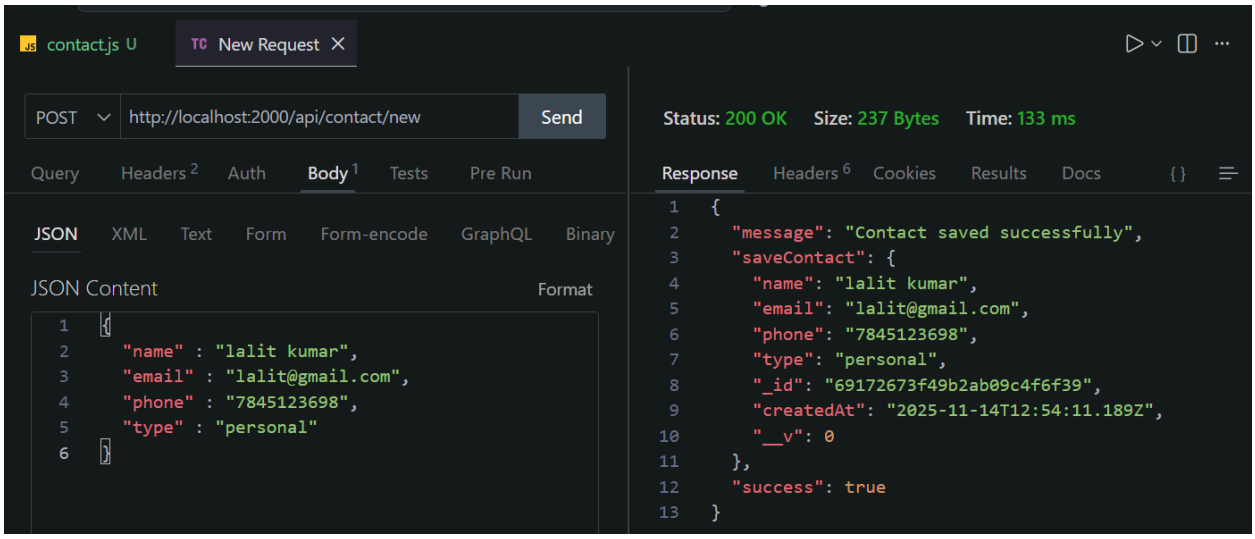


Suppose save hone ke baad yhi tool pe mai wo data dekhna bhi chahta, to response mai bhej do bas,

```
// 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
})
```

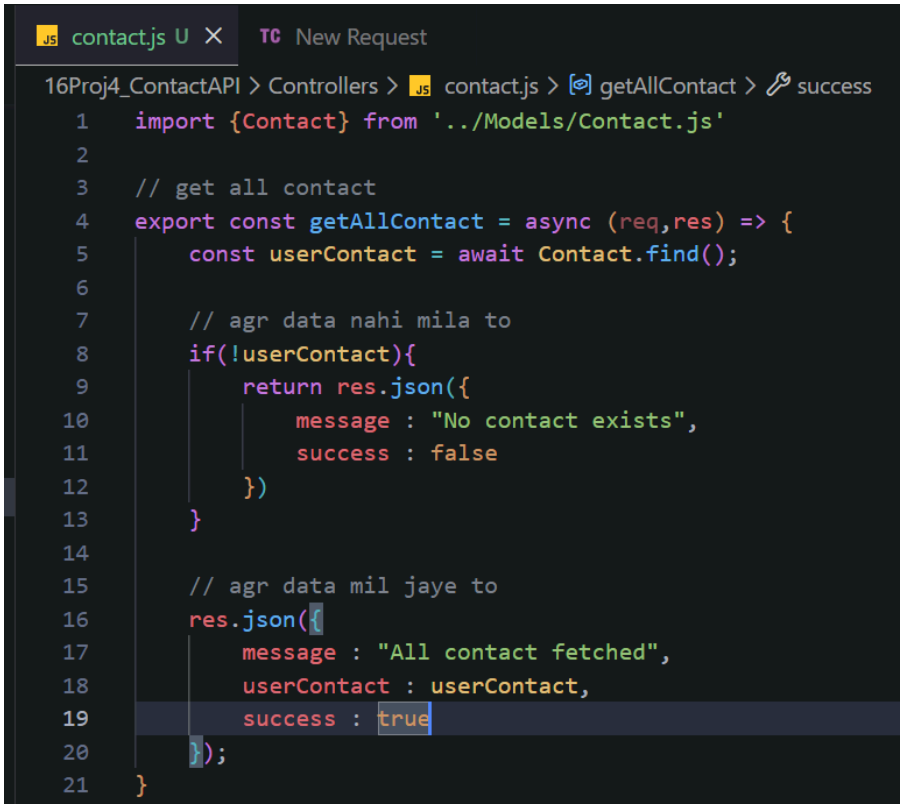
Response mai saveContact bhej diya.

- Ab new record daala



• Get All Contact - API

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



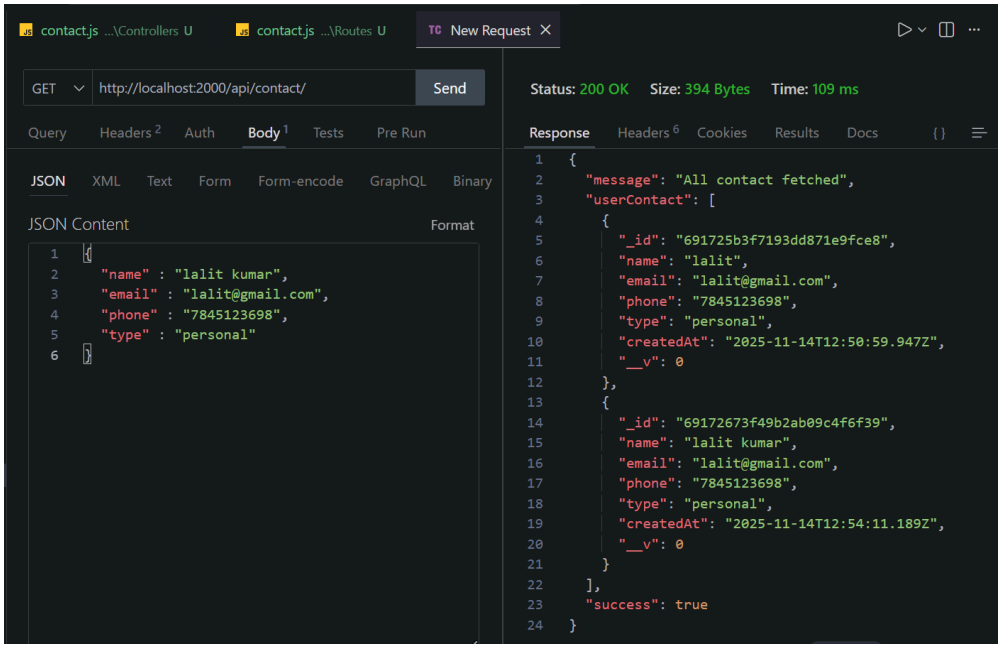
Get ka poora controller ka code.
Ab iska route banana hoga, us route pe jake hi to check hoga na? Us route ko hit karenge to ye gata get waala show hoga.

-
```
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



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
    });
}
```

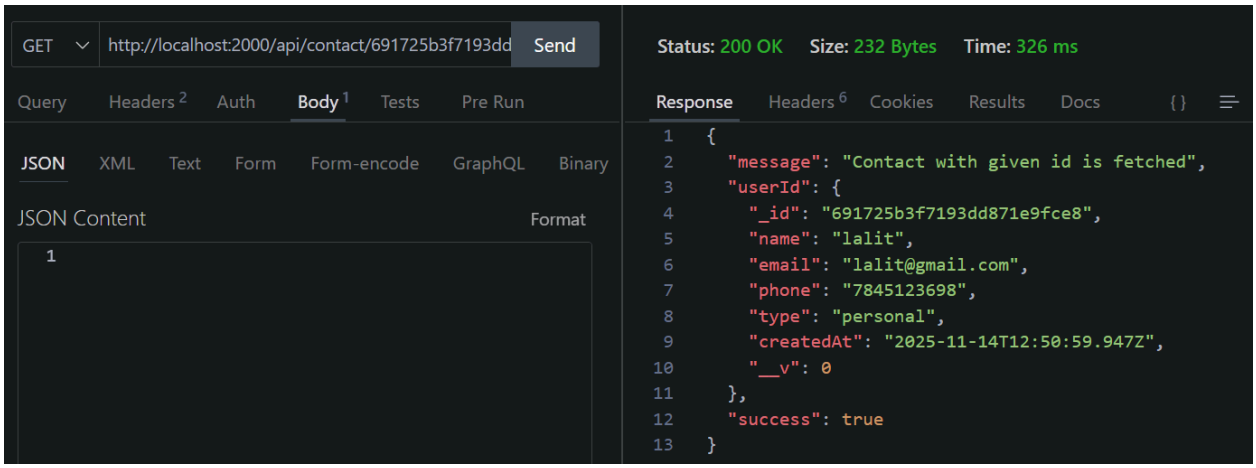
    pehle id nikaal li parameter mai se,cuz parameter hi to pass hota hai id waala
  - **Note :** maine req.params.id, likha hai, yaha jo id likha hai, wo important hai, ye hi naam se ab route banega, to route ke end pont ka nam bhi ab id hona chhaiye.

- req.parama.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 jarrori tha and id naam same hona chahie.
Now test karo.



- 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

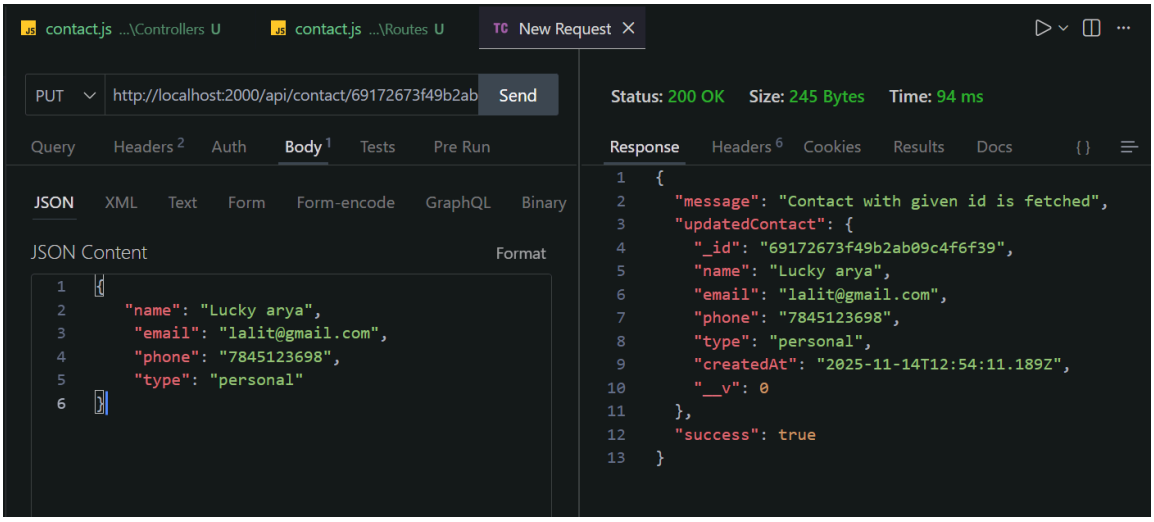
- Update krne ke liye ek method hota hai : findByIdAndUpdate()
  findByIdAndUpdate 3 cheeze leta hai,
  1. Id
  2. data jo ham update krna chhahte hai.
  3. new:true, means new field agr aaye to use bhi set kardo

```
// 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 hoga
    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)
```

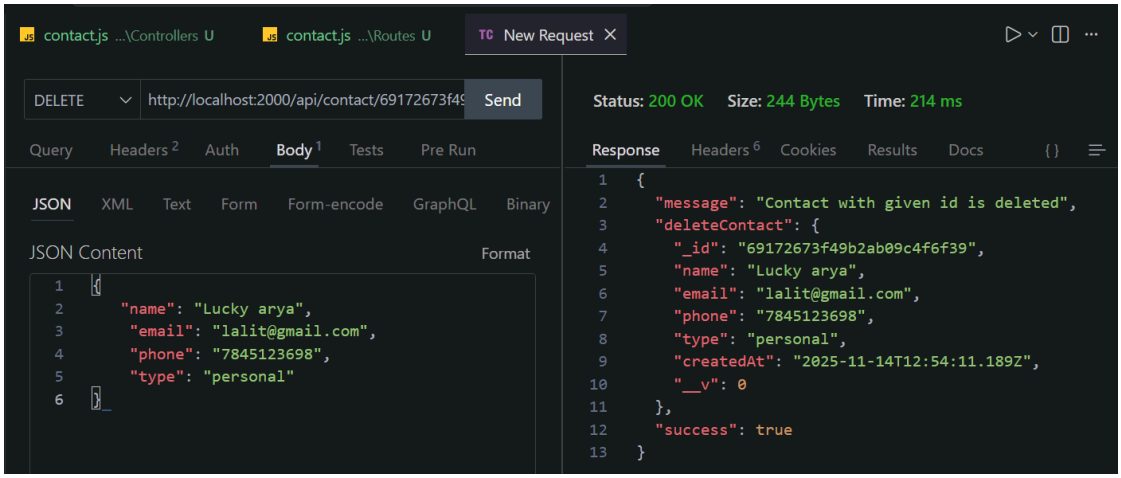
- Test karlo



- **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



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

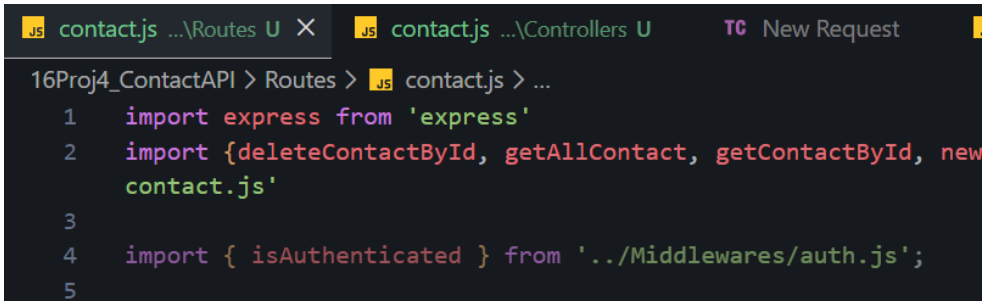
Auth capital mai rakhna. Authentatication ka middlware h ye.

- 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.

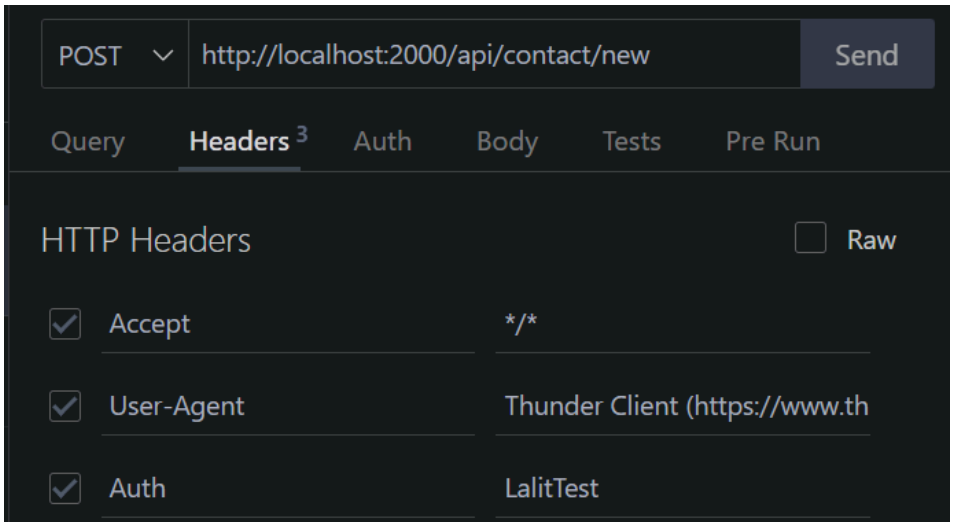


- And create contact waali route mai ye karo

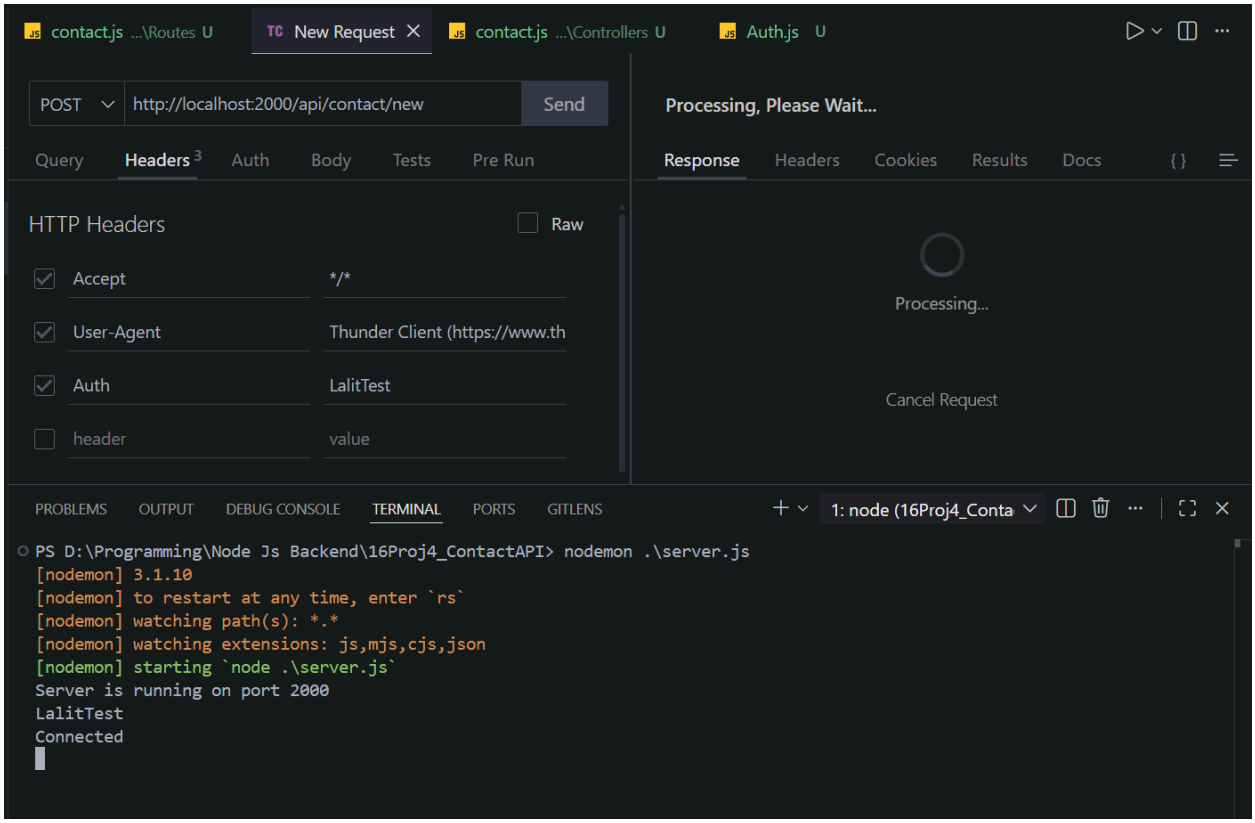
```
// new contact
// @api description :- creating contact
// @api method : post
// @api endPoint : /api/contact/new
router.post('/new' , isAuthenticated , newContact)
```

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 hoga

- Ab test karne chalo



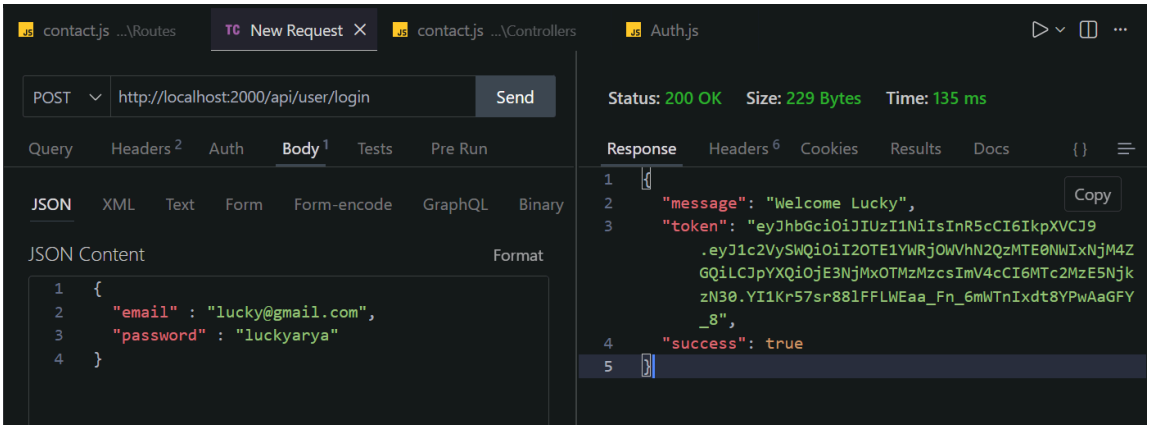
Sbse pehle header mai jake maine Auth naam se middleware banaya tha, wo exact naam diya and LalitTest ki jgh kuch bhi de sakte hai.
And mai ek new contact create kr raha hoon to create nahi hora, cuz authenticate nahi hai, and console pe LalitTest print hora hai token.



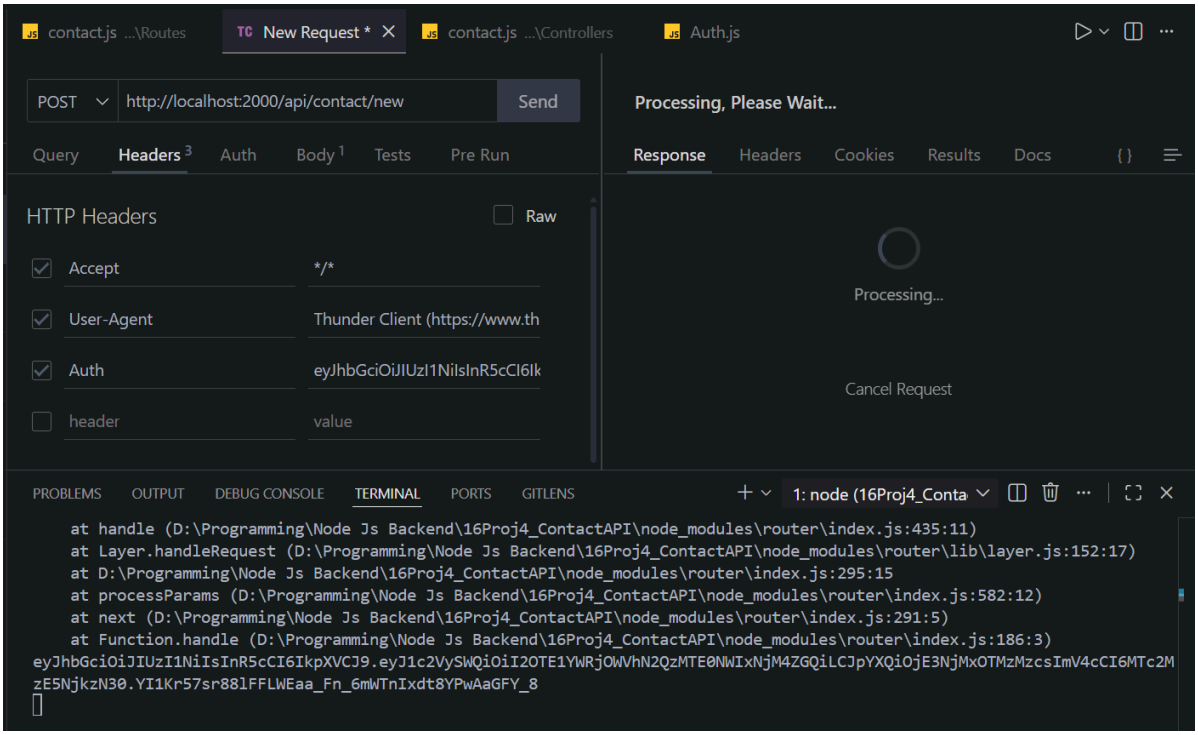
RHS mai sirf processing aara and console pe print hora.

- 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



Maine login kiya or mera token generate ho gaya.
Is token ko ab aap copy karlo and Auth jaha banaya tha waha paste kardo



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, verfiy 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 tareeke se verification ho jayegga

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

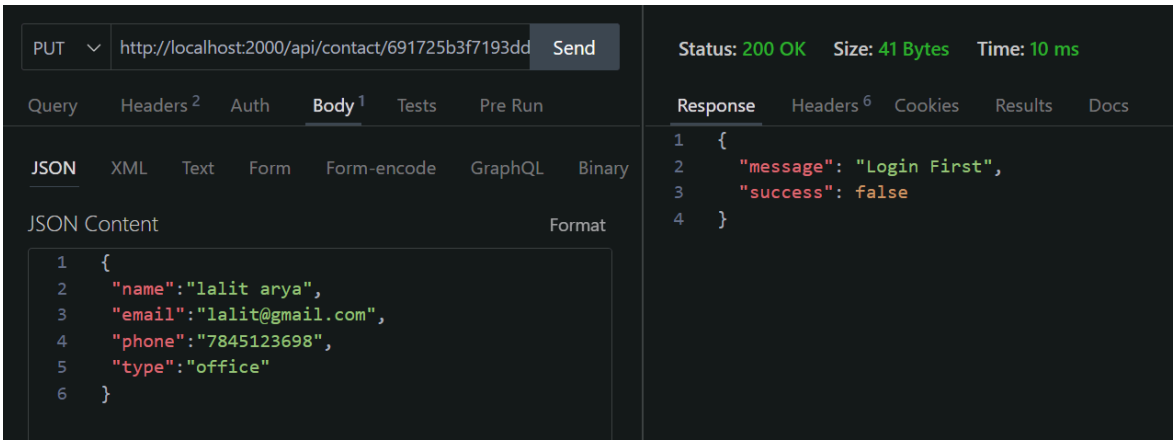
```
// agr saare fields de diye to save kara do db mai
let saveContact = await Contact.create({name,email,phone,type , user:req.userGlobal});
```

- 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,



Bolra hai login first

- **Ab banega user specific contact**

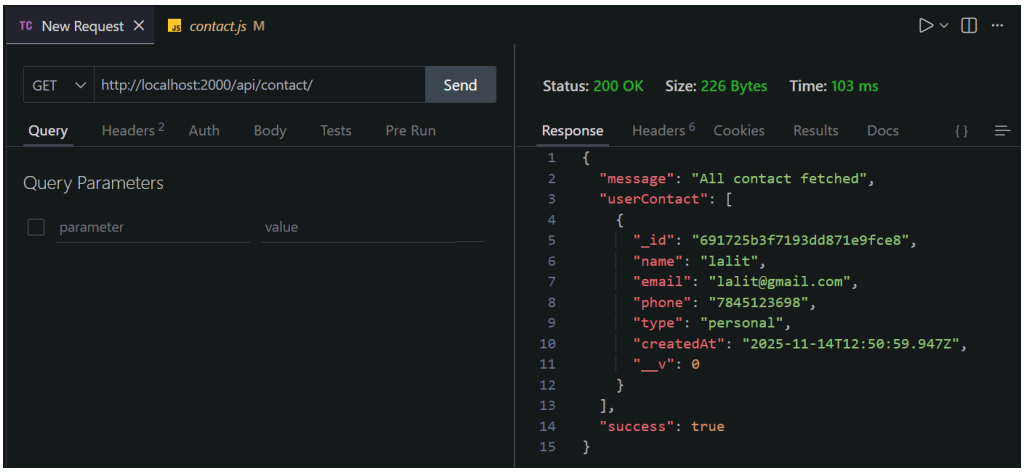
```
// get contact by user id
export const getUserSpecificData = async (req , res) => {
    const id = req.params.id
    const userId = await Contact.find({user:id});
    if(!userId){
        return res.json({
            message : "No contact exists with this id",
            success : false
      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

    // 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

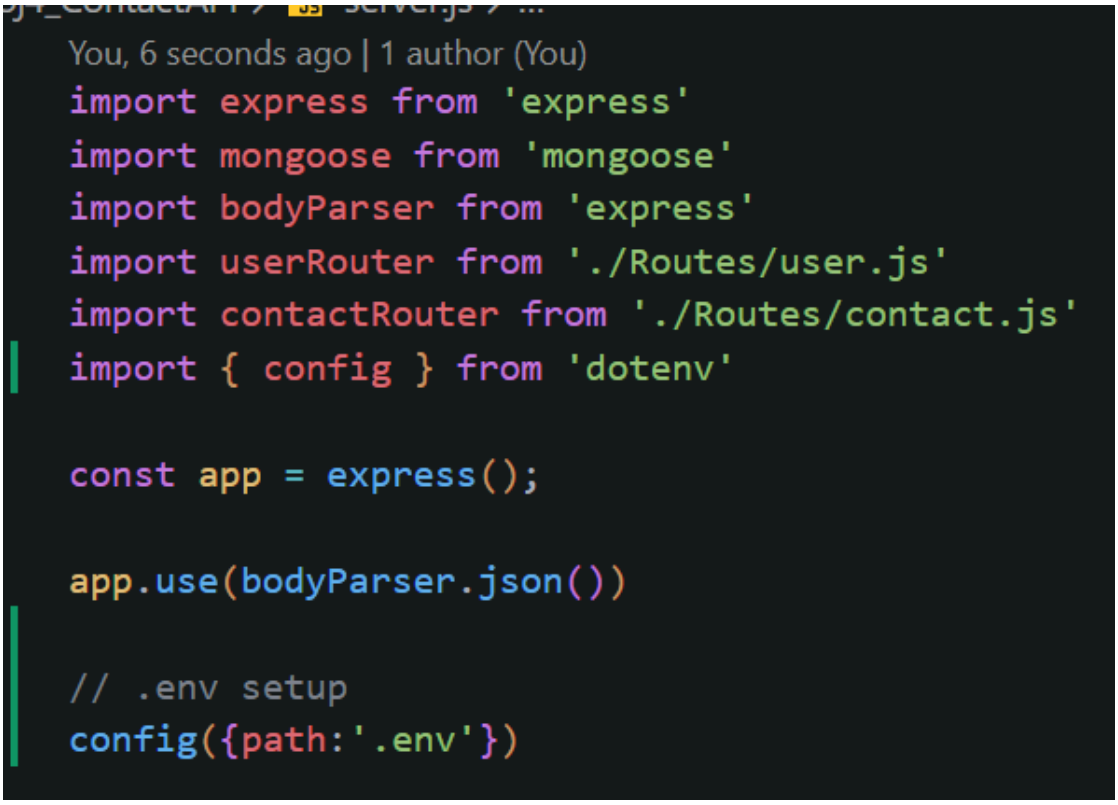


- Sbse pehle mere pass ek bande ka contact pada hai.

- **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
  - 

  - Ab jaha se remove kiya h waha pe kya rakhna hai?
  - server.js mai : 

    config({path:'.env'})

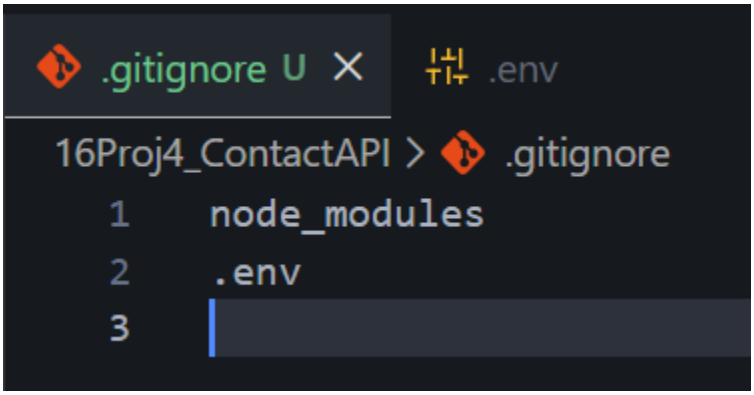
- Ab neeche

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

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

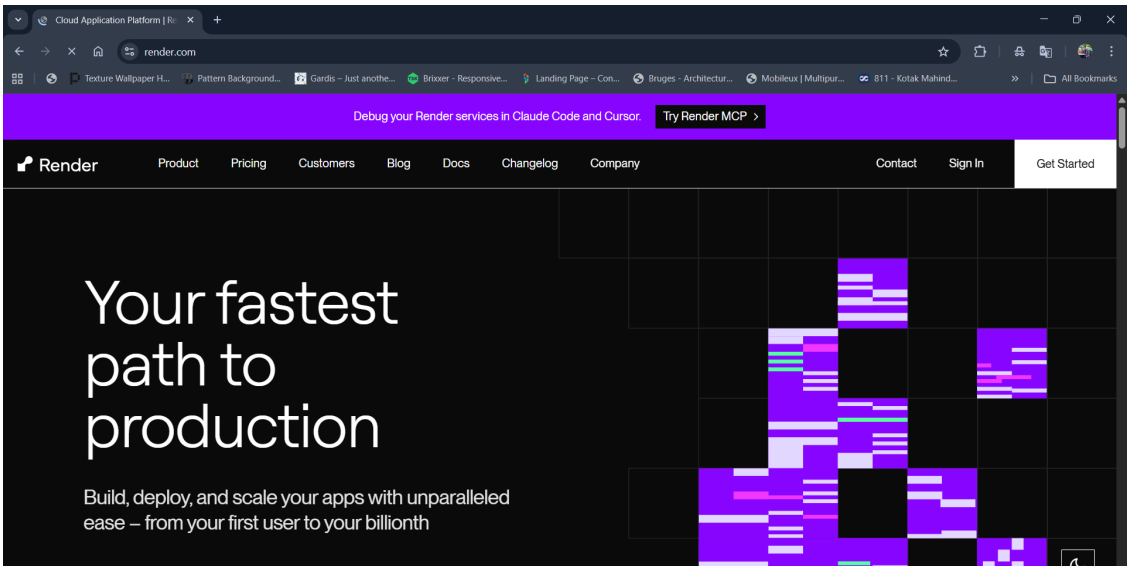
PORT or MONGOOSE pe diya process.env.____

- **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);
```

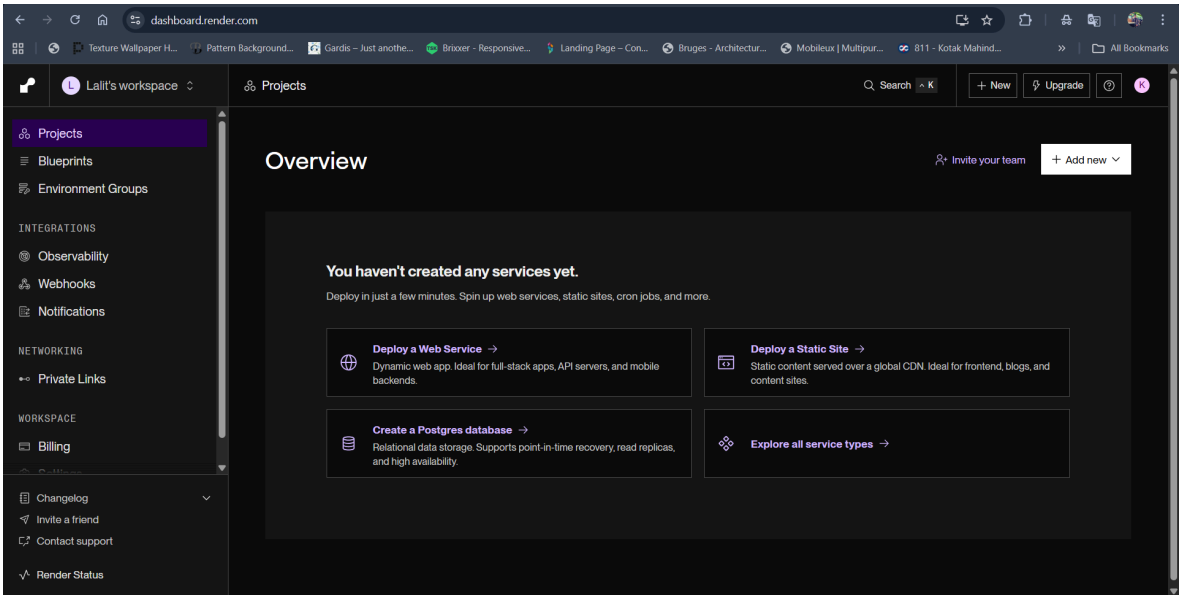
- .gitignore file banao, or jo jo cheeze github pe push nahi karni wo wo cheeze likh do isme.
  - 

- Saare files ko github pe daal do pehle
- rendor.com pe jao, signup karo

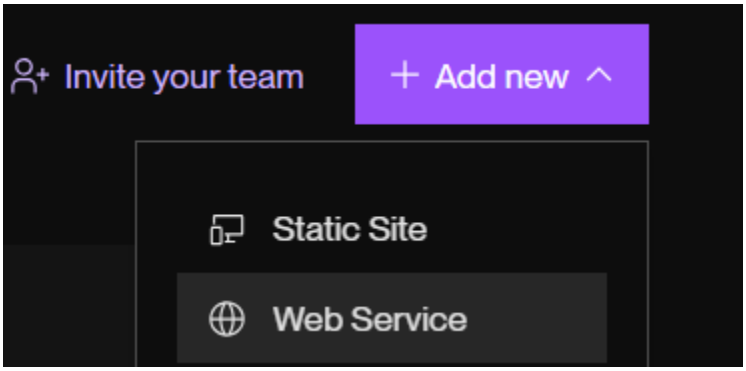


-

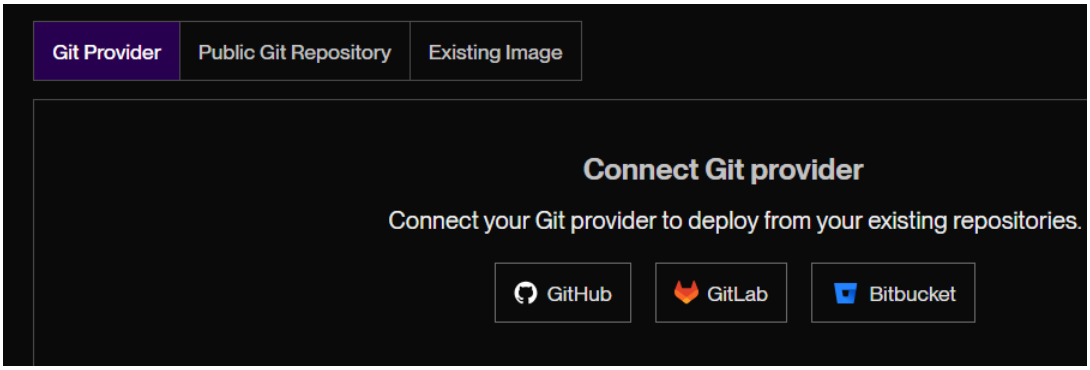
- Dashboard pe aa jao



Top pe add new

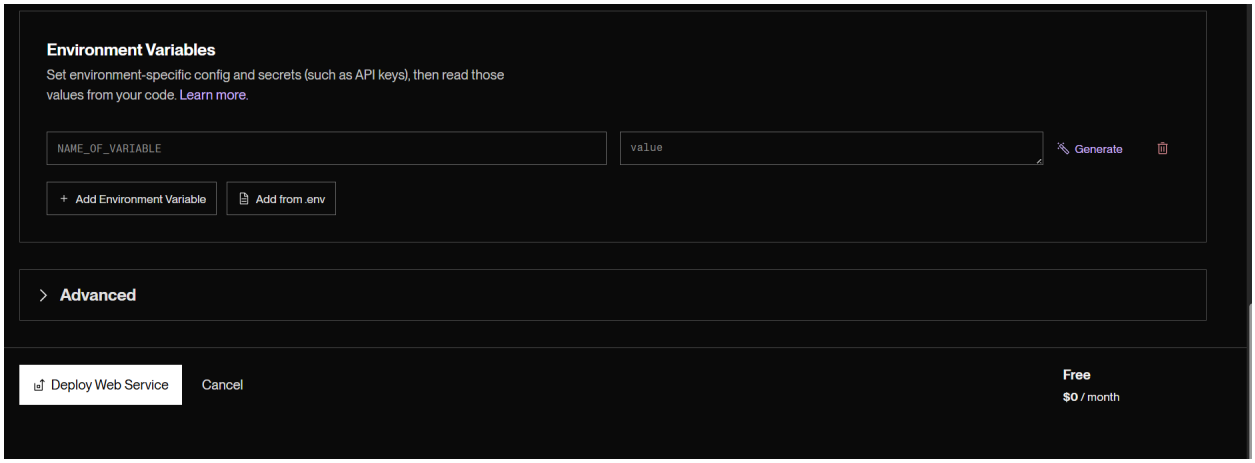


Webservce

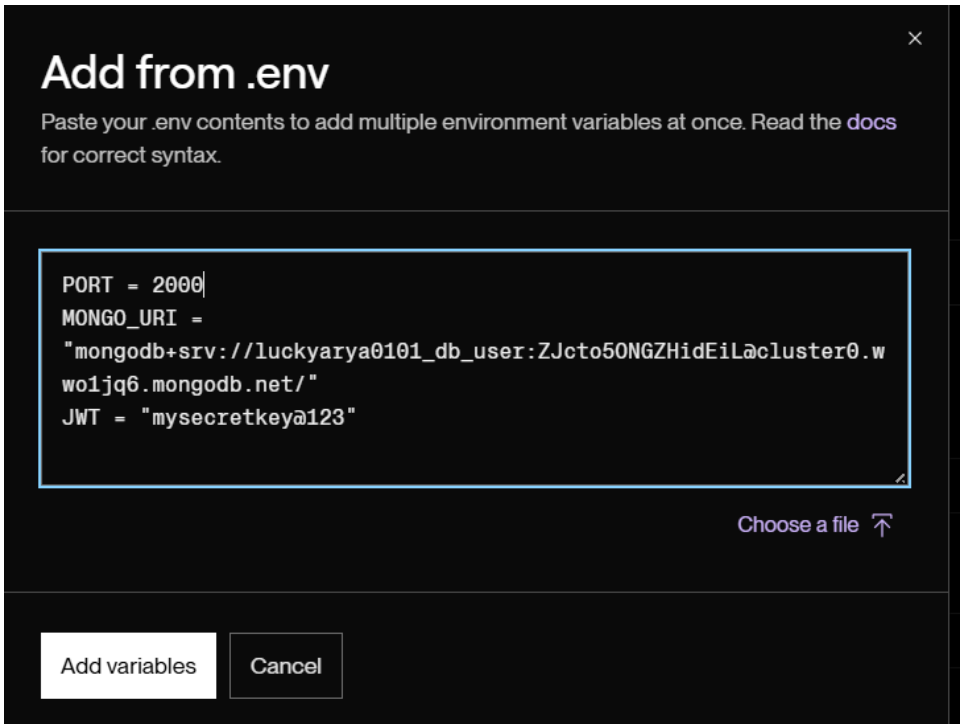


Link your github here

-

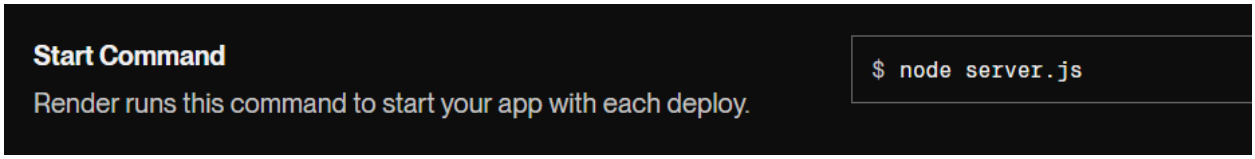


Poora page as it is rehndo, yaha env variables daalo, Add from .env pe click karo



Poori env file copy paste krdi
Add variables pe click kardo

- Yaha starting command de dena



Then deploy webservices

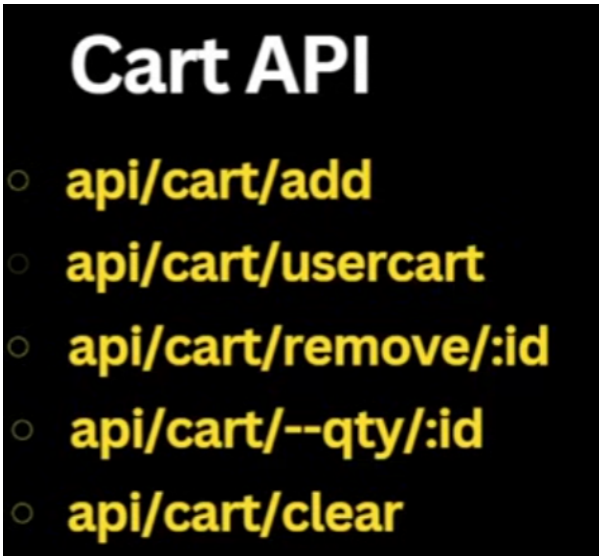
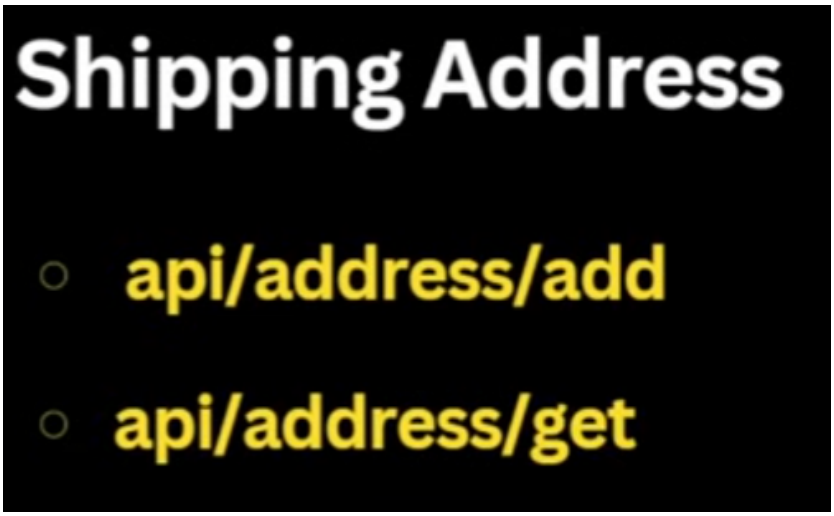
- Itna krne ke baad link mil jayega. Hamara deploye nahi ho skta cuz hamne to poora proj hi github pe daala hai, usme kaafi saari server.js file hai. Specific proj ko daaloge to chal jayega.

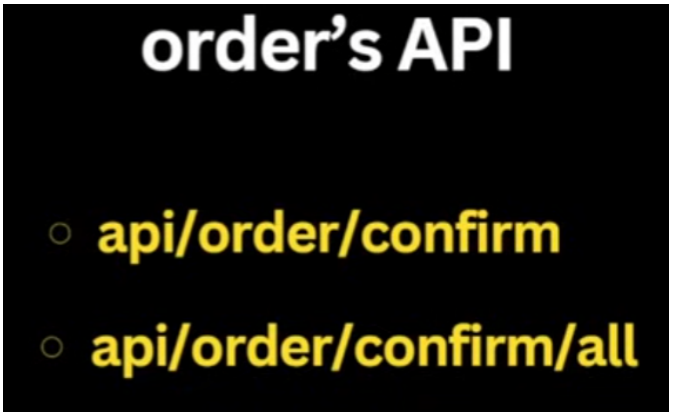
Ecommerce project

- Flow

1. 

2. 

3. 
Cart API
api/cart/add
api/cart/usercart
api/cart/remove/:id
api/cart/--qty/:id
api/cart/clear
4. 
Shipping Address
api/address/add
api/address/get
5. 
Payment API
api/payment/pay
api/payment/verify
Razorpay

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 .env U

17Proj5_EcommerceApp > server.js > ...

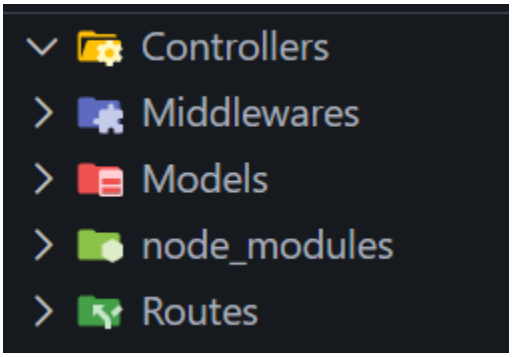
```
1  import express from 'express'
2  import mongoose from 'mongoose'
3  import bodyParser from 'express'
4  import { config } from 'dotenv'
5
6  const app = express();
7
8  app.use(bodyParser.json())
9
10 // .env setup
11 config({path:'.env'})
12
13 // home route
14 app.get('/' , (req,res) => {
15     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}`))
```

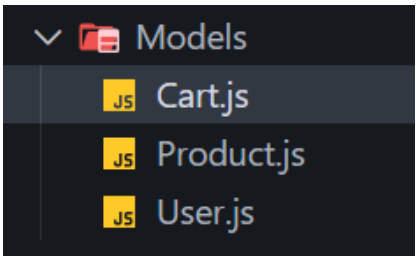
Is tarah hamari file dikh rahi hai and .env ka poora code bhi maine .env waali file mai paste kr diya as it is

server.js U .env U

17Proj5_EcommerceApp > .env

```
PORT = 2000

MONGO_URI = "mongodb+srv://luckyarya0101_db_user:ZJcto5ONGZHidEiL@cluster0.wwo1jq6.mongodb.net/"

JWT = "mysecretkey@123"
```

- **User MVC**
  - 

    Itne folder bana lo
  - 

    ye 3 model hamare hone waale hai
  - **User schema :**

User.js

Models > User.js > User

```
import mongoose from "mongoose";

const userEcomSchema = new mongoose.Schema({
    name:{type : String , require : true},
    email:{type : String , require : true},
    password:{type : String , require : true},
    createdAt:{type : Date , default : Date.now},
})

export const User = mongoose.model("userEcom" , userEcomSchema)
```

- **User Controller**

```
user.js
Controllers > user.js > 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

```
user.js Controllers    user.js Routes
Routes > user.js > 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;
```

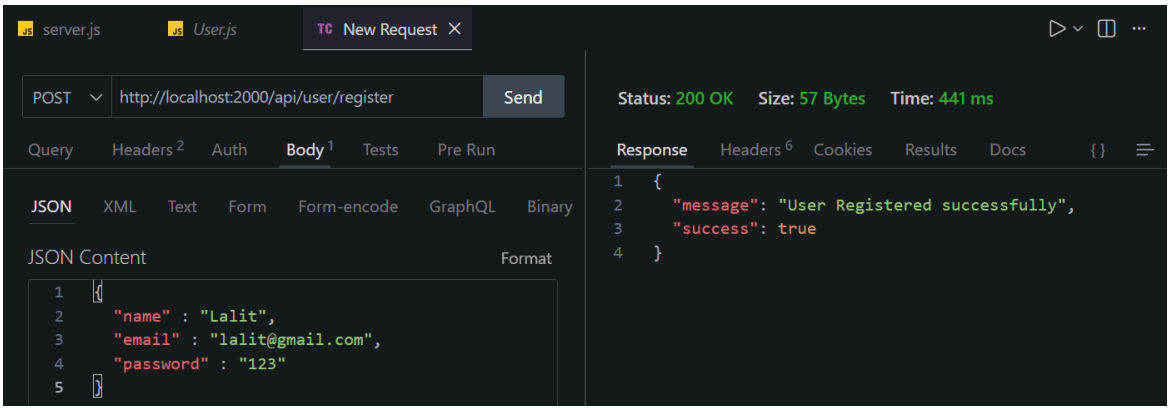
- Ab isko server.js se link krna hoga

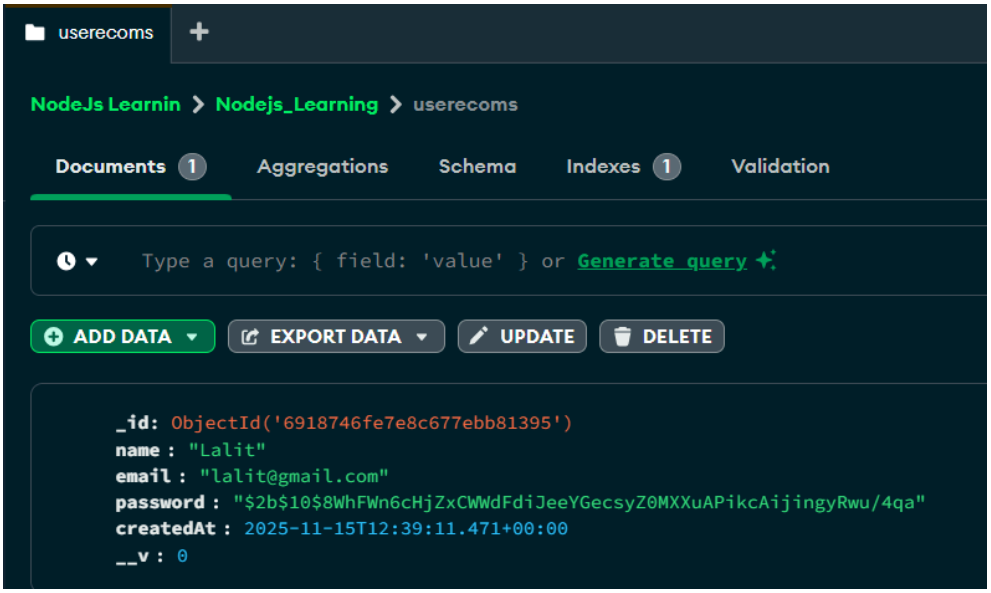
Ye import kiya :

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

Or ye route :

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

- Testing :
  - 

  - 

Password hash karke store hua hai.

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

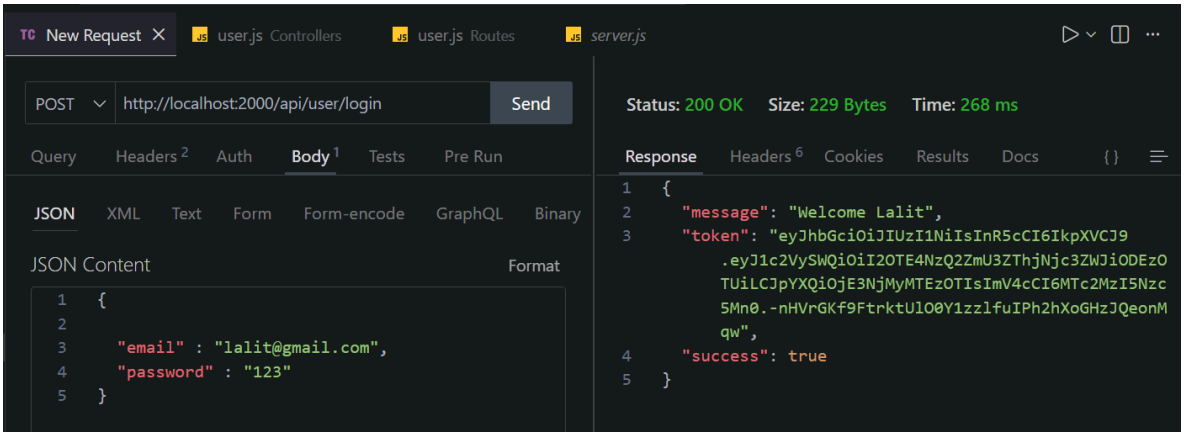
```
// user login
export const login = async(req,res) =>{
    const {email,password} = req.body;

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

    // check password
    const validPass = await bcrypt.compare(password, user.password) // usertextbox pe jo password likhra usko
    compare kar re DB mai jo password save hai

    if (!validPass) {
        return res.json({
            message : "Invalid Password",
            success : false
        })
    }
```

```
    // ab email and pass dono sahi ho to token banayenge.
    const token = jwt.sign({userId:user._id} , process.env.JWT , {expiresIn:'1d'})

    // token banne ke baad user ko welcome bolo
    res.json({
        message : `Welcome ${user.name}`,
        token : token,
        success : true
    })
}
```

- Ab router mai jake route banao.
-

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

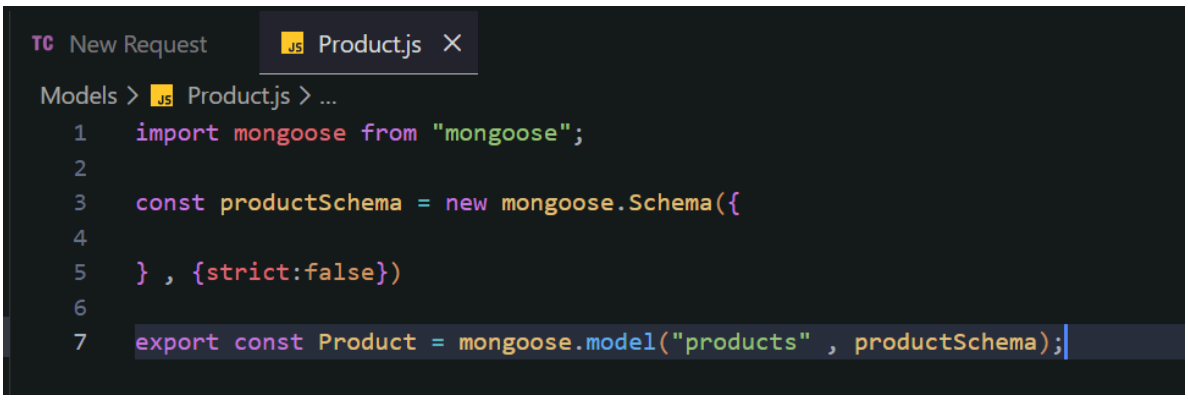
- Ab test karlo

- 

- 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 feautres hone like camara, 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 hoga.
  - Pr kuch cheeze sabhi products mai common hoti hai like : title, description, price and category.



{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
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     }
13     catch (error) {
14         res.json(error.message)
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   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

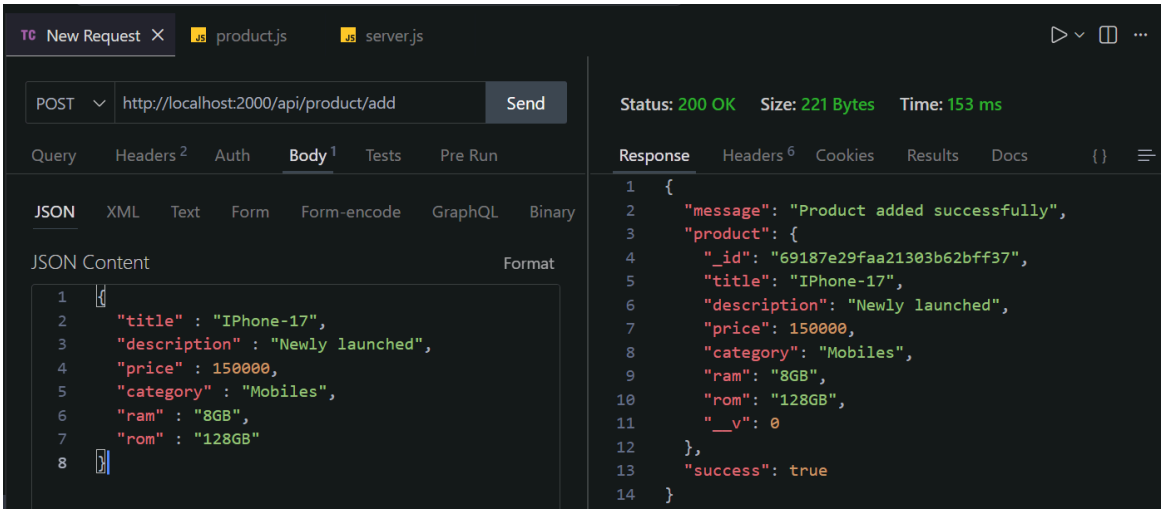
- **Server.js**
  -

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

  -

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

- **Testing**



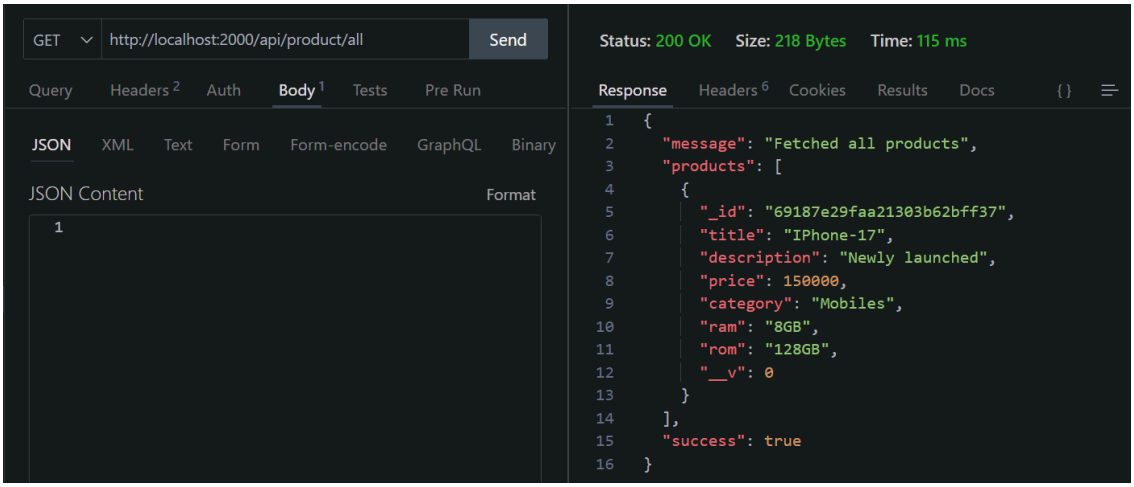
- **Get product :**
  - **Controller :**

```
// 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)
    }
}
```

  - **Routes :**

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

- **Test :**



- **Get Product by Id**

  - **Controller :**

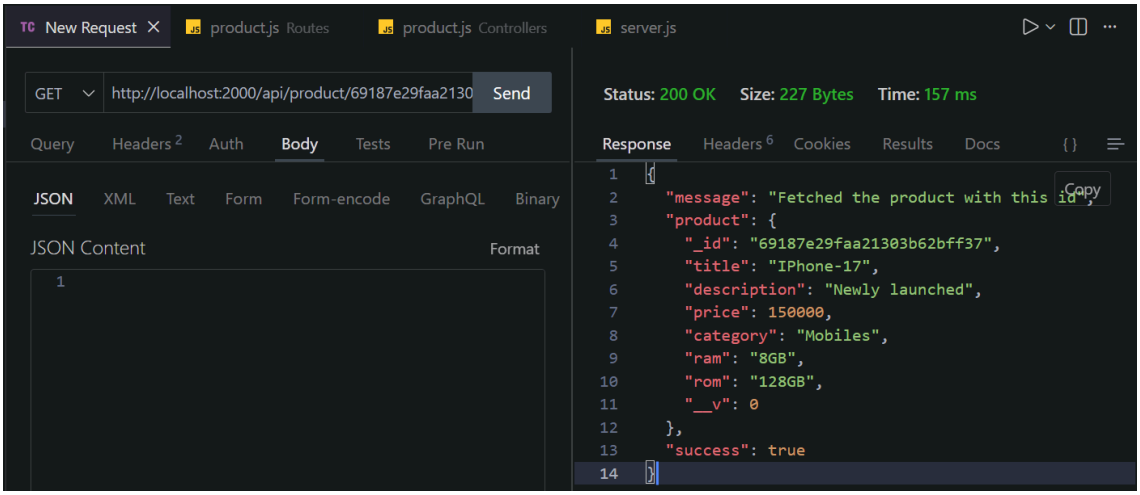
```
// get product by id
export const getProductById = async (req,res) => {

    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);
    }
}
```

- **Route :**

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

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

- **Testing :**
  -



- **Update product**

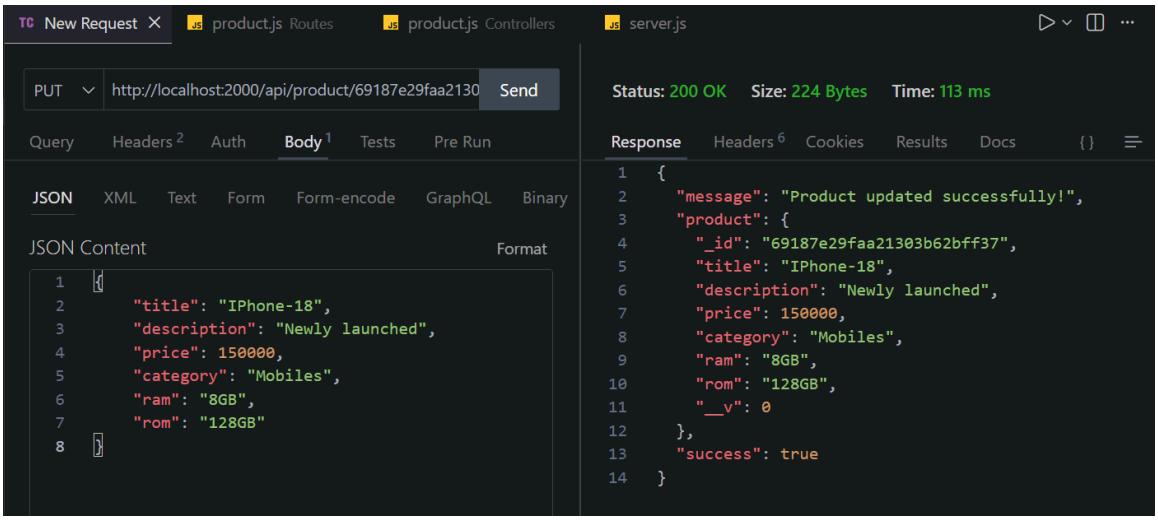
  - **Controller : findByIdAndUpdate(prodId , req.body , {new:true}) —>** ye fun ka use krenge. Ye id lera hai jiske basis pe update hoga, req.body mai jo bhi data mile, sabko update kardo, and new:true means koi new field add karna chahe to wo bhi kar sakte.
    -

```
// 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);
    }
}
```

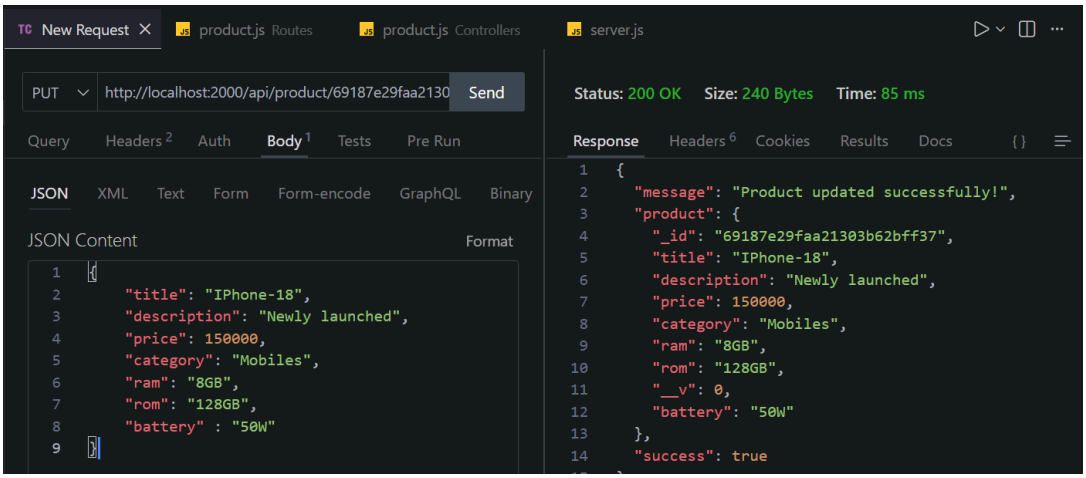
- **Router :**

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

- **Testing :**



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



- **Delete Product**

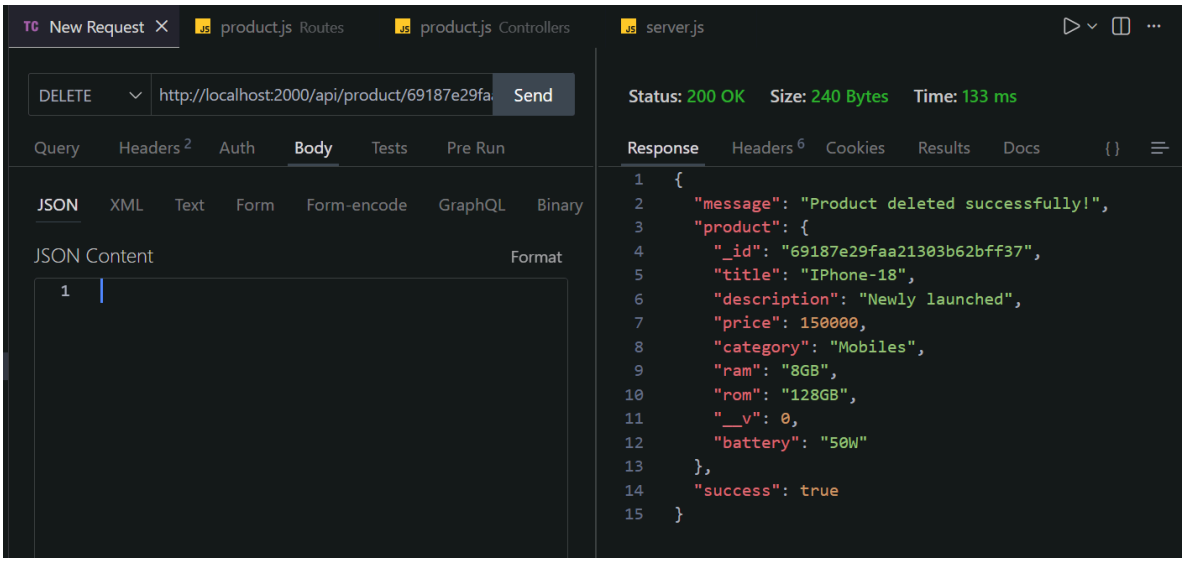
- **Controller :**

```
// 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);
    }
}
```

- **Route :**

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

- **Testing :**



- **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.

```
Auth.js
Middlewares > 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 find",
23      });
24    }
25    req.user = user;
26    next();
27  };
28
```

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

- Ab banao cart

- # Cart MVC

  - **Schema :**

    - Models > 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 > ...

```
import mongoose from "mongoose";

const cartItemSchema = new mongoose.Schema({

    productId : {
        type : mongoose.Schema.Types.ObjectId,
        ref : "Product",
        require : true
    },
    title : {type : String, require : true},
    price : {type : Number, require : true},
    qty : {type : Number, require : true},
})

const cartSchema = new mongoose.Schema({
    userId : {
        type : mongoose.Schema.Types.ObjectId,
        ref : "User",
        require : true
    },
    items : [cartItemSchema]
})

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

Controllers > cart.js > addToCart

```
import {Cart} from "../Models/Cart.js"

// add to cart
export const addToCart = async (req,res) => {
    const {productId, title, price, qty} = req.body;

    const userId = req.user; // ye globall se aari, auth file se

    // now we'll check ki is user id se pehle koi cart available h ya nahi, agr available hai to ham naya cart nahi banayenge, usi mai item ko add kr denge.
    let cart = await Cart.findOne({userId})
    if (!cart) {
        // to naya cart banayenge
        cart = new Cart({userId , items:[]}); // items ko array ki form mai dere cuz schema mai hame array ki form mai liya hai
    }
    else{
        // ab us product ka index find karna hoga ki wo product already h ya nahi - suppose iphone ko maine add to cart kiya, then dobara use add to cart kiya to uski qty bad jati hai na ki wo dobara add hota hai.
        const itemIndex = cart.items.array.findIndex((item) => item.productId.toString() == productId);
    }
```

-

```
    // agr already item hai , to jo cart mai already items hai uski qty ko increase kardo
    if (itemIndex >- 1) {
        cart.items[itemIndex].qty += qty
        cart.items[itemIndex].price += price * qty
    }
    else{
        cart.items.push({productId, title, price, qty})
    }

    await cart.save();
    res.json({
        message : "Item added to cart",
        cart : cart,
        success : true
    })
}
```

- **Route**

cart.js Controllers | cart.js Routes | Auth.js

Routes > cart.js > default

```
import express from 'express'
import { addToCart } from '../Controllers/cart.js';
import {isAuthenticated} from '../Middlewares/Auth.js'

const router = express.Router();

// add to cart
router.post('/add' ,isAuthenticated ,addToCart)

export default router;
```

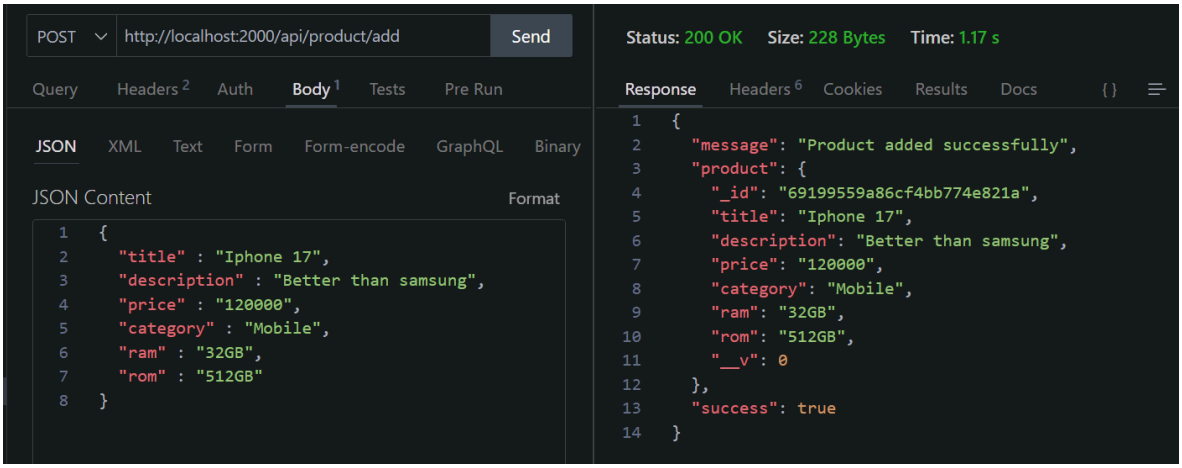
  - Ab isko server se link karna hoga

- **Server**

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

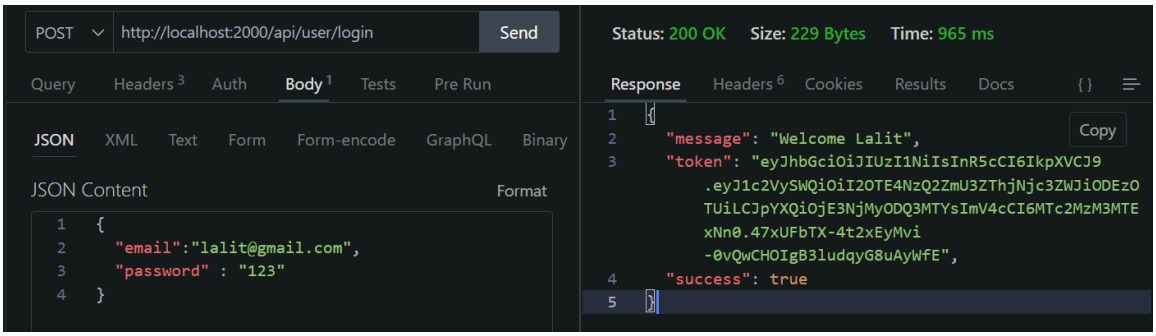
```
// cart route - middleware
app.use('/api/cart' , cartRouter)
```

- **Testing :**
  - Sbse pehle maine kuch product add kar liye



- Ye dekh sakte ho aap :

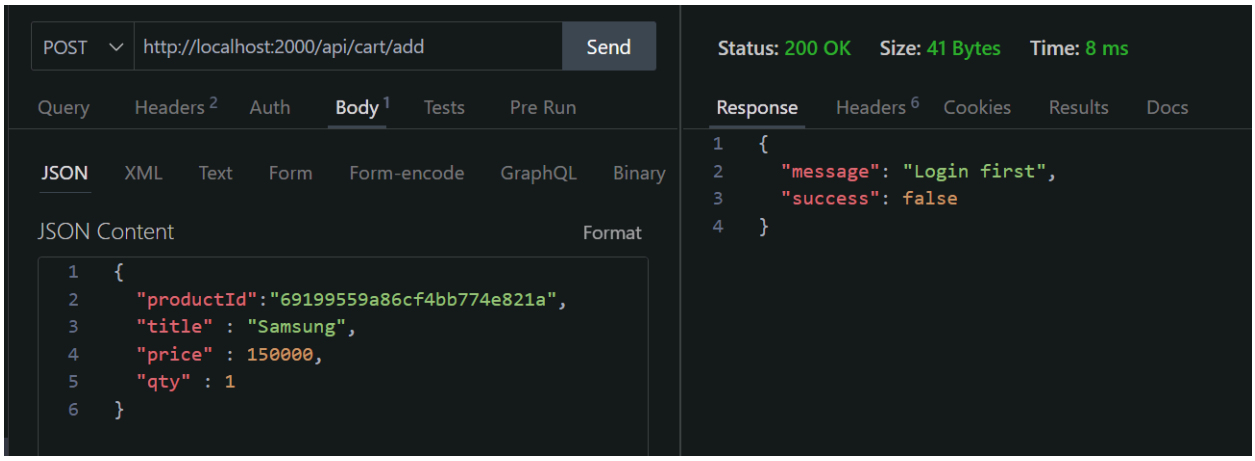
-
    - Yaha se product id uth lo uski need hai.
- Ab login token utha lo



-

- Ye token uthao or chalo cart mai

- Ab cart ko chalate hai :



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

Controllers > cart.js > 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 chhod 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)
```