

# Creating a RESTful API using express.js and creating a database and index in MongoDB.

NAME : KSHATRI AKHILESWARI BAI

EMAIL ID : [208X1a0537@khitguntur.ac.in](mailto:208X1a0537@khitguntur.ac.in)

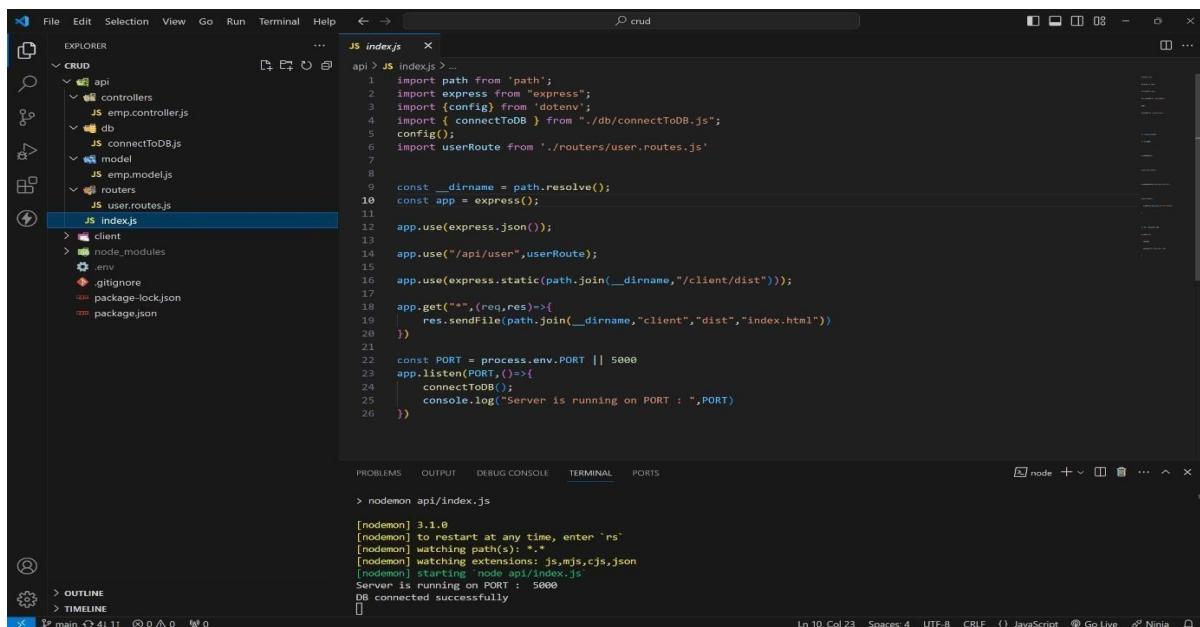
PHONE NO : 8712363393

ROLL NO : 208X1A0537

COLLEGE : KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY, GUNTUR

source code :

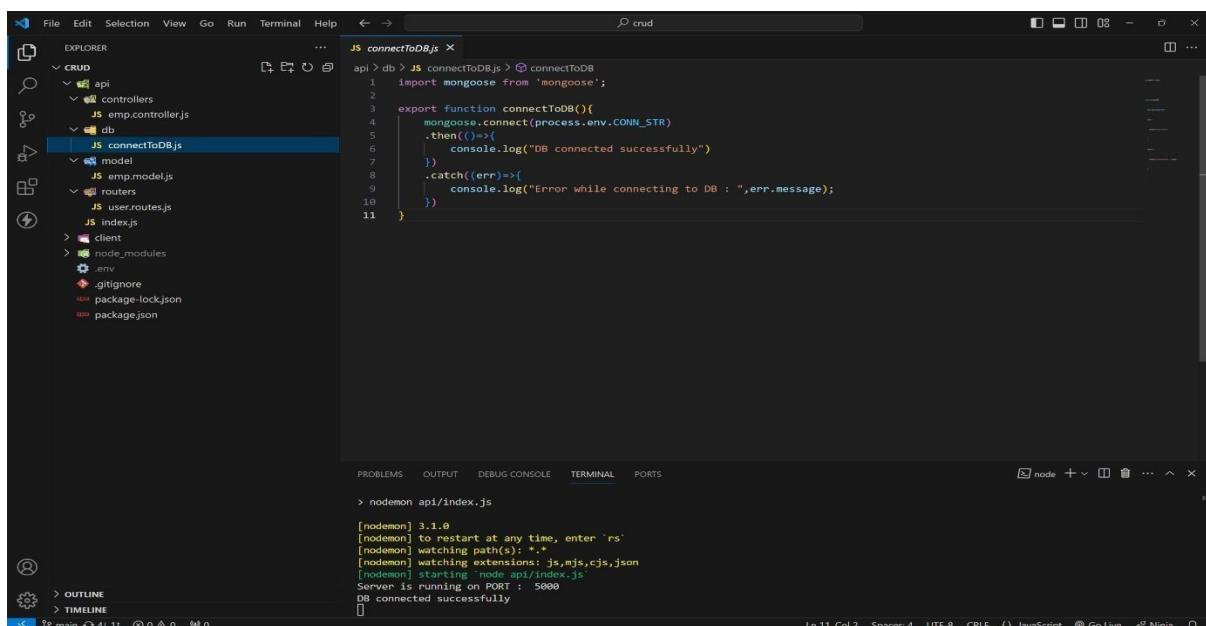
index.js file :



```
api > JS index.js > ...
1 import path from 'path';
2 import express from 'express';
3 import {config} from 'dotenv';
4 import { connectToDB } from './db/connectToDB.js';
5 config();
6 import userRoute from './routes/user.routes.js'
7
8
9 const __dirname = path.resolve();
10 const app = express();
11
12 app.use(express.json());
13
14 app.use("/api/user",userRoute);
15
16 app.use(express.static(path.join(__dirname,"/client/dist")));
17
18 app.get("*",(req,res)=>{
19   res.sendFile(path.join(__dirname,"client","dist","index.html"))
20 })
21
22 const PORT = process.env.PORT || 5000
23 app.listen(PORT,()=>{
24   connectToDB();
25   console.log("Server is running on PORT : ",PORT)
26 })
```

```
> node api/index.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node api/index.js`
Server is running on PORT : 5000
DB connected successfully
```

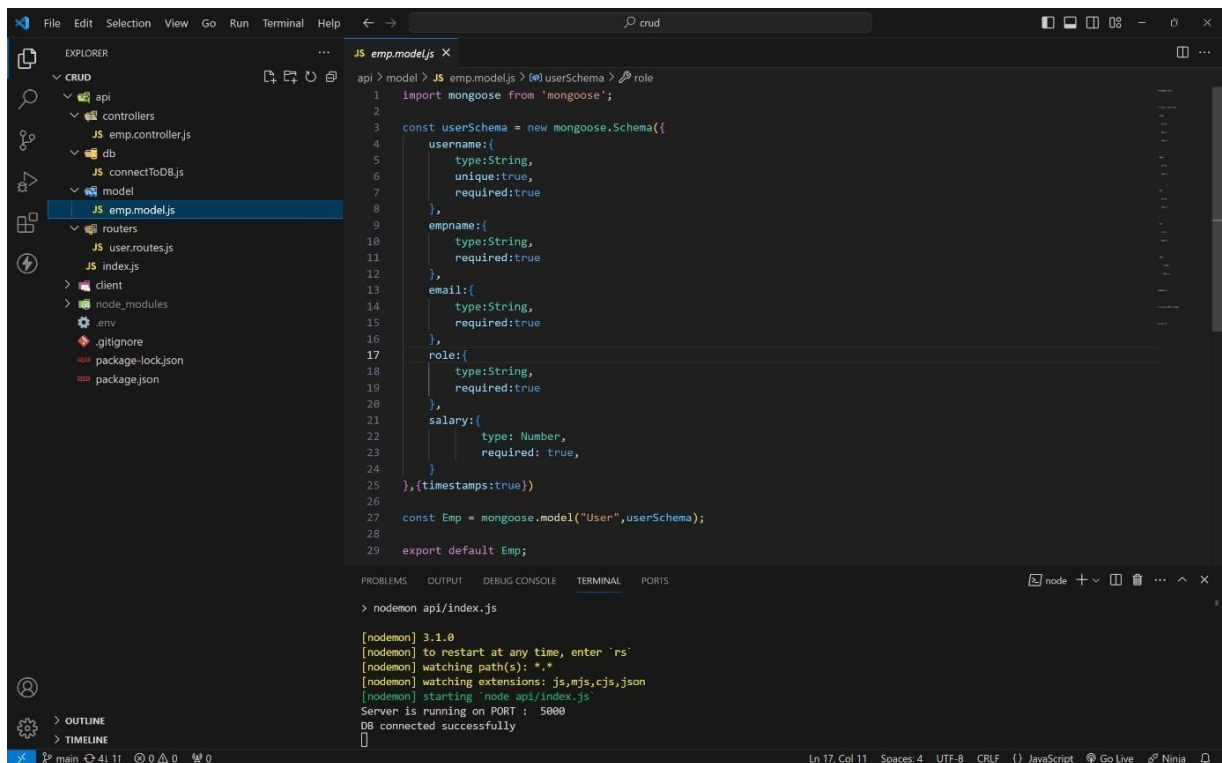
MONGODB CONNECTION :



```
api > db > JS connectToDB.js > @ connectToDB
1 import mongoose from 'mongoose';
2
3 export function connectToDB(){
4   mongoose.connect(process.env.CONN_STR)
5   .then(()=>{
6     console.log("DB connected successfully")
7   })
8   .catch((err)=>{
9     console.log("Error while connecting to DB : ",err.message);
10   })
11 }
```

```
> node api/index.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node api/index.js`
Server is running on PORT : 5000
DB connected successfully
```

## MODEL :



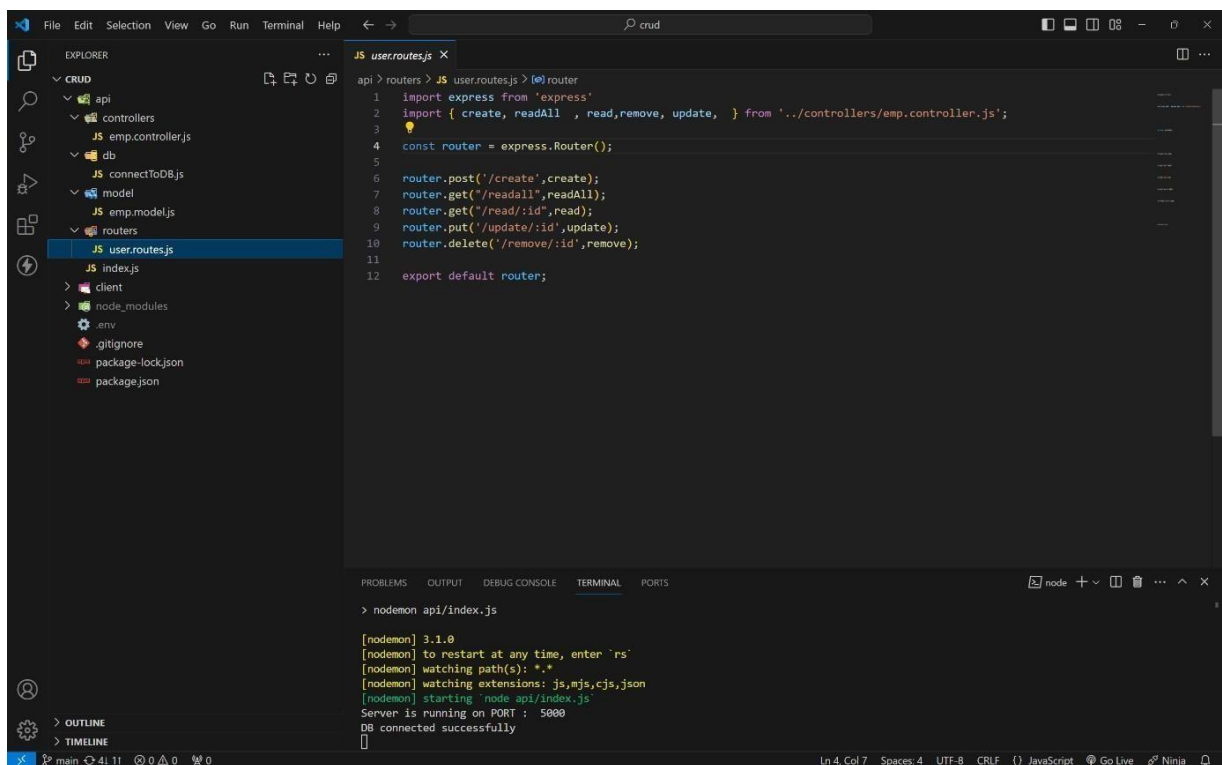
The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like api, controllers, db, model, routers, and client. The file model/emp.model.js is selected. The editor displays the following code:

```
1 import mongoose from 'mongoose';
2
3 const userSchema = new mongoose.Schema({
4   username: {
5     type: String,
6     unique: true,
7     required: true
8   },
9   empname: {
10    type: String,
11    required: true
12  },
13  email: {
14    type: String,
15    required: true
16  },
17  role: {
18    type: String,
19    required: true
20  },
21  salary: {
22    type: Number,
23    required: true,
24  }
25 }, { timestamps: true })
26
27 const Emp = mongoose.model("User", userSchema);
28
29 export default Emp;
```

The terminal at the bottom shows the output of running the application:

```
> nodemon api/index.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node api/index.js`
Server is running on PORT : 5000
DB connected successfully
```

## ROUTES:



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with folders like api, controllers, db, model, routers, and client. The file routers/user.routes.js is selected. The editor displays the following code:

```
1 import express from 'express'
2 import { create, readAll, read, remove, update, } from '../controllers/emp.controller.js';
3
4 const router = express.Router();
5
6 router.post('/create', create);
7 router.get('/readall', readAll);
8 router.get('/read/:id', read);
9 router.put('/update/:id', update);
10 router.delete('/remove/:id', remove);
11
12 export default router;
```

The terminal at the bottom shows the output of running the application:

```
> nodemon api/index.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node api/index.js`
Server is running on PORT : 5000
DB connected successfully
```

## CONTROLLERS :

### CREATE :

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `emp.controller.js` file. The code implements the `create` function, which takes a request and response object. It attempts to find an employee by username. If found, it returns a 400 status with an error message. If not found, it creates a new employee object with fields `username`, `empname`, `email`, `role`, and `salary`, saves it, and returns a 201 status with the employee details. Error handling is included for both the database operations and the function itself.

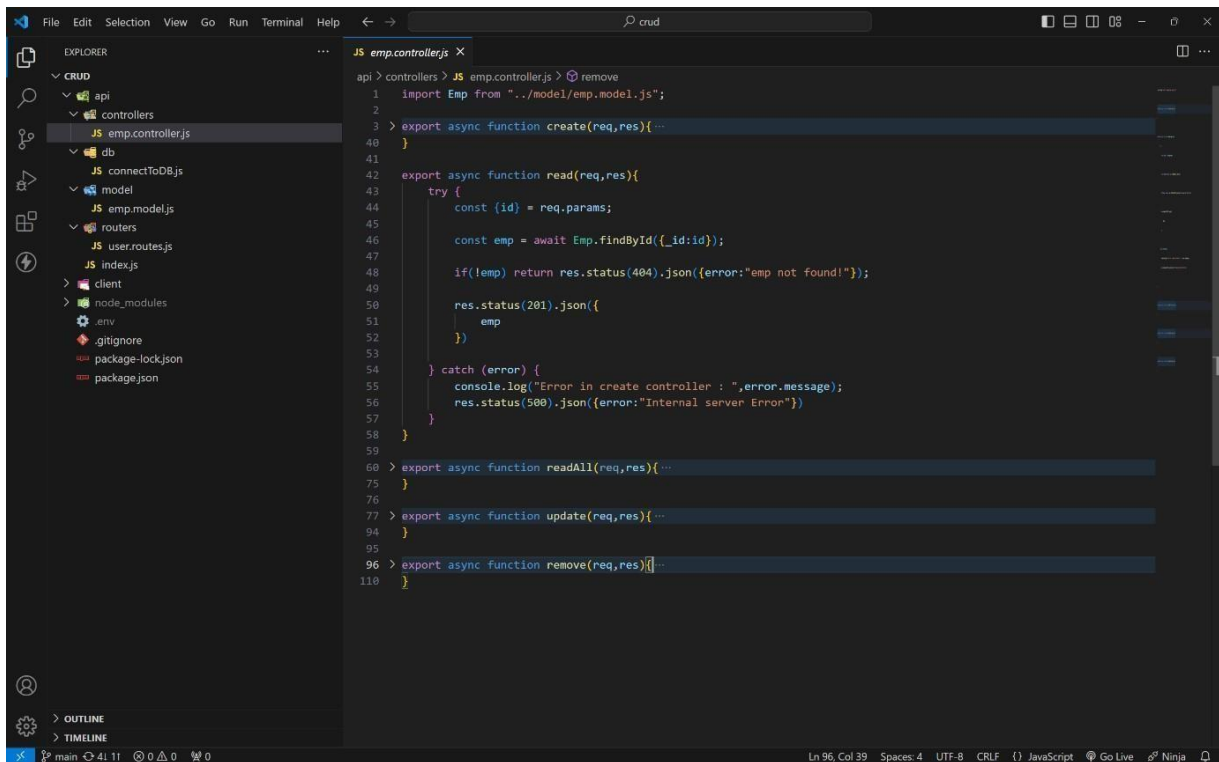
```
api > controllers > JS emp.controller.js > remove
1 import Emp from "../model/emp.model.js";
2
3 export async function create(req,res){
4   try {
5     const {username,empname,email,role,salary} = req.body;
6
7     console.log(req.body);
8     const emp = await Emp.findOne({username});
9
10    if(emp) return res.status(400).json({error:"username is already exists"});
11
12    const newEmp = new Emp({
13      username,
14      empname,
15      email,
16      role,
17      salary
18    });
19
20    if(newEmp){
21
22      await newEmp.save();
23
24      res.status(201).json({
25        _id : newEmp._id,
26        username : newEmp.username,
27        empname : newEmp.empname,
28        email : newEmp.email,
29        role : newEmp.role,
30        salary : newEmp.salary
31      })
32    }else{
33      res.status(400).json({error:"Invalid emp data"});
34    }
35
36  } catch (error) {
37    console.log("Error in create controller : ",error.message);
38    res.status(500).json({message : error.message})
39  }
40 }
41
```

**READALL:**

The screenshot shows the VS Code editor with the file explorer on the left. The main editor window shows the `emp.controller.js` file. The code implements the `readAll` function, which takes a request and response object. It attempts to find all employees. If no employees are found, it returns a 404 status with an error message. If employees are found, it returns a 201 status with the employee details. Error handling is included for both the database operations and the function itself.

```
api > controllers > JS emp.controller.js > remove
1 import Emp from "../model/emp.model.js";
2
3 > export async function create(req,res){ ...
40 }
41
42 > export async function read(req,res){ ...
58 }
59
60 export async function readAll(req,res){
61   try {
62
63     const emps = await Emp.find();
64
65     if(!emps || !emps.length ) return res.status(404).json({error:" no emp data found!"});
66
67     res.status(201).json({
68       emps
69     })
70
71   } catch (error) {
72     console.log("Error in create controller : ",error.message);
73     res.status(500).json({error:"Internal server Error"})
74   }
75 }
76
77 > export async function update(req,res){ ...
94 }
95
96 > export async function remove(req,res){ ...
118 }
```

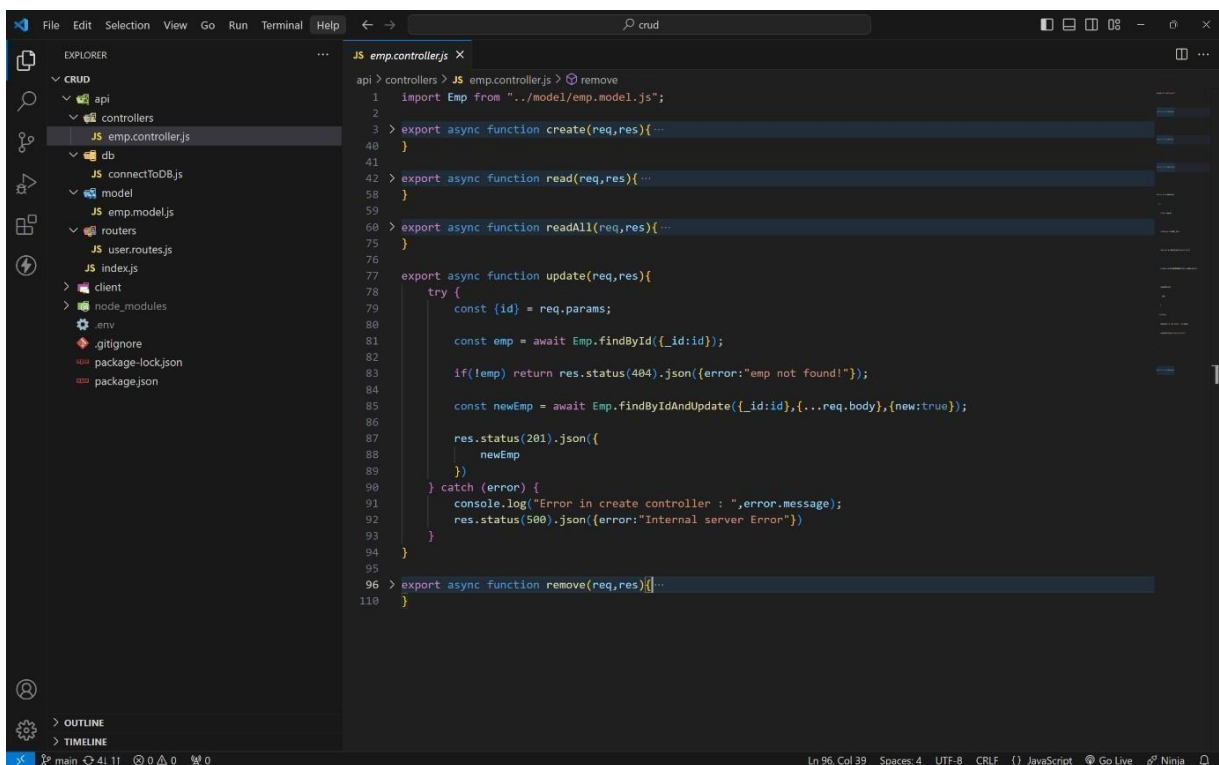
**READONE :**



The screenshot shows a VS Code editor with the file explorer on the left displaying a project structure for a CRUD application. The main editor window shows the `emp.controller.js` file. The code includes imports for `Emp` from `../model/emp.model.js` and defines three asynchronous functions: `create`, `read`, and `remove`. The `read` function uses `req.params` to find an employee by ID and returns a 404 status if not found. The `remove` function is partially visible at the bottom of the file.

```
api > controllers > JS emp.controller.js > remove
1  import Emp from "../model/emp.model.js";
2
3  > export async function create(req,res){ ...
40 }
41
42 export async function read(req,res){
43   try {
44     const {id} = req.params;
45
46     const emp = await Emp.findById({_id:id});
47
48     if(!emp) return res.status(404).json({error:"emp not found!"});
49
50     res.status(201).json({
51       emp
52     })
53   } catch (error) {
54     console.log("Error in create controller : ",error.message);
55     res.status(500).json({error:"Internal server Error"})
56   }
57 }
58
59
60 > export async function readAll(req,res){ ...
75 }
76
77 > export async function update(req,res){ ...
94 }
95
96 > export async function remove(req,res){ ...
110 }
```

## UPDATE :



This screenshot shows the same VS Code editor, but the `update` function in `emp.controller.js` has been modified. It now uses `Emp.findByIdAndUpdate` to update an employee's details, including a `new:true` option. The `remove` function remains at the bottom of the file.

```
api > controllers > JS emp.controller.js > remove
1  import Emp from "../model/emp.model.js";
2
3  > export async function create(req,res){ ...
40 }
41
42 > export async function read(req,res){ ...
58 }
59
60 > export async function readAll(req,res){ ...
75 }
76
77 export async function update(req,res){
78   try {
79     const {id} = req.params;
80
81     const emp = await Emp.findById({_id:id});
82
83     if(!emp) return res.status(404).json({error:"emp not found!"});
84
85     const newEmp = await Emp.findByIdAndUpdate({_id:id},{...req.body},{new:true});
86
87     res.status(201).json({
88       newEmp
89     })
90   } catch (error) {
91     console.log("Error in create controller : ",error.message);
92     res.status(500).json({error:"Internal server Error"})
93   }
94 }
95
96 > export async function remove(req,res){ ...
110 }
```

## DELETE :

This screenshot shows the VS Code editor with the file explorer on the left displaying a project structure for a CRUD application. The main editor window is open to `api > controllers > JS emp.controller.js`, showing the `remove` function implementation. The function uses `Emp.findByIdAndDelete` to remove a record and returns a 201 status with the deleted record's details. Error handling is implemented with `catch` blocks for logging and returning 500 status for internal server errors.

```
1 import Emp from "../model/emp.model.js";
2
3 export async function create(req,res){ ...
40 }
41
42 export async function read(req,res){ ...
58 }
59
60 export async function readAll(req,res){ ...
75 }
76
77 export async function update(req,res){ ...
94 }
95
96 export async function remove(req,res){
97   try {
98     const {id} = req.params;
99
100    await Emp.findByIdAndDelete({_id:id});
101
102    res.status(201).json({
103      id,
104      message : 'deleted successfully..',
105    })
106  } catch (error) {
107    console.log("Error in create controller : ",error.message);
108    res.status(500).json({error:"Internal server Error"})
109  }
110 }
```

This screenshot shows the VS Code editor with the `.env` file open, containing configuration for the application's port and MongoDB connection string. Below the editor, the terminal window displays the output of the application, showing a successful creation of a user record with a salary of 60000. The terminal also shows the available shells (powershell, powershell, node) and the current status of the application (main\*).

```
1 PORT = 5000
2 CONN_STR =mongodb://localhost:27017/<database>
```

```
username: 'jack',
empname: 'jack rider',
email: 'jack@gmail.com',
role: 'Front End Developer',
salary: 60000
}
```

OUTPUT :

create      update      read      delete

bakshu6
ALLAH BAKSH
bakshu@gmail.co
manager
400000
submit

## operation

get all data  
clear data

```
{
  "_id": "65f93d40b8c40b11b7011b3",
  "username": "stark",
  "empname": "tony",
  "email": "stark@3000",
  "role": "dev",
  "salary": 10000,
  "createdAt": "2024-03-19T07:14:08.002Z",
  "updatedAt": "2024-03-19T07:14:08.002Z",
  "v": 0
},
{
  "_id": "65fdb22ee2c8876fb3d0eb48",
  "username": "000",
  "empname": "divya",
  "email": "208x1a0598@khitguntur.ac.in",
  "role": "software",
  "salary": 500000,
  "createdAt": "2024-03-22T16:30:38.600Z",
  "updatedAt": "2024-03-22T16:31:38.756Z",
  "v": 0
},
{
  "_id": "65fe6eef181812bc6e66d5cb",
  "username": "bakshu6",
  "empname": "ALLAH BAKSHU",
  "email": "bakshu@gmail.com",
  "role": "manager",
  "salary": 400000,
  "createdAt": "2024-03-23T05:55:59.568Z",
  "updatedAt": "2024-03-23T05:55:59.568Z",
  "v": 0
}
```