

# ASSIGNMENT-3

NAME : BHAI RAPUNENI CHANDRA SEKHAR

EMAIL : 208X1A4259@khitguntur.ac.in

MOBILE NO : 8074916828

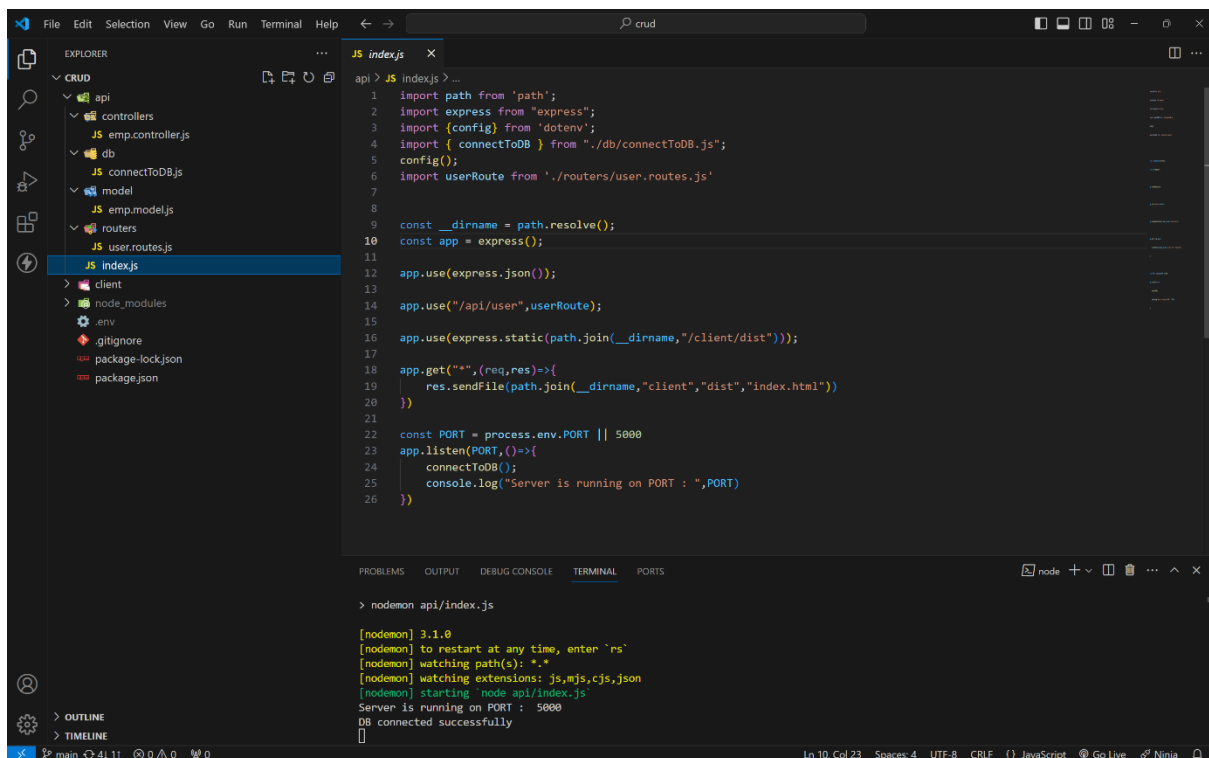
Roll NO : 208x1a4259

College Name : KALLAM

HARANADHAREDDY INSTITUTE OF  
TECHNOLOGY

source code :

index.js file :



The screenshot displays the Visual Studio Code editor with a project named 'crud'. The Explorer sidebar on the left shows the file structure, with 'api/index.js' selected. The main editor window shows the content of 'index.js', which is a Node.js application using Express.js. The code imports necessary modules, configures the app, and sets up routes for user management and static client files. The terminal at the bottom shows the command 'nodemon api/index.js' being executed, with output indicating that the server is running on port 5000 and the database is connected successfully.

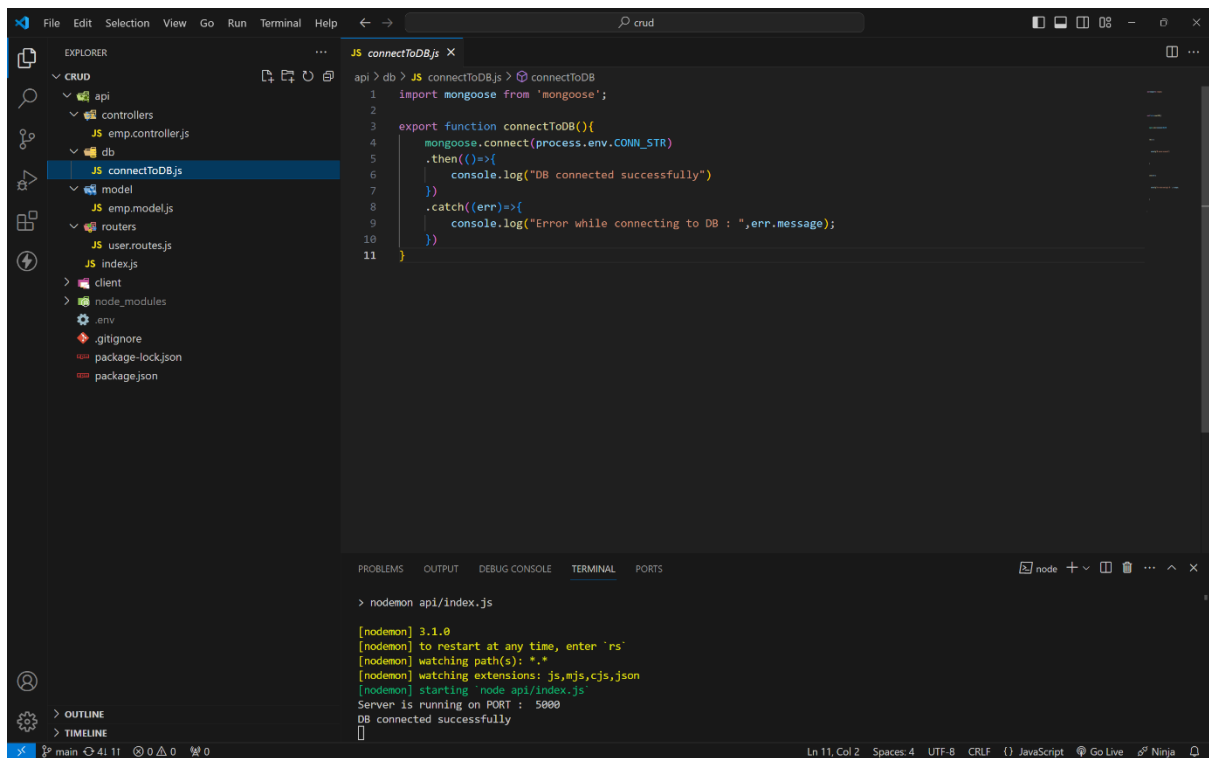
```
api > JS index.js > ...
1  import path from 'path';
2  import express from 'express';
3  import {config} from 'dotenv';
4  import { connectToDB } from './db/connectToDB.js';
5  config();
6  import userRoute from './routes/user.routes.js'
7
8
9  const __dirname = path.resolve();
10 const app = express();
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 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
> 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
```

# MONGODB CONNECTION :



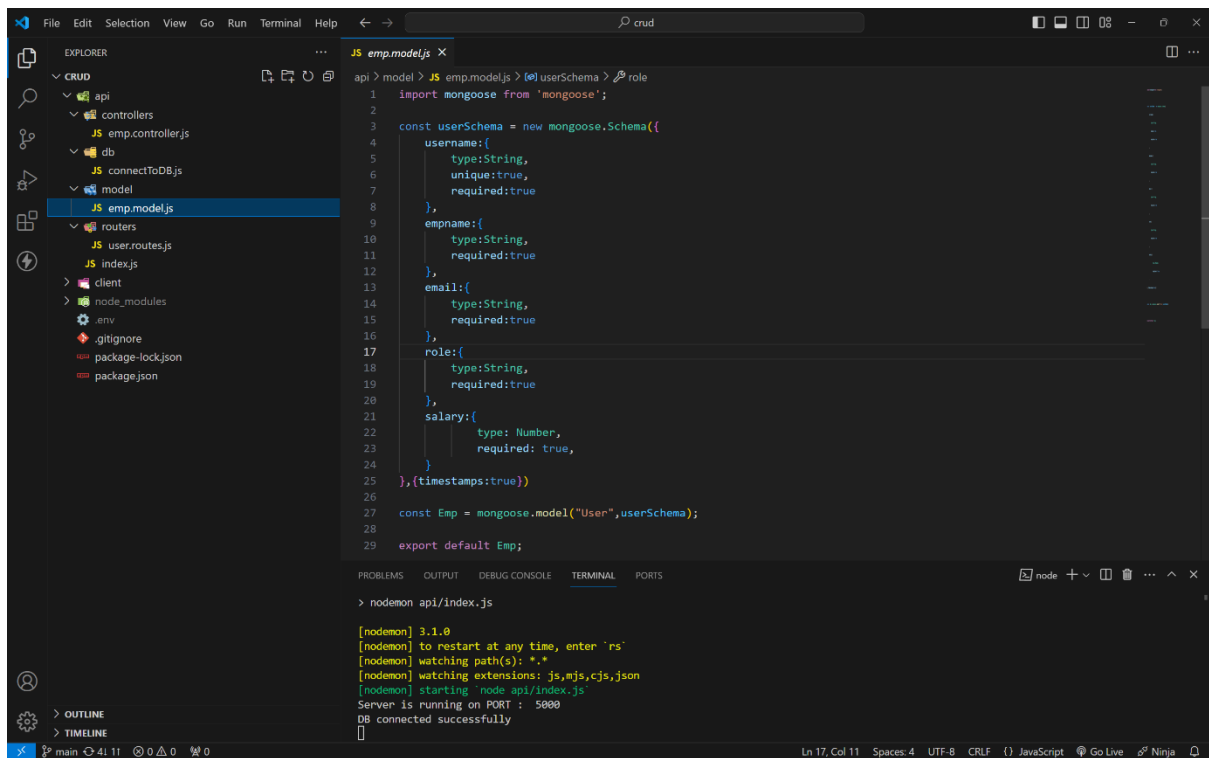
The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure with folders like 'api', 'controllers', 'db', 'model', 'routers', and 'index.js'. The file 'connectToDB.js' is selected in the 'db' folder. The main editor shows the code for 'connectToDB.js'.

```
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 }
```

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

```
> 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
```

# MODEL :



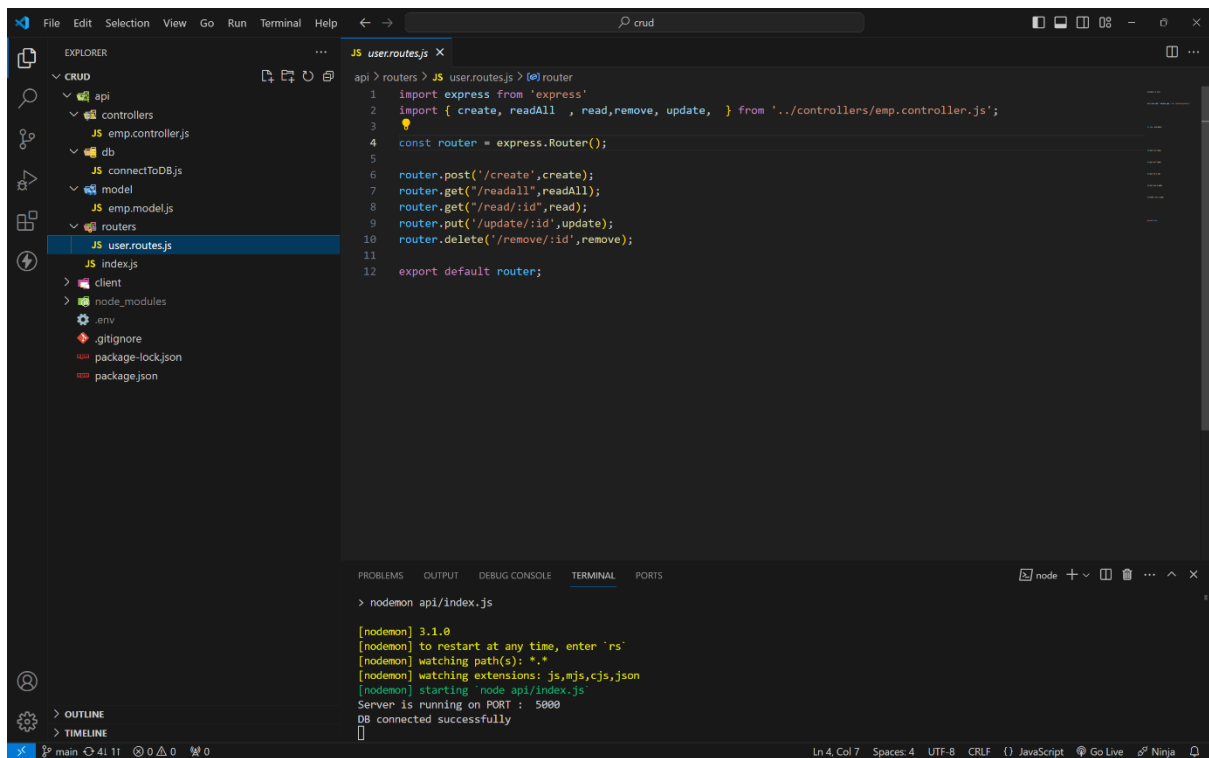
The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure with folders like 'api', 'controllers', 'db', 'model', 'routers', and 'index.js'. The file 'emp.model.js' is selected in the 'model' folder. The main editor shows the code for 'emp.model.js'.

```
api > model > JS emp.model.js > userSchema > role
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 with nodemon:

```
> 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 a Visual Studio Code editor with a project structure on the left and a code editor in the center. The project structure includes a `crud` folder with subfolders `api`, `controllers`, `db`, `model`, and `routes`. The `api` folder contains `user.routes.js`, which is the file being edited. The code in `user.routes.js` defines an Express router with routes for `create`, `readAll`, `read`, `update`, and `remove`. The terminal at the bottom shows the command `nodemon api/index.js` being executed, and the output indicates that the server is running on port 5000 and the database is connected successfully.

```
api > routes > JS user.routes.js > @ router
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;
```

```
> 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
```

## HOW TO RUN ON LOCALLY :

- 1 . Create a folder as any name.
- 2 . Open that folder in any code editor (vs code).
- 3 . Open terminal ( ctrl + ~ ) on code editor.
- 4 .

Type this code to get code locally.

**git clone <https://github.com/4727yesuraju/crud.git>**

- 5 . Now move to crud folder (cd crud in terminal)
- 6 .

Ignore client folder.

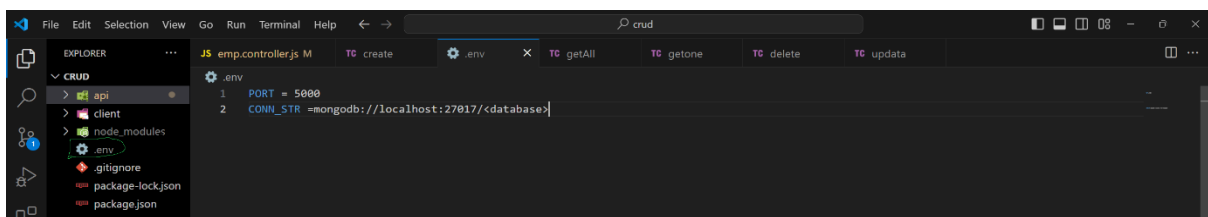
- 7 . Here crud is root folder.

- 8 . In root folder create a .env file and create a PORT and

**CONN\_STR** variables and assign value.

**ex : PORT = 3000** ( commonly any number between 3000 - 8080).

**CONN\_STR = your mongodb\_connection\_string.**



**OUTPUT :**

**CREATE :**

The screenshot shows the Thunder Client interface with a POST request to `http://localhost:5000/api/user/create` successfully executed. The request body is a JSON object representing a user. The response is a 201 status code with a JSON object containing the created user's details.

**Request Details:**

- Method: POST
- URL: `http://localhost:5000/api/user/create`
- Body (JSON):

```
{  "username": "rose",  "empname": "rose143",  "email": "rose@gmail.com",  "role": "software developen",  "salary": 90000}
```

**Response Details:**

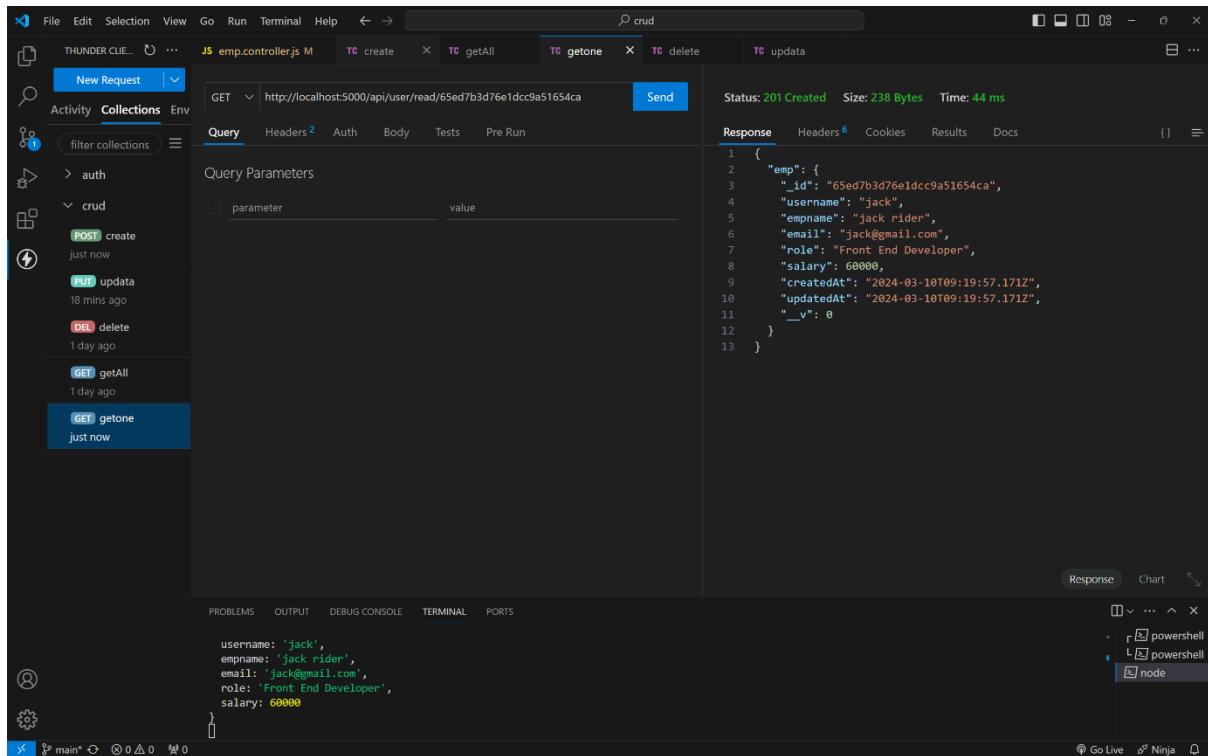
- Status: 201 Created
- Size: 140 Bytes
- Time: 241 ms
- Response (JSON):

```
{  "_id": "65ed7a9d76e1dcc9a51654c6",  "username": "rose",  "empname": "rose143",  "email": "rose@gmail.com",  "role": "software developer",  "salary": 90000}
```

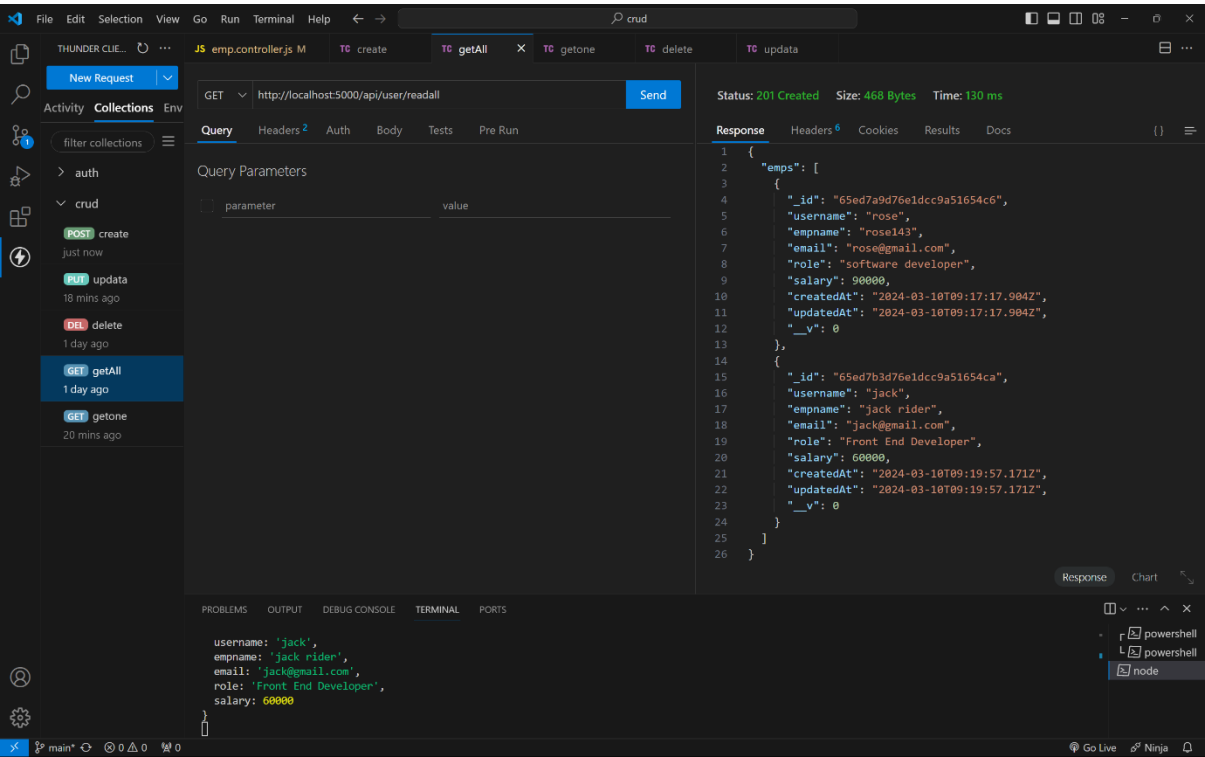
**Terminal Output:**

```
Error in create controller : Cast to ObjectId failed for value "{ _id: '65ed625bbd510e515f6767d' }" (type Object) at path "_id" for model "User"
Error in create controller : Cast to ObjectId failed for value "{ _id: '65ed625bbd510e515f6767d' }" (type Object) at path "_id" for model "User"
{
  username: 'rose',
  empname: 'rose143',
  email: 'rose@gmail.com',
  role: 'software developen',
  salary: 90000
}
```

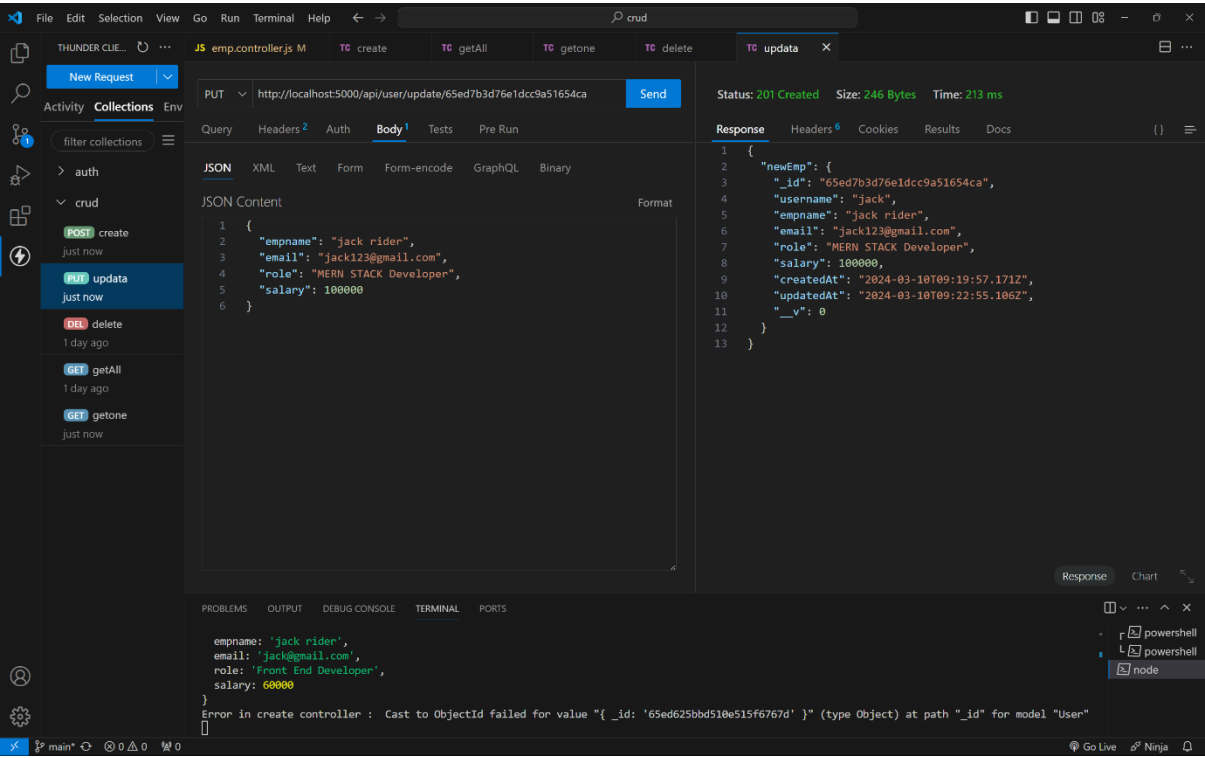
# READ ONE :



READ ALL :



UPDATE :



## DELETE :

The screenshot displays the Thunder Client interface with a DELETE request configured and executed. The request is sent to the URL `http://localhost:5000/api/user/remove/65ed7b3d76e1dcc9a51654ca`. The response is a 201 Created status with a size of 68 Bytes and a time of 111 ms. The response body is a JSON object: `{ "id": "65ed7b3d76e1dcc9a51654ca", "message": "deleted successfully.." }`.

**Request Details:**

- Method: DELETE
- URL: `http://localhost:5000/api/user/remove/65ed7b3d76e1dcc9a51654ca`
- Query Parameters: None

**Response Details:**

- Status: 201 Created
- Size: 68 Bytes
- Time: 111 ms
- Response Body: 

```
{
  "id": "65ed7b3d76e1dcc9a51654ca",
  "message": "deleted successfully.."
}
```

**Terminal Output:**

```
Node.js v20.11.0
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting node api/index.js
Server is running on PORT : 5000
DB connected successfully
```