**Creating an API:**

Npm init- creates a package.json file.

To make api – install => npm installexpress nodemon. => now we got nodemodules and packagelock.json files.

**App.js file:**

Const express=require(‘express’)

Const app=express();

App.listen(5000,()=>{  
console.log(‘server started’)

})

Lets create api now.

There are many methods to create api.- get,post,patch,delete.

We create simple api first.

In get- u can just receive response from server.

In post- u can send a request to server and can receive a response.

const express = require("express");

const app = express();

app.listen(3000, () => {

  console.log("server started");

});

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

})

Now when user enters email and paswd and clicks submit, then that data should be received by api.

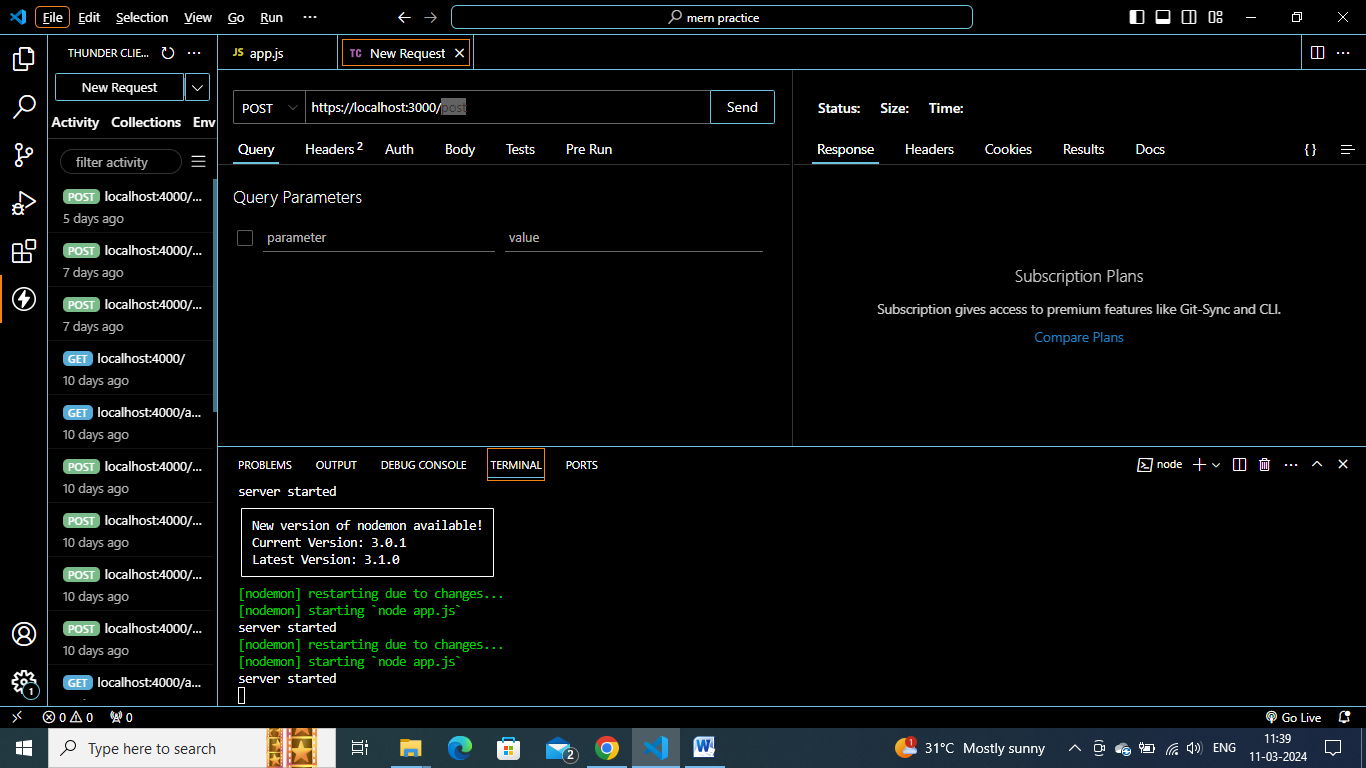
How to receive that.

* When user clicks submit, that website sends request to my api, to receive that , we can use req.body.
* To receive that data – console.log(req.body)

app.post("/post", async (req, res) => {

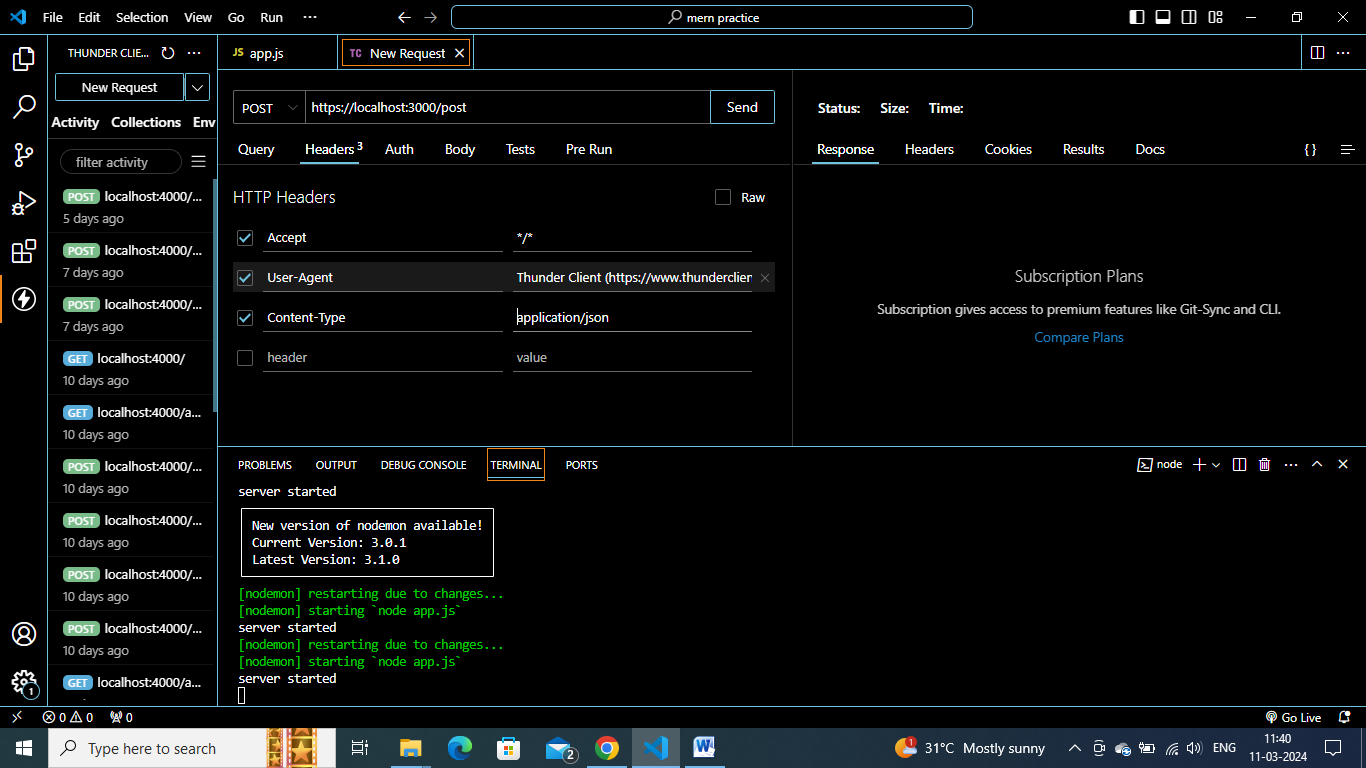
  console.log(req.body);

});

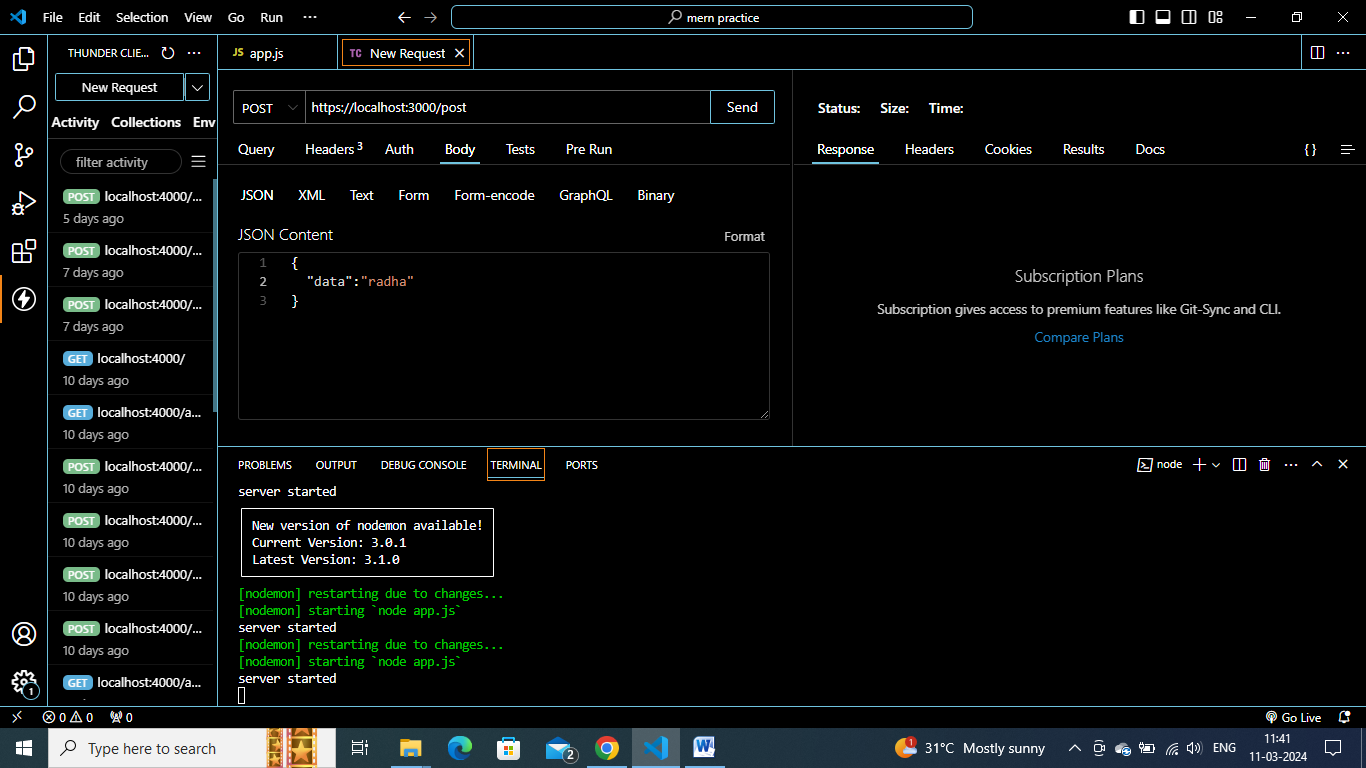


/post is the name of the api.

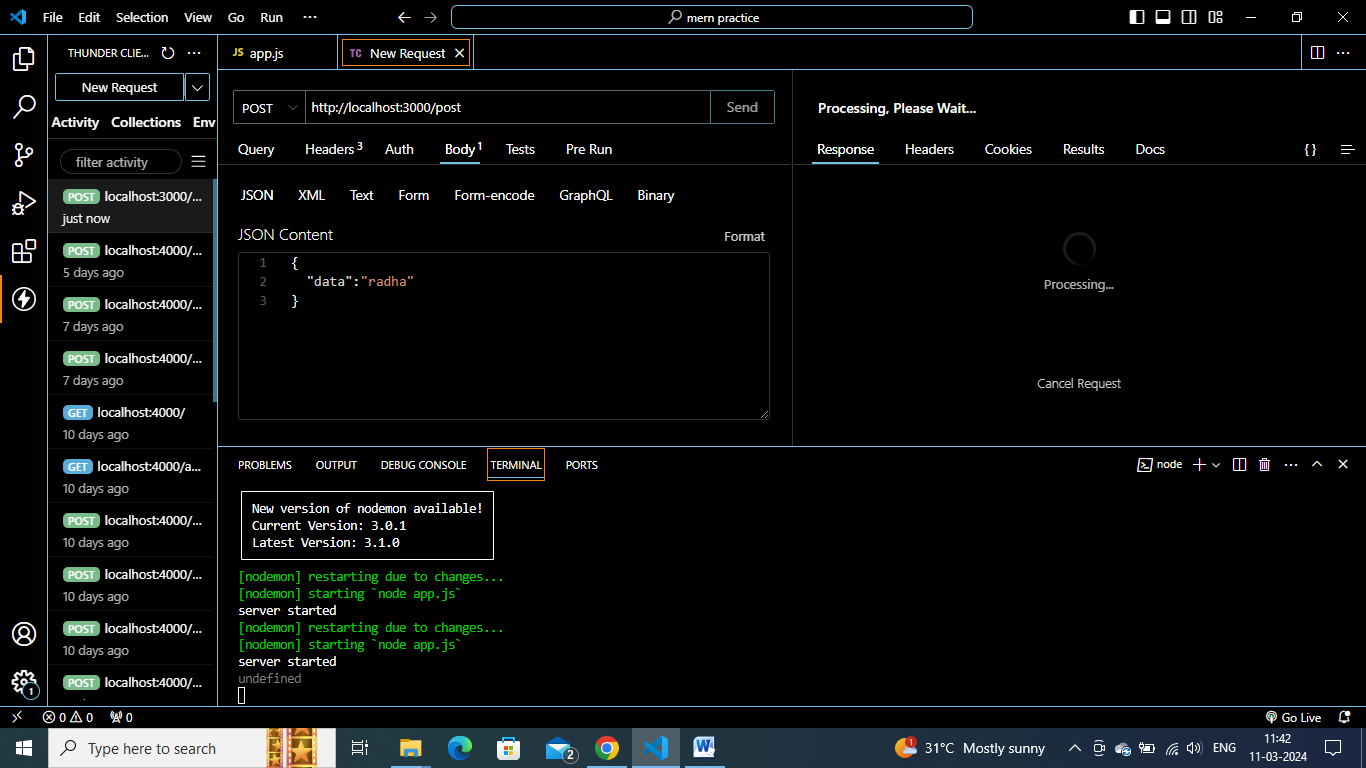
To receive json format –



Added content type header.



In body – we are writing data.

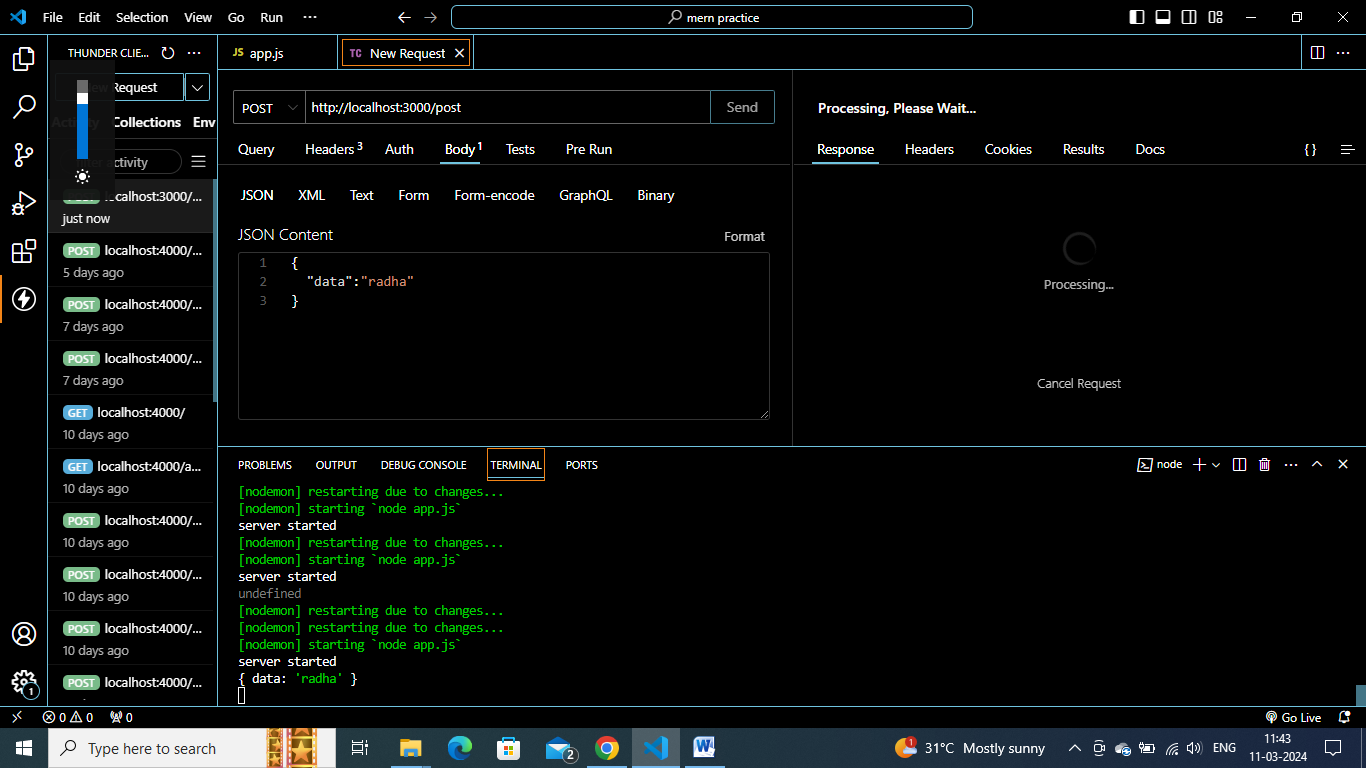


See in console , we got undefined.

As we are sending data in json format.

Use middleware

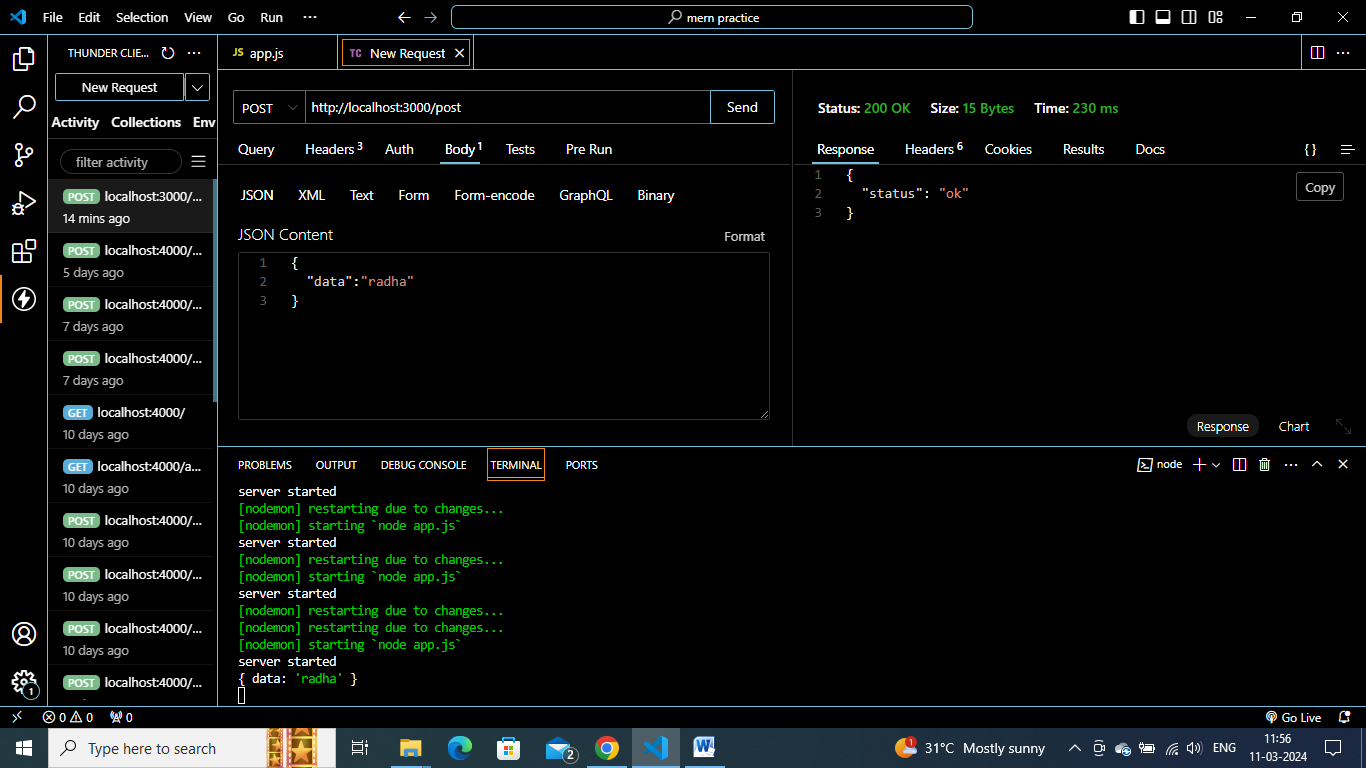
app.use(express.json())



Now see got the data. But the req in thunderclient is keep on loading , as request is never end, we didn’t received the response.

In this way, we can receive the data sent by user through api,

Now lets work on these data.



Now see im getting response as status ok.

This is the response from the server, in this way, u can work with request and response.

So overall, we created an api,we send data/req a api, now we are receiving response.

**Better Way of writing api, to handle errors:**

U can use try and catch block .

app.post("/post", async (req, res) => {

  console.log(req.body);

  const { data } = req.body; //destructure the data

  try {

    if (data === "radha") {

      res.send({ status: "ok" });

    } else {

      res.send({ status: "user not found" });

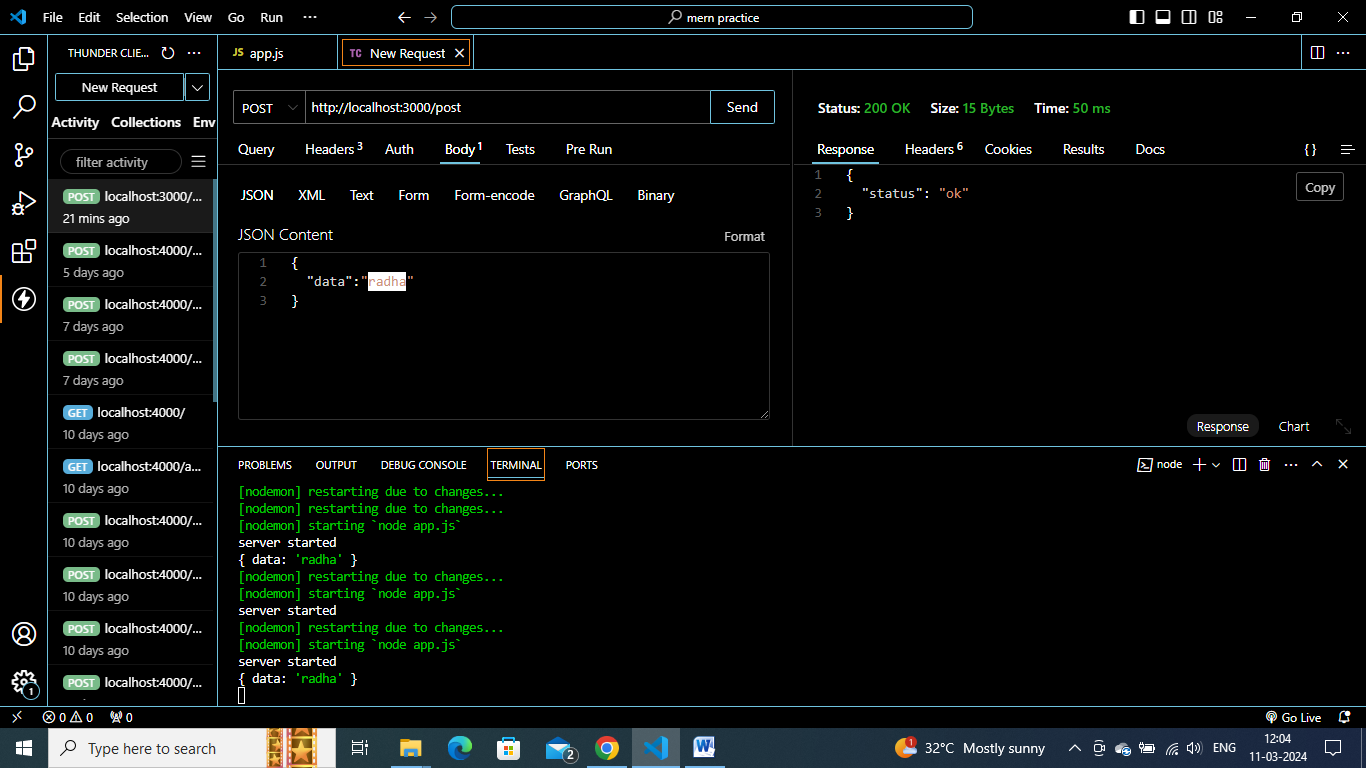
    }

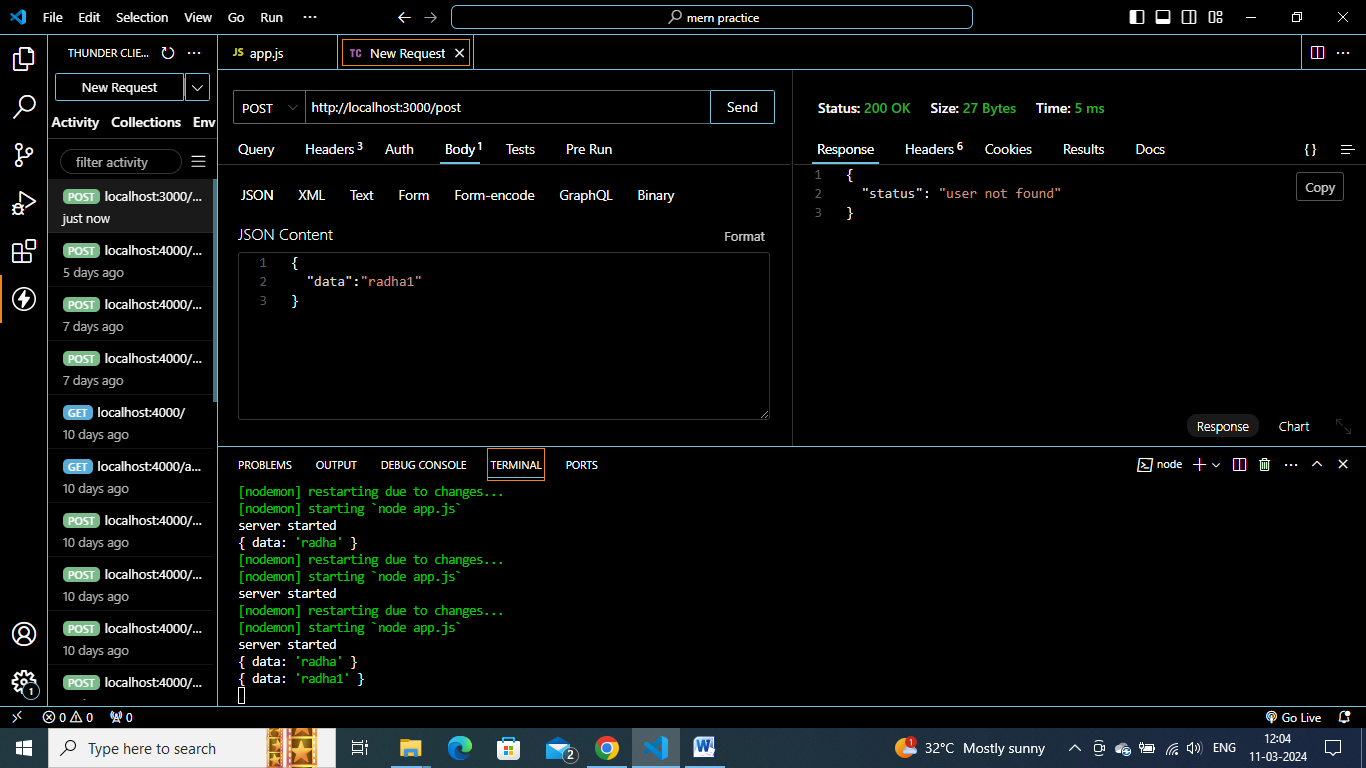
  } catch (err) {

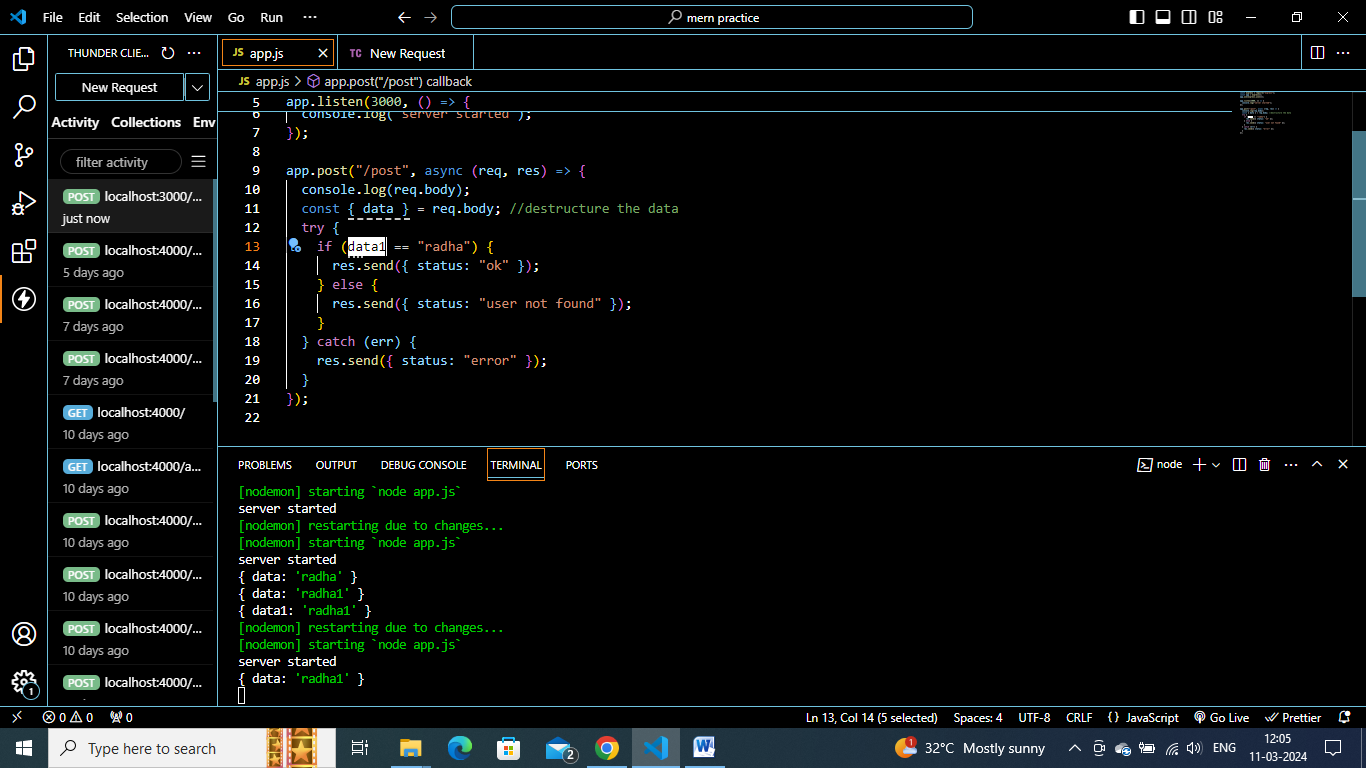
    res.send({ status: "error" });

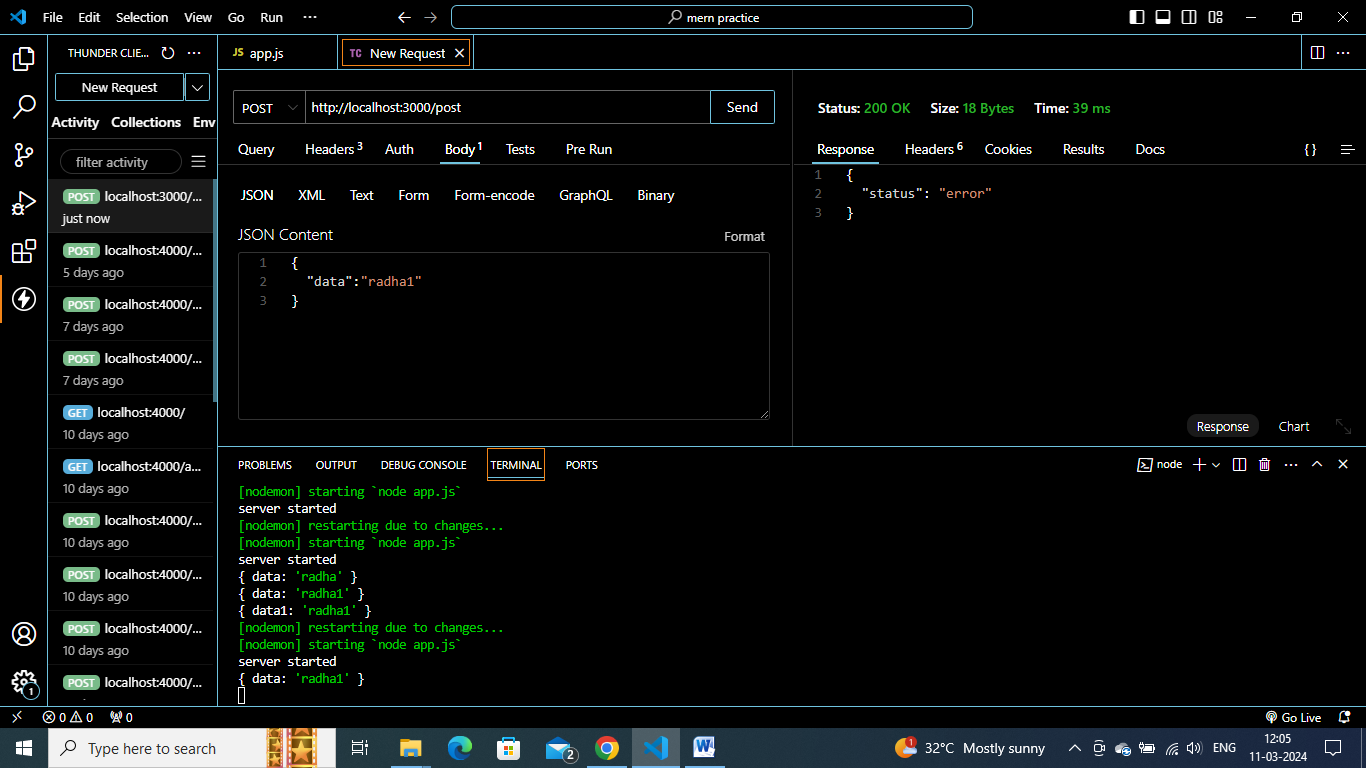
  }

});









app.post("/post", async (req, res) => {

  console.log(req.body);

  const { data } = req.body; //destructure the data

  try {

    if (data1 == "radha") {

      res.send({ status: "ok" });

    } else {

      res.send({ status: "user not found" });

    }

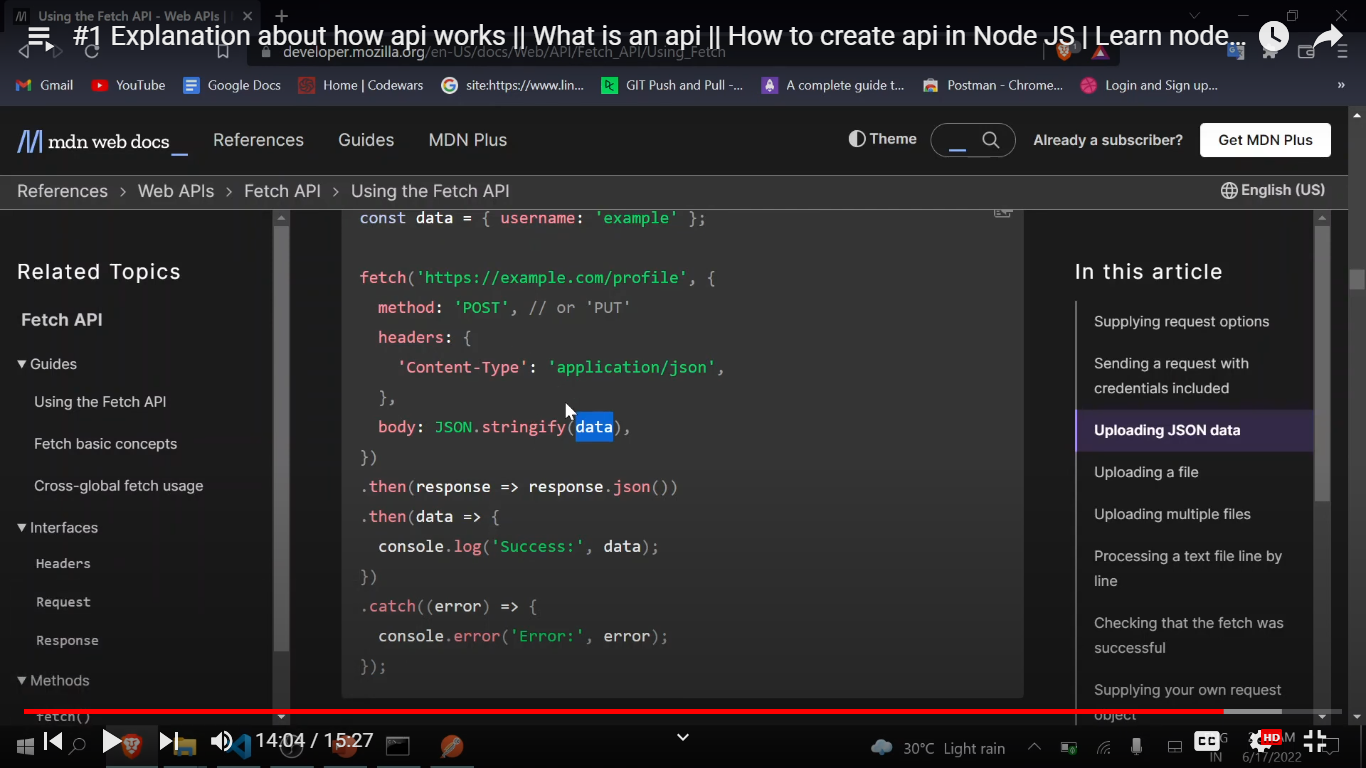
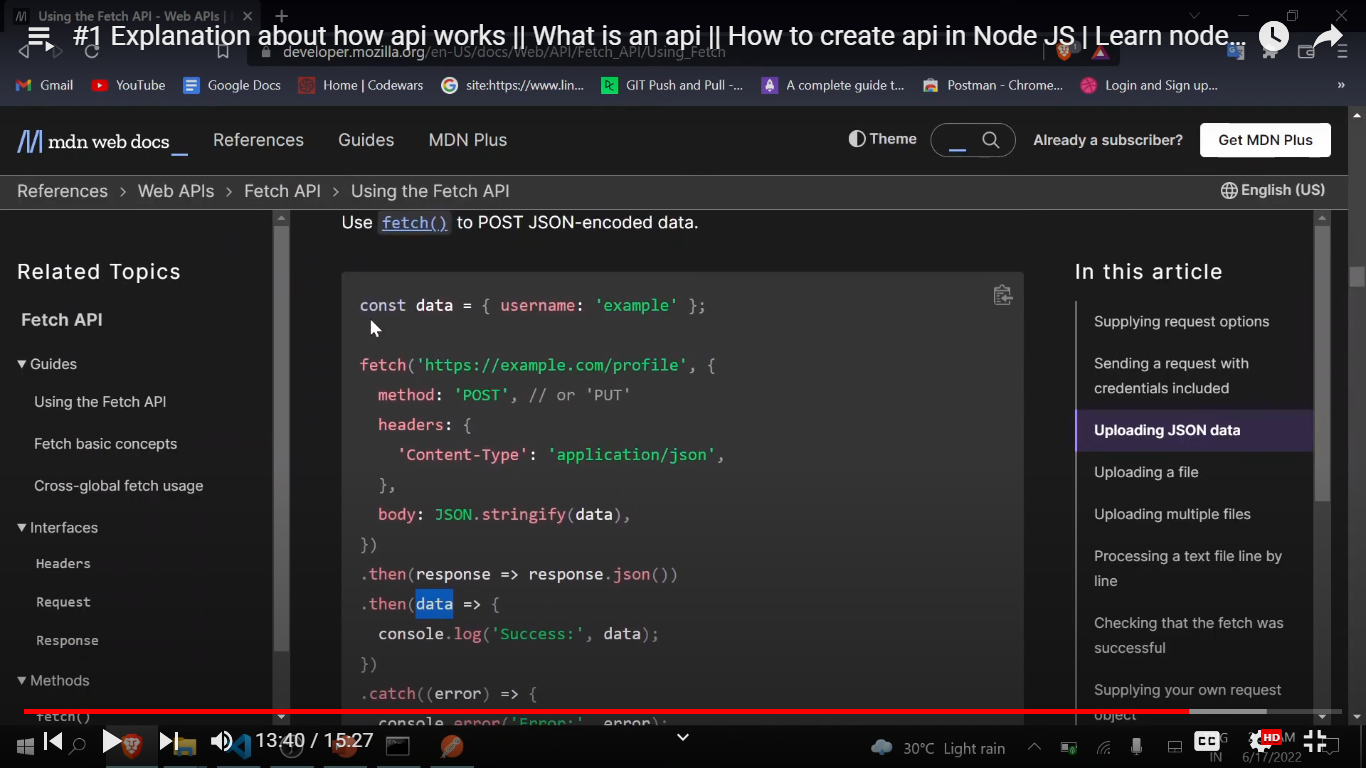
  } catch (err) {

    res.send({ status: "something went wrong try again !!!" });

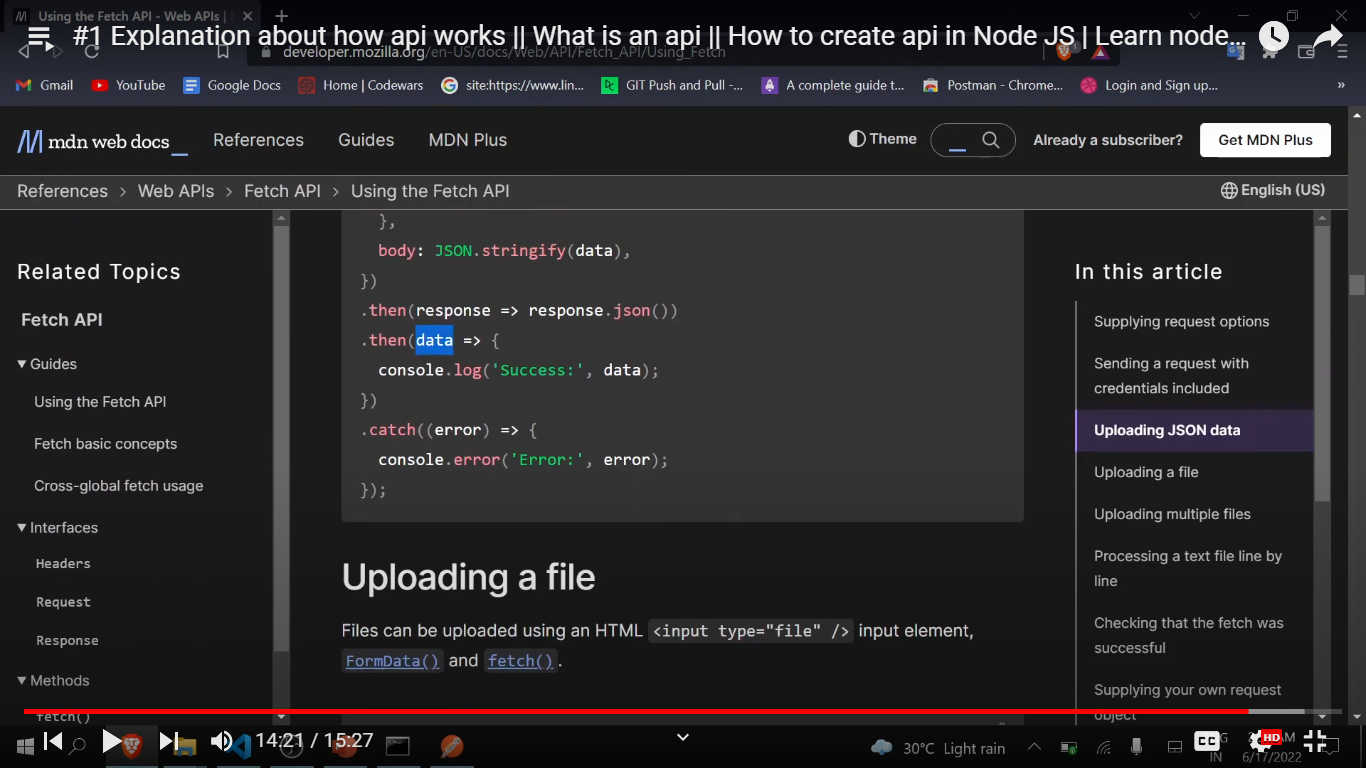
  }

});

Now we can create login form and call fetch function.



In place of data here in body, we will send our actual data.



We will receive that response like status:”ok” or anything here .

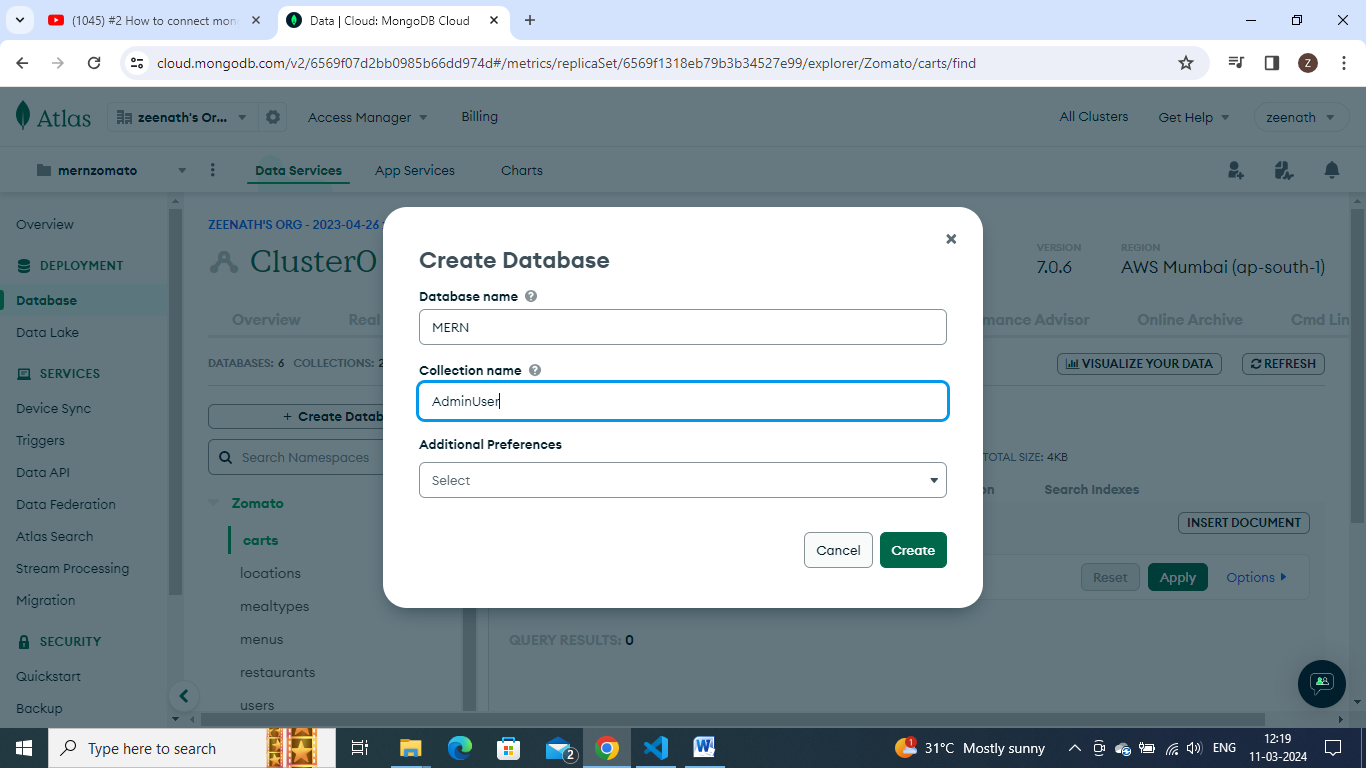
So we can check as if data.status===ok then we can alert – successful login .

In this way we can use api between react form and db.

**Fetch/axios and connecting to mongodb:**

Now making this api dynamic. For that we use mongodb.

Creating a db:



Npm I mongoose.

const mongoose = require("mongoose");

const mongourl =

  "mongodb+srv://login:login@cluster0.xlrs6ke.mongodb.net/Mern?retryWrites=true&w=majority&appName=Cluster0";

mongoose

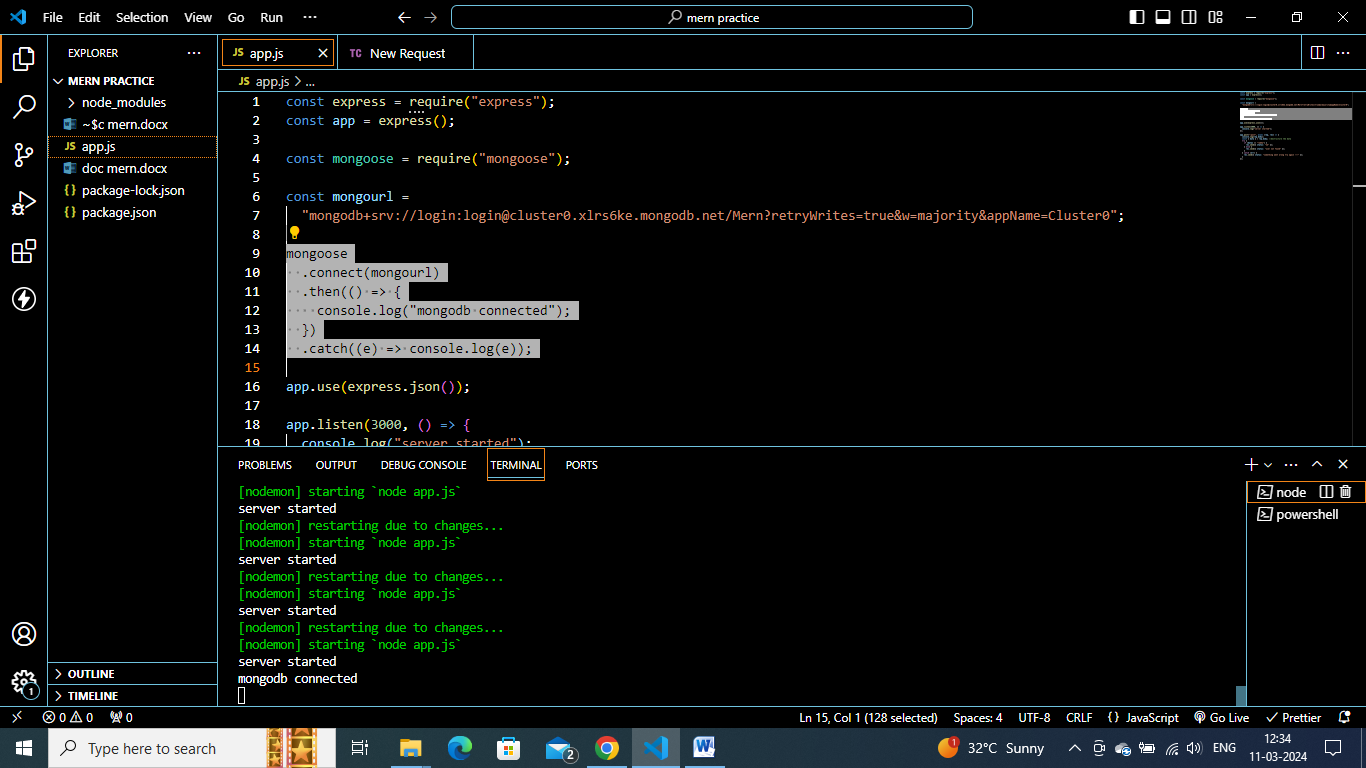
  .connect(mongourl)

  .then(() => {

    console.log("mongodb connected");

  })

  .catch((e) => console.log(e));



Now our mongodb is successfully got connected to nodejs server.

So now we can make any type of changes in our mongodb through our api, we can insert user data, remove user data ,we can create schema etc.

**Creating schema and add user to mongodb:  
schema:**  structure for data or blueprint for data.

Creating schema for user details- variables needed can be name,email ,user id, pwd etc

**Userdetails.js:**

We can use mongoose.Schema () function

const mongoose=require('mongoose');

const userDetailsSchema=new mongoose.Schema({

    name:String,

    email:String,

    phoneno:String

})

This is creating schema

Then storing it in a collection.

const mongoose = require("mongoose");

const userDetailsSchema = new mongoose.Schema(

  {

    uname: String,

    email: String,

    phoneno: String,

  },

  {

    collection: "userInfo",

  }

);

Then creating a model.

mongoose.model("userInfo",userDetailsSchema);

this schema will get developed in this particular collection userinfo in mongodb.

Now import it in app.js

**App.js:**

require('./userDetails');

const User=mongoose.model("UserInfo")

storing all model and schema data into variable user.

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

    try{

    }

    catch{

    }

})

To register need some data form user, we can access it using req.body.

    const {name,email,mobileNo}=req.body;

        await User.create({

            uname:name,

            email,

            phoneno:mobileNo

        });

Left side names- from schema and right side names from req.body.

**App.js:**

require('./userDetails');

const User=mongoose.model("userInfo");

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

    const {name,email,mobileNo}=req.body;

    try{

        await User.create({

            uname:name,

            email,

            phoneno:mobileNo

        });

        res.send({status:"ok"})

    }

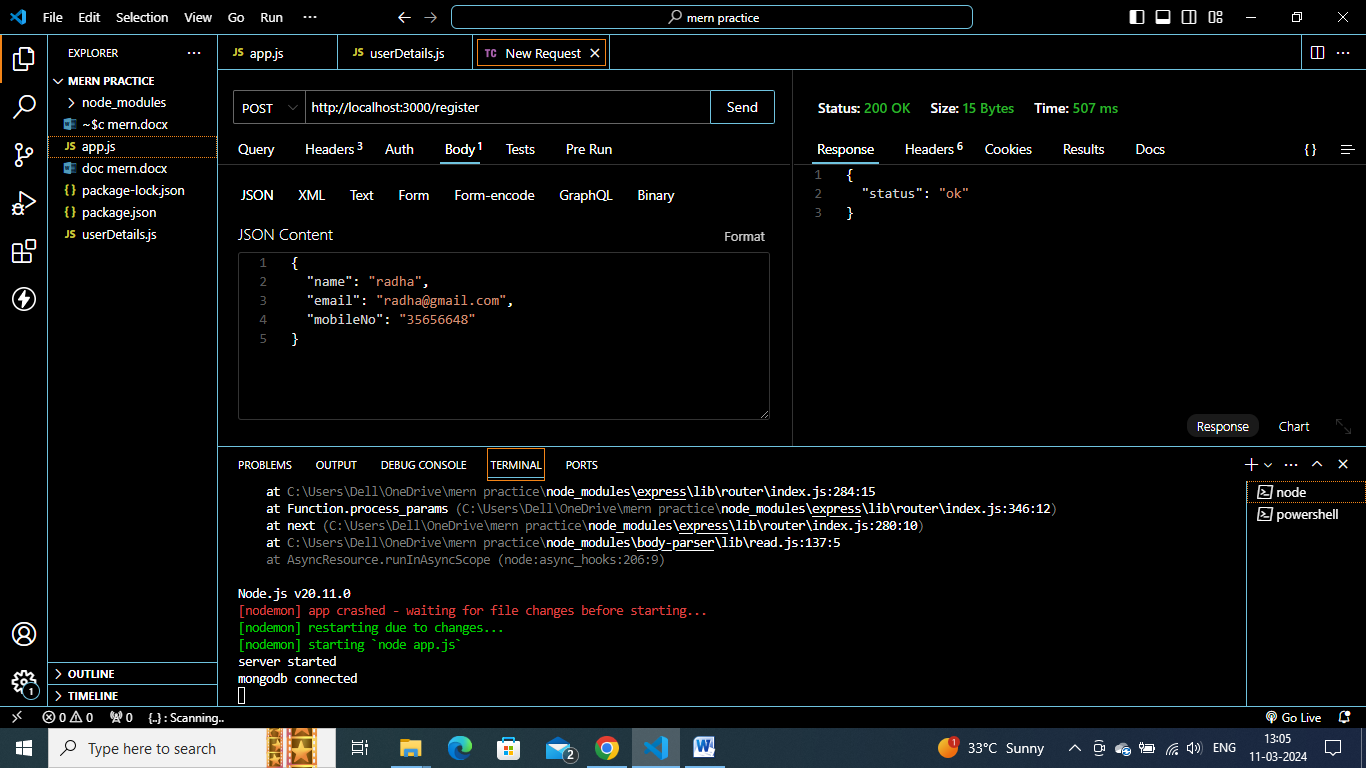
    catch{

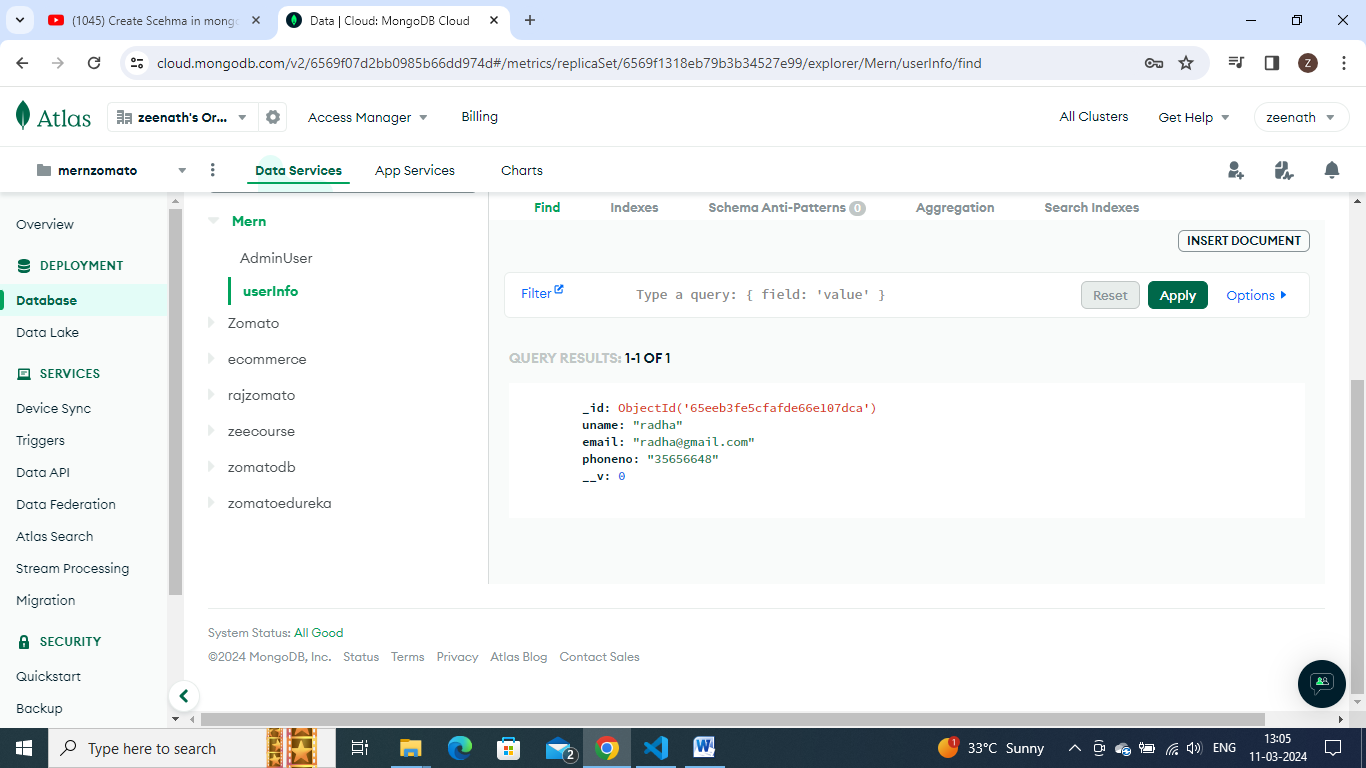
        res.send({status:"error"})

    }

})

Check in postman, keep content-type in headers .





In this way, we can create new user in mongodb with nodejs.

**Creating login and registration in react and testing it now with fetch instead of postman:**

Combining react, node ,mongodb

<https://github.com/SinghDigamber/react-login-signup-ui-template>

just login and signup static form in react.

**Signupcomponent.jsx:**

import React, { Component } from 'react'

export default class SignUp extends Component {

  render() {

    return (

      <form>

        <h3>Sign Up</h3>

        <div className="mb-3">

          <label>First name</label>

          <input

            type="text"

            className="form-control"

            placeholder="First name"

          />

        </div>

        <div className="mb-3">

          <label>Last name</label>

          <input type="text" className="form-control" placeholder="Last name" />

        </div>

        <div className="mb-3">

          <label>Email address</label>

          <input

            type="email"

            className="form-control"

            placeholder="Enter email"

          />

        </div>

        <div className="mb-3">

          <label>Password</label>

          <input

            type="password"

            className="form-control"

            placeholder="Enter password"

          />

        </div>

        <div className="d-grid">

          <button type="submit" className="btn btn-primary">

            Sign Up

          </button>

        </div>

        <p className="forgot-password text-right">

          Already registered <a href="/sign-in">sign in?</a>

        </p>

      </form>

    )

  }

}

**Logincomponent.jsx:**

import React, { Component } from 'react'

export default class Login extends Component {

  render() {

    return (

      <form>

        <h3>Sign In</h3>

        <div className="mb-3">

          <label>Email address</label>

          <input

            type="email"

            className="form-control"

            placeholder="Enter email"

          />

        </div>

        <div className="mb-3">

          <label>Password</label>

          <input

            type="password"

            className="form-control"

            placeholder="Enter password"

          />

        </div>

        <div className="mb-3">

          <div className="custom-control custom-checkbox">

            <input

              type="checkbox"

              className="custom-control-input"

              id="customCheck1"

            />

            <label className="custom-control-label" htmlFor="customCheck1">

              Remember me

            </label>

          </div>

        </div>

        <div className="d-grid">

          <button type="submit" className="btn btn-primary">

            Submit

          </button>

        </div>

        <p className="forgot-password text-right">

          Forgot <a href="#">password?</a>

        </p>

      </form>

    )

  }

}

**App.jsx:**

import React from 'react'

import '../node\_modules/bootstrap/dist/css/bootstrap.min.css'

import './App.css'

import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom'

import Login from './components/login.component'

import SignUp from './components/signup.component'

function App() {

  return (

    <Router>

      <div className="App">

        <nav className="navbar navbar-expand-lg navbar-light fixed-top">

          <div className="container">

            <Link className="navbar-brand" to={'/sign-in'}>

              positronX

            </Link>

            <div className="collapse navbar-collapse" id="navbarTogglerDemo02">

              <ul className="navbar-nav ml-auto">

                <li className="nav-item">

                  <Link className="nav-link" to={'/sign-in'}>

                    Login

                  </Link>

                </li>

                <li className="nav-item">

                  <Link className="nav-link" to={'/sign-up'}>

                    Sign up

                  </Link>

                </li>

              </ul>

            </div>

          </div>

        </nav>

        <div className="auth-wrapper">

          <div className="auth-inner">

            <Routes>

              <Route exact path="/" element={<Login />} />

              <Route path="/sign-in" element={<Login />} />

              <Route path="/sign-up" element={<SignUp />} />

            </Routes>

          </div>

        </div>

      </div>

    </Router>

  )

}

export default App

**signup component:**

to get input box values – creating state.

First converted signup from class component to functional component-

import React from "react";

export const SignUp = () => {

  return (

    <form>

      <h3>Sign Up</h3>

      <div className="mb-3">

        <label>First name</label>

        <input type="text" className="form-control" placeholder="First name" />

      </div>

      <div className="mb-3">

        <label>Last name</label>

        <input type="text" className="form-control" placeholder="Last name" />

      </div>

      <div className="mb-3">

        <label>Email address</label>

        <input

          type="email"

          className="form-control"

          placeholder="Enter email"

        />

      </div>

      <div className="mb-3">

        <label>Password</label>

        <input

          type="password"

          className="form-control"

          placeholder="Enter password"

        />

      </div>

      <div className="d-grid">

        <button type="submit" className="btn btn-primary">

          Sign Up

        </button>

      </div>

      <p className="forgot-password text-right">

        Already registered <a href="/sign-in">sign in?</a>

      </p>

    </form>

  );

};

**Creating states in signup component:**

  const [fname, setFname] = useState("");

  const [lname, setLname] = useState("");

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

        <input type="text" className="form-control" placeholder="First name" onChange={(e)=>setFname(e.target.value)}/>

**Similarly make for lname,email,passwords input boxes.**

function handleSubmit() {

    console.log(fname, lname, email, password);

  }

    <form onSubmit={handleSubmit}>

    <form onSubmit={handleSubmit}>

**run and check by submitting signup form- check in console.**

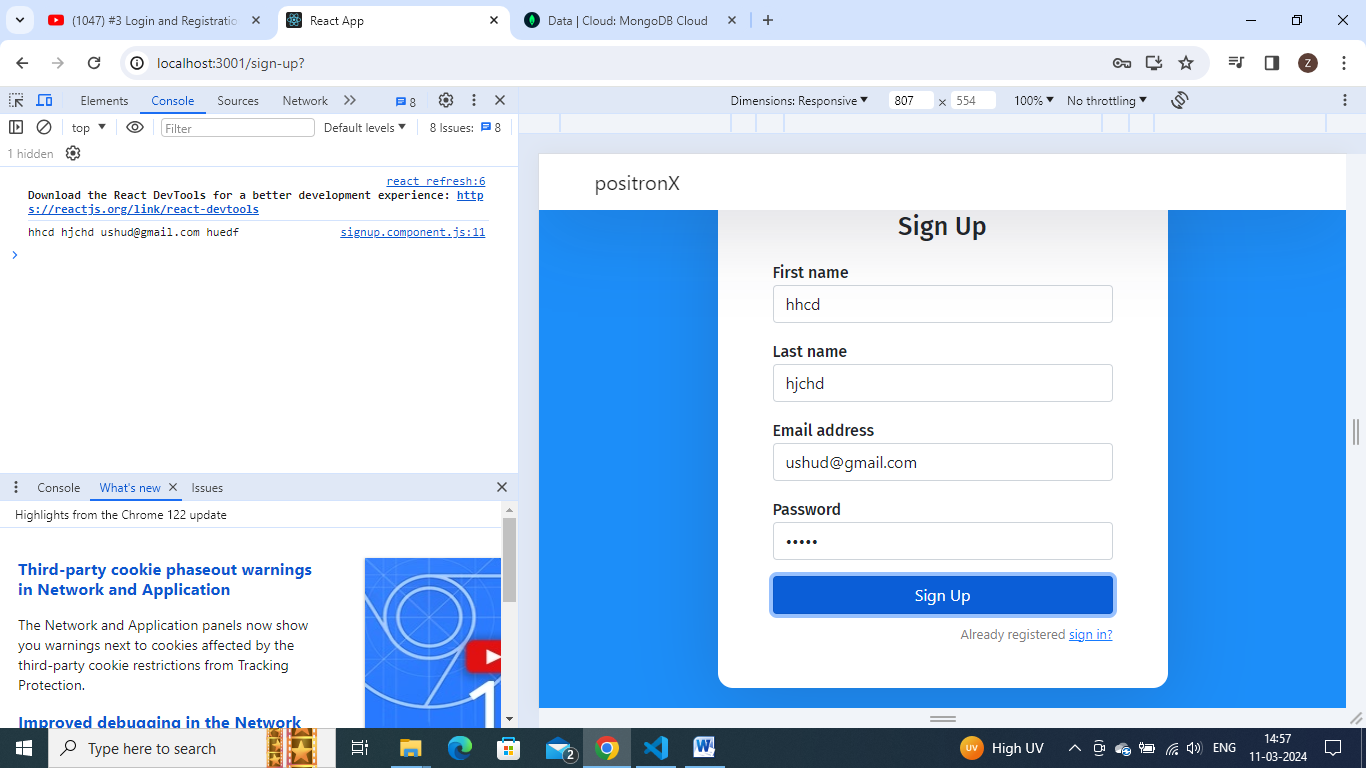
  function handleSubmit(e) {

    e.preventDefault()

    console.log(fname, lname, email, password);

  }

**Add**  e.preventDEfault to avoid refresh of pageon submit.



Now we are successfully getting data.

Now lets move to nodejs part.

**Making new register api in app.js:**

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

    //taking data- name,email,pwds from req.body

    const {fname,lname,email,password}=req.body;

    try{

    }

    catch(e){

    }

})

Now before moving further need to create schema

**userDetails.js:**

const userDetailsSchema = new mongoose.Schema(

  {

    fname: String,

    lname:String,

    email: String,

    phoneno: String,

  },

  {

    collection: "userInfo",

  }

);

**App.js**

Impoting schema

require('./userDetails');

const User=mongoose.model('userInfo')

app.post("/register", async (req, res) => {

  //taking data- name,email,pwds from req.body

  const { fname, lname, email, password } = req.body;

  try {

  } catch (e) {}

});

app.post("/register", async (req, res) => {

  //taking data- name,email,pwds from req.body

  const { fname, lname, email, password } = req.body;

  try {

    await User.create({

        fname,

        lname,

        email,

        password

    });

    res.send({status:"ok"})

  } catch (e) {

    res.send(

        {

            status:"error"

        }

    )

  }

});

Now our nodejs api creation is done.

Now lets try to register it in react js using fetch api.

**Frontend>signup.js:**

  function handleSubmit(e) {

    e.preventDefault();

    console.log(fname, lname, email, password);

    fetch("http://localhost:3000/register", {

      method: "POST",

      crossDomain: true,

      headers: {

        "Content-Type": "application/json",

        Accept: "application/json",

        "Access-Control-Allow-Origin": "\*",

      },

      body: JSON.stringify({

        fname,

        lname,

        email,

      }),

    })

      .then((res) => res.json())

      .then((data) => console.log(data, "User Registered successfully"));

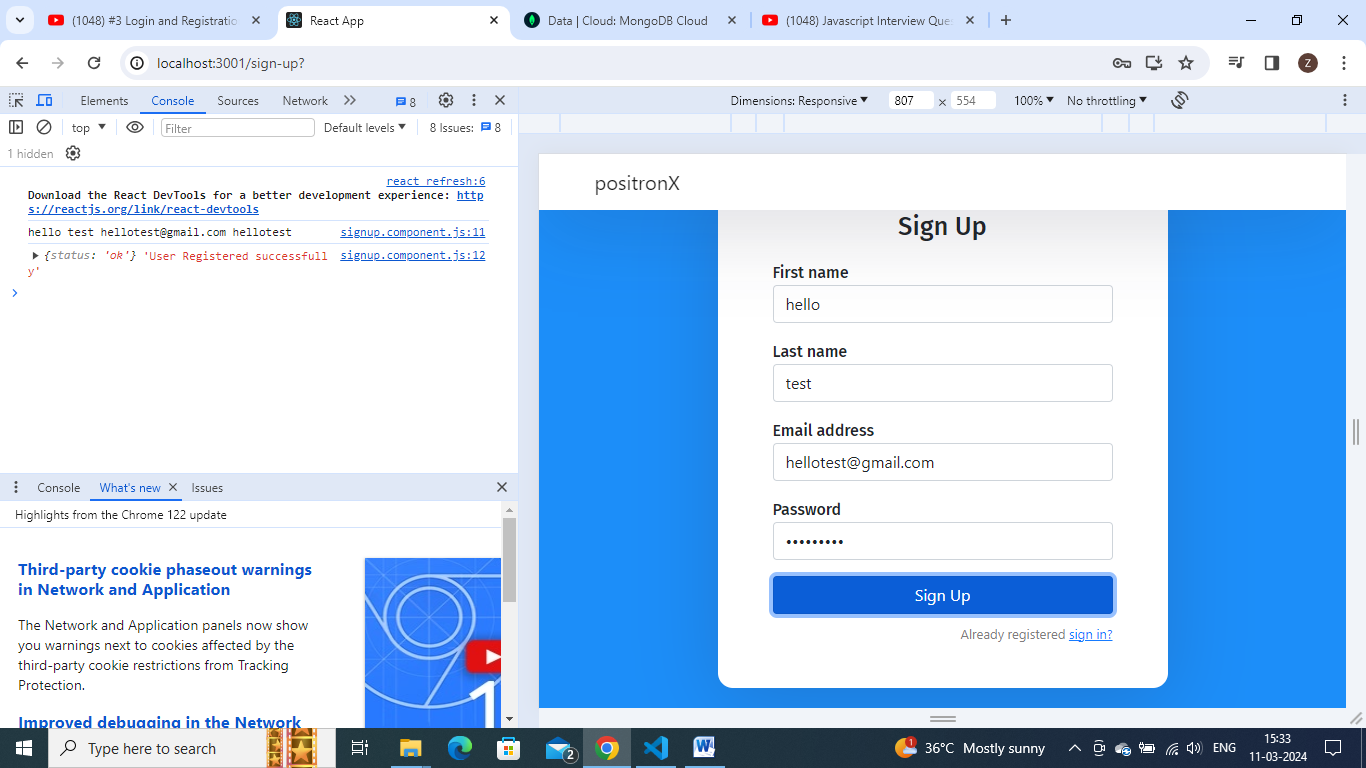
  }

Npm I cors

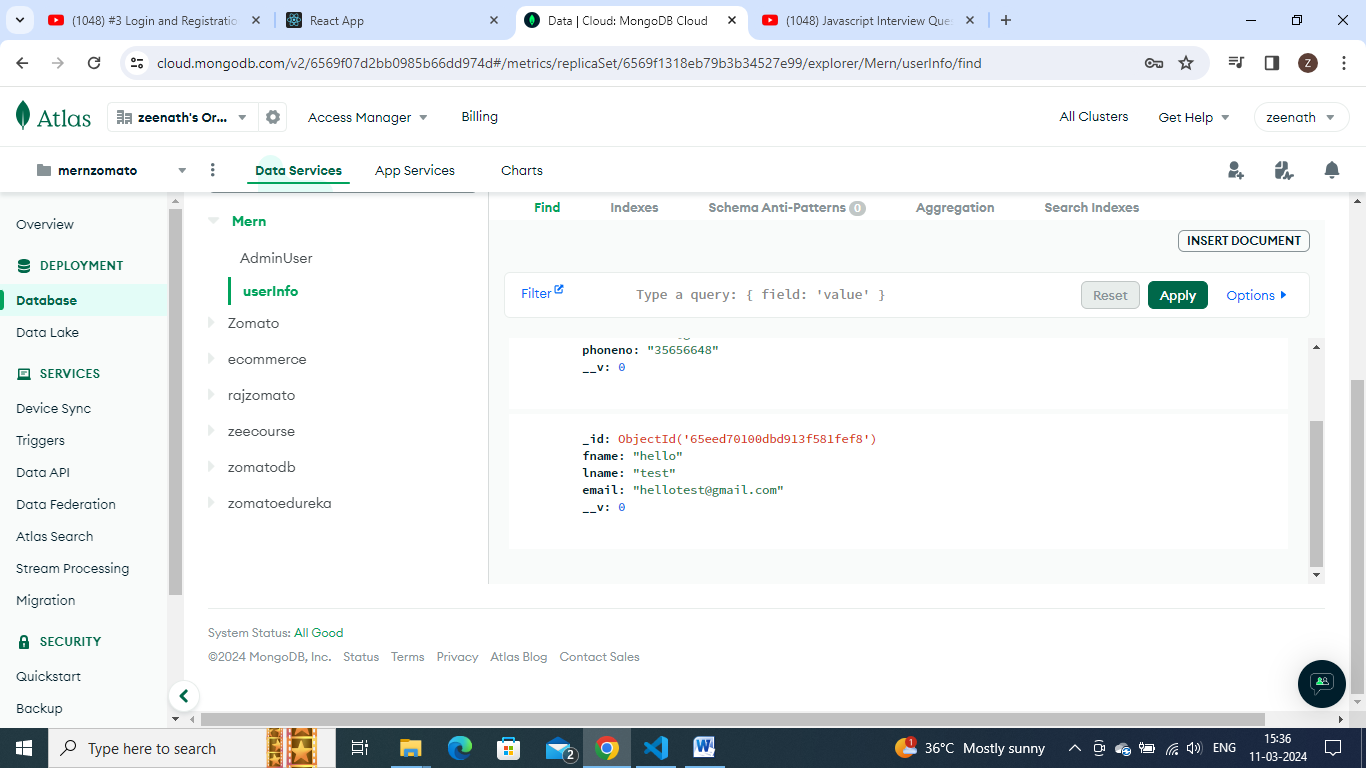
**App.js:**

const cors=require('cors');

app.use(cors());

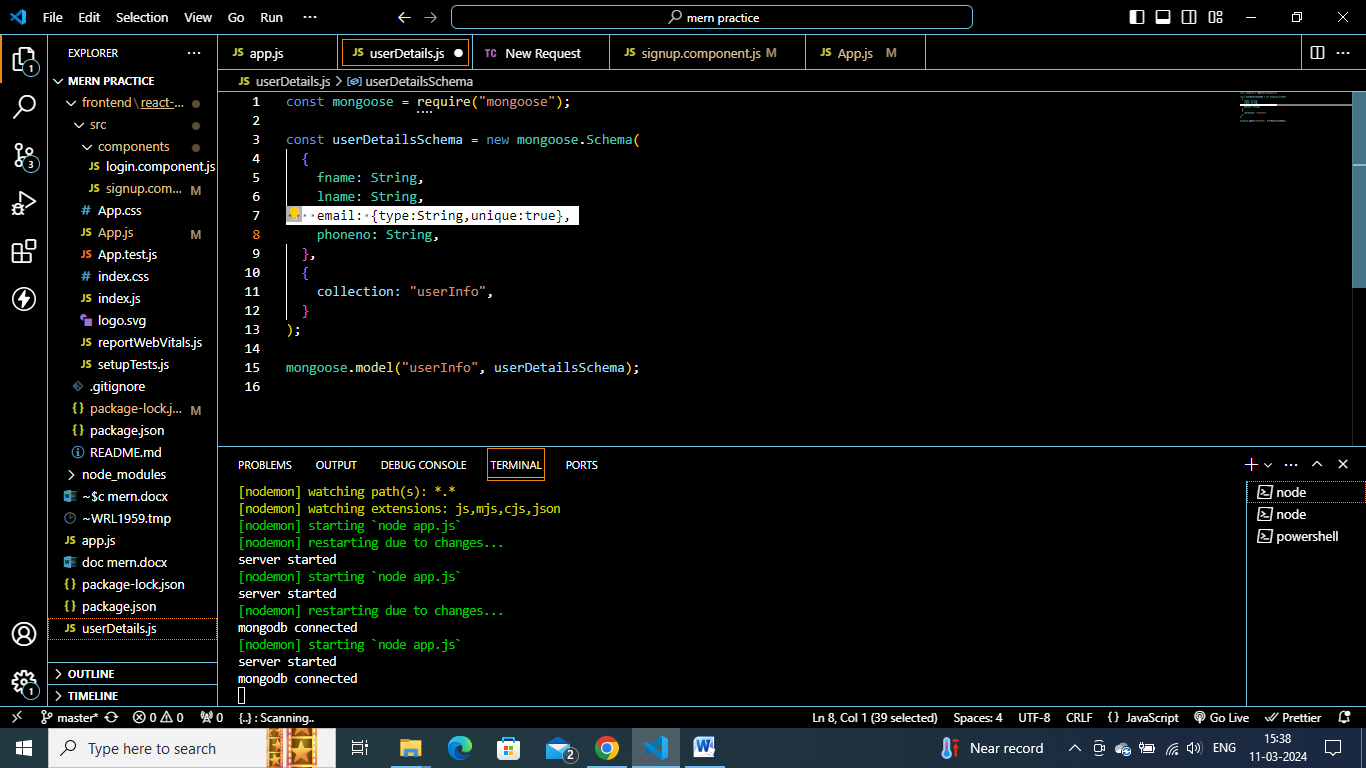


We got status –ok => it means data got stored into mongodb successfully.



Data has been stored in mongodb.

Making sure that one user with same email id should not exits.



Added unique to email in schema

Now all users email ids should be unique.

  try {

    const olduser=User.findOne({email});

    if(olduser){

        res.send({ error : "User already exists"})

    }

    await User.create({

      fname,

      lname,

      email,

      password,

    });

    res.send({ status: "ok" });

  } catch (e) {

    res.send({

      status: "error",

    });

  }

});

Using bcrypt.js to crypt the data.

Npm I bcryptjs

**App.js:**

const bcrypt = require("bcryptjs");

app.post("/register", async (req, res) => {

  //taking data- name,email,pwds from req.body

  const { fname, lname, email, password } = req.body;

  const encryptedPassword = await bcrypt.hash(password, 10);

  try {

    const olduser = await User.findOne({ email });

    if (olduser) {

      res.send({ error: "User already exists" });

    }

    await User.create({

      fname,

      lname,

      email,

      password:encryptedPassword,

    });

    if (olduser) {

      return res.send({ error: "User already exists" });

    }

Adding return, as we cant send two responses.

