Name : Vivek Lakhlani

Roll no : 025

Subject : Applicatipon developed using fullstack (705)

Semester : 7th

Division : A

Date : 30/7/2023

**Practical Assignment : 1**

GITHUB LINK : <https://github.com/Vivek2425/Practical_Assignment_1_25_705.git>

1. Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.

- Handle POST request.

Program :

const http = require('http');

const url = require('url');

const fs = require('fs');

const static = require('node-static');

var fileserver = new static.Server('files');

const server = http.createServer((req,res)=>{

    var url1 = url.parse(req.url, true);

    if(url1.pathname=="/file.html"){

        fileserver.serve(req,res);

    }else if(url1.pathname=="/index.html"){

        fileserver.serve(req,res);

    }else if(url1.pathname=="/processData" && req.method=="GET"){

        res.write("Name : " + url1.query.name + " Email : " + url1.query.email + " Passowrd : " + url1.query.password);

        res.end()

    }else if(url1.pathname=="/processData" && req.method=="POST"){

        let body = '';

        req.on("data" ,chunk=>{

            body += chunk.toString();

        })

        req.on("end" ,()=>{

            res.end('ok => '+ body)

        })

    }else{

        res.end("get lost");

    }

    // req.on("/",()=>{

    //     console.log("hello world")

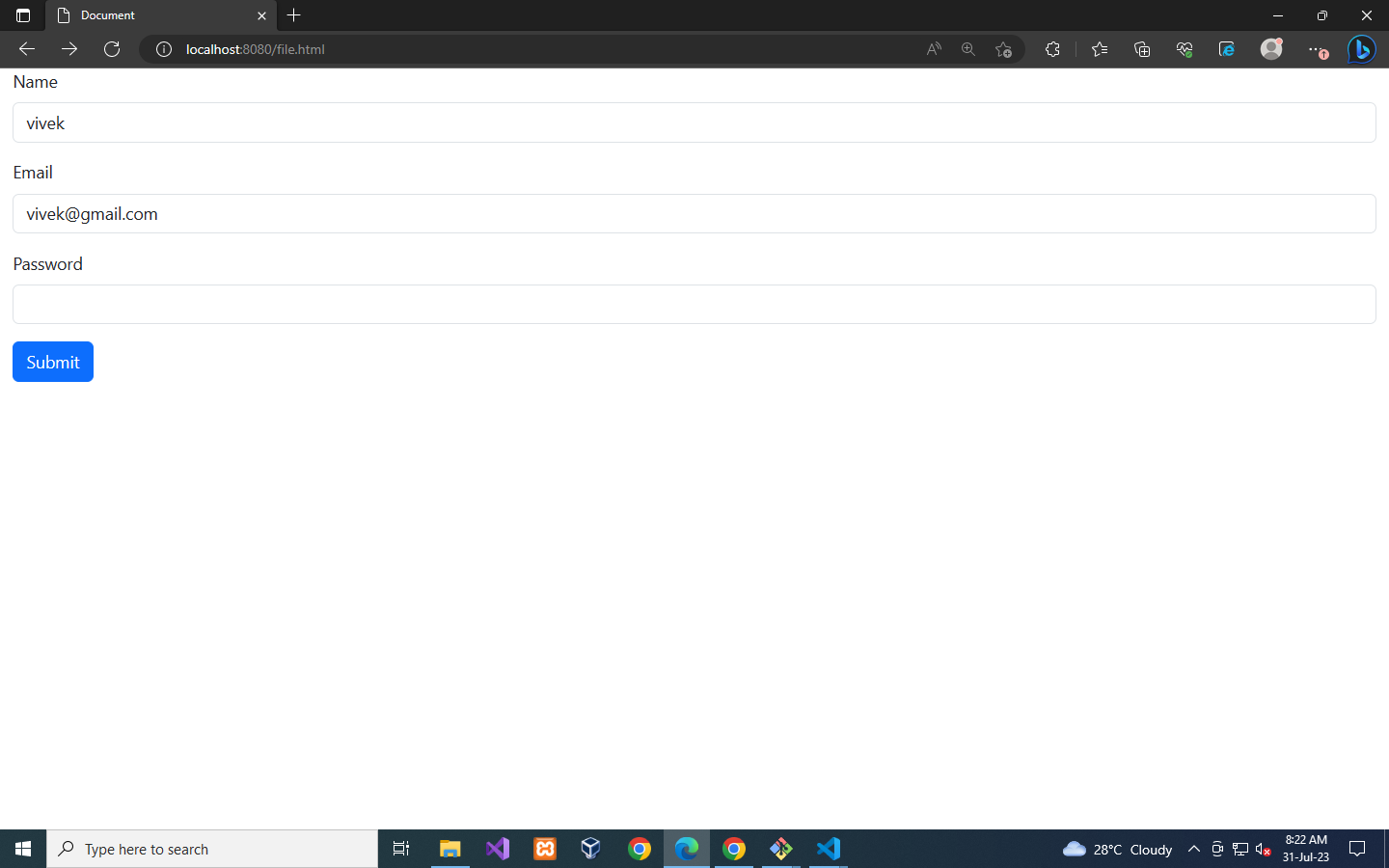
    // })

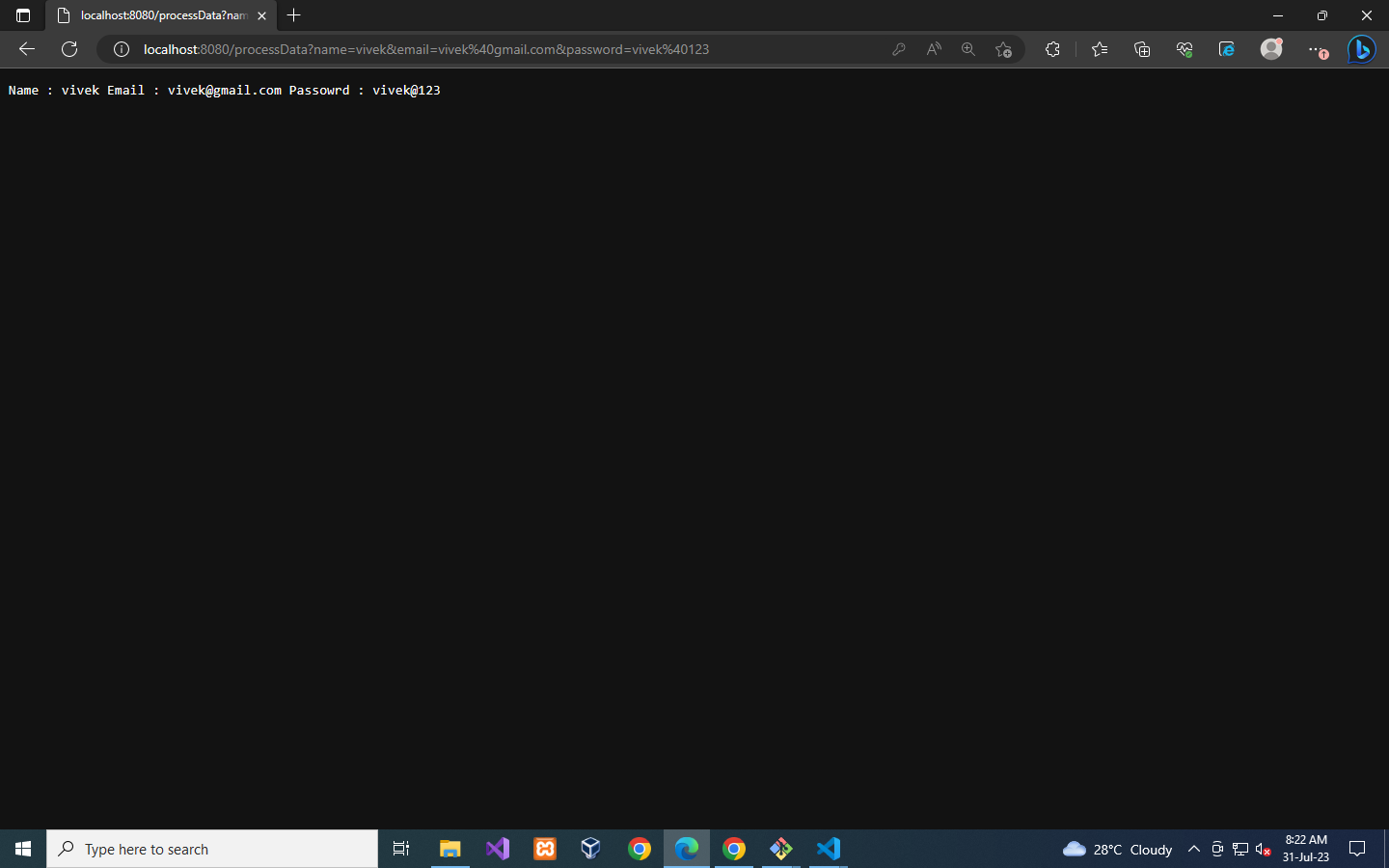
})

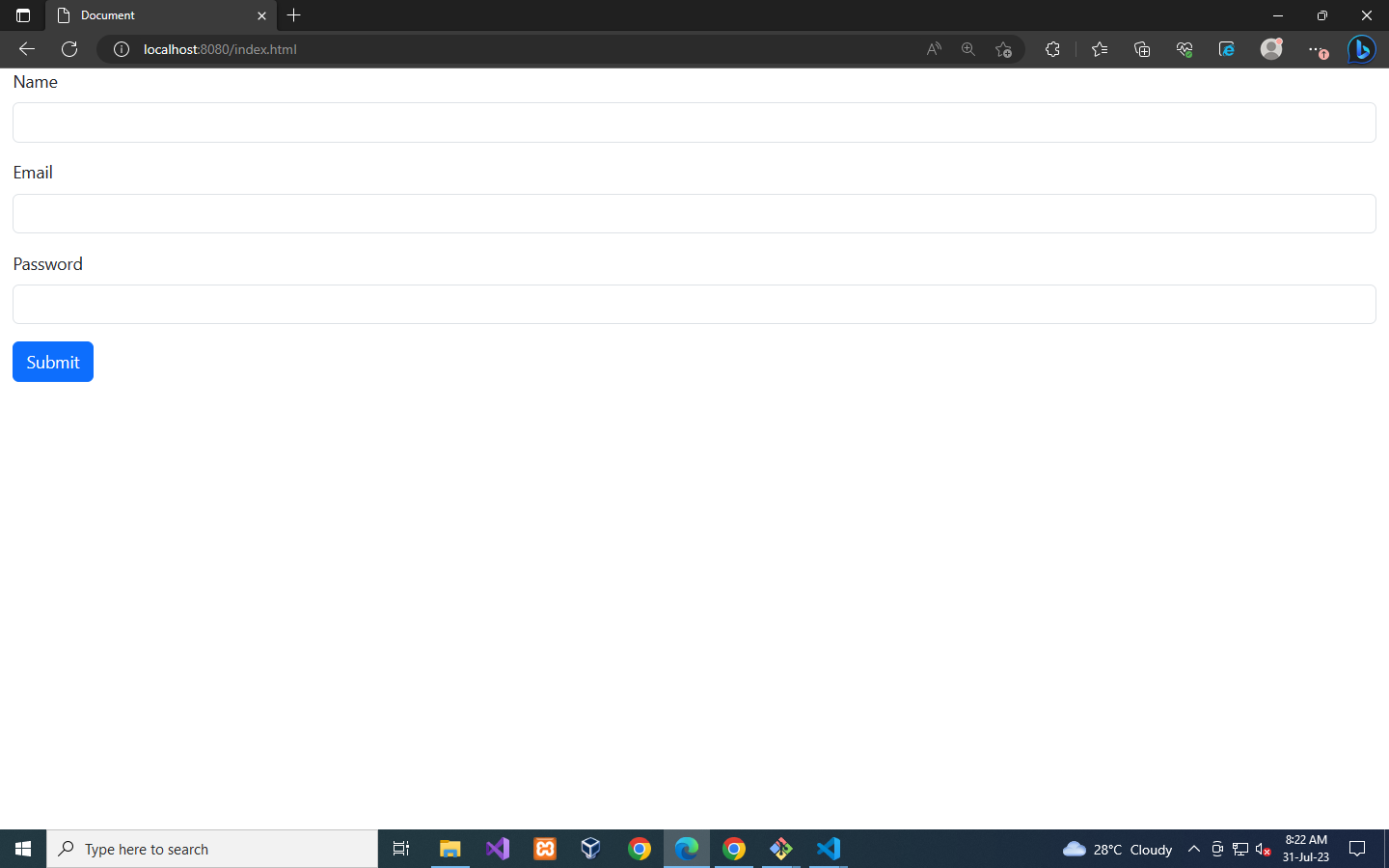
server.listen(8080,()=>{

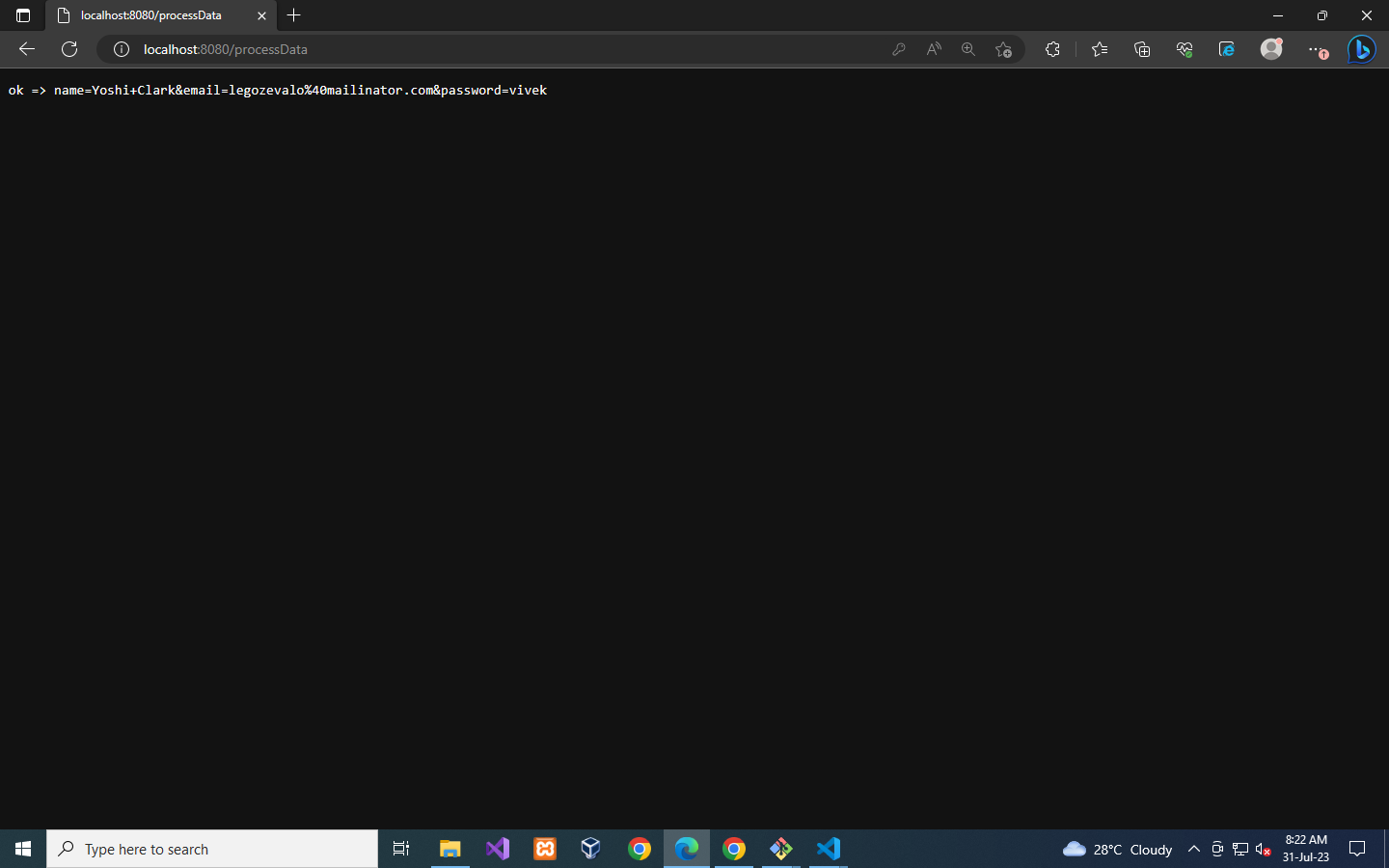
    console.log("Server starts on port 8080 port: http://localhost:8080")

})









2. Develop nodejs application with following requirements:

- Develop a route "/gethello" with GET method. It displays "Hello NodeJS!!" as response.

- Make an HTML page and display.

- Call "/gethello" route from HTML page using AJAX call. (Any frontend AJAX call API can be

used.)

Program :

const http = require('http');

const url = require('url');

const static = require('node-static')

// var fileserver = static.Server('static')

// const server = http.createServer((req,res)=>{

//     var url2 = url.parse(req.url,true);

//     if(url2.pathname=="/"){

//         fileserver.serve(req,res);

//     }else if(url2.pathname == "/getHello"){

//         res.end("Hello Node js..!!");

//     }else{

//         res.end("Do nothing")

//     }

// })

// server.listen(8080,()=>{

//     console.log("http://localhost:8080");

// })

var fileserver =new static.Server('static');

const server = http.createServer((req, res) => {

   var url2 = url.parse(req.url, true);

   if (url2.pathname == "/") {

      fileserver.serve(req, res);

   } else if (url2.pathname == "/getHello" && req.method=="GET") {

      res.end("Hello Node js..!!");

      // res.end();

   } else {

      res.end("Do nothing");

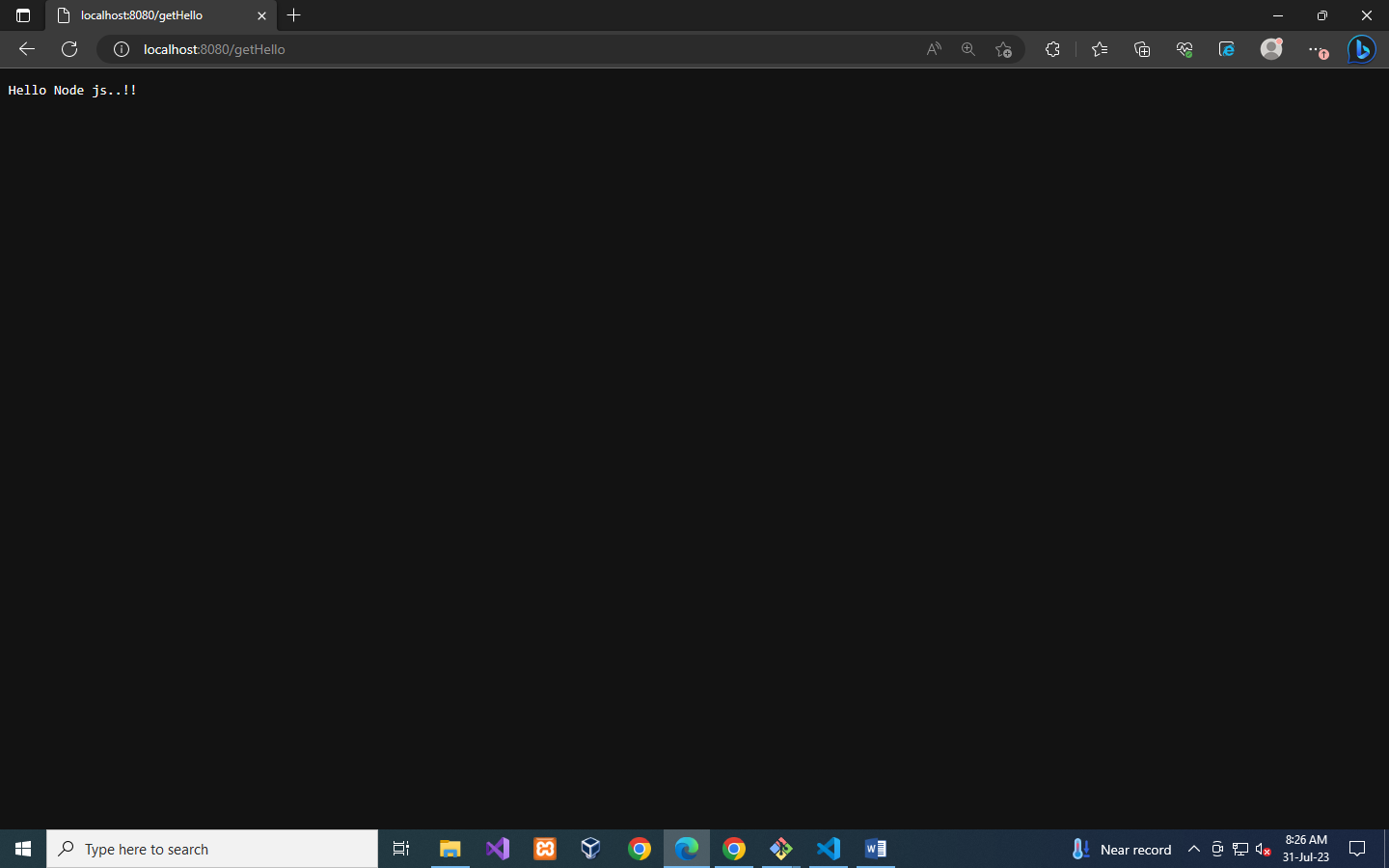
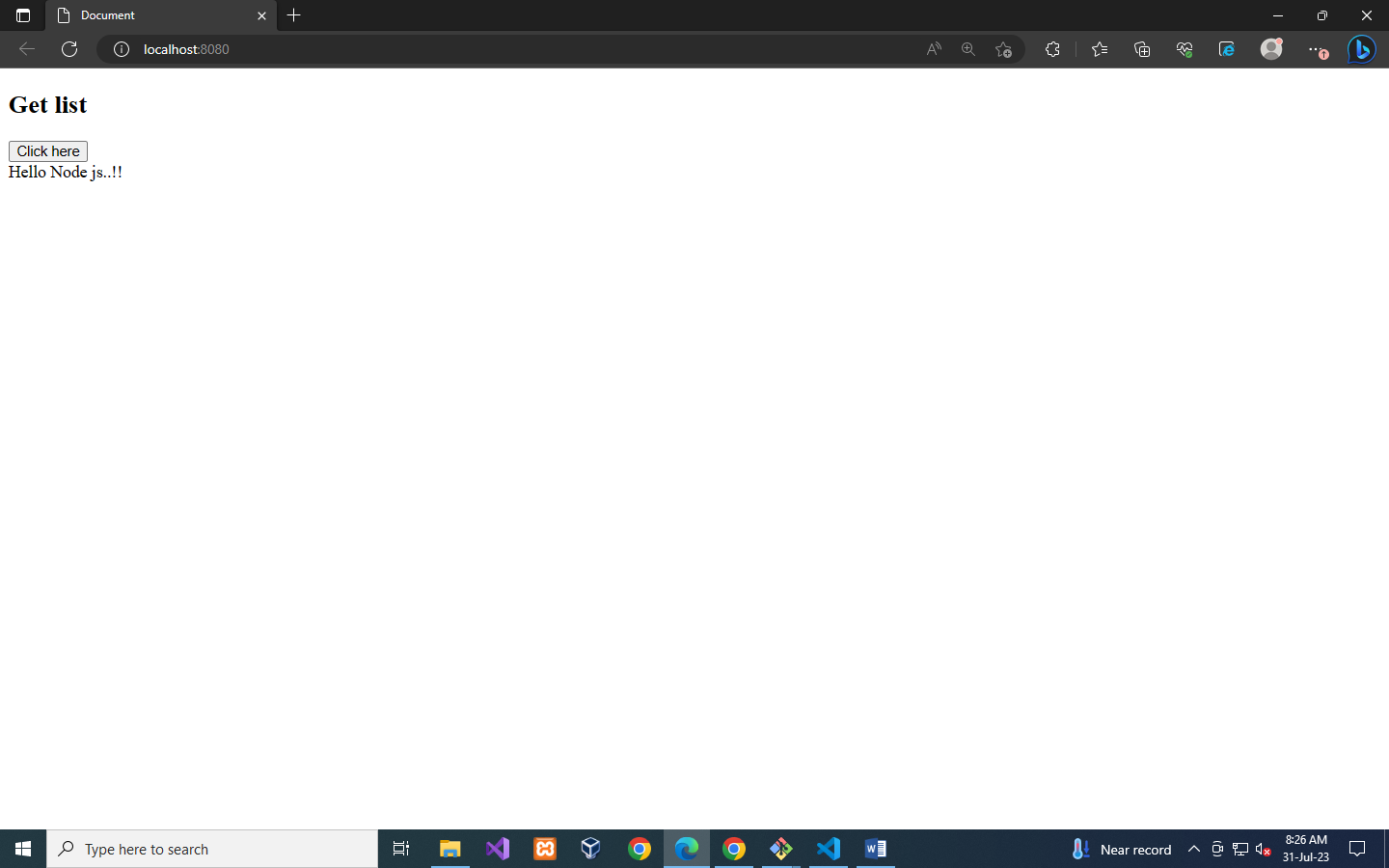
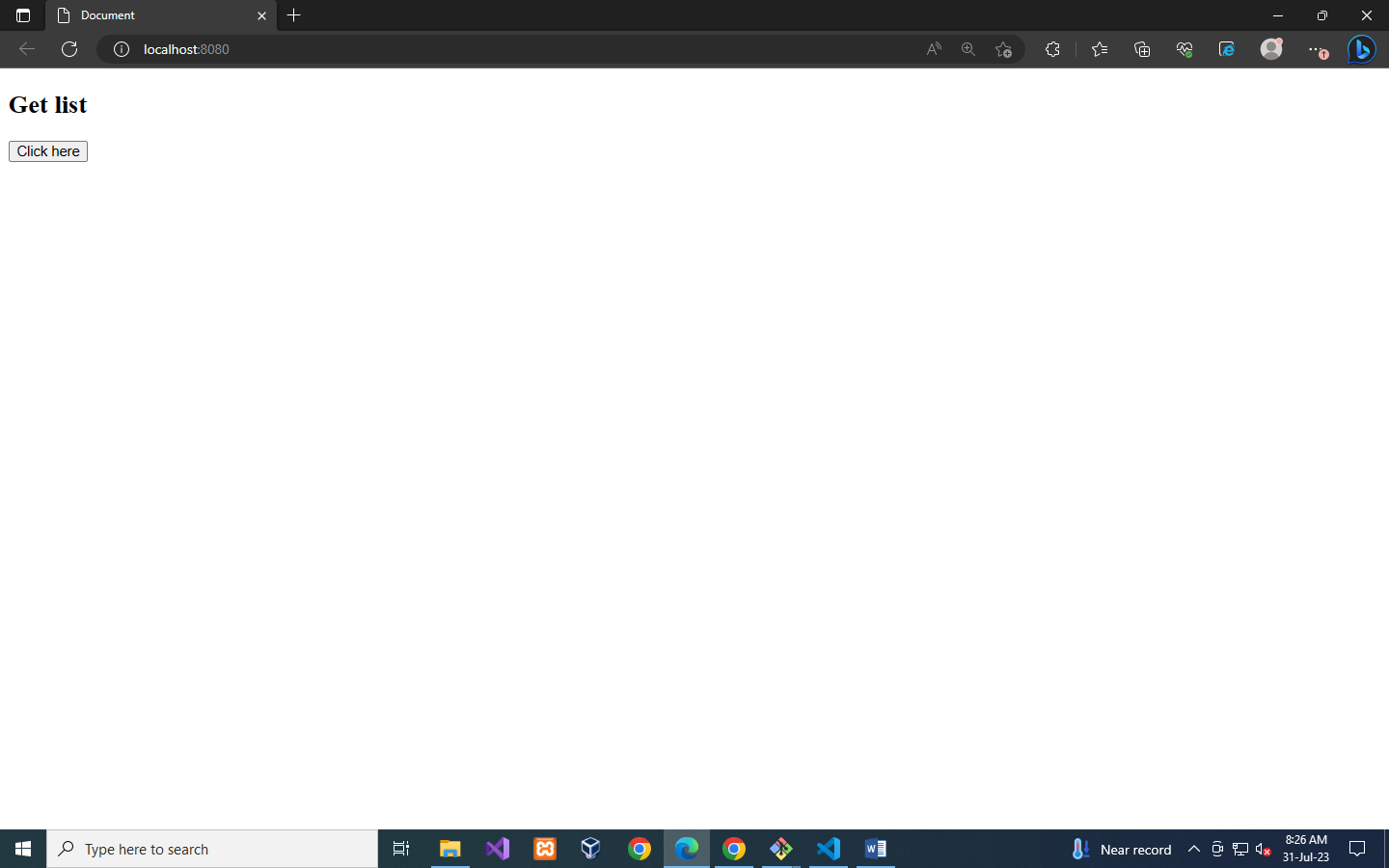
   }

});

server.listen(8080, () => {

    console.log("http://localhost:8080");

 });



1. Develop a module for domain specific chatbot and use it in a command line application.

Program :

App.js

var chatbot = require('./chat');

var readline = require('readline');

var r1 = readline.createInterface(process.stdin,process.stdout);

r1.setPrompt("You==> ");

r1.prompt();

r1.on('line',function(message){

    console.log('Bot==> '+chatbot.chatbotreply(message));

    r1.prompt();

}).on('close',function(){

    process.exit(0);

})

Chat.js

module.exports.chatbotreply = function(message){

    this.age = 25;

    this.name = 'chatGpt';

    this.university = 'vnsgu';

    this.country = 'india';

    message = message.toLowerCase()

    if(message.indexOf('hi')  > -1){

        return "hii,How are you ..!!";

    }else if(message.indexOf('fine')  > -1){

        return "Thats greate..!!\n How can i help you..!!";

    }else if(message.indexOf('can you tell me about your self ?')  > -1){

        return "Sure ..! , I am a chatBot \n, basically developed by Vivek lakhlani. My age is "+this.age  + " Thats all about me.";

    }else if(message.indexOf('bye')  > -1){

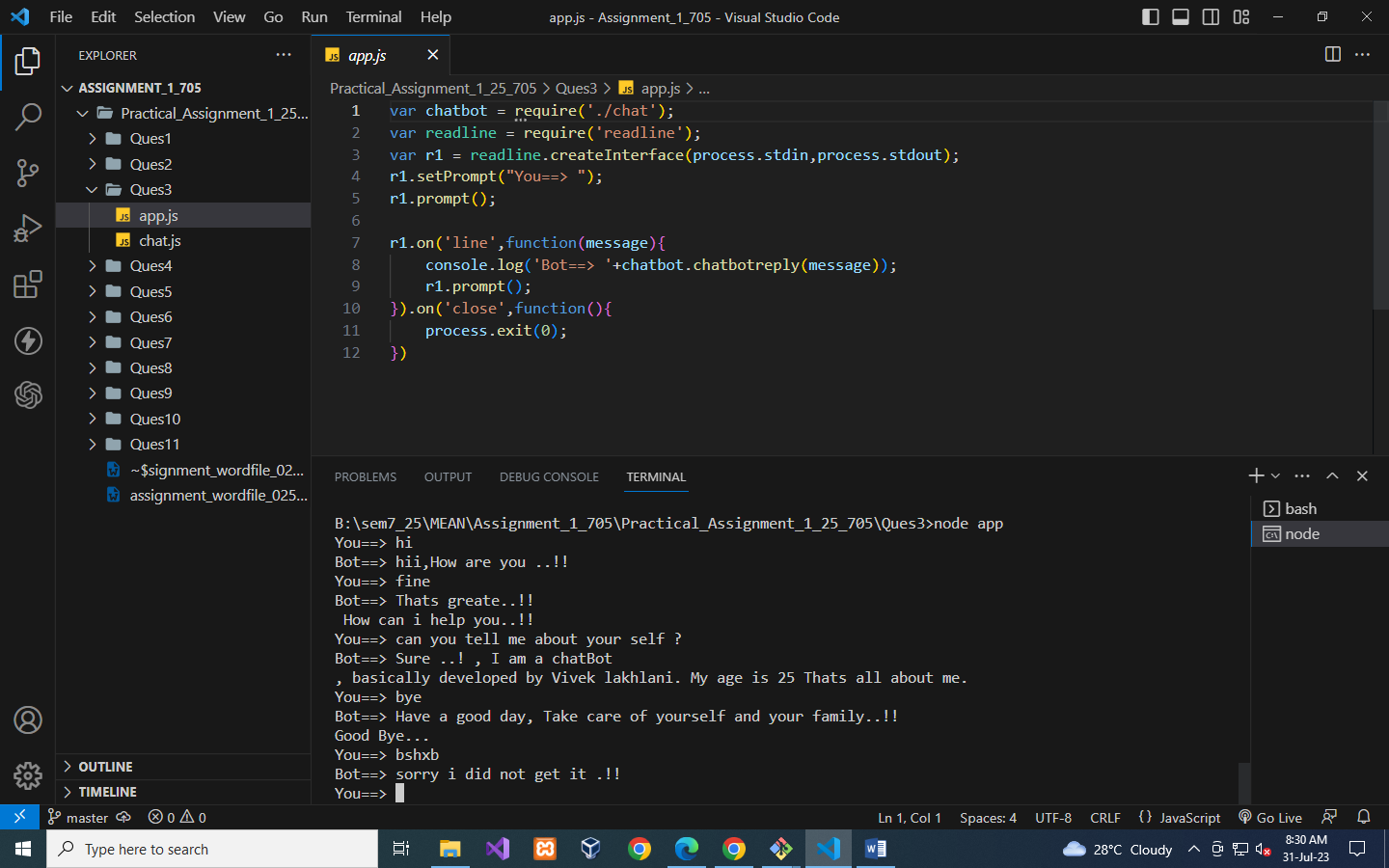
        return "Have a good day, Take care of yourself and your family..!!\nGood Bye...";

    }else{

        return "sorry i did not get it .!!";

    }

}



1. Use above chatbot module in web based chatting of websocket.

Program :

const websocket = require('ws');

const http = require('http');

const url = require('url');

const st = require('node-static');

const fileserver = new st.Server('./public');

const httpserver = http.createServer((req,res)=>{

    req.on('end',()=>{

        var get  = url.parse(req.url,true).query;

        fileserver.serve(req,res);

    }).resume();

}).listen(8080,()=>{

    console.log("http://localhost:8080");

})

const wss = new websocket.Server({server:httpserver});

wss.on('connection',(ws)=>{

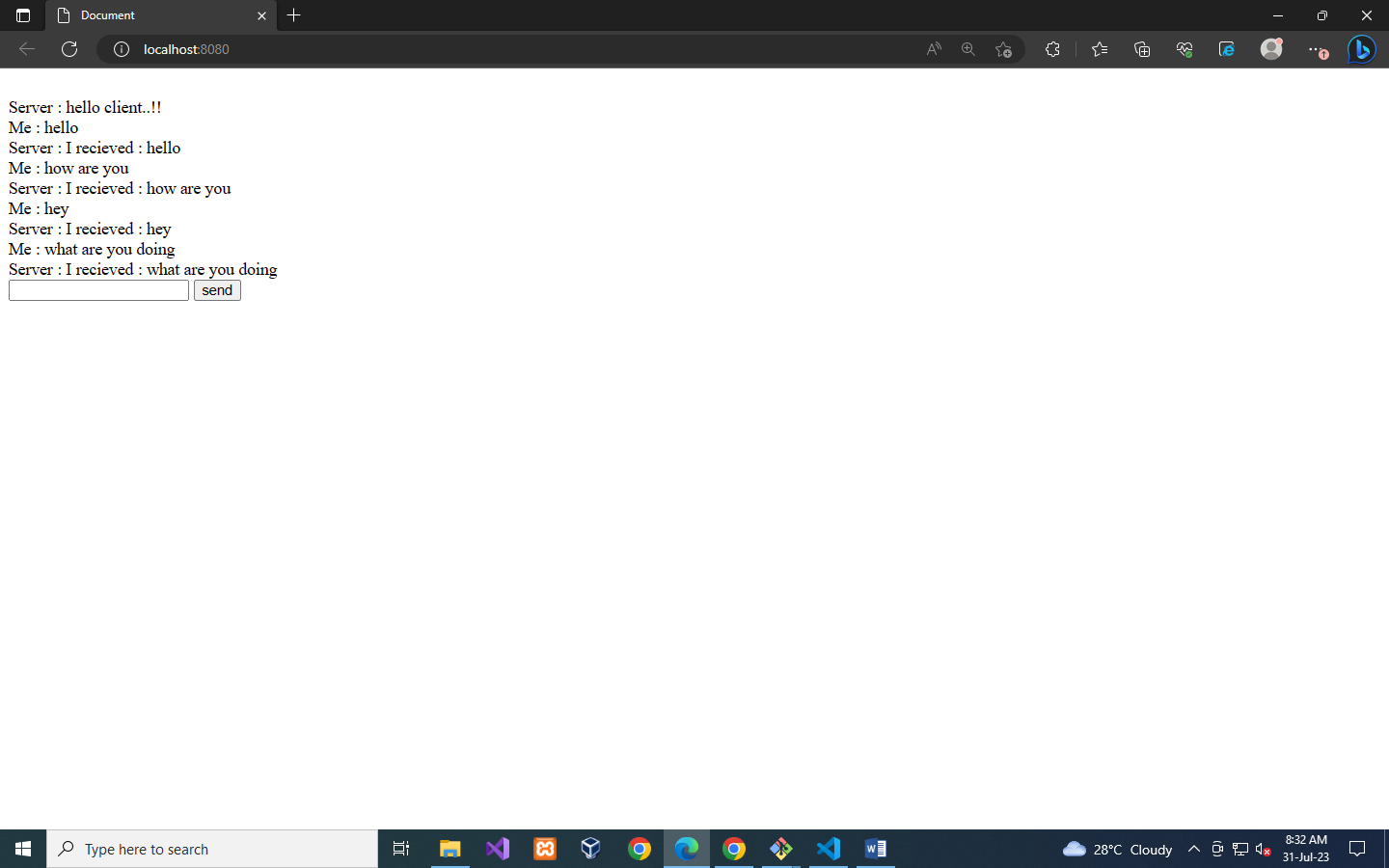
    ws.send("hello client..!!");

    ws.on('message',messgae=>{

        ws.send('I recieved : ' + messgae)

    })

})



1. Write a program to create a compressed zip file for a folder.

Program :

const zlib=require('zlib');

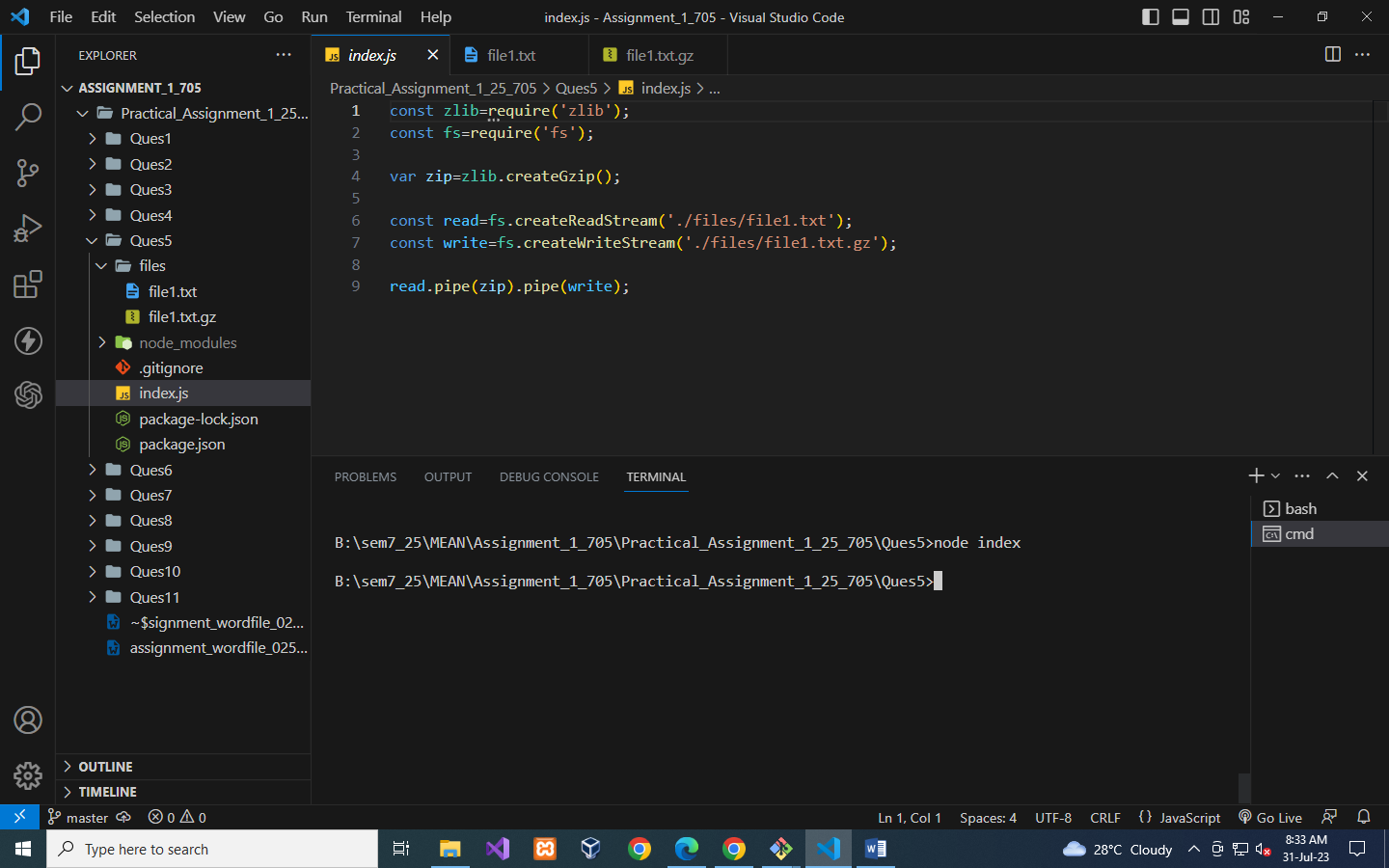
const fs=require('fs');

var zip=zlib.createGzip();

const read=fs.createReadStream('./files/file1.txt');

const write=fs.createWriteStream('./files/file1.txt.gz');

read.pipe(zip).pipe(write);



1. Write a program to extract a zip file.

Program :

const zlib=require('zlib');

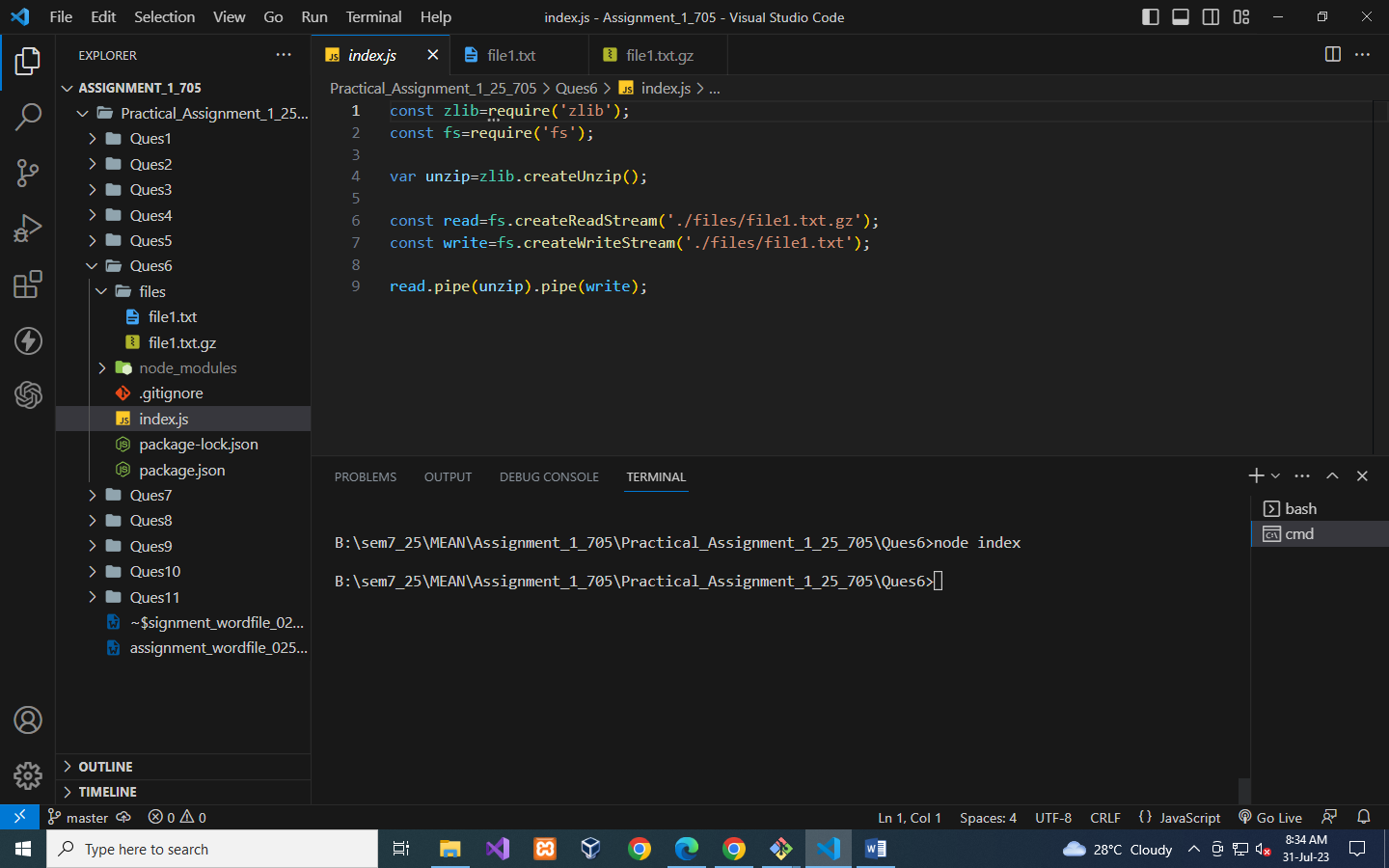
const fs=require('fs');

var unzip=zlib.createUnzip();

const read=fs.createReadStream('./files/file1.txt.gz');

const write=fs.createWriteStream('./files/file1.txt');

read.pipe(unzip).pipe(write);



1. Write a program to promisify fs.unlink function and call it.

Program :

const fs = require("fs")

const removeFile = (file\_path) => {

    return new Promise((resolve, reject) => {

        fs.unlink(file\_path, (err) => {

            if (err) {

                return reject(err)

            }

            else {

                return resolve('file removed successfully.')

            }

        })

    })

}

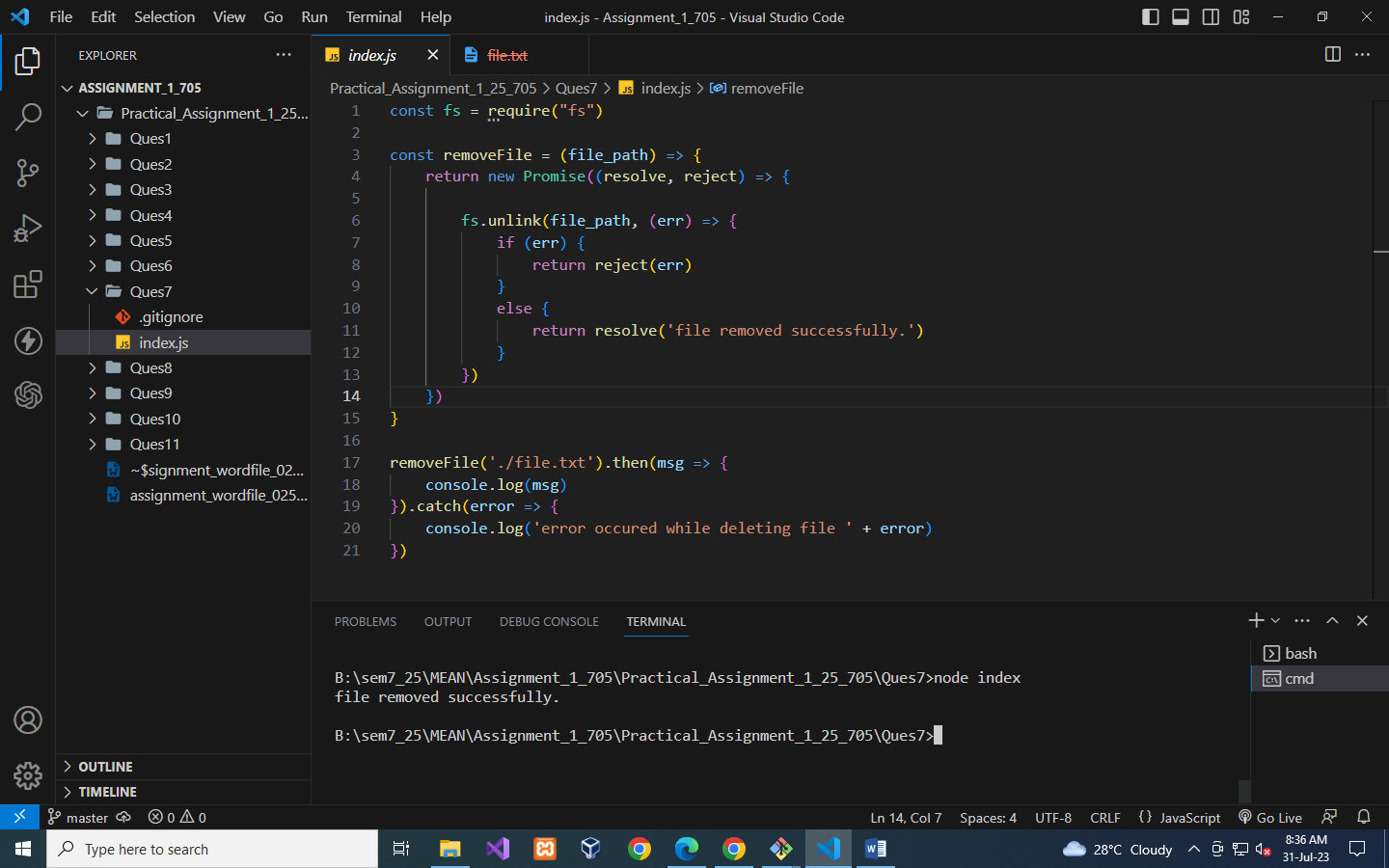
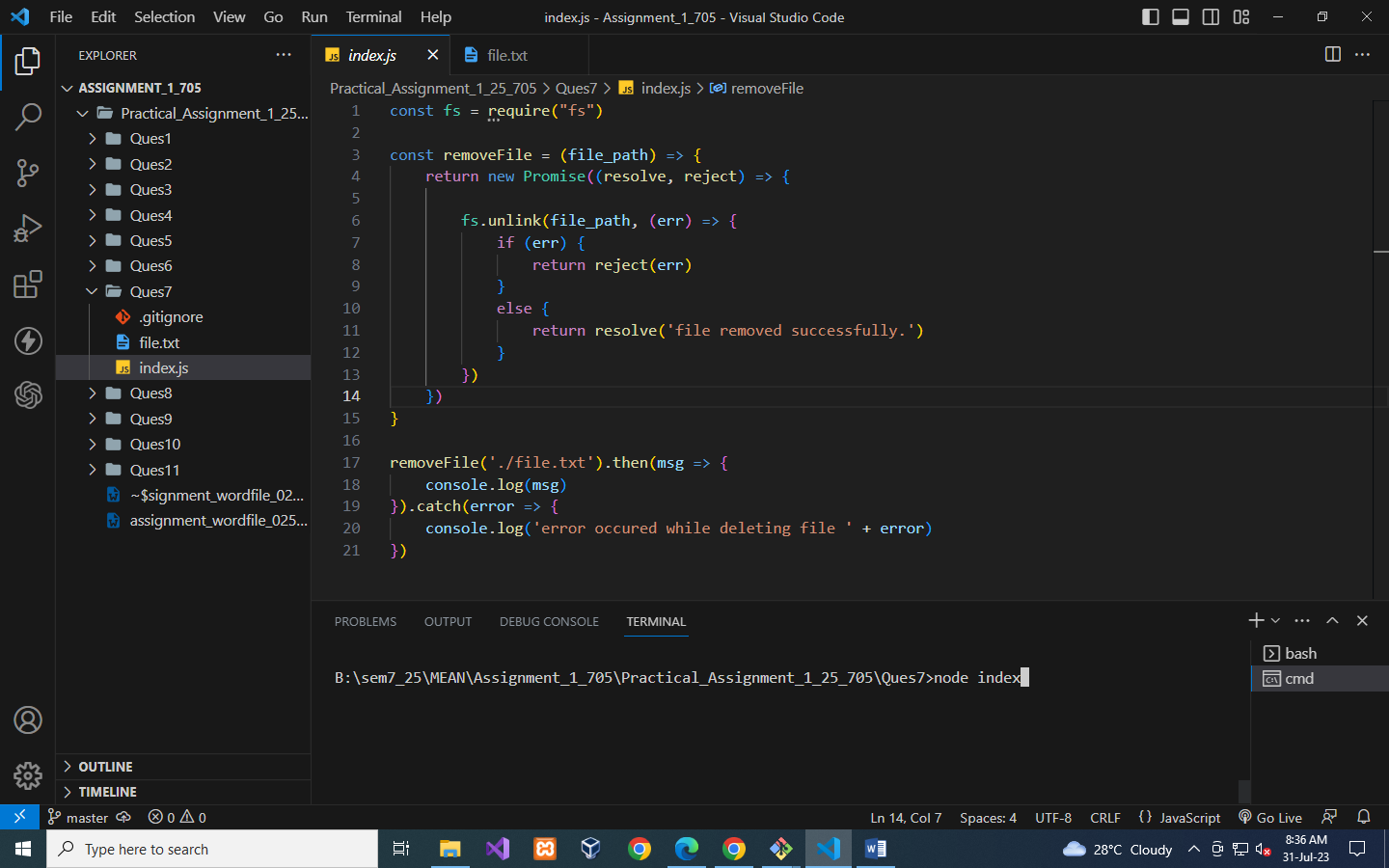
removeFile('./file.txt').then(msg => {

    console.log(msg)

}).catch(error => {

    console.log('error occured while deleting file ' + error)

})



1. Fetch data of google page using note-fetch using async-await model.

Program :

(async () => {

  try {

    const response = await fetch("https://www.google.com/");

    const text = await response.text();

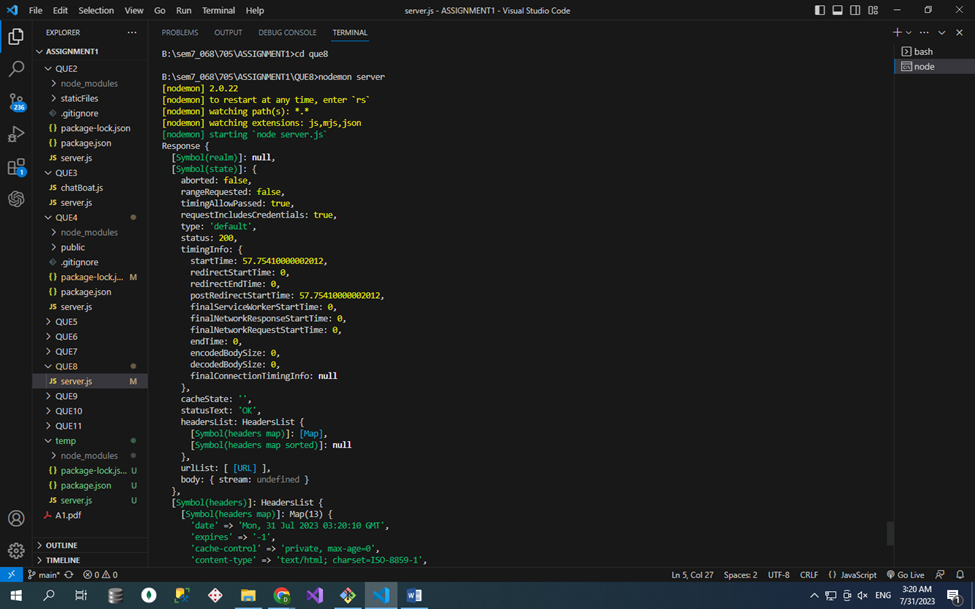
    console.log(text);

  } catch (error) {

    console.log(error.response.body);

  }

})();



9. Write a program that connect Mysql database, Insert a record in employee table and

display all records in employee table using promise based approach.

Program :

const http=require("http");

const mysql=require("mysql");

const static=require("node-static");

var fileserver=new static.Server("./public");

var conn=mysql.createConnection({

    host:"localhost",

    user:"root",

    password:"root",

    database:"employeedb"

});

conn.connect((err)=>{

    if(err){

        console.log(err);

    }else{

        console.log("connected")

    }

})

async function getData(){

}

var server=http.createServer((req,res)=>{

    console.log(req.url);

    if(req.url=="/"){

        fileserver.serve(req,res);

    }

    if(req.url=="/getData"){

        conn.query("SELECT \* FROM `emptb`",(err,data)=>{

            if(err){

                return "err";

            }

            res.end(JSON.stringify(data));

        })

    }

    if(req.url=="/insert\_emp\_data" && req.method==="POST"){

        let data = '';

        req.on('data', (chunk) => {

            data += chunk;

        });

        req.on("end",()=>{

            var fd=JSON.parse(data);

            // console.log(fd.name)

            var sql=`INSERT INTO emptb(emp\_name, emp\_email, emp\_pwd) VALUES ('${fd.ename}','${fd.eEmail}','${fd.epwd}')`;

            conn.query(sql,(err,data)=>{

                if(err){

                    console.log(err);

                }else{

                    res.end("success");

                }

            })

        })

        // res.end();

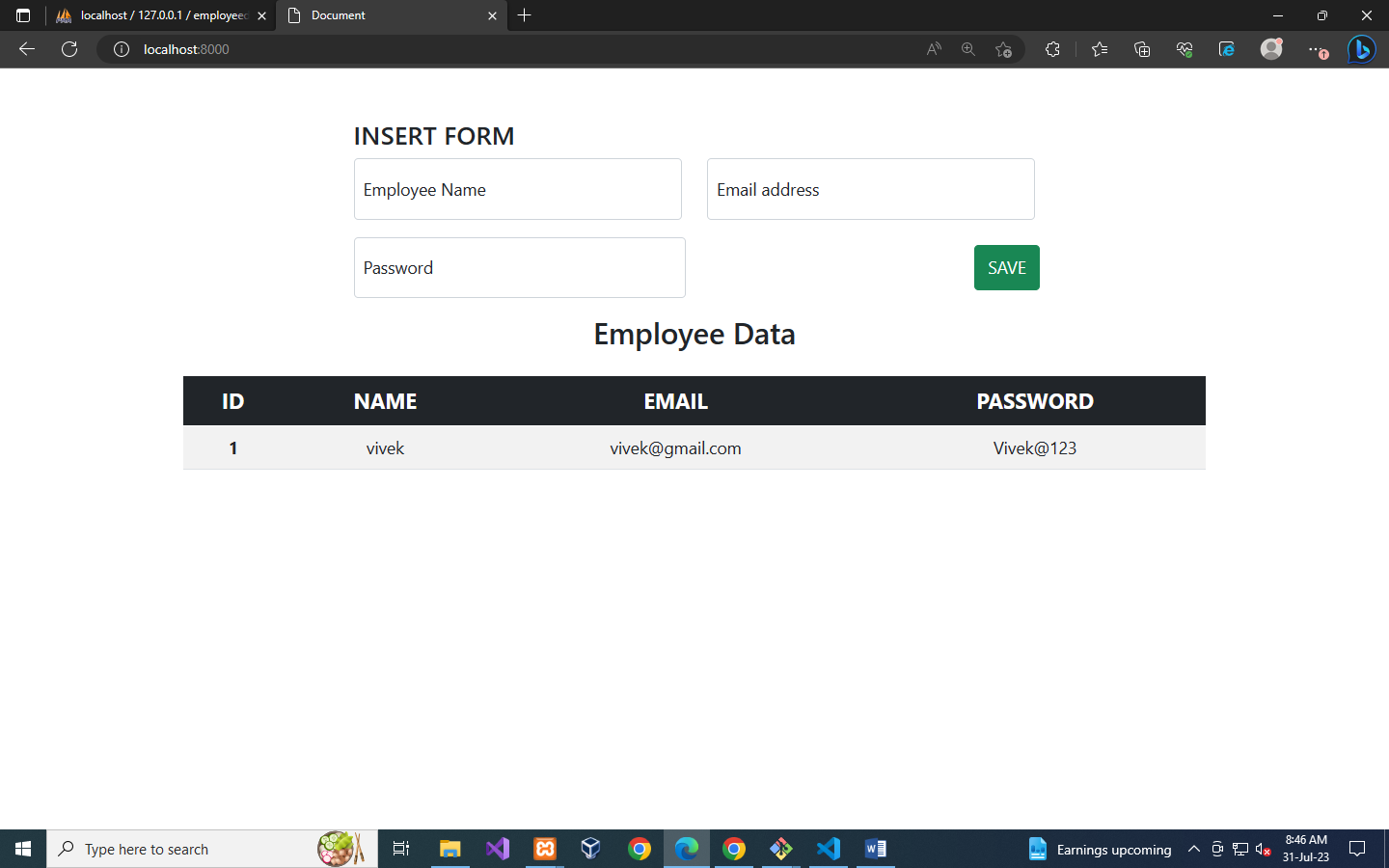
    }

})

server.listen(8000,()=>{

    console.log("server listening on port 8000");

})



10.Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs.

Program :

server 1:

var http=require("http");

var server=http.createServer((req,res)=>{

    console.log("server1")

})

server.listen(8000,()=>{

    console.log("server listening on port 8000");

})

Server2 :

var http=require("http");

var server=http.createServer((req,res)=>{

    console.log("server2")

})

server.listen(8080,()=>{

    console.log("server listening on port 8080");

})

server3 :

var http=require("http");

var server=http.createServer((req,res)=>{

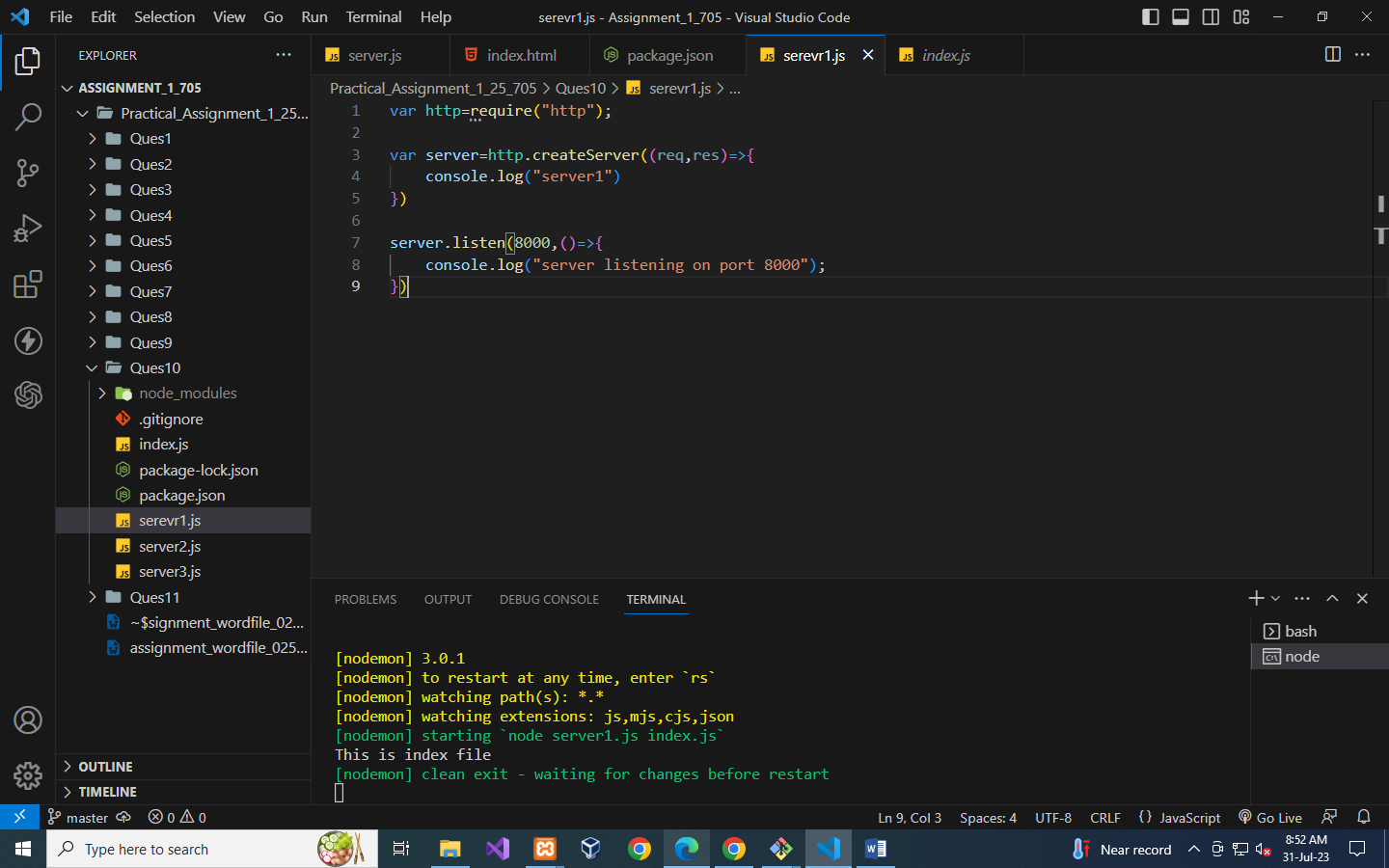
    console.log("server3")

})

server.listen(3000,()=>{

    console.log("server listening on port 3000");

})



11. Develop an application to show live cricket score.

Program :

const axios = require("axios");

const http = require("http");

const static = require("node-static");

const url = require("url");

const websocket = require("ws");

var fileServer = new static.Server("./public");

var server = http.createServer((req, res) => {

    fileServer.serve(req, res);

})

var latestData = null;

server.listen(8000, () => {

    console.log("server listening on port 8000");

})

async function fetchMatchScore() {

    try {

        var response = await axios.get("https://api.cricapi.com/v1/currentMatches?apikey=0bf9e0f5-5333-4925-912f-5a5511d62c19&offset=0");

        return response.data;

    } catch (err) {

        console.log(err)

    }

}

var wss = new websocket.Server({ server: server });

wss.on("connection", async (ws) => {

    var data = await fetchMatchScore();

    ws.send(JSON.stringify(data));

})

async function updateDataAndBroadcast() {

    latestData = await fetchMatchScore();

    if (latestData !== null) {

        wss.clients.forEach((client) => {

            if (client.readyState === websocket.OPEN) {

                client.send(JSON.stringify(latestData));

            }

        });

    }

}

setInterval(updateDataAndBroadcast, 5000);

