PRACTICAL ASSIGNMENT-1

NAME: KRISHI TAILOR

ROLL NO: 75

SEMESTER: 7TH

SUBJECT:

GITHUB LINK:

1. Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.

- Handle POST request.

Server.js—

const http = require('http');

const fs = require('fs');

const url = require('url');

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

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

    if(req.url == '/')

    {

        var path = "./Que-1/index.html";

        fs.readFile(path, (err, data) => {

        if(err)

        {

            res.writeHead(404, {'Content-Type': 'text/html'});

            return res.end("404 page not found...");

        }

        res.writeHead(200, {'Content-Type': 'text/html'});

        res.write(data);

        res.end();

        });

    }

    else if(ul.pathname == '/process' && req.method == 'GET')

    {

        res.write("FIRST NAME: " + ul.query.fname + " \nLAST NAME: " +

        ul.query.lname);

        res.end();

    }

    else if(ul.pathname == '/process' && req.method == 'POST')

    {

        let body = '';

        req.on('data', chunk => {

        body += chunk.toString();

    });

    req.on('end', () => {

    res.end(body);

    });

    }

});

server.listen(8080);

console.log("The Server is Running on 8080");

index.html—

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-

scale=1.0">

<title>My-Form</title>

</head>

<body>

<h1>GET Request</h1>

<form action="/process" method="GET">

FIRST NAME <input type="text" name="fname"/><br/><br/>

LAST NAME <input type="text" name="lname"/><br/><br/>

<input type="submit" value="Submit"/><br/><br/>

</form>

<h1>POST Request</h1>

<form action="/process" method="POST">

FIRST NAME <input type="text" name="fname"/><br/><br/>

LAST NAME <input type="text" name="lname"/><br/><br/>

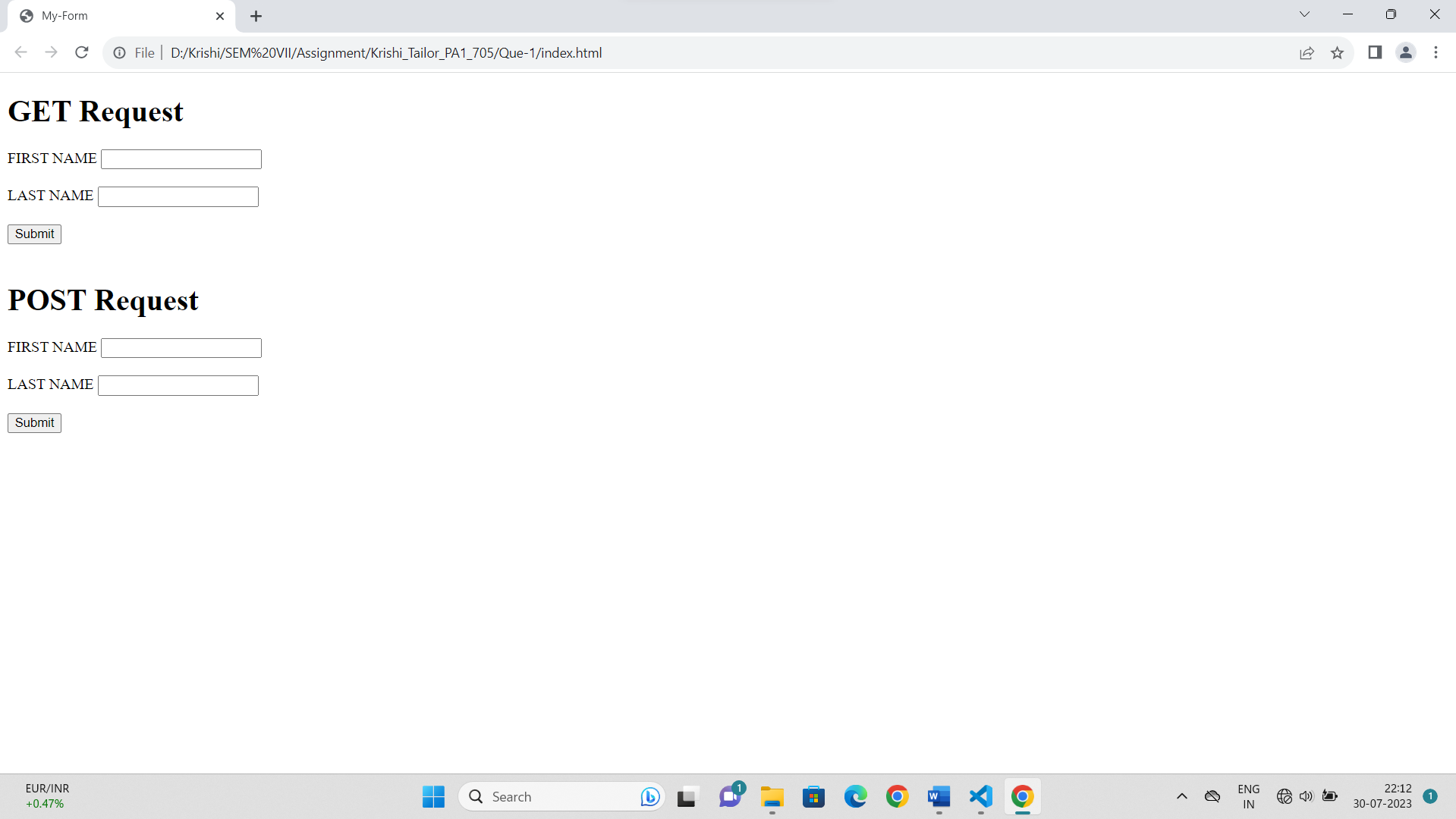
<input type="submit" value="Submit"/>

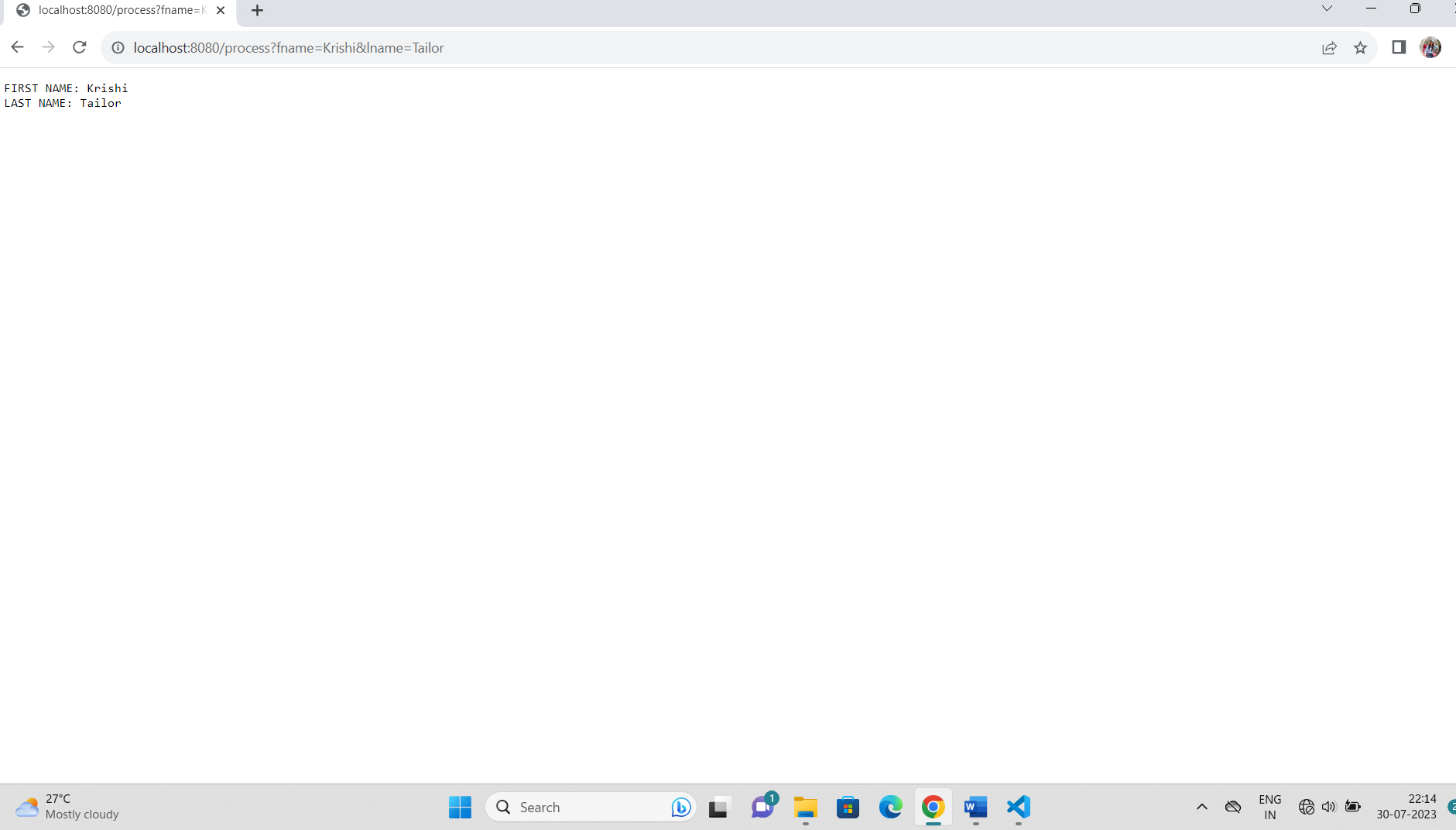
</form>

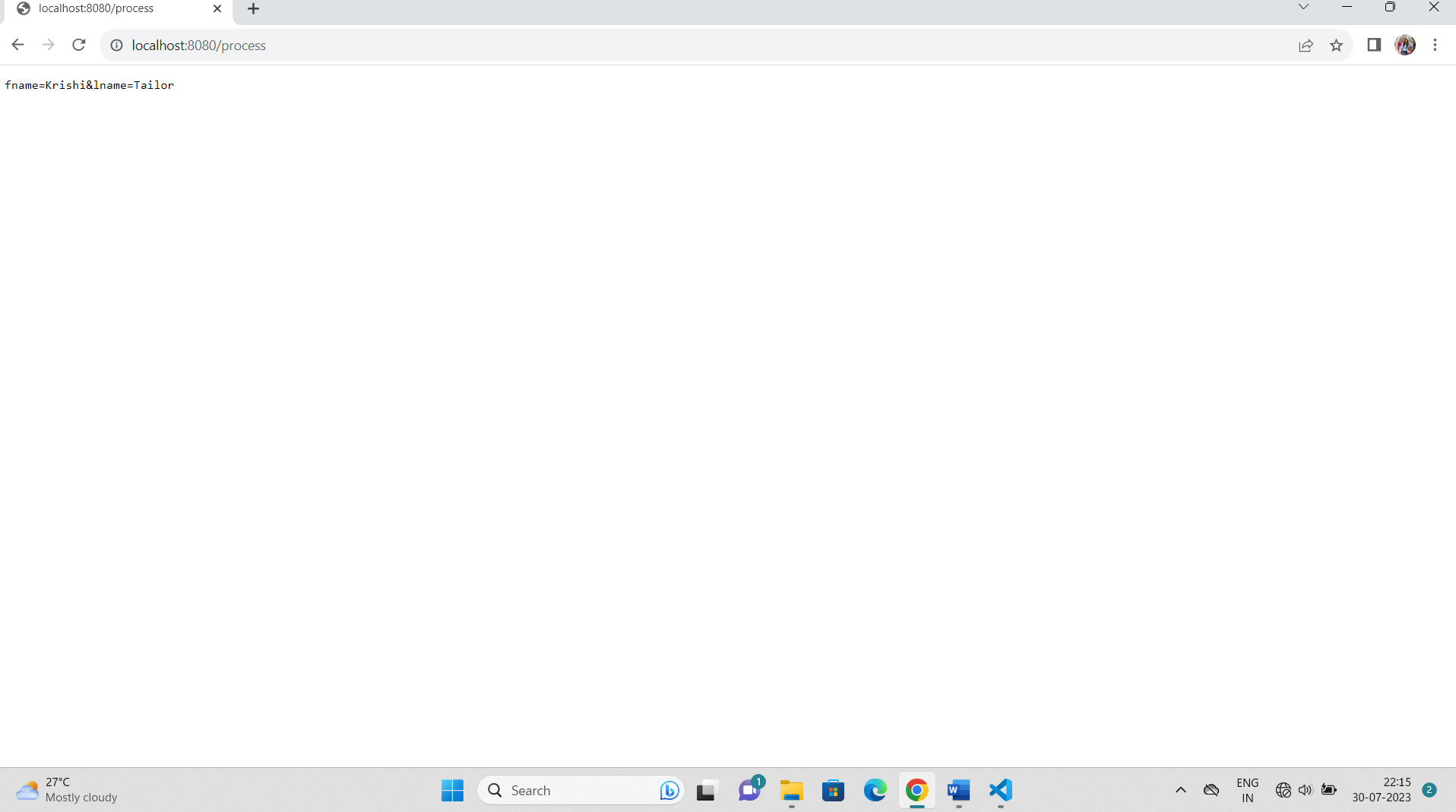
</body>

</html>

Output—







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.)

server.js—

const http = require('http');

const fs=require('fs');

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

if (req.method === 'GET') {

if (req.url === '/') {

res.end("Home Page");

}

if (req.url === '/gethello') {

fs.readFile('./public/hello.html',(err,data)=>{

if(err)

{

return res.send("Something went wrong!!");

}

else{

res.writeHead(200,{

'Content-Type':'text/html'});

res.write(data);

return res.end();

}

})

}

if (req.url === '/ajaxcall') {

fs.readFile('./public/ajaxcall.html',(err,data)=>{

if(err)

{

return res.send("Something went wrong!!");

}

else{

res.writeHead(200,{

'Content-Type':'text/html'});

res.write(data);

return res.end();

}

})

}

}

}).listen(8000, () => {

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

})

hello.html—

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-

scale=1.0">

<title>Document</title>

</head>

<body>

<h4>Hello NodeJS!!</h4>

</body>

</html>

ajaxcall.html—

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-

scale=1.0">

<title>Document</title>

</head>

<body>

<div id="page\_content">

</div>

<button onclick="loadData()">Fetch Page</button>

<script>

function loadData() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function () {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("page\_content").innerHTML =

this.responseText;

}

};

xhttp.open("GET", "/gethello", true);

xhttp.send();

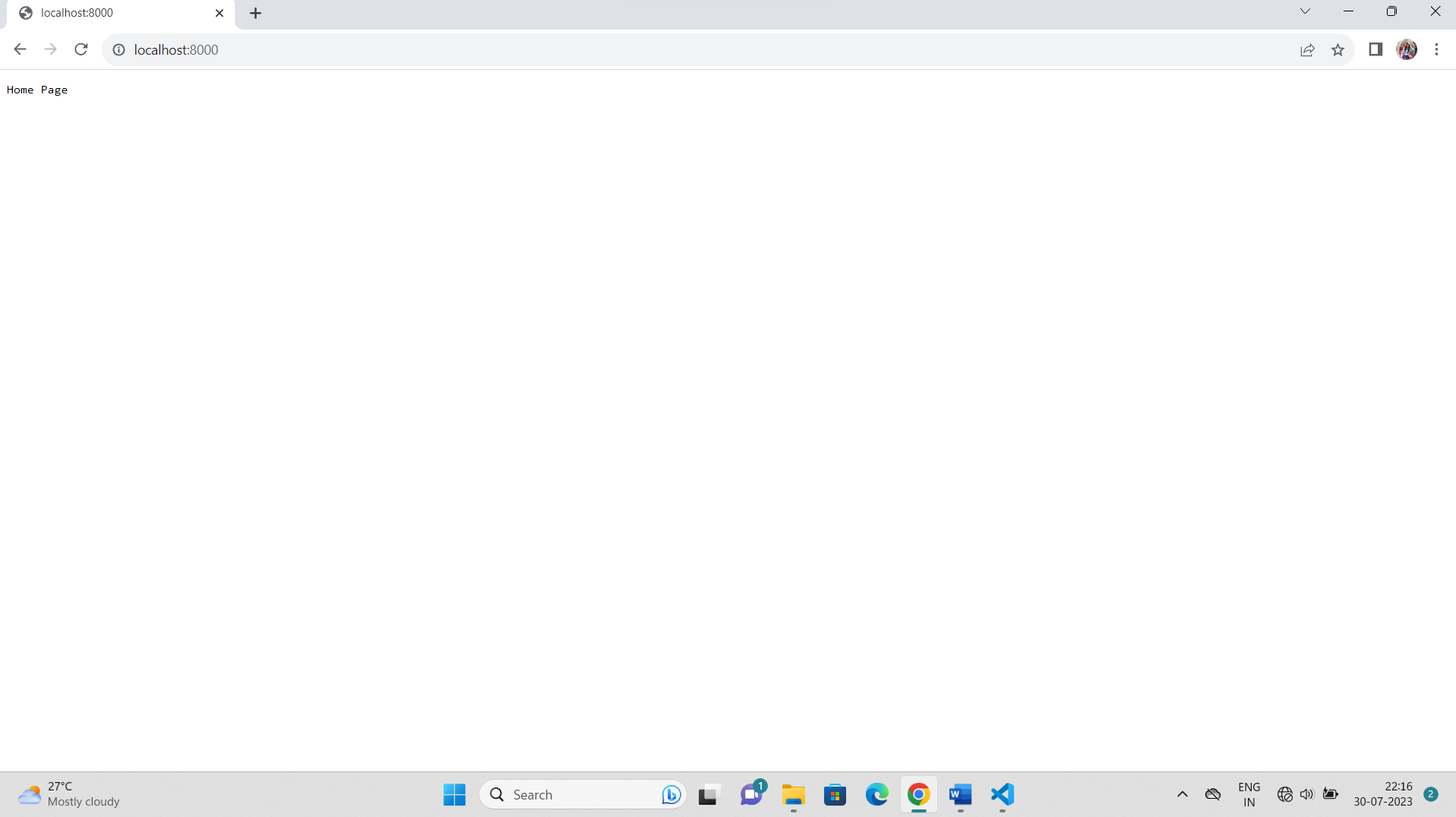
}

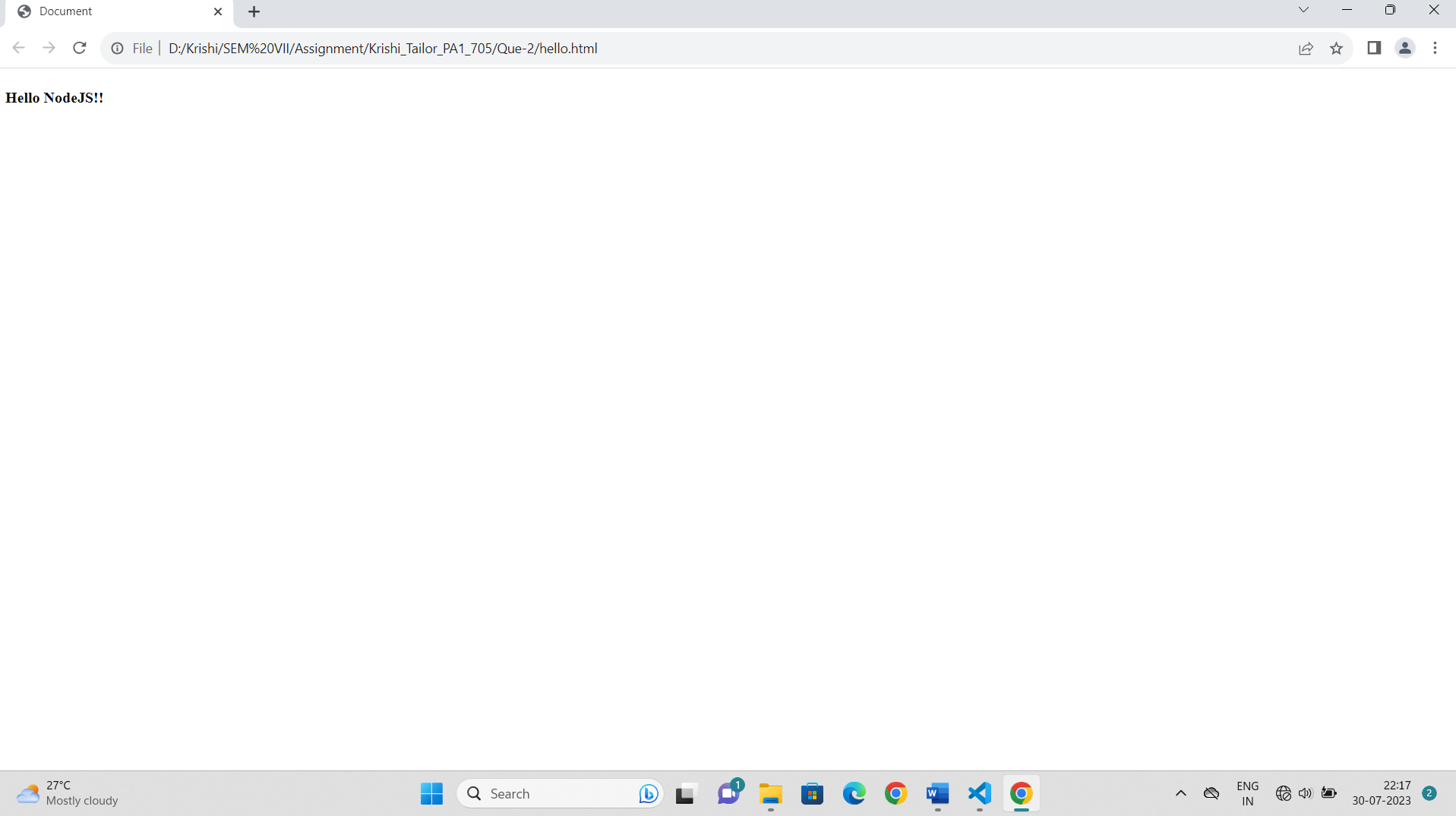
</script>

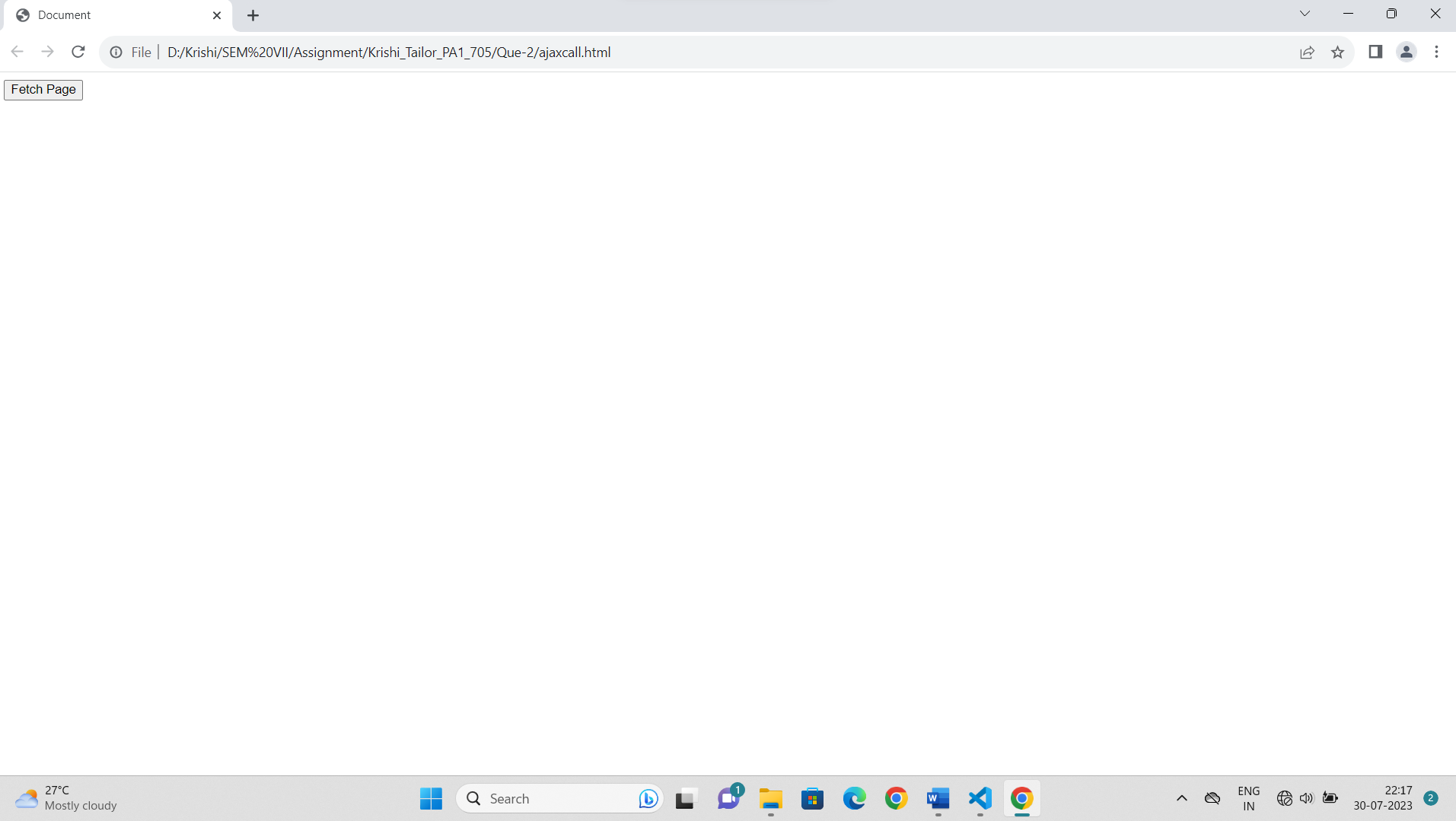
</body>

</html>

Output—







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

Bot.js—

const readline = require('readline');

var rl = readline.createInterface({input: process.stdin, output:

process.stdout, terminal: false});

const app = require('app-for-chat');

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

Server.js—

const WebSocket = require('ws')

var http = require('http');

var fs = require('fs');

var httpserver = http.createServer(function(request, response)

{

if(request.url=="/")

{

fs.readFile("./Que-4/index.html",(err,data)=>{

response.write(data)

response.end();

})

}

}).listen(8080, function() {

console.log((new Date()) +

' Server is listening on port 8080');

});

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

wss.on("connection",(clientws)=>{

clientws.send("Hello Client")

clientws.on("message",(msg)=>{

console.log("Received "+msg)

clientws.send("Received "+ msg)

})

})

Index.html—

<!DOCTYPE html >

<html>

<body>

<script language="javascript">

var ws = new WebSocket('ws://localhost:8080');

ws.addEventListener("message", function(msg1) {

var msg = msg1.data;

document.getElementById('chatlog').innerHTML+='<br>Server: '+ msg;

});

function sendMessage(){

var message = document.getElementById('message').value;

document.getElementById('chatlog').innerHTML+='<br> Me: '+ message;

ws.send(message);

}

</script>

<h2>Data from server</h2>

<div id="chatlog"></div>

<hr/>

<h2>Data from client</h2>

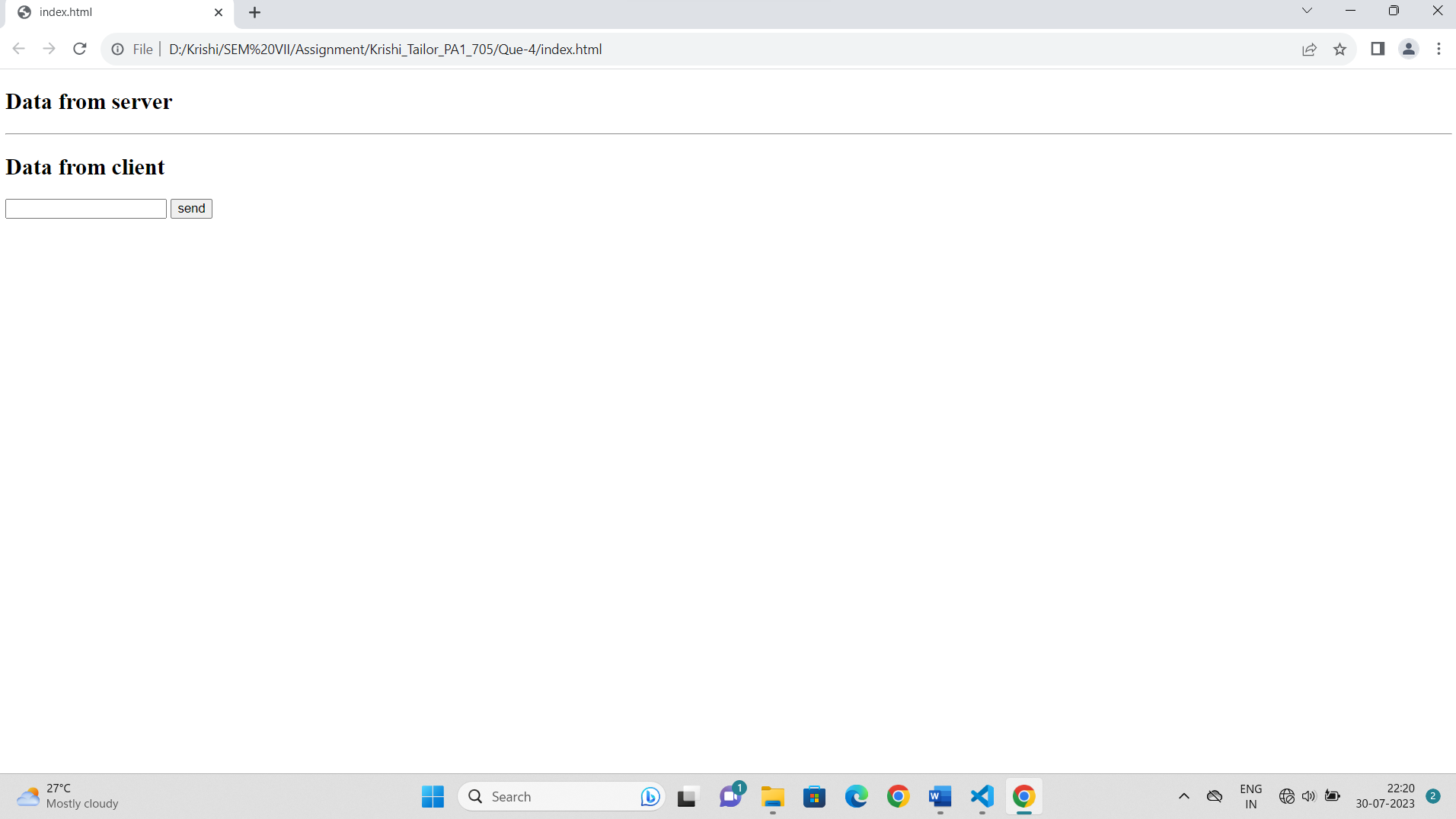
<input type="text" id="message" />

<input type="button" id="b1" onclick="sendMessage()"

value="send" />

</body>

</html>



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

App.js—

var fs = require("fs");

var zlib = require('zlib');

// Compress the file input.txt to input.txt.gz

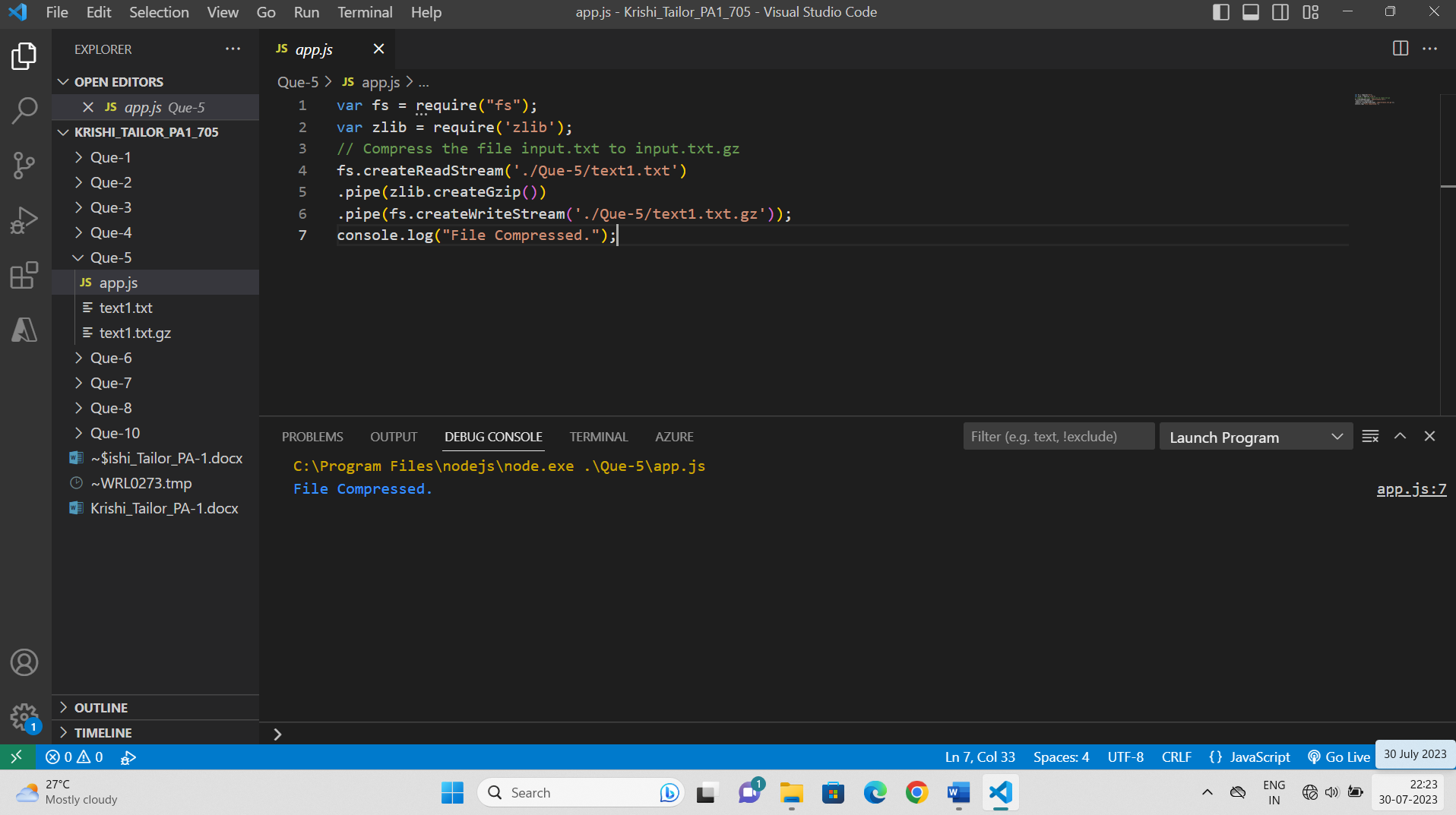
fs.createReadStream('./Que-5/text1.txt')

.pipe(zlib.createGzip())

.pipe(fs.createWriteStream('./Que-5/text1.txt.gz'));

console.log("File Compressed.");

Output—



6. Write a program to extract a zip file.

Index.js—

var fs = require("fs");

var zlib = require('zlib');

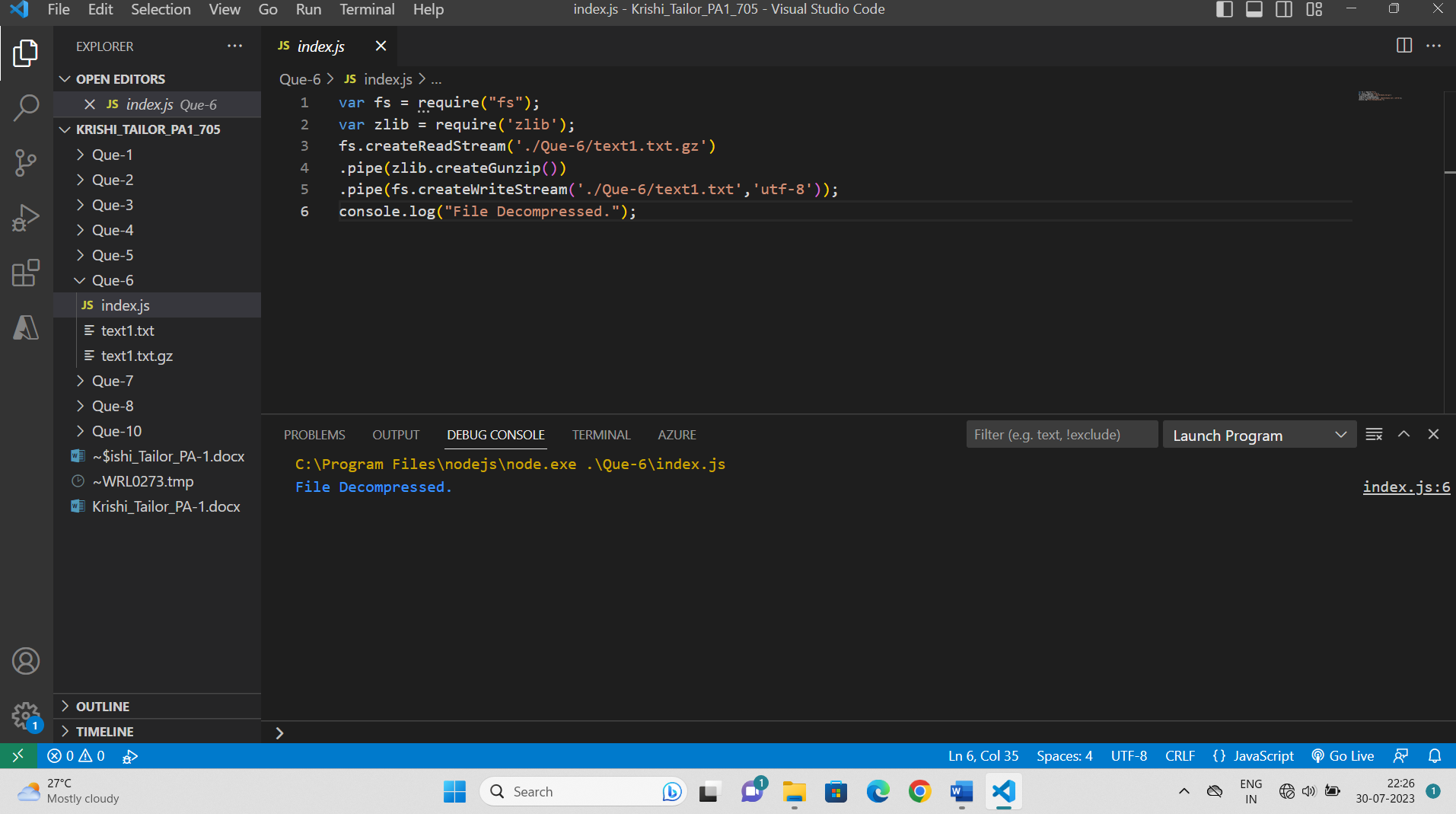
fs.createReadStream('./Que-6/text1.txt.gz')

.pipe(zlib.createGunzip())

.pipe(fs.createWriteStream('./Que-6/text1.txt','utf-8'));

console.log("File Decompressed.");

Output—



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

Main.js—

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('./Que-7/text1.txt').then(msg => {

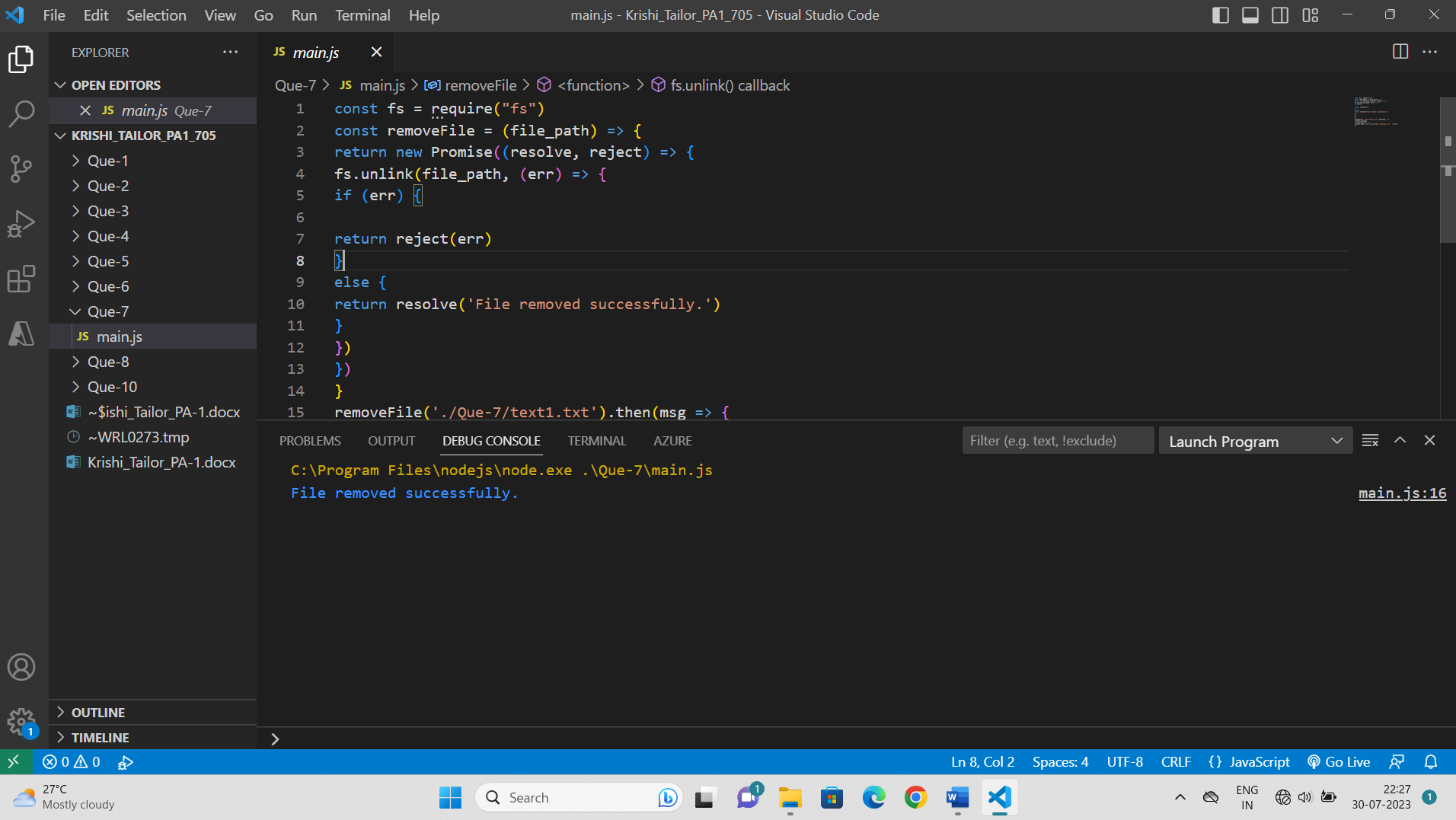
console.log(msg)

}).catch(error => {

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

})

Output—



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

Server.js—

import fetch from 'node-fetch'

globalThis.fetch = fetch

async function callfunc() {

fetch('https://google.com')

.then(res => res.text())

.then(text => printData(text));

}

callfunc();

function printData(res)

{

console.log(res);

}

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.

Serve.js—

import { createConnection } from "mysql";

var con = createConnection({

    host: "localhost",

    user: "root",

    password: "",

    database: 'empDB'

});

const selectAllEmployees = () => {

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

        con.query("SELECT \* FROM emptb", (err, result, fields) => {

            if (err) {

                reject(err);

            }

            else {

                resolve(result);

            }

        })

    })

}

con.connect((err) => {

    if (err) {

        console.log("error: " + err)

    } else {

        //inserting record in employee table

        con.query("INSERT INTO empTB values('Krishi','Tailor')", (err, result) => {

if (err) {

            console.log("error: " + err)

        } else {

            console.log("record inserted")

        }

    })

selectAllEmployees().then(result => {

    console.log(result)

}).catch(err => {

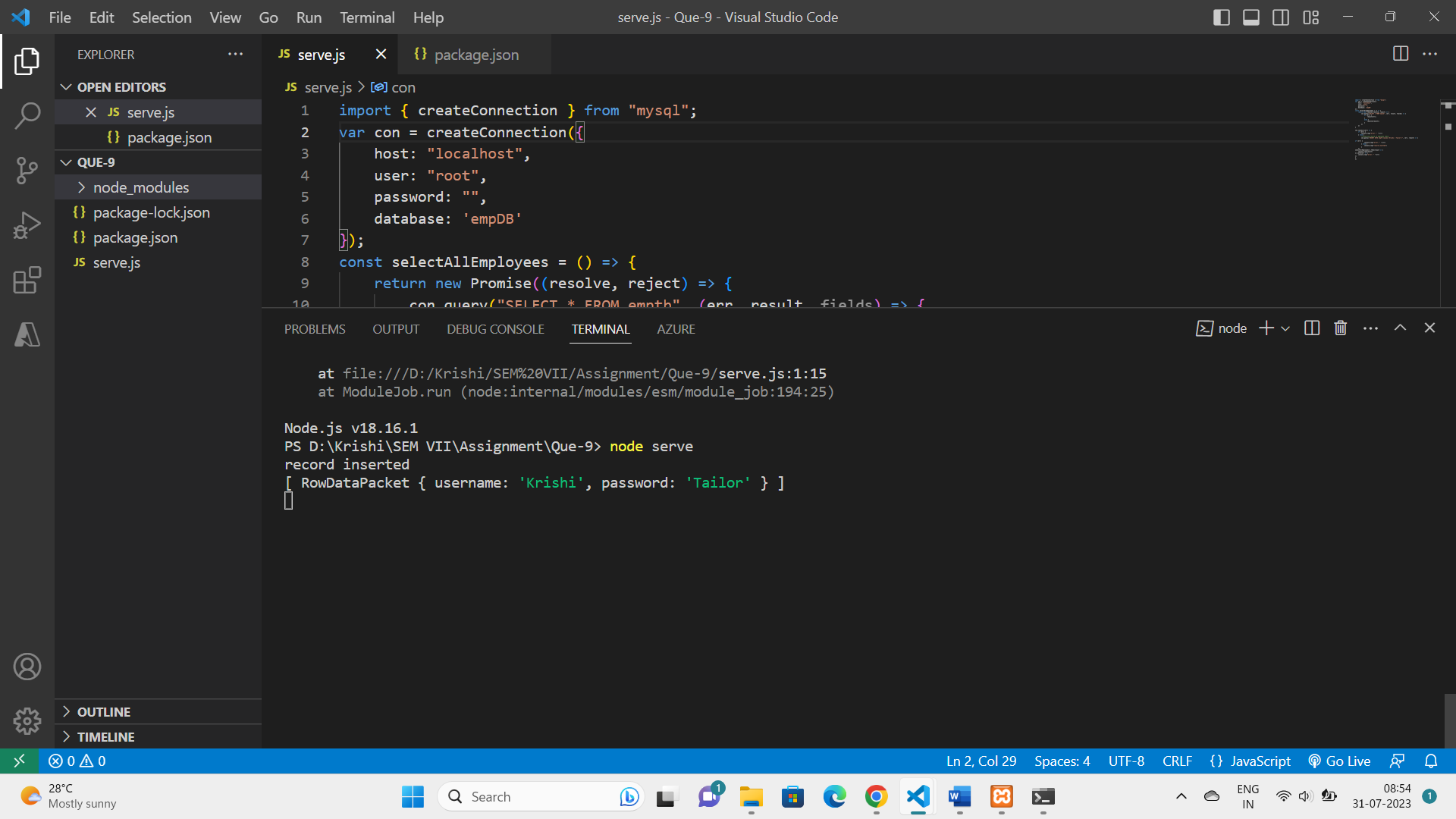
    console.log("error: " + err)

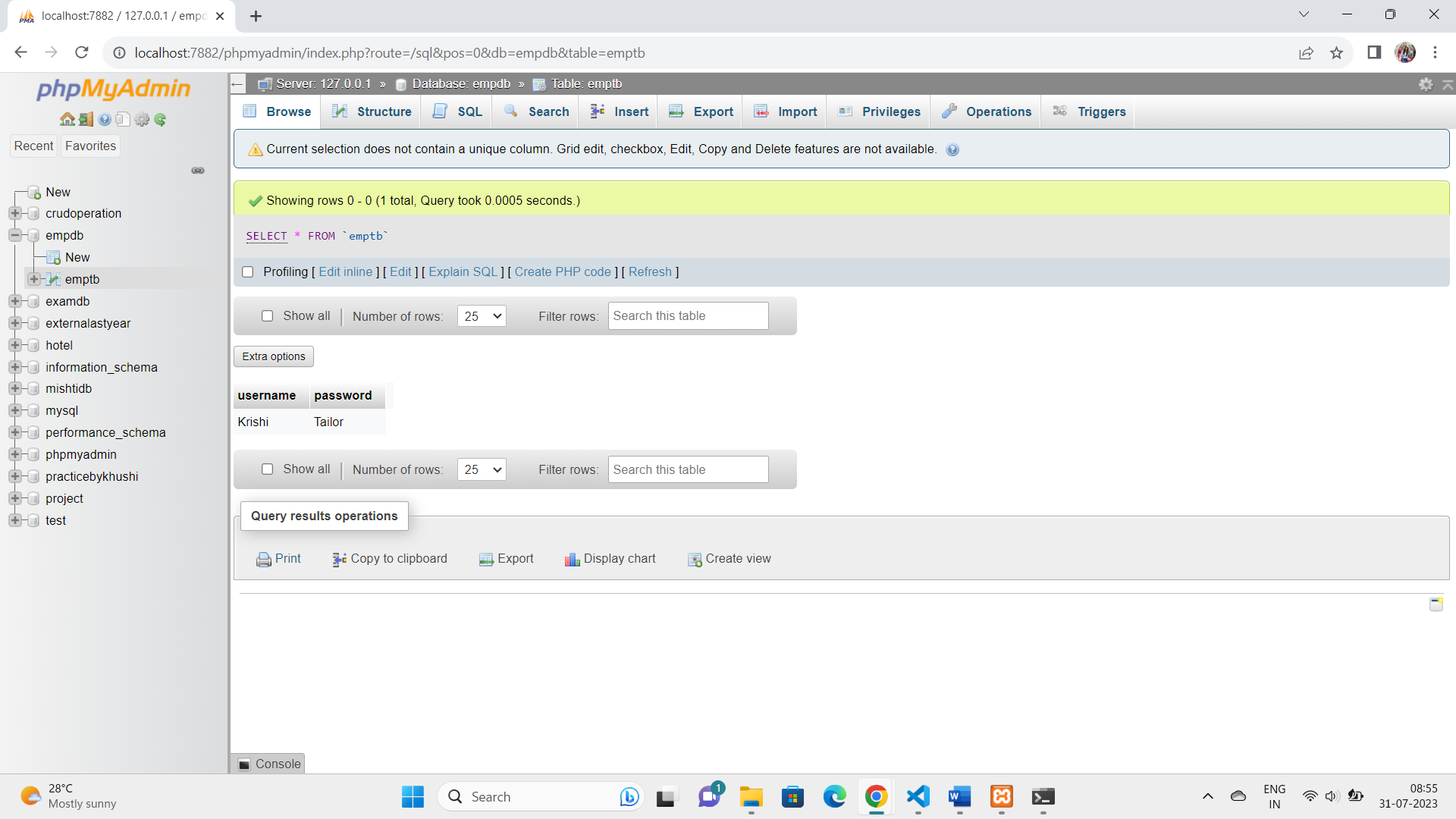
})

}

})

Output—





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

Server.js—

console.log("Hello from question 10");

Output—

