Name: Nidhi B. Valand

Roll no:72

Sub:701(Full Stack )

Assignment – 1

Que-1:

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>My Web Server</title>

</head>

<body>

    <h1>Welcome to My Web Server!</h1>

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

        <input type="text" name="data" placeholder="Enter some information" required>

        <button type="submit">Submit</button>

    </form>

</body>

</html>

………………………………………………………………………………………………………………………………………………

Server.js

const http = require('http');

const fs = require('fs');

const path = require('path');

const url = require('url');

const querystring = require('querystring');

const PORT = 3000;

// Function to serve static files

const serveStaticFile = (res, filePath, contentType) => {

    fs.readFile(filePath, (error, content) => {

        if (error) {

            res.writeHead(500);

            res.end(`Sorry, there was an error: ${error.code} ..\n`);

        } else {

            res.writeHead(200, { 'Content-Type': contentType });

            res.end(content, 'utf-8');

        }

    });

};

// Create the server

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

    const parsedUrl = url.parse(req.url, true);

    // Handle GET request

    if (req.method === 'GET') {

        if (parsedUrl.pathname === '/') {

            serveStaticFile(res, path.join(\_\_dirname, 'public', 'index.html'), 'text/html');

        } else if (parsedUrl.pathname === '/submit') {

            // Handle form submission here if needed

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

            res.end('Form submitted successfully!');

        } else {

            // Serve other static files

            const filePath = path.join(\_\_dirname, 'public', parsedUrl.pathname);

            const extname = String(path.extname(filePath)).toLowerCase();

            const mimeTypes = {

                '.html': 'text/html',

                '.js': 'text/javascript',

                '.css': 'text/css',

                '.json': 'application/json',

                '.png': 'image/png',

                '.jpg': 'image/jpg',

                '.gif': 'image/gif',

                '.svg': 'image/svg+xml',

                '.wav': 'audio/wav',

                '.mp4': 'video/mp4',

                '.woff': 'application/font-woff',

                '.ttf': 'application/font-ttf',

                '.eot': 'application/vnd.ms-fontobject',

                '.otf': 'application/font-otf',

                '.txt': 'text/plain',

                '.xml': 'application/xml',

                '.pdf': 'application/pdf',

                '.zip': 'application/zip',

                '.css': 'text/css',

            };

            const contentType = mimeTypes[extname] || 'application/octet-stream';

            serveStaticFile(res, filePath, contentType);

        }

    }

    // Handle POST request

    else if (req.method === 'POST' && parsedUrl.pathname === '/submit') {

        let body = '';

        req.on('data', chunk => {

            body += chunk.toString(); // Convert Buffer to string

        });

        req.on('end', () => {

            const postData = querystring.parse(body);

            console.log('Received data:', postData.data);

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

            res.end('You entered:  ' + postData.data);

        });

    } else {

        res.writeHead(404);

        res.end('404 Not Found');

    }

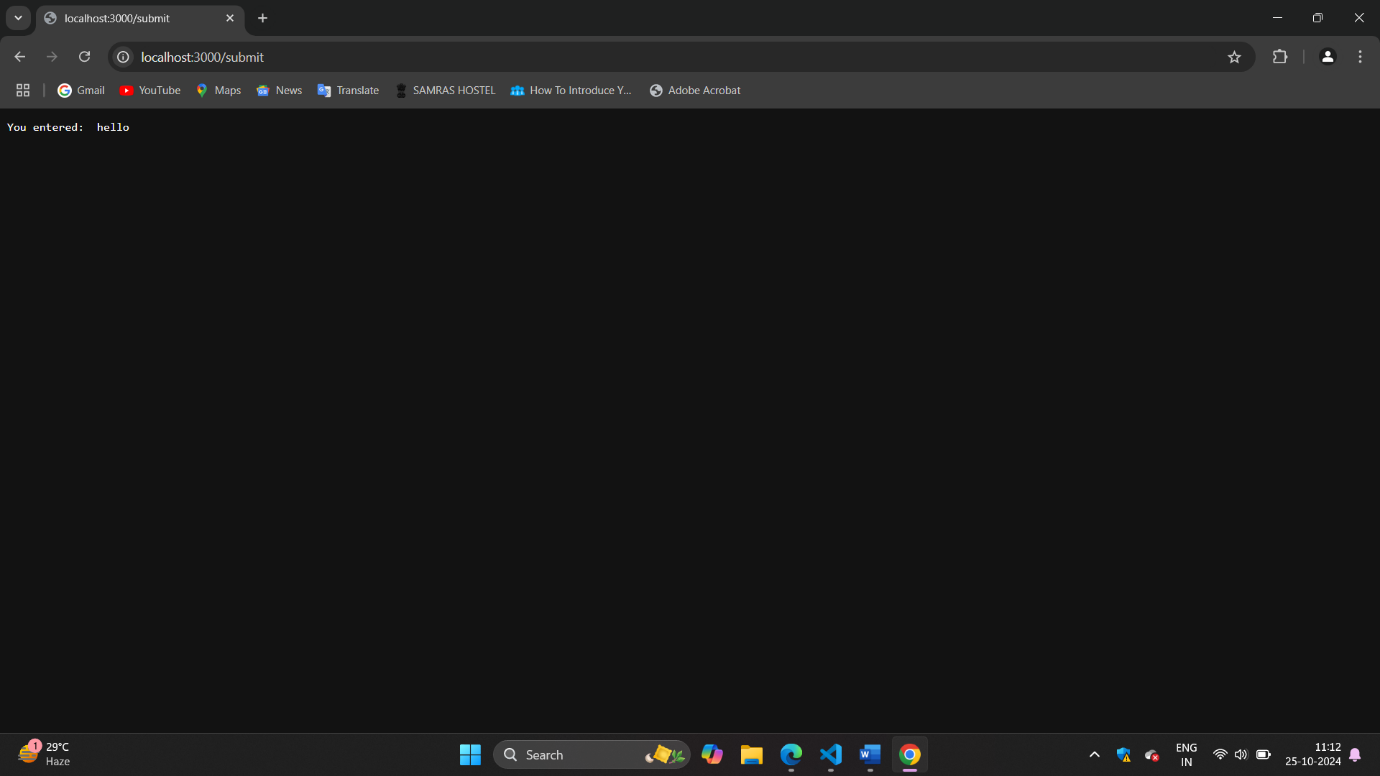
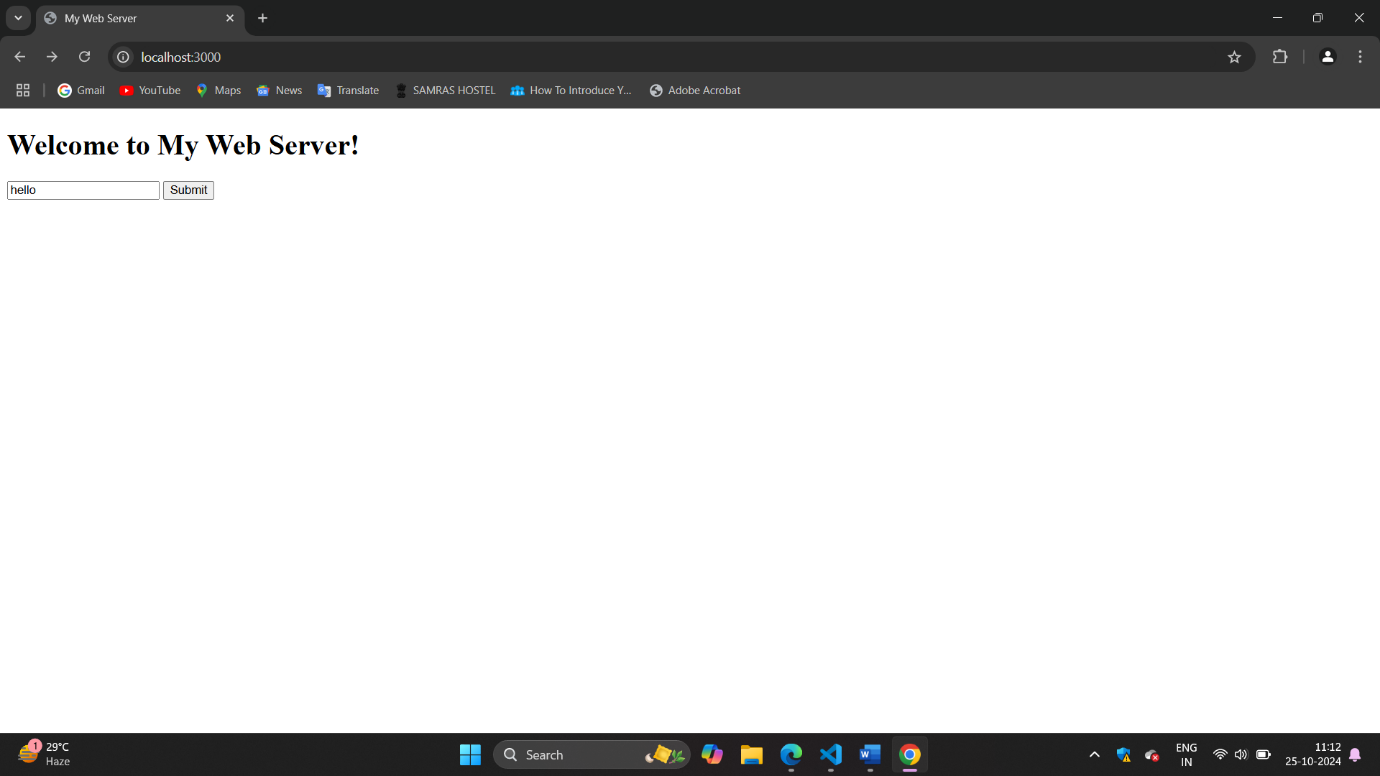
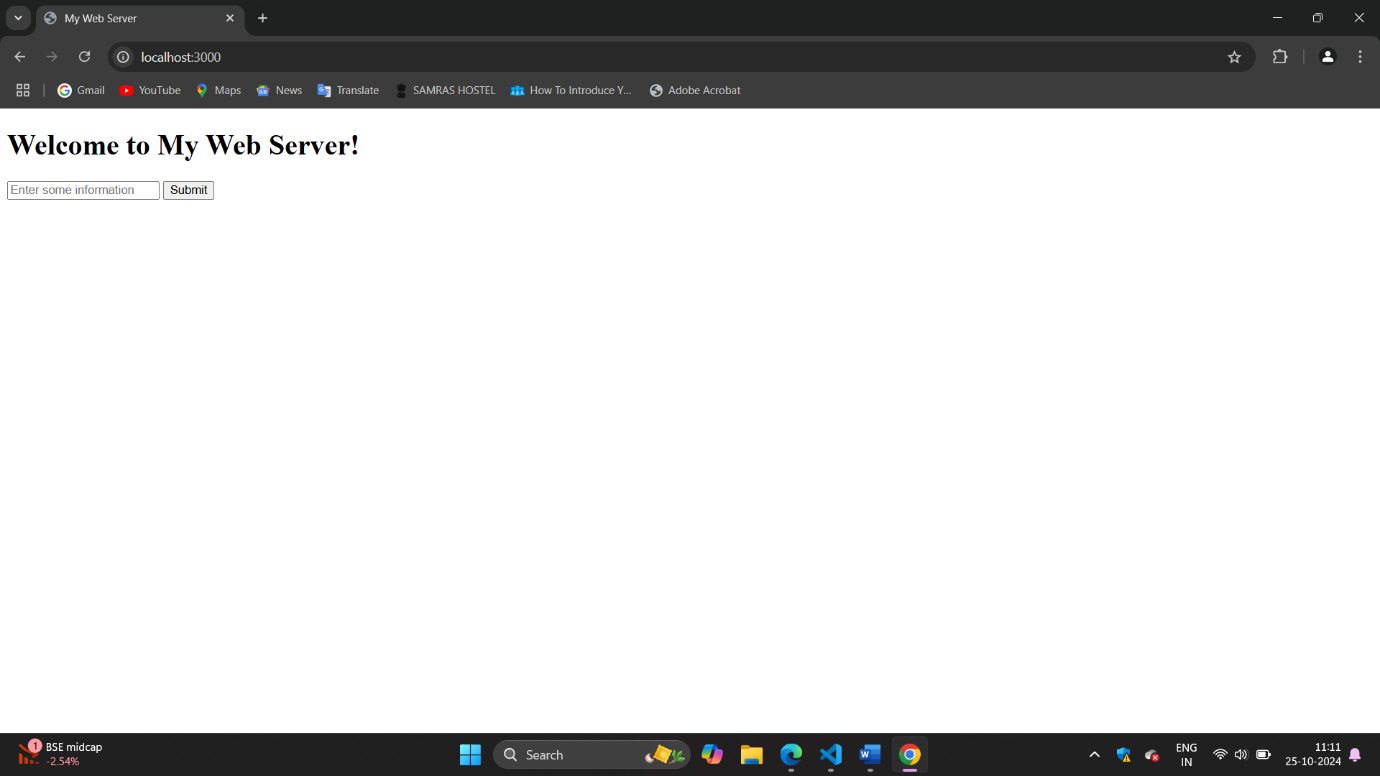
});

// Start the server

server.listen(PORT, () => {

    console.log(`Server is listening on http://localhost:${PORT}`);

});



……………………………………………………………………………………………………………………………………………..

Que-2:

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Hello NodeJS</title>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

    <h1>Welcome to the NodeJS App</h1>

    <button id="getHelloButton">Get message</button>

    <div id="response"></div>

    <script>

        $(document).ready(function() {

            $('#getHelloButton').click(function() {

                $.ajax({

                    url: '/gethello',

                    method: 'GET',

                    success: function(data) {

                        $('#response').text(data);

                    },

                    error: function() {

                        $('#response').text('Error occurred while fetching data.');

                    }

                });

            });

        });

    </script>

</body>

</html>

server.js

const express = require('express');

const path = require('path');

const app = express();

const PORT = 3000;

app.use(express.static('public'));

app.get('/gethello', (req, res) => {

    res.send('Hello Nodejs!!');

});

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

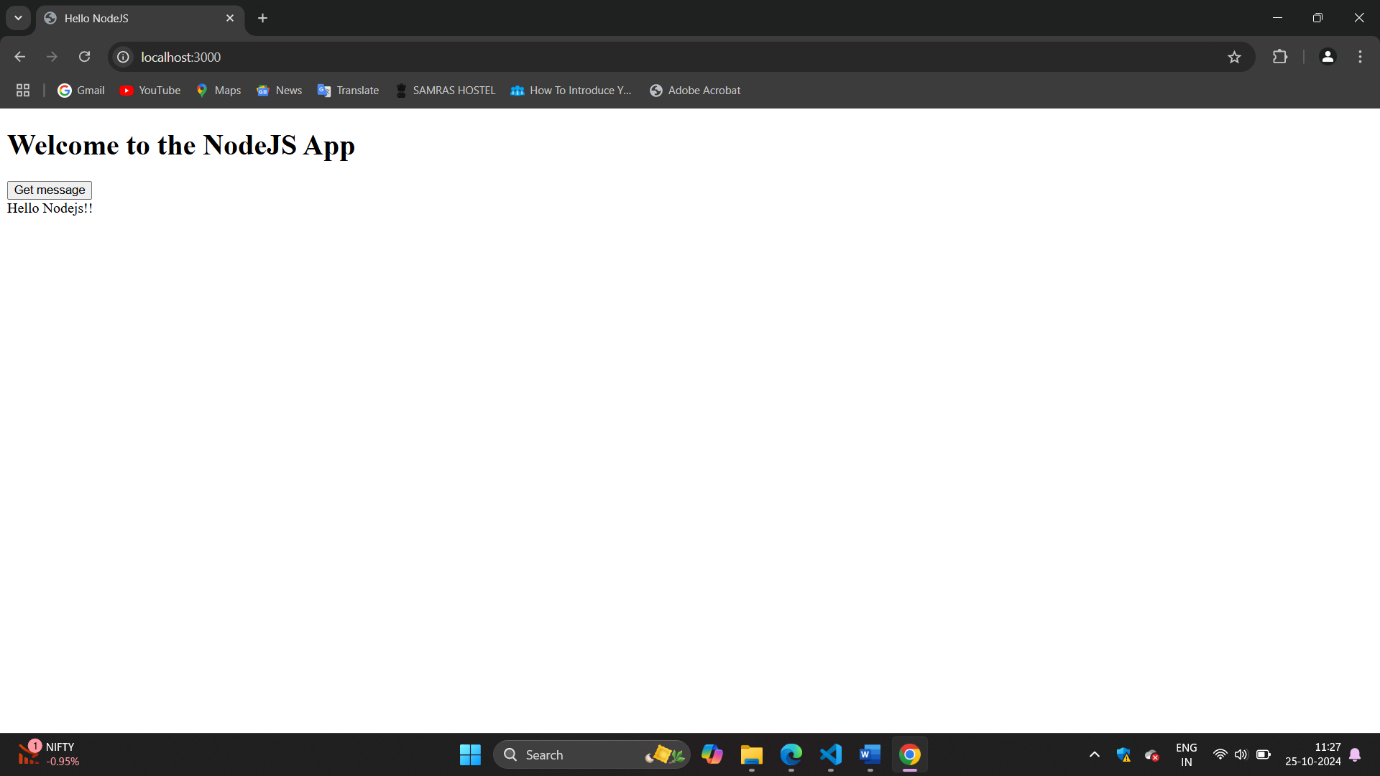
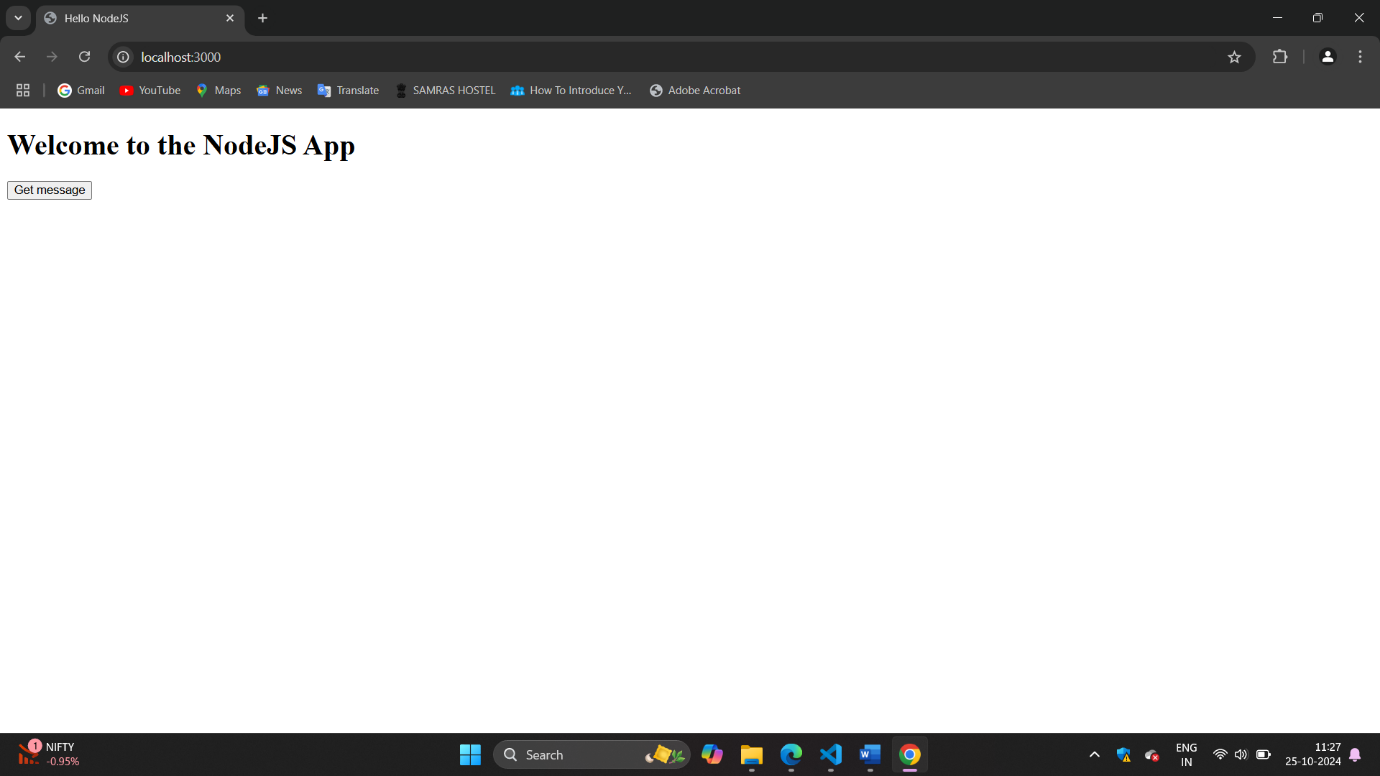
    res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

app.listen(PORT, () => {

    console.log(`Server is listening on http://localhost:${PORT}`);

});



Que-3:

app.js

const readline = require('readline');

const Chatbot = require('./chatbot');

const chatbot = new Chatbot('Customer Support');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

console.log('Welcome to the chatbot application!');

console.log('Type "exit" to quit.\n');

const askQuestion = () => {

rl.question('You: ', (input) => {

if (input.toLowerCase() === 'exit') {

console.log('Chatbot: Goodbye!');

rl.close();

return;

}

const response = chatbot.respond(input);

console.log(`Chatbot: ${response}\n`);

askQuestion();

});

};

askQuestion();

chatbot.js

class Chatbot {

    constructor(domain) {

        this.domain = domain;

        this.responses = {

            greeting: `Hello! I'm a chatbot specialized in ${this.domain}. How can I assist you today?`,

            farewell: `Goodbye! If you need any further assistance in ${this.domain}, feel free to ask!`,

            hours: `Our hours of operation are 9 AM to 5 PM, Monday to Friday.`,

            services: `We offer a variety of services including customer support, product inquiries, and technical assistance.`,

            faq: `You can ask me about our services, hours of operation, or any other questions you might have!`,

            default: `I'm sorry, I didn't understand that. Can you please rephrase your question?`

        };

    }

    respond(message) {

        const lowerMessage = message.toLowerCase();

        if (lowerMessage.includes('hello') || lowerMessage.includes('hi')) {

            return this.responses.greeting;

        } else if (lowerMessage.includes('bye') || lowerMessage.includes('goodbye')) {

            return this.responses.farewell;

        } else if (lowerMessage.includes('hours')) {

            return this.responses.hours;

        } else if (lowerMessage.includes('services') || lowerMessage.includes('what do you offer')) {

            return this.responses.services;

        } else if (lowerMessage.includes('faq') || lowerMessage.includes('questions')) {

            return this.responses.faq;

        } else {

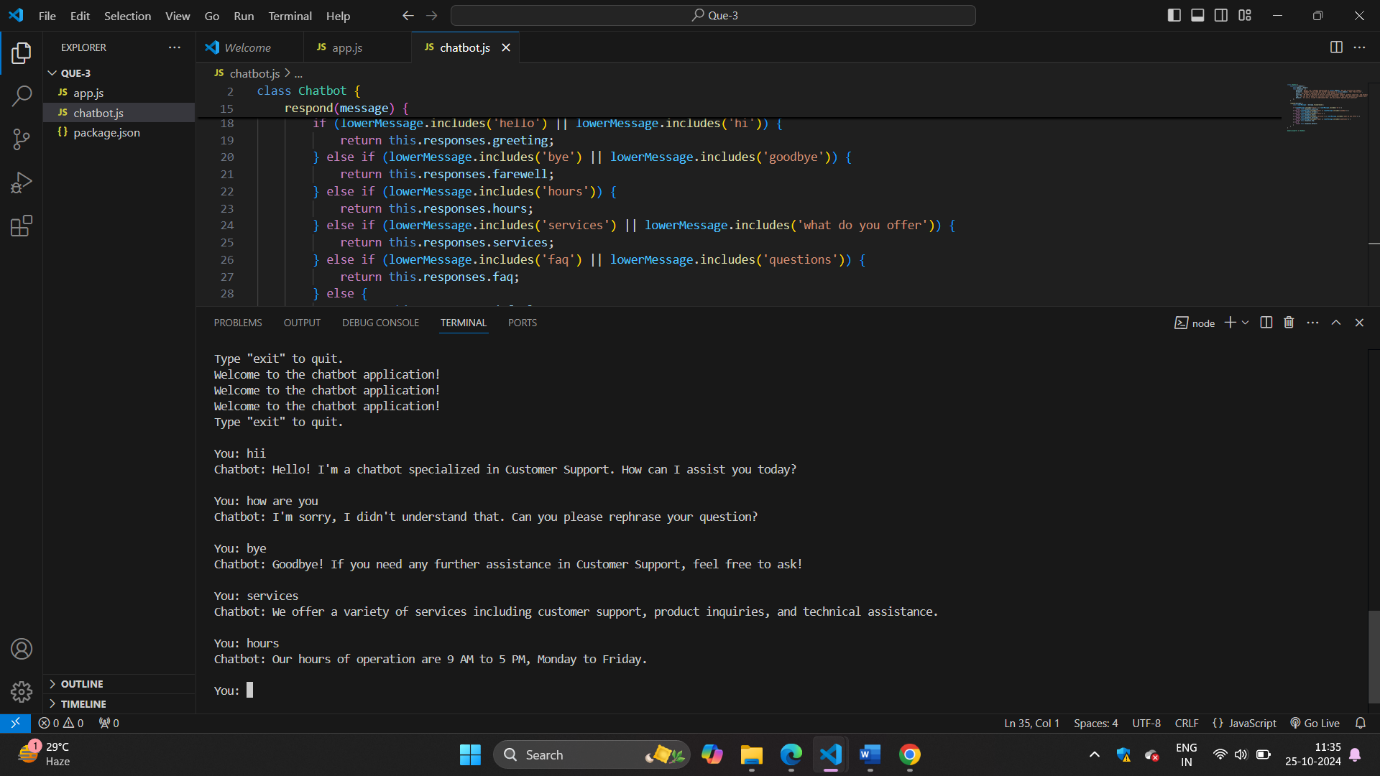
            return this.responses.default;

        }

    }

}

module.exports = Chatbot;

Output:

Que-4:

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>WebSocket Chatbot</title>

    <style>

        body { font-family: Arial, sans-serif; }

        #chat { max-width: 600px; margin: auto; }

        #messages { border: 1px solid #ccc; height: 300px; overflow-y: scroll; padding: 10px; }

        #input { width: 100%; padding: 10px; }

    </style>

</head>

<body>

    <div id="chat">

        <h1>Chatbot</h1>

        <div id="messages"></div>

        <input type="text" id="input" placeholder="Type your message..." />

    </div>

    <script>

        const messagesDiv = document.getElementById('messages');

        const inputField = document.getElementById('input');

        const socket = new WebSocket('ws://localhost:3000');

        socket.onopen = function() {

            console.log('WebSocket connection established.');

        };

        socket.onmessage = function(event) {

            const message = document.createElement('div');

            message.textContent = `Chatbot: ${event.data}`;

            messagesDiv.appendChild(message);

            messagesDiv.scrollTop = messagesDiv.scrollHeight; // Scroll to the bottom

        };

        inputField.addEventListener('keypress', function(event) {

            if (event.key === 'Enter') {

                const userMessage = inputField.value;

                const message = document.createElement('div');

                message.textContent = `You: ${userMessage}`;

                messagesDiv.appendChild(message);

                socket.send(userMessage);

                inputField.value = ''; // Clear input

            }

        });

    </script>

</body>

</html>

server.js

const express = require('express');

const WebSocket = require('ws');

const http = require('http');

const Chatbot = require('./chatbot');

const app = express();

const server = http.createServer(app);

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

const chatbot = new Chatbot('Customer Support');

app.use(express.static('public'));

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

    console.log('New client connected');

    ws.on('message', (message) => {

        console.log(`Received: ${message}`);

        const response = chatbot.respond(message);

        ws.send(response);

    });

    ws.on('close', () => {

        console.log('Client disconnected');

    });

});

const PORT = 3000;

server.listen(PORT, () => {

    console.log(`Server is listening on http://localhost:${PORT}`);

});

Chatbot.js

class Chatbot {

    constructor(domain) {

        this.domain = domain;

        this.responses = {

            greeting: `Hello! I'm a chatbot specialized in ${this.domain}. How can I assist you today?`,

            farewell: `Goodbye! If you need any further assistance in ${this.domain}, feel free to ask!`,

            hours: `Our hours of operation are 9 AM to 5 PM, Monday to Friday.`,

            services: `We offer a variety of services including customer support, product inquiries, and technical assistance.`,

            faq: `You can ask me about our services, hours of operation, or any other questions you might have!`,

            default: `I'm sorry, I didn't understand that. Can you please rephrase your question?`

        };

    }

    respond(message) {

        const lowerMessage = message.toLowerCase();

        if (lowerMessage.includes('hello') || lowerMessage.includes('hi')) {

            return this.responses.greeting;

        } else if (lowerMessage.includes('bye') || lowerMessage.includes('goodbye')) {

            return this.responses.farewell;

        } else if (lowerMessage.includes('hours')) {

            return this.responses.hours;

        } else if (lowerMessage.includes('services') || lowerMessage.includes('what do you offer')) {

            return this.responses.services;

        } else if (lowerMessage.includes('faq') || lowerMessage.includes('questions')) {

            return this.responses.faq;

        } else {

            return this.responses.default;

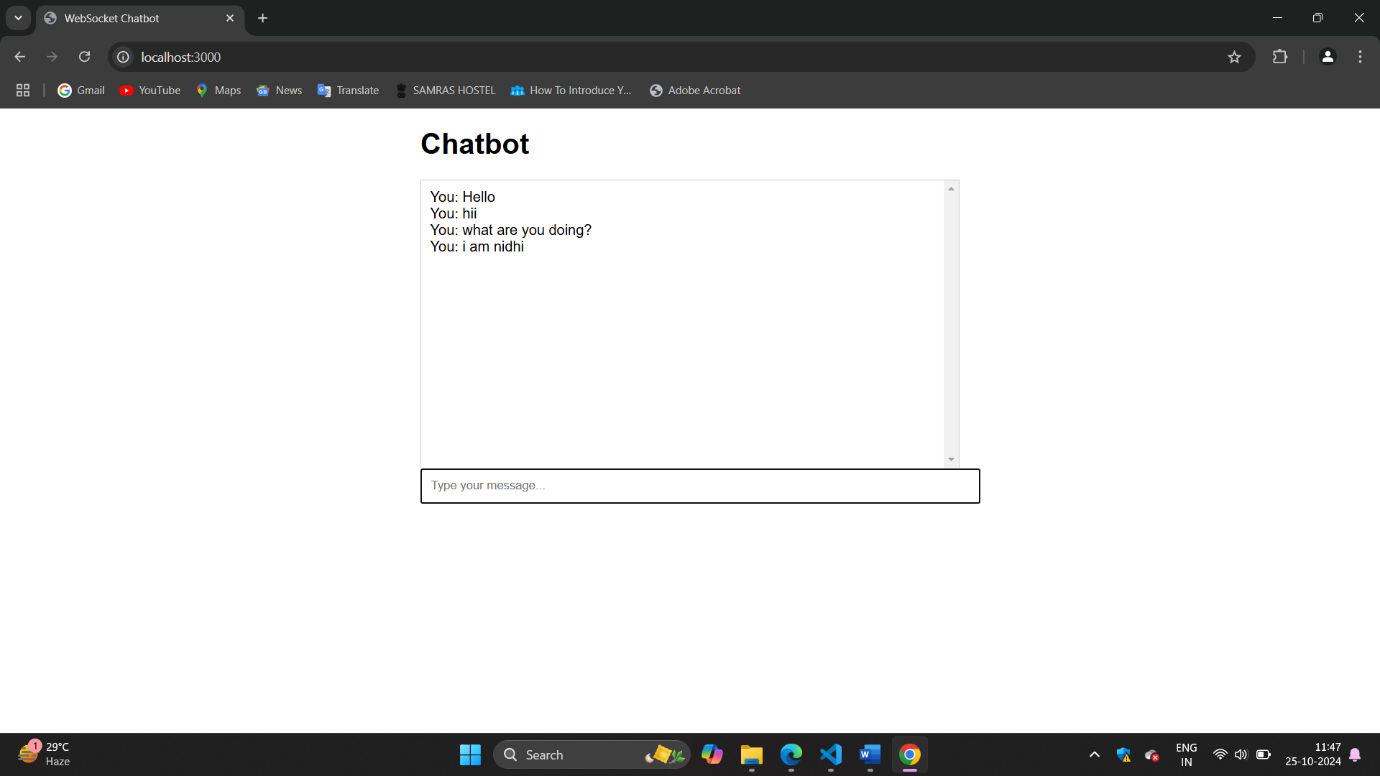
        }

    }

}

module.exports = Chatbot;

Output:



Que-5:

Zipfolder.js

const fs = require('fs-extra');

const archiver = require('archiver');

function zipFolder(sourceFolder, outPath) {

    const output = fs.createWriteStream(outPath);

    const archive = archiver('zip', {

        zlib: { level: 9 } // Set the compression level

    });

    output.on('close', () => {

        console.log(`ZIP file created: ${outPath} (${archive.pointer()} total bytes)`);

    });

    archive.on('error', (err) => {

        throw err;

    });

    archive.pipe(output);

    archive.directory(sourceFolder, false); // Include all files in the folder

    archive.finalize();

}

// Example usage

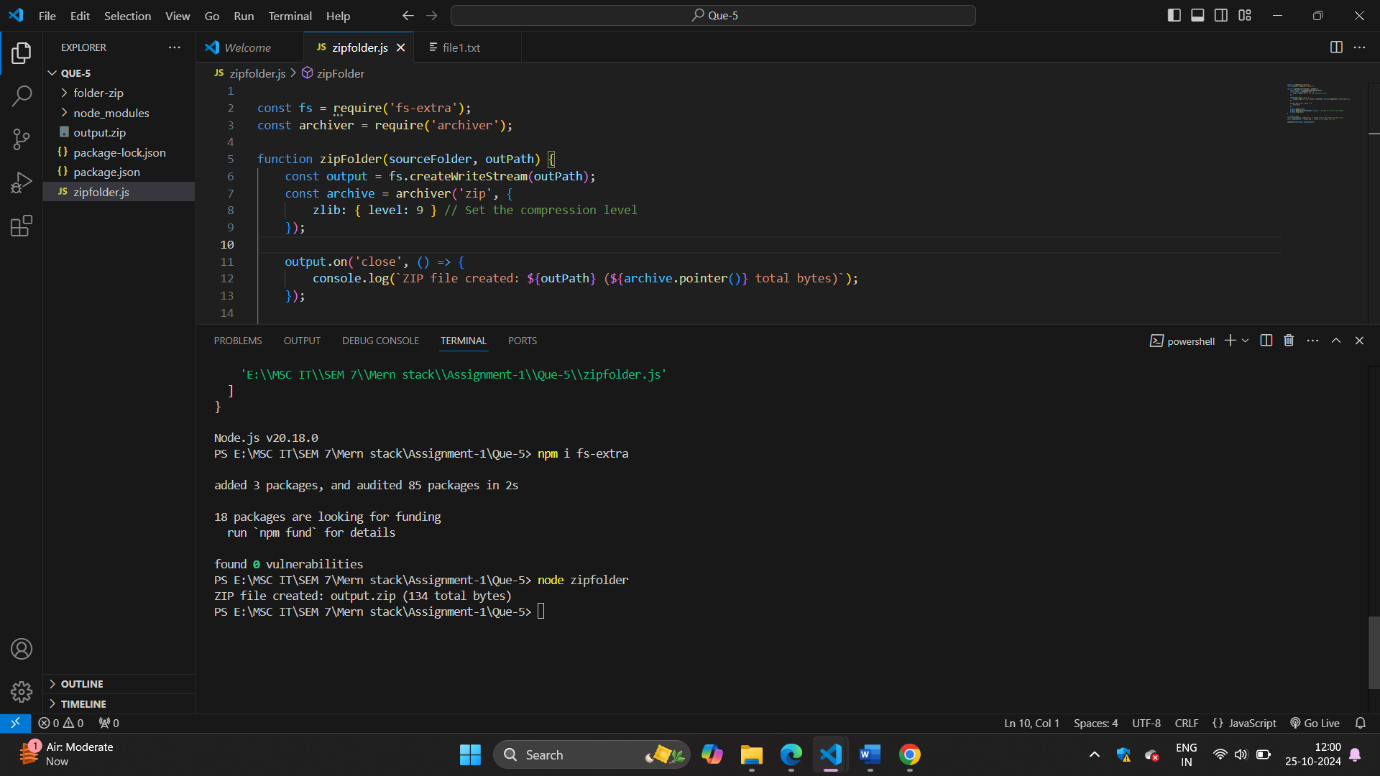
const folderToZip = 'folder-zip'; // Change this to the folder you want to zip

const outputZipPath = 'output.zip'; // Name of the output zip file

zipFolder(folderToZip, outputZipPath);

folder-zip => file1.txt

Output:



Que-5:

Extract\_zip\_file.js

const fs = require('fs');

const unzipper = require('unzipper');

function extractZip(zipFilePath, outputFolder) {

    fs.createReadStream(zipFilePath)

        .pipe(unzipper.Extract({ path: outputFolder }))

        .on('close', () => {

            console.log(`Extraction completed: ${outputFolder}`);

        })

        .on('error', (err) => {

            console.error(`Error during extraction: ${err.message}`);

        });

}

// Example usage

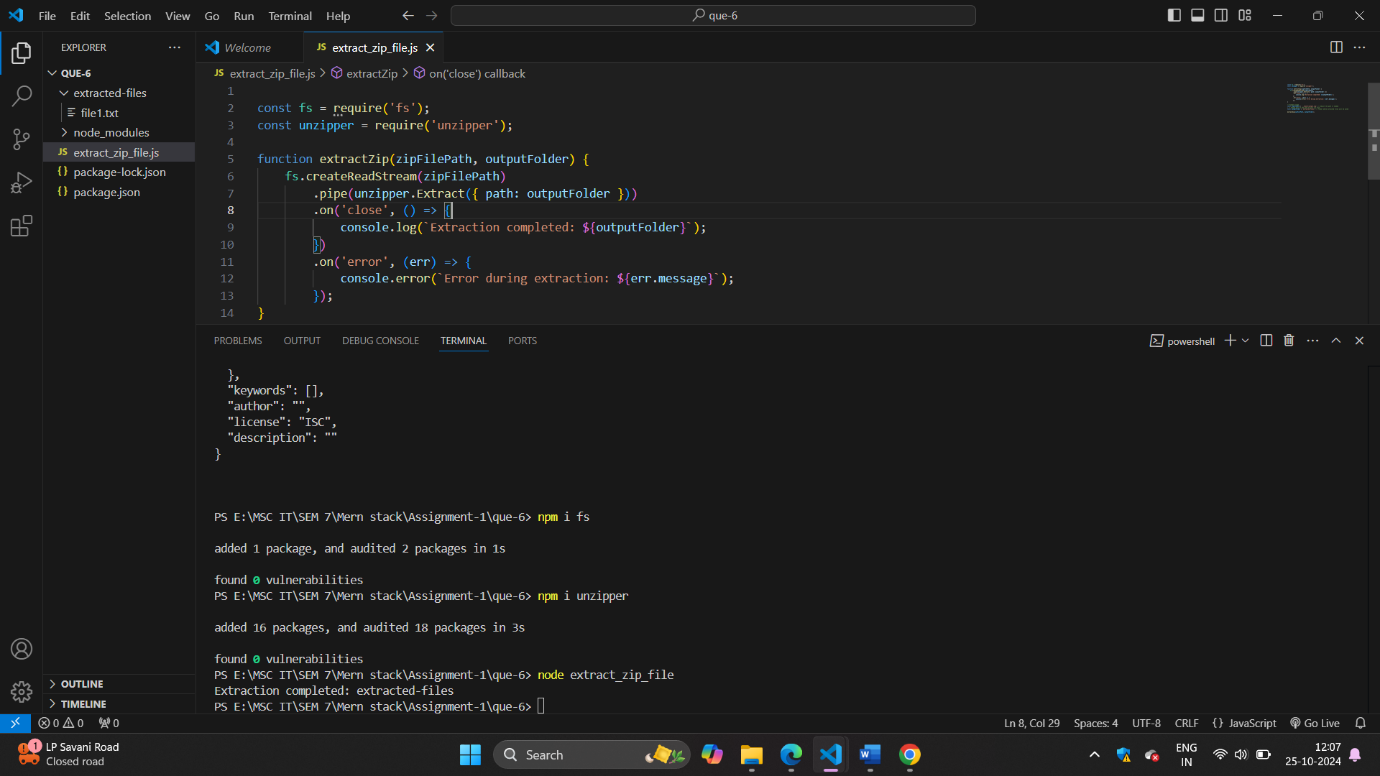
const zipFilePath = '../Que-5/output.zip'; // Adjust the path if needed

; // Change this to the path of your zip file

const outputFolder = 'extracted-files'; // Folder where extracted files will be saved

extractZip(zipFilePath, outputFolder);

Output:



Que-6:

Promishifi.js

const fs = require('fs');

const util = require('util');

const unlink = util.promisify(fs.unlink);

async function deleteFile(filePath) {

    try {

        await unlink(filePath);

        console.log(`File deleted: ${filePath}`);

    } catch (err) {

        console.error(`Error deleting file: ${err.message}`);

    }

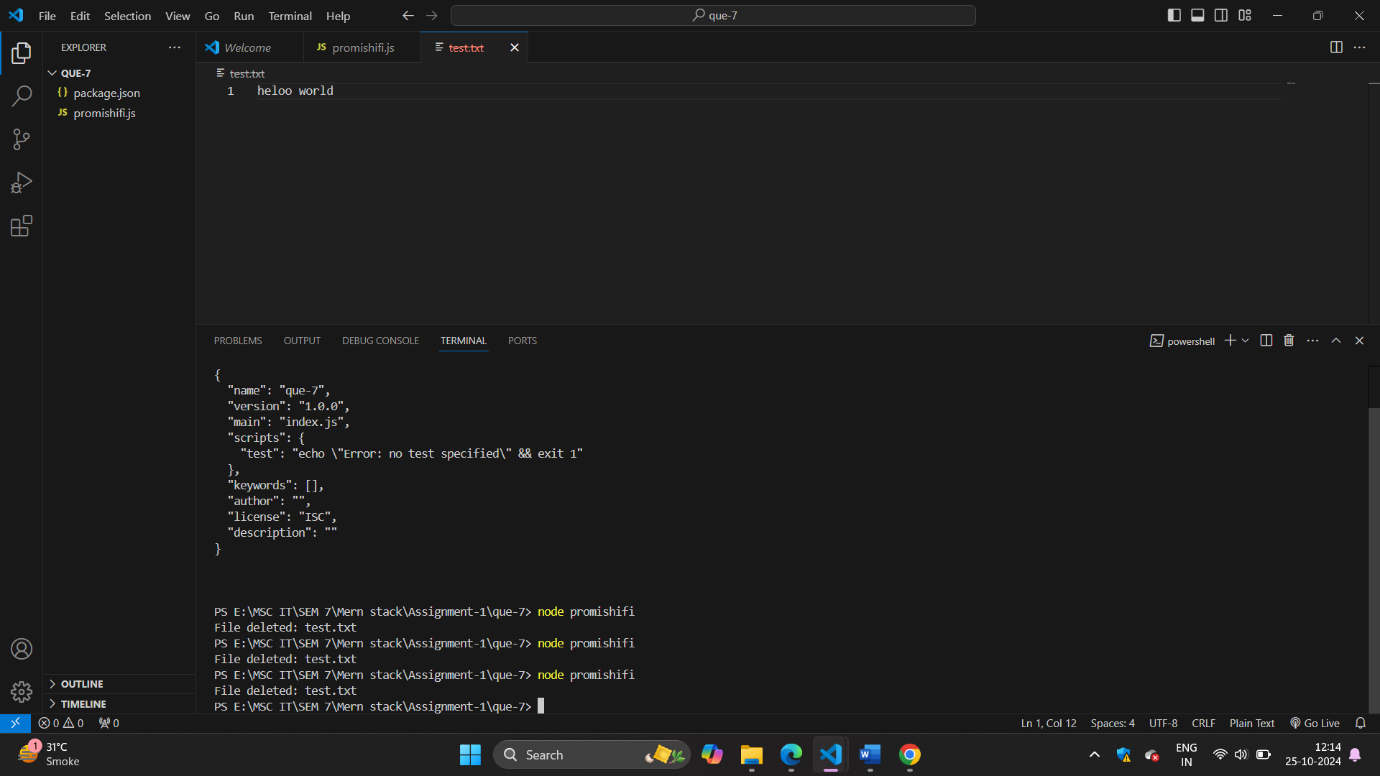
}

const fileToDelete = 'test.txt';

fs.writeFileSync(fileToDelete, 'This is a test file.');

deleteFile(fileToDelete);

Output:



Que-8:

fetchgoogle.mjs

import fetch from 'node-fetch';

import \* as cheerio from 'cheerio'; // Use named import

async function fetchGooglePage() {

    try {

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

        if (!response.ok) {

            throw new Error(`HTTP error! Status: ${response.status}`);

        }

        const data = await response.text();

        const $ = cheerio.load(data);

        const title = $('title').text();

        console.log(`Title: ${title}`);

        const firstLink = $('a').first().attr('href');

        console.log(`First link: ${firstLink}`);

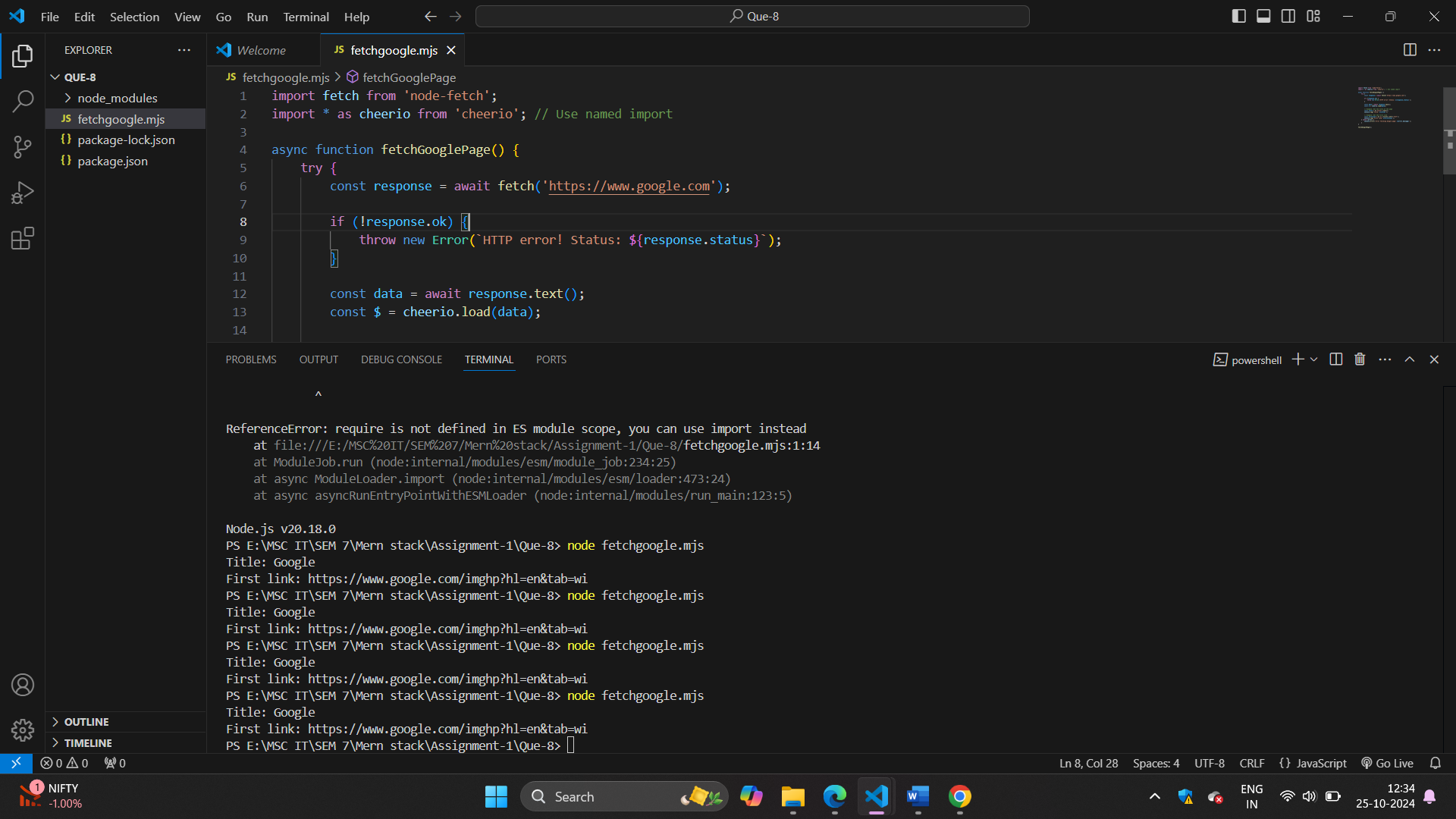
    } catch (error) {

        console.error(`Error fetching Google page: ${error.message}`);

    }

}

fetchGooglePage();



Que-10:

Server.js

const express = require('express');

const app = express();

const PORT = process.env.PORT || 3000;

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

    res.send('Hello, World!');

});

app.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

In tests folder=>

server.test.js

const request = require('supertest');

const express = require('express');

const app = express();

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

    res.send('Hello, World!');

});

describe('GET /', () => {

    it('should respond with Hello, World!', async () => {

        const response = await request(app).get('/');

        expect(response.text).toBe('Hello, World!');

    });

});

Package.json

{

  "name": "que-10",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "start": "node server.js",

    "test": "jest",

    "user-script-1": "echo 'This is user-defined script 1'",

    "user-script-2": "echo 'This is user-defined script 2'",

    "user-script-3": "echo 'This is user-defined script 3'"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "dependencies": {

    "express": "^4.21.1",

    "supertest": "^7.0.0"

  },

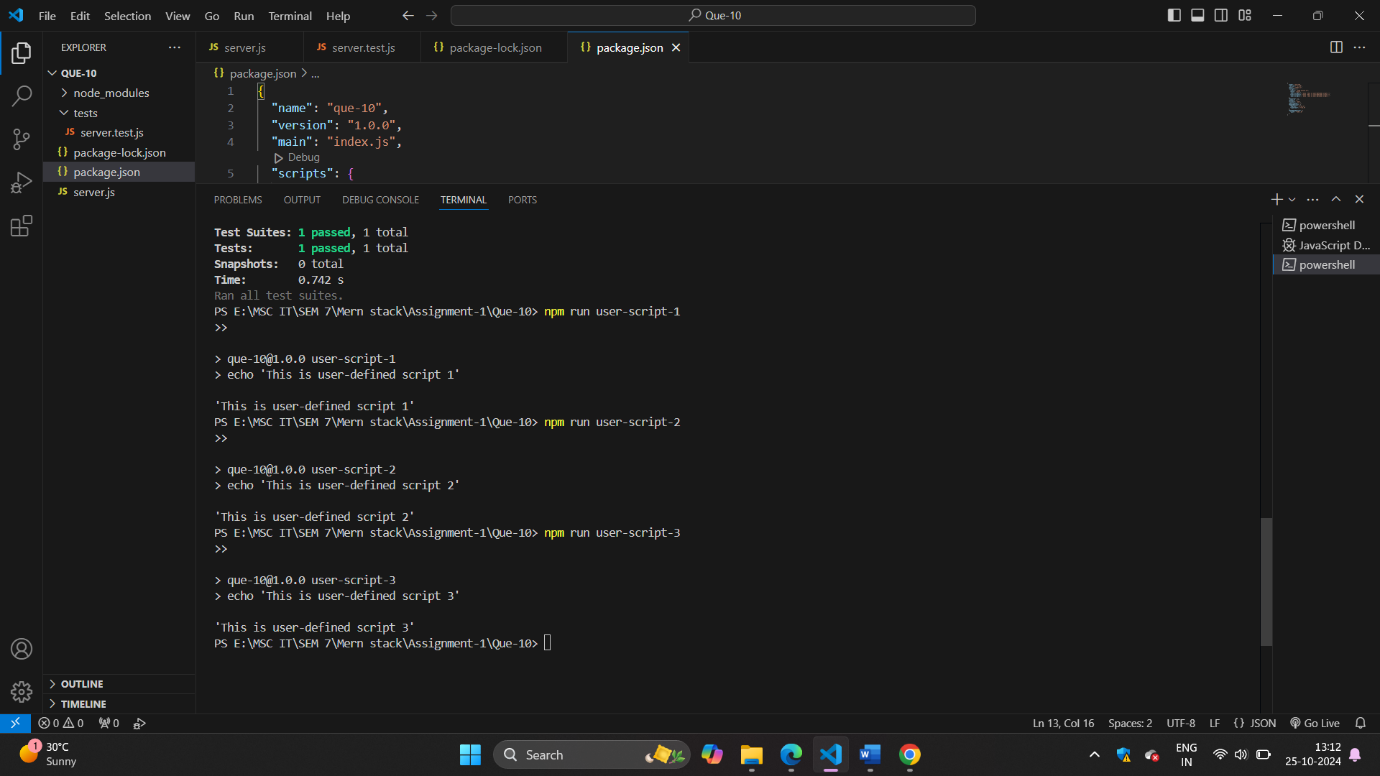
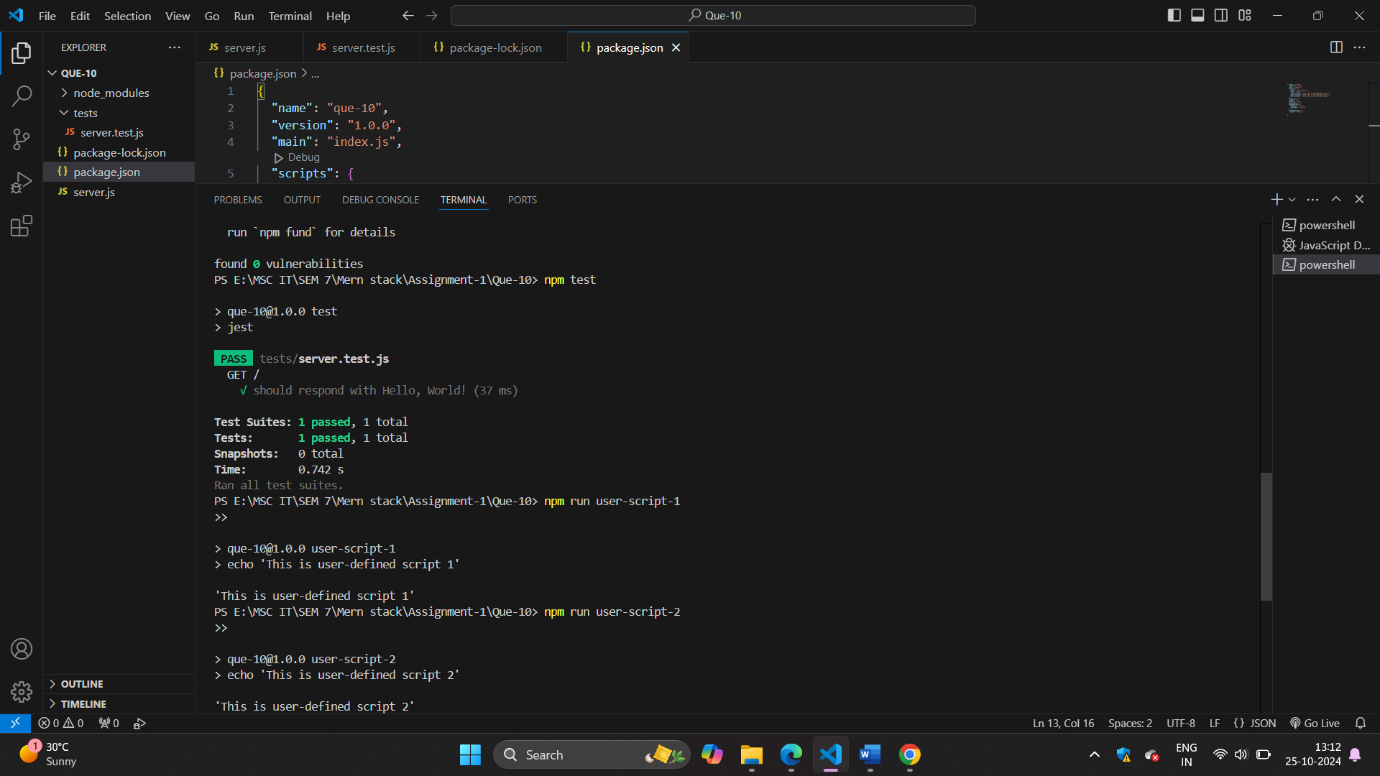
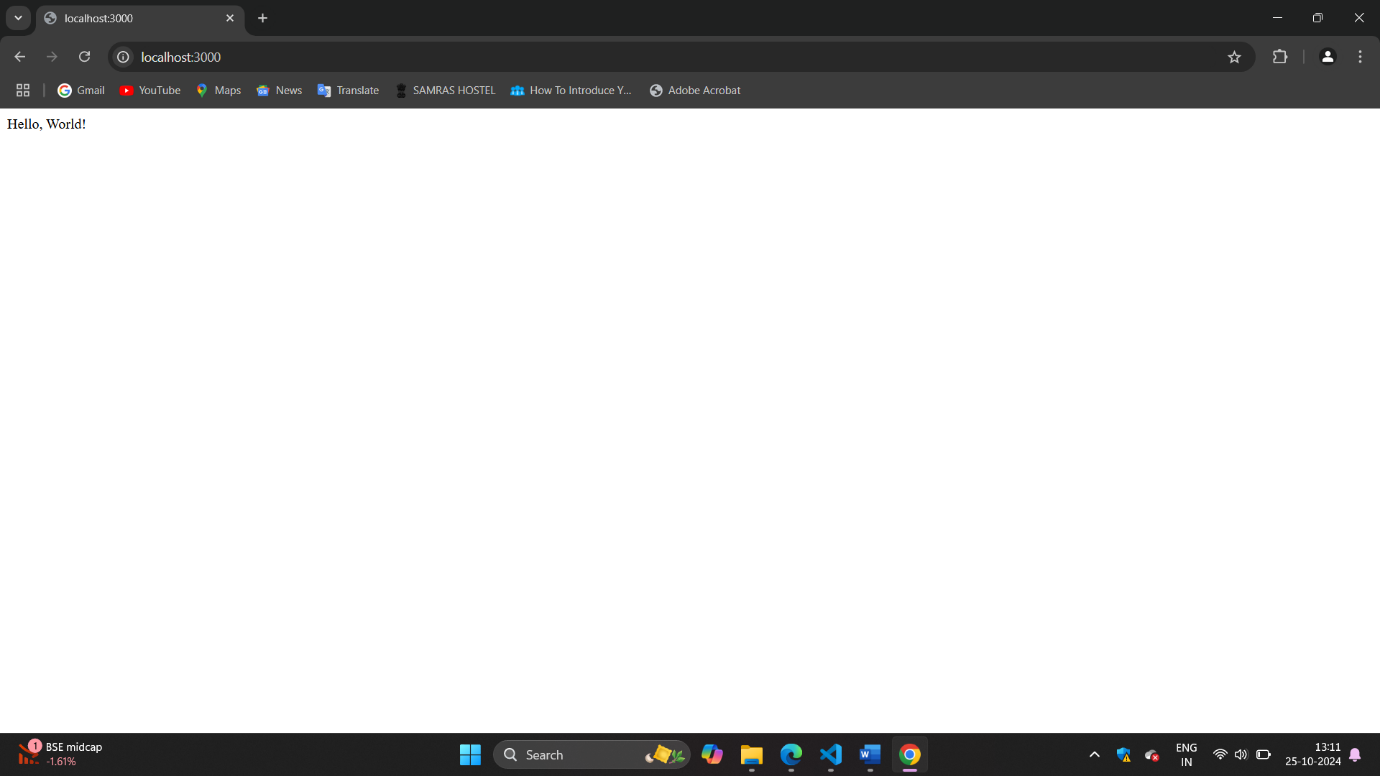
  "devDependencies": {

    "jest": "^29.7.0"

  }

}

Output:



Que-11:

Server.js

const express = require('express');

const app = express();

const PORT = process.env.PORT || 8000;

// Set EJS as the templating engine

app.set('view engine', 'ejs');

// Serve static files

app.use(express.static('public'));

// Sample static cricket scores

const scores = [

    {

        series: { name: 'IPL 2023' },

        team1: { name: 'Team A' },

        team2: { name: 'Team B' },

        status: 'Team A: 150/5 (18.0 overs) - Team B: 155/2 (17.0 overs) - Team B won by 8 wickets'

    },

    {

        series: { name: 'ODI Series' },

        team1: { name: 'Team C' },

        team2: { name: 'Team D' },

        status: 'Team C: 200/10 (40.0 overs) - Team D: 201/3 (35.0 overs) - Team D won by 7 wickets'

    }

];

// Home route

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

    res.render('index');

});

// Scores route

app.get('/scores', (req, res) => {

    res.render('scores', { scores });

});

// Start the server

app.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

Views folder

Scores.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Live Cricket Scores</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-top: 20px;

        }

        th, td {

            padding: 12px;

            text-align: left;

            border-bottom: 1px solid #ddd;

        }

        th {

            background-color: #f2f2f2;

        }

        tr:hover {

            background-color: #f5f5f5;

        }

        h1 {

            color: #333;

        }

    </style>

</head>

<body>

    <h1>Live Cricket Scores</h1>

    <a href="/">Back to Home</a>

    <table>

        <thead>

            <tr>

                <th>Series</th>

                <th>Teams</th>

                <th>Status</th>

            </tr>

        </thead>

        <tbody>

            <% if (scores.length > 0) { %>

                <% scores.forEach(match => { %>

                    <tr>

                        <td><%= match.series.name %></td>

                        <td><%= match.team1.name %> vs <%= match.team2.name %></td>

                        <td><%= match.status %></td>

                    </tr>

                <% }) %>

            <% } else { %>

                <tr>

                    <td colspan="3">No live matches at the moment.</td>

                </tr>

            <% } %>

        </tbody>

    </table>

</body>

</html>

Index.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Live Cricket Score</title>

</head>

<body>

    <h1>Welcome to Live Cricket Score</h1>

    <a href="/scores">View Live Scores</a>

</body>

</html>

Output:

