Assignment 1

Name:Patel Krupa Umeshbhai

Sem 7th

Roll No.47

Subject:FullStack

Practical Assignment1

Question1

1. Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.

- Handle POST request.

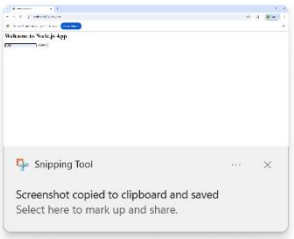
->server.js

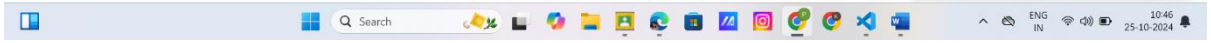
```
const express = require('express');
const fetch = require('node-fetch');

const app = express();
const PORT = 3000;

// Route to fetch live cricket score
app.get('/live-score', async (req, res) => {
  try {
    const response = await fetch(`https://cricapi.com/api/matches?apikey=API_KEY`);
    const data = await response.json();
    res.json(data);
  } catch (error) {
    res.status(500).send('Error fetching live score');
```

```
  }
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

->Index.html

```
<html>
  <head>
    <title>Static Resource</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Welcome to Node.js App</h1>
    <form action="/submit" method="POST">
      <input type="text" name="username" placeholder="Enter your name">
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```





Question 2

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

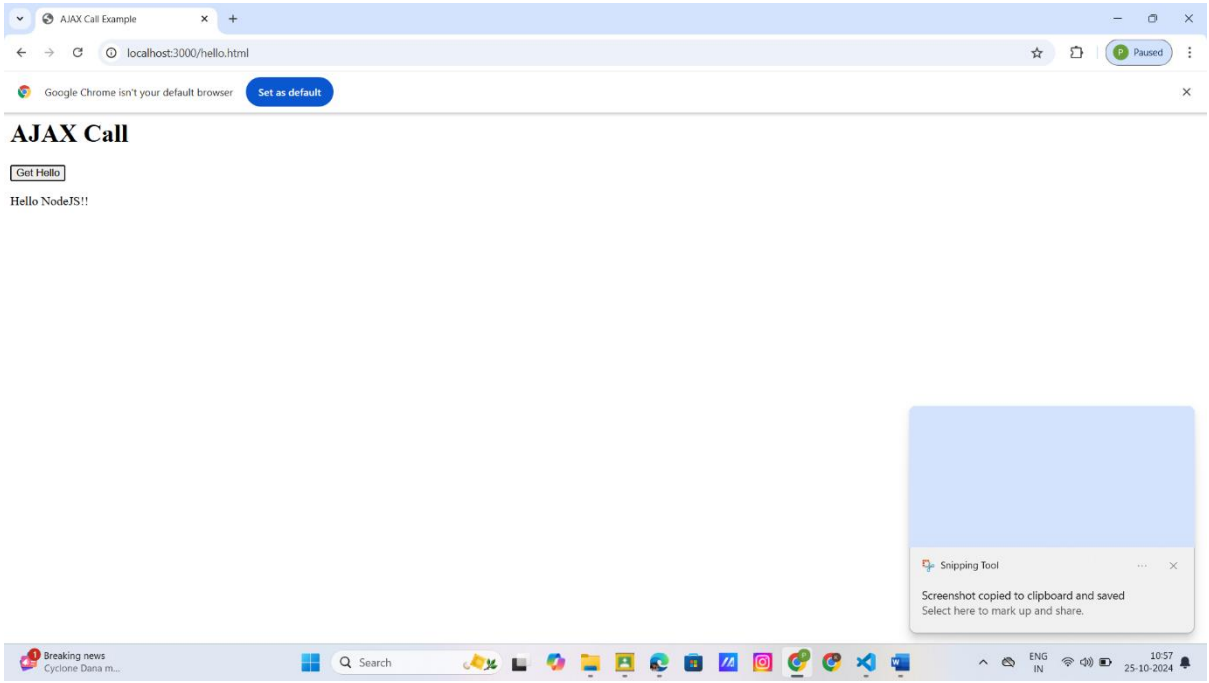
```
const express = require('express');
const path = require('path');

const app = express();
const PORT = 3000;

app.use(express.static(path.join(__dirname, 'public')));

// Route /gethello
app.get('/gethello', (req, res) => {
  res.send('Hello NodeJS!!');
});

app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

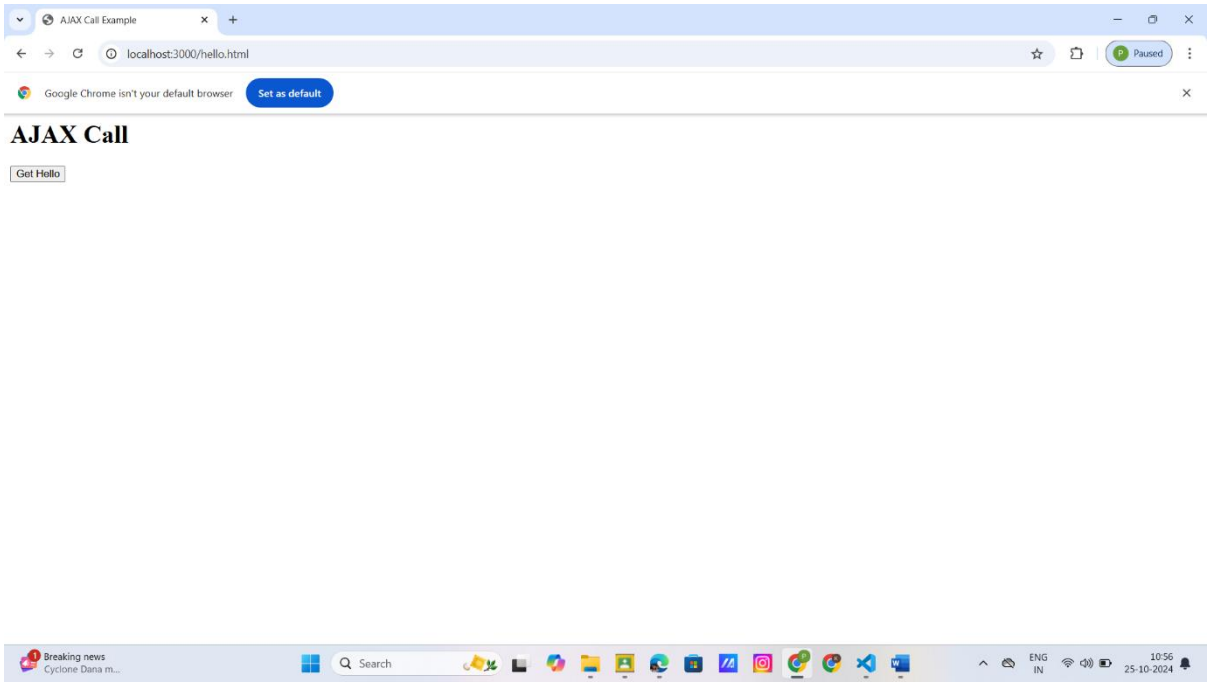
hello.html

```
<html>
  <head>
    <title>AJAX Call Example</title>
  </head>
  <body>
    <h1>AJAX Call</h1>
    <button id="btn">Get Hello</button>
    <p id="response"></p>
    <script src="script.js"></script>
  </body>
</html>
```

→script.js

```
document.getElementById('btn').addEventListener('click', () => {
  fetch('/gethello')
    .then(response => response.text())
    .then(data => {
      document.getElementById('response').innerText = data;
    })
    .catch(error => console.error('Error:', error));
});
```





Question 3

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

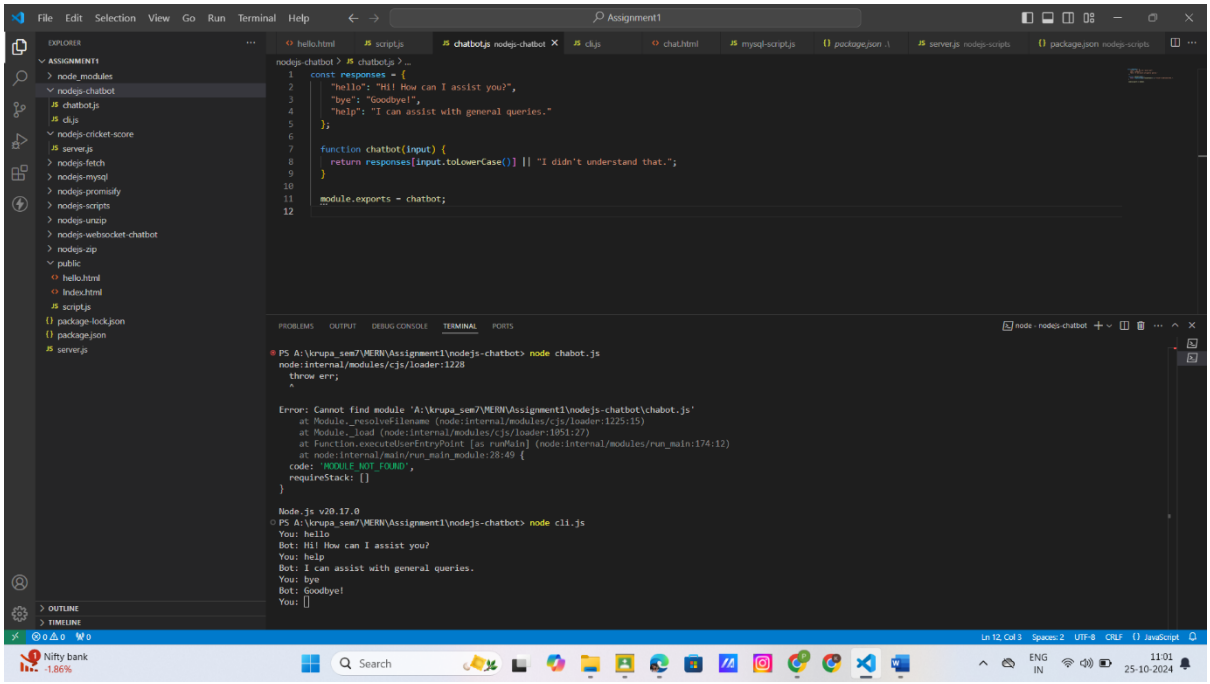
→cli.js

```
const readline = require('readline');
const chatbot = require('./chatbot');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
```

```
});

rl.setPrompt('You: ');
rl.prompt();

rl.on('line', (input) => {
  const response = chatbot(input);
  console.log(`Bot: ${response}`);
  rl.prompt();
});
```

→chatbot.js

```
const responses = {
    "hello": "Hi! How can I assist you?",
    "bye": "Goodbye!",
    "help": "I can assist with general queries."
  };

  function chatbot(input) {
    return responses[input.toLowerCase()] || "I didn't understand that.";
  }

  module.exports = chatbot;
```



Question4

Use above chatbot module in web based chatting of websocket.

→server.js

```
// server.js
const express = require('express');
const path = require('path');
const WebSocket = require('ws');
const chatbot = require('./chatbot');

const app = express();
const PORT = 3000;

// Serve static HTML file
app.use(express.static(path.join(__dirname, 'public')));

// Create HTTP server and WebSocket server
const server = app.listen(PORT, () => {
```

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

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

// WebSocket connection handler
wss.on('connection', (ws) => {
  console.log('New client connected!');

  ws.on('message', (message) => {
    console.log(`Received: ${message}`);
    const response = chatbot(message);
    ws.send(`Bot: ${response}`);
  });

  ws.on('close', () => {
    console.log('Client disconnected.');
  });
});
```

→chatbot.js

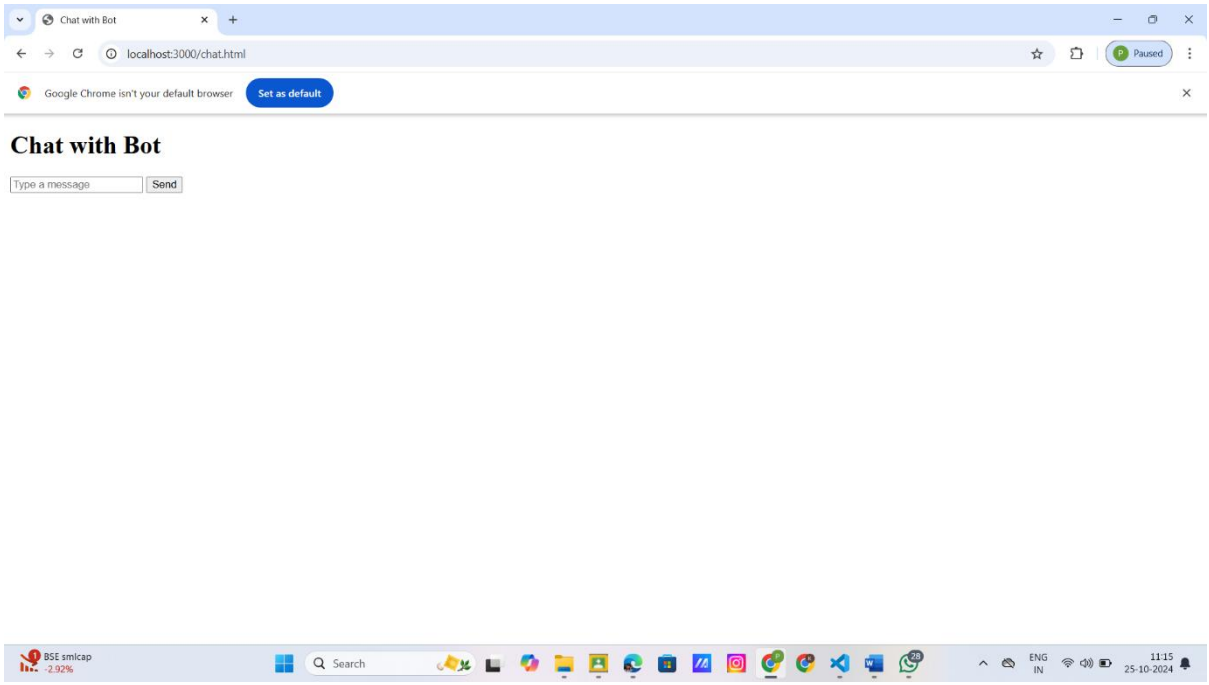
```
// chatbot.js
const responses = {
    "hello": "Hi! How can I assist you?",
    "bye": "Goodbye!",
    "help": "I can assist with general queries."
  };

  function chatbot(input) {
    return responses[input.toLowerCase()] || "I didn't understand that.";
  }
```

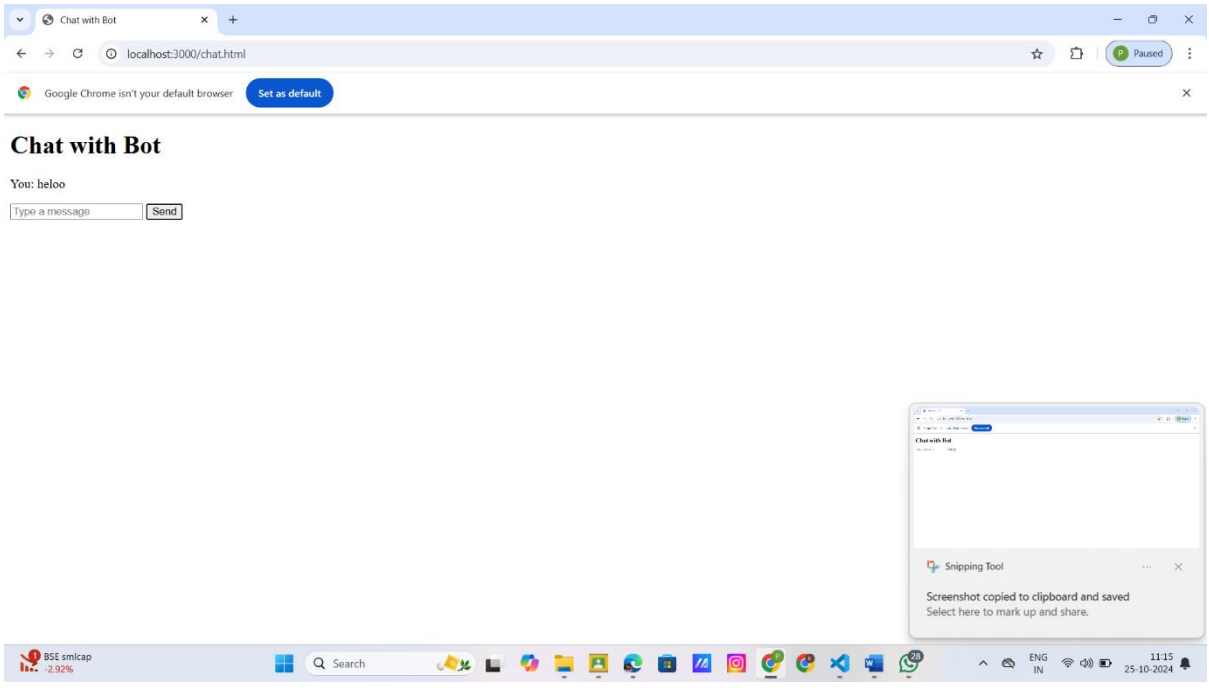
```
module.exports = chatbot;
```

→chat.html

```
<!-- /public/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chat with Bot</title>
</head>
<body>
  <h1>Chat with Bot</h1>
  <div id="chatbox"></div>
  <input type="text" id="input" placeholder="Type a message" />
  <button onclick="sendMessage()">Send</button>

  <script>
    const ws = new WebSocket('ws://localhost:3000');

    ws.onmessage = function(event) {
      const chatbox = document.getElementById('chatbox');
      chatbox.innerHTML += `<p>${event.data}</p>`;
    };

    function sendMessage() {
      const input = document.getElementById('input');
      ws.send(input.value);
      const chatbox = document.getElementById('chatbox');
      chatbox.innerHTML += `<p>You: ${input.value}</p>`;
```

```
    input.value = ''; // Clear input field after sending
  }
 </script>
</body>
</html>
```



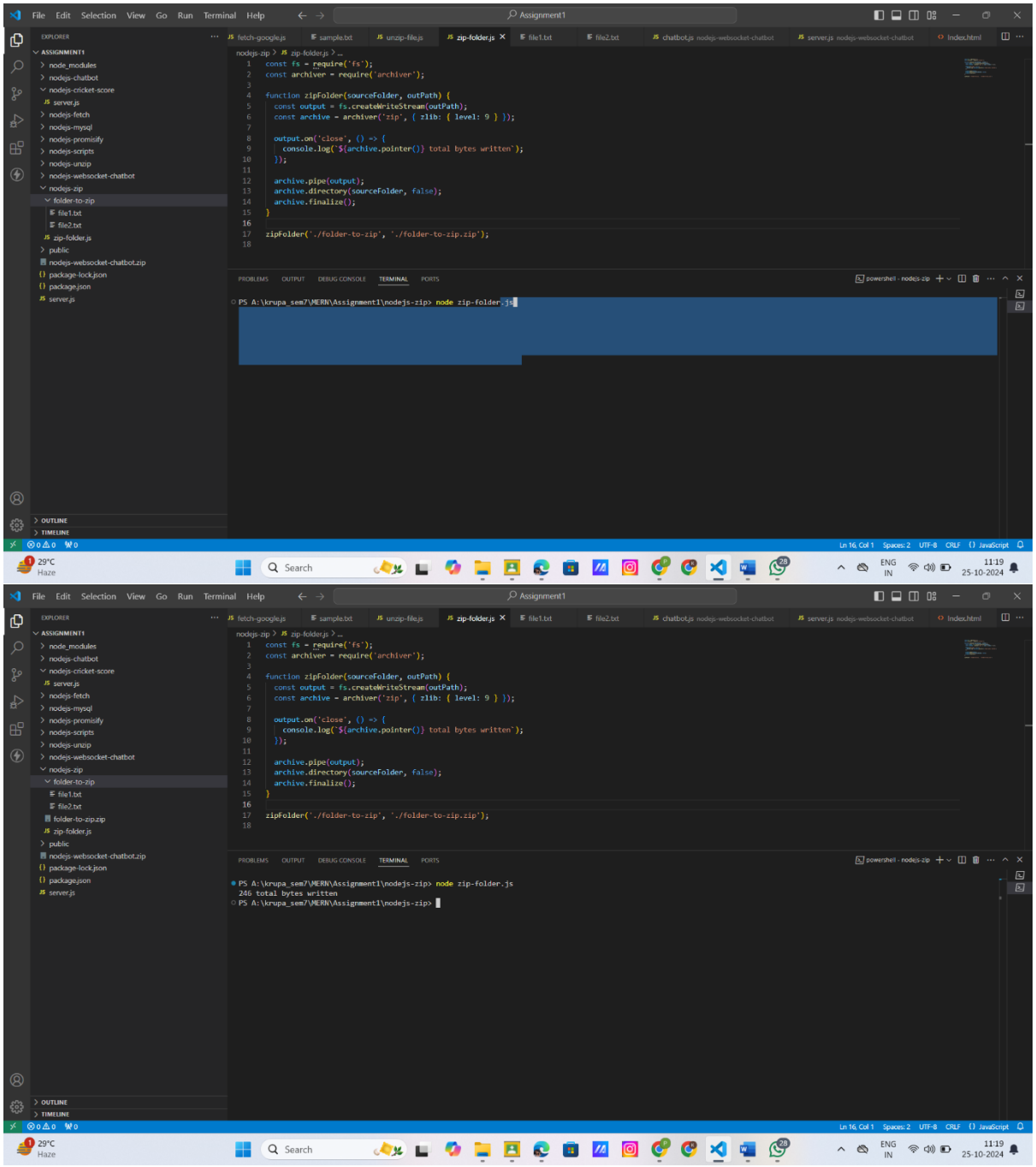


Question 5

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

→zip-folder.js

```
const fs = require('fs');
const archiver = require('archiver');

function zipFolder(sourceFolder, outPath) {
  const output = fs.createWriteStream(outPath);
  const archive = archiver('zip', { zlib: { level: 9 } });

  output.on('close', () => {
    console.log(`${archive.pointer()} total bytes written`);
  });

  archive.pipe(output);
  archive.directory(sourceFolder, false);
  archive.finalize();
}

zipFolder('./folder-to-zip', './folder-to-zip.zip');
```



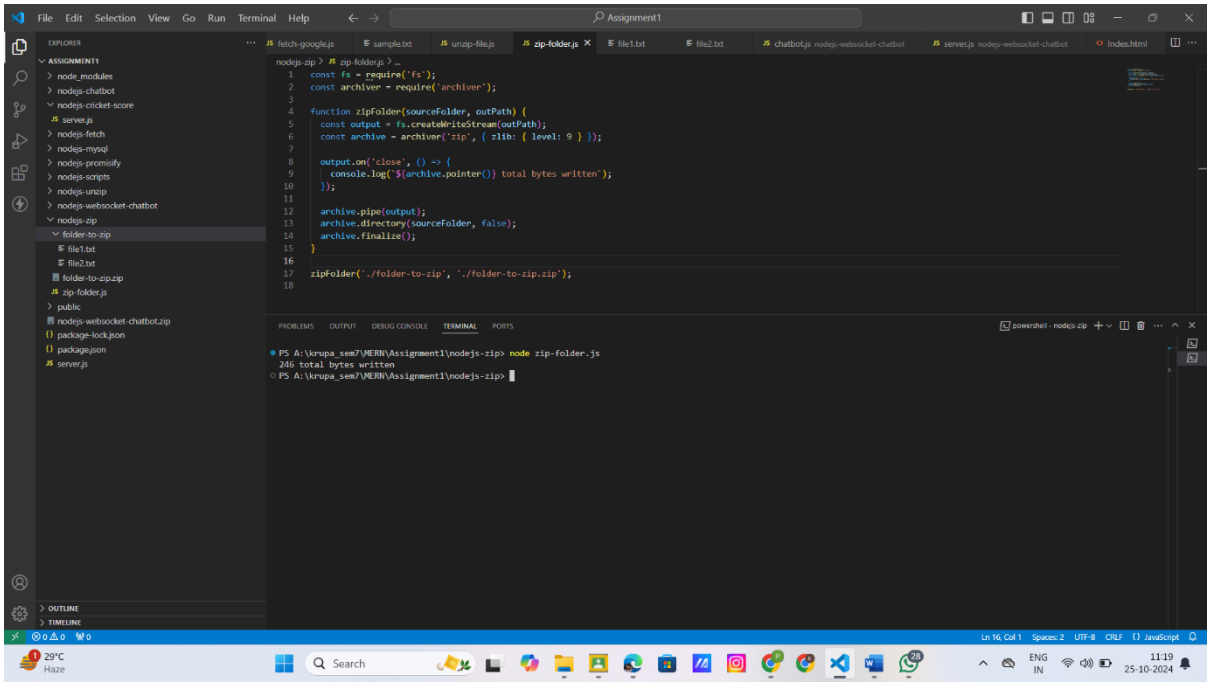
Question6

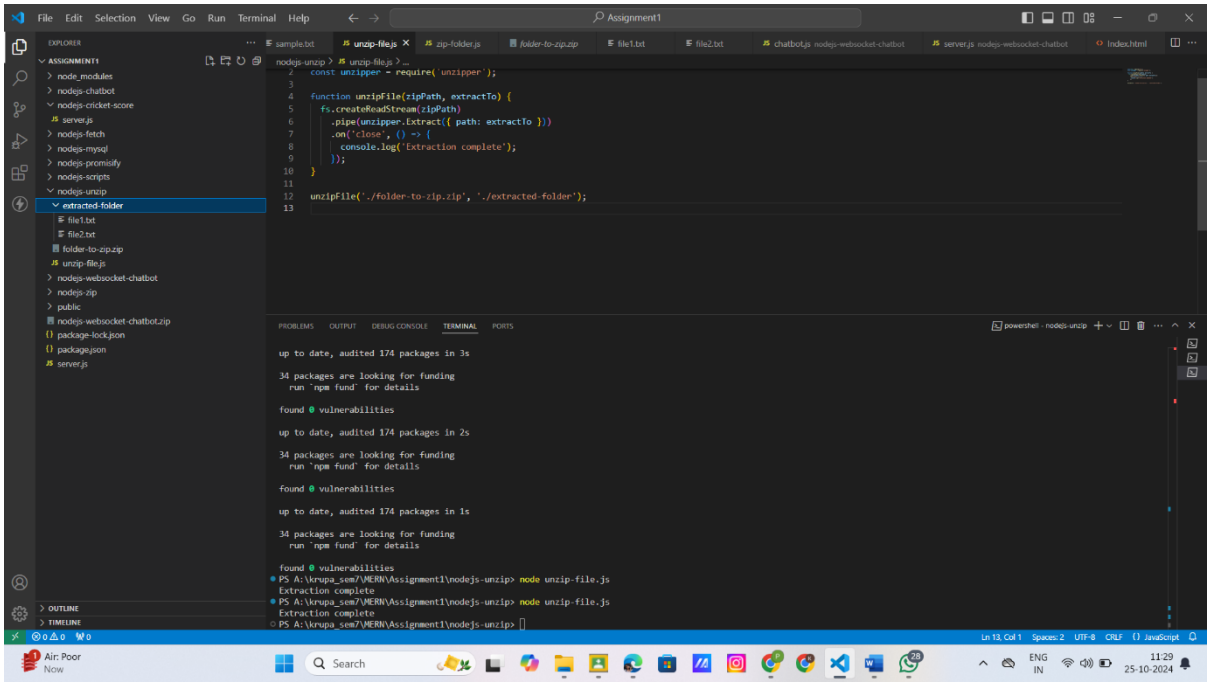
6. Write a program to extract a zip file.

→unzip-file.js

```
const fs = require('fs');

const unzipper = require('unzipper');


function unzipFile(zipPath, extractTo) {

  fs.createReadStream(zipPath)
```

```
    .pipe(unzipper.Extract({ path: extractTo }))
    .on('close', () => {
      console.log('Extraction complete');
    });
}

unzipFile('./folder-to-zip.zip', './extracted-folder');
```

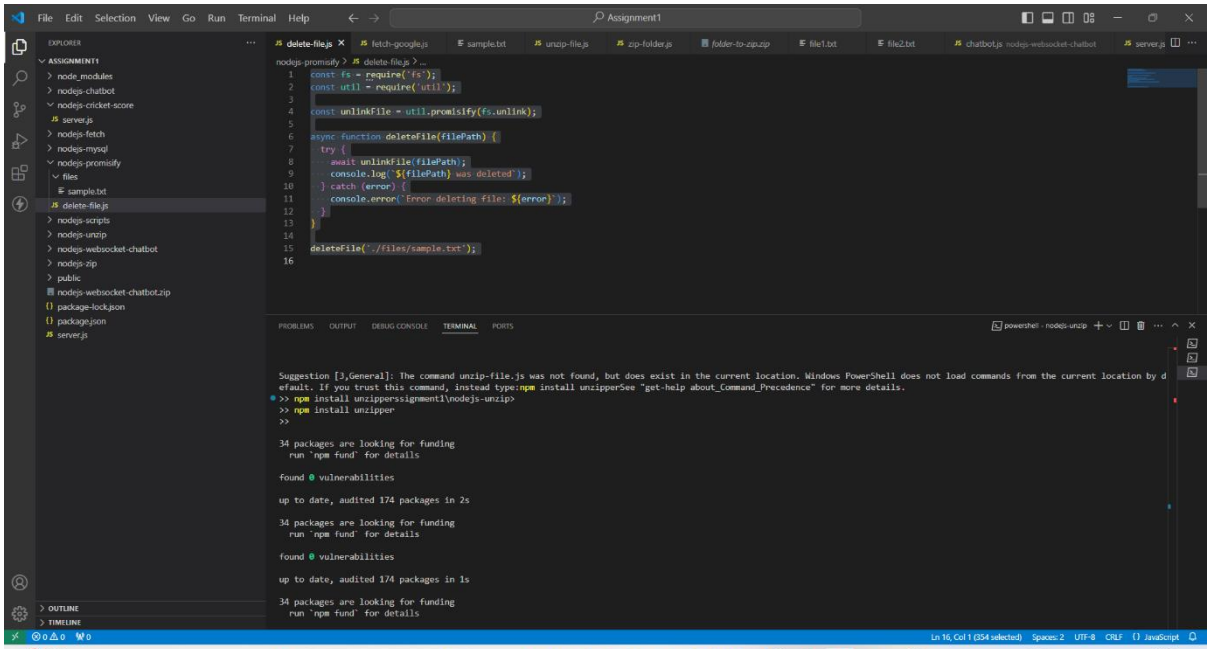


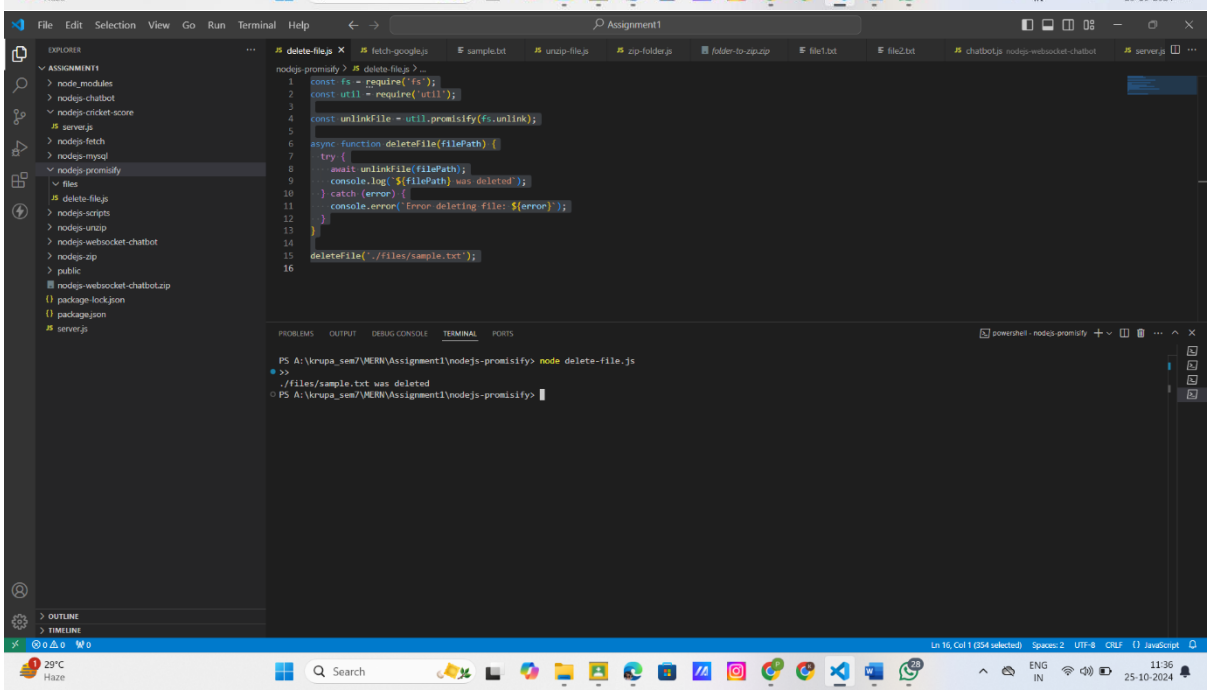


Question7
7. Write a program to promisify fs.unlink function and call it.

→delete-file.js

```
const fs = require('fs');

const util = require('util');


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


async function deleteFile(filePath) {
  try {
    await unlinkFile(filePath);
    console.log(`${filePath} was deleted`);
  } catch (error) {
    console.error(`Error deleting file: ${error}`);
  }
}


deleteFile('./files/sample.txt');
```





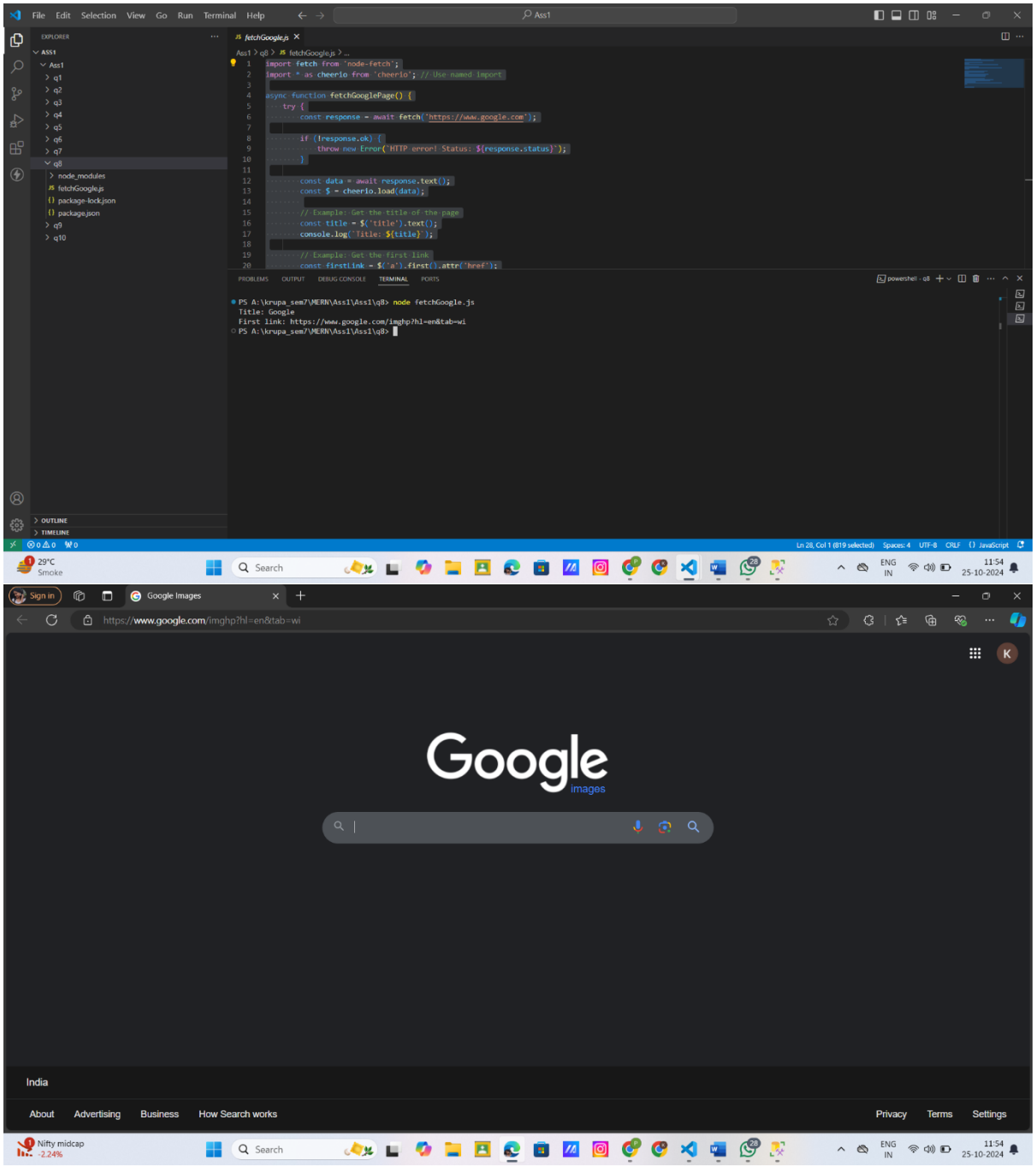
Question8

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

→fetchGoogle.js

```
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);

    // Example: Get the title of the page
    const title = $('title').text();
    console.log(`Title: ${title}`);

    // Example: Get the first link
    const firstLink = $('a').first().attr('href');
    console.log(`First link: ${firstLink}`);
  } catch (error) {
    console.error(`Error fetching Google page: ${error.message}`);
  }
}

fetchGooglePage();
```



Question10

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

→package.json

{

"name": "q10",

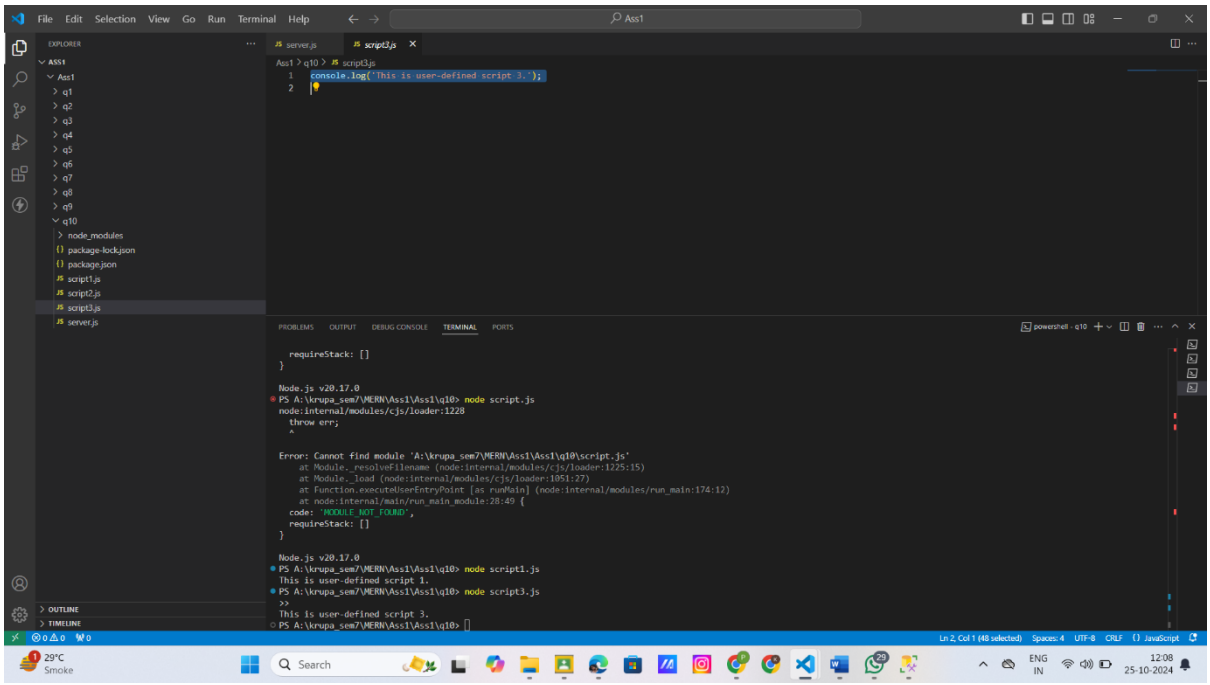
"version": "1.0.0",

"description": "Node.js Application with Scripts",

```
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node server.js",
    "test": "echo \"No tests specified\" && exit 0",
    "user-script-1": "node script1.js",
    "user-script-2": "node script2.js",
    "user-script-3": "node script3.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.21.1"
  }
}
```

→script3.js

```
console.log('This is user-defined script 3.');
```



Question11
→server.js

```
// 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}`);
});
```

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;
            background-color: #f9f9f9;
            color: #333;
        }
        h1 {
            color: #007bff;
            text-align: center;
            margin-bottom: 20px;
```

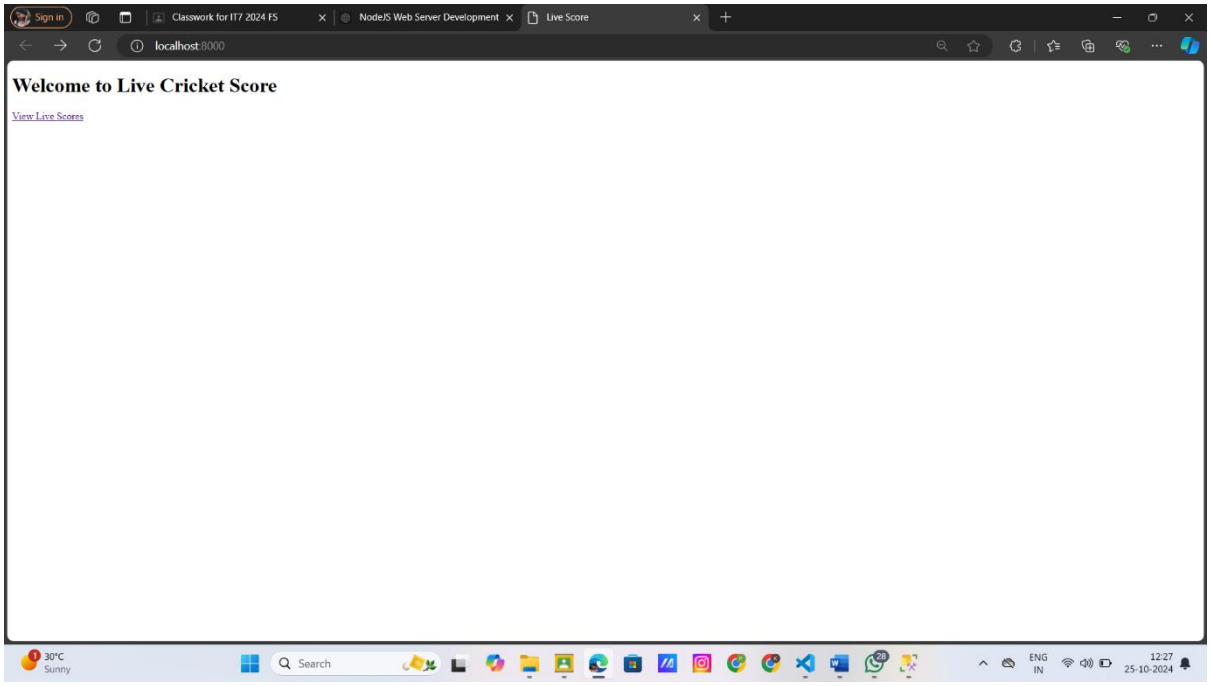
```
}
a {
    display: block;
    text-align: center;
    margin-bottom: 20px;
    text-decoration: none;
    color: #fff;
    background-color: #007bff;
    padding: 10px;
    border-radius: 5px;
    transition: background-color 0.3s;
}
a:hover {
    background-color: #0056b3;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    overflow: hidden;
}
th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}
th {
    background-color: #007bff;
    color: #fff;
```

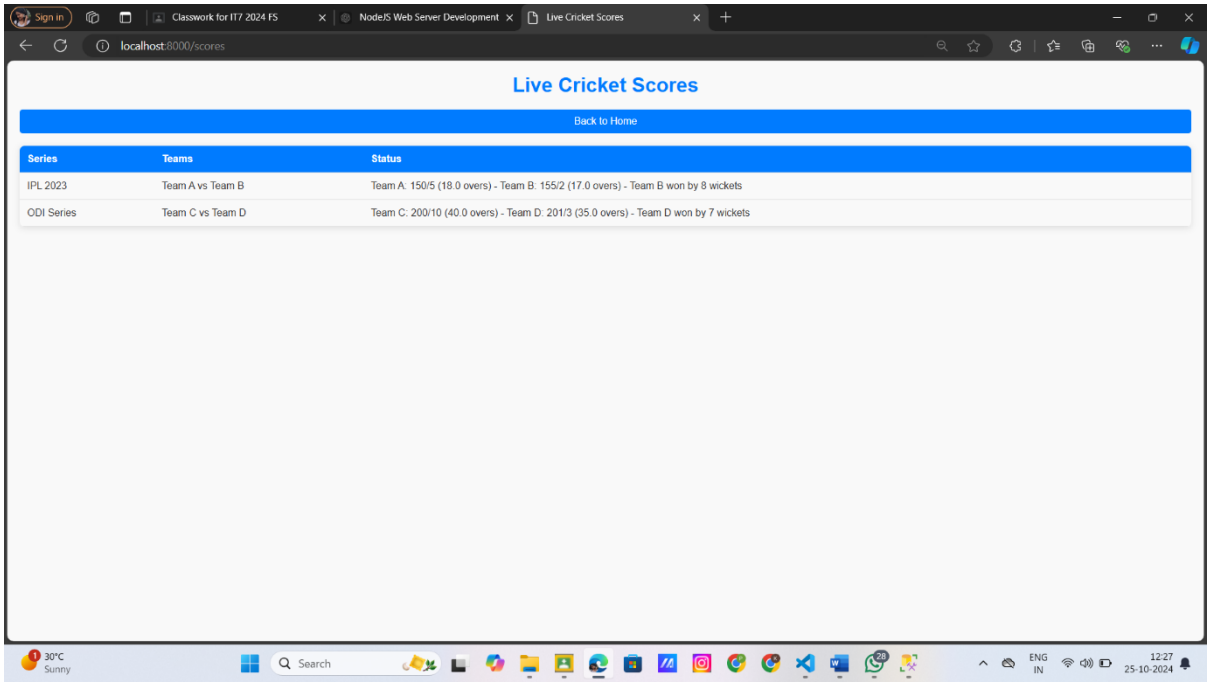
```
        }
        tr:hover {
            background-color: #f1f1f1;
        }
        .no-matches {
            text-align: center;
            padding: 20px;
            color: #888;
        }
    </style>
</head>
<body>
    <h1>Live Cricket Scores</h1>
    <a href="/">Back to Home</a>
    <table>
        <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" class="no-matches">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  Score</title>
</head>
<body>
    <h1>Welcome to Live Cricket Score</h1>
    <a href="/scores">View Live Scores</a>
</body>
</html>
```


Sign in
Classwork for IT7 2024 FS
NodeJS Web Server Development
Live Score
localhost:8000
Welcome to Live Cricket Score
View Live Scores
30°C
Sunny
Search
ENG
IN
12:27
25-10-2024


Sign in
Classwork for IT7 2024 FS
NodeJS Web Server Development
Live Cricket Scores
localhost:8000/scores
Live Cricket Scores
Back to Home
Series
Teams
Status
IPL 2023
Team A vs Team B
Team A: 150/5 (18.0 overs) - Team B: 155/2 (17.0 overs) - Team B won by 8 wickets
ODI Series
Team C vs Team D
Team C: 200/10 (40.0 overs) - Team D: 201/3 (35.0 overs) - Team D won by 7 wickets
30°C
Sunny
Search
ENG
IN
12:27
25-10-2024