RollNo : 07
Name : Shivangi N. Chavda
Semester : 7th
Subject : 705

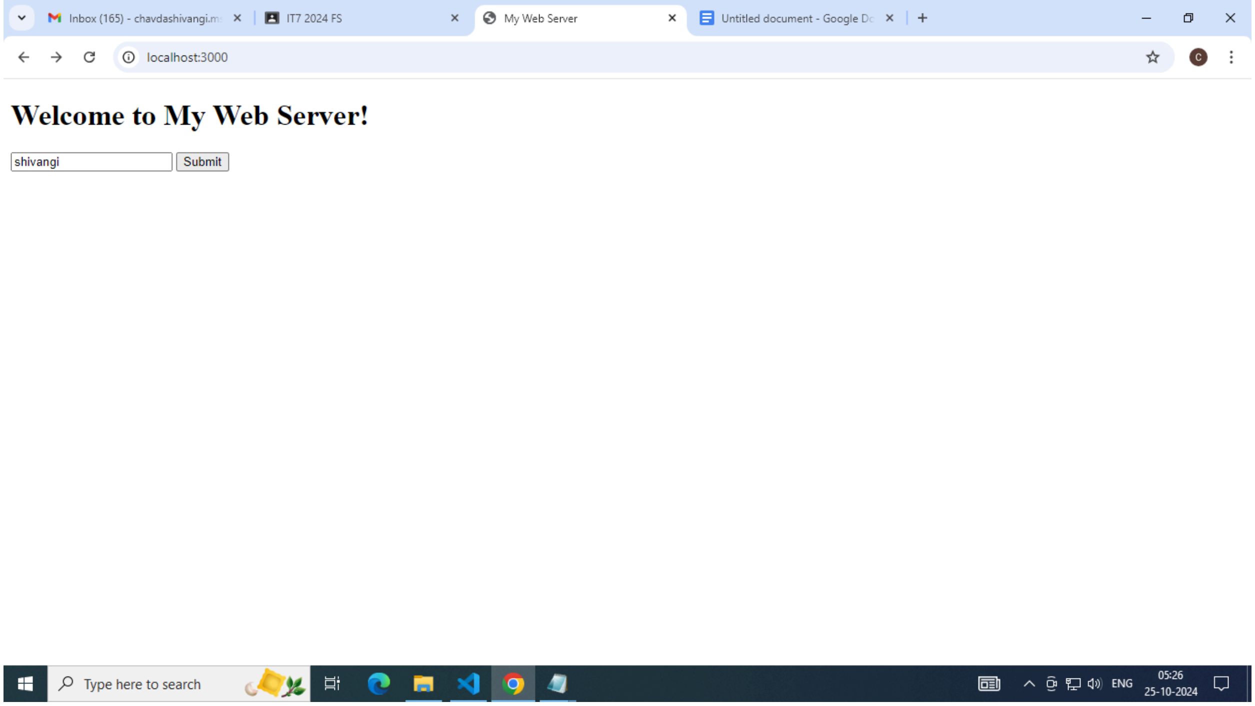
Assignment 1

Start Date : 25 october 2024

Q:1

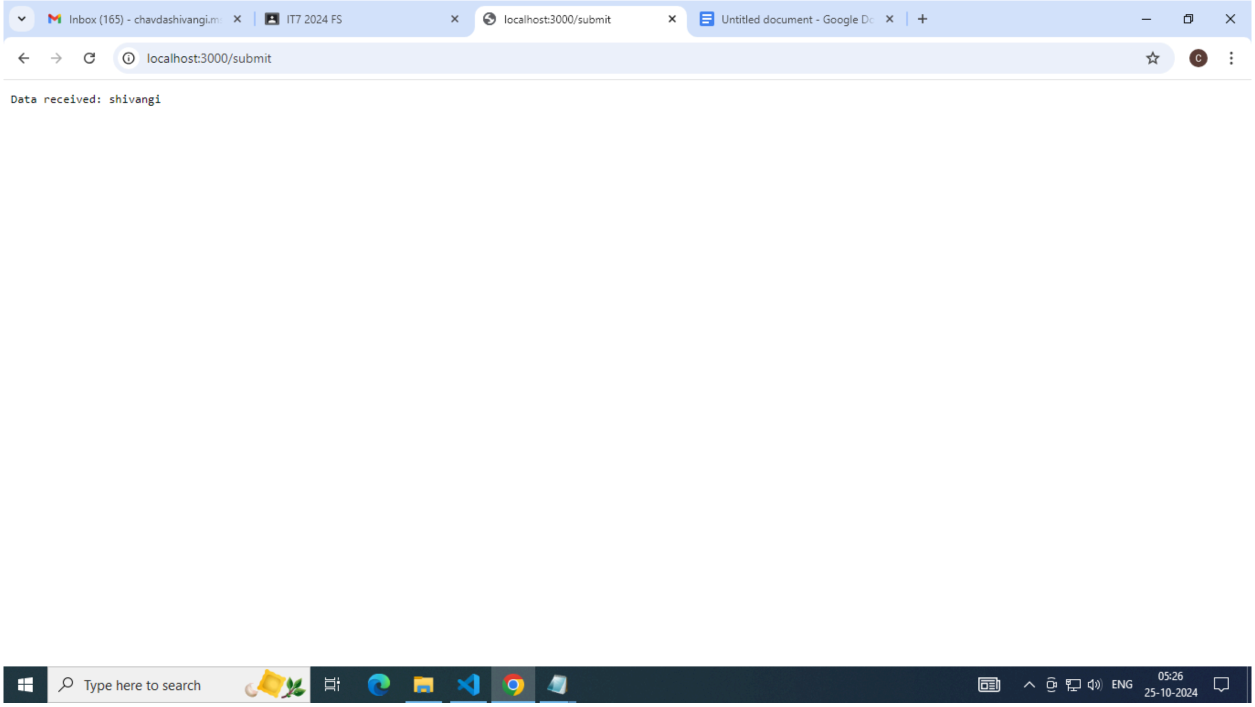
```
// sever.js

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('Data received: ' + 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}`);
});


//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 data" required>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```



Welcome to My Web Server!

shivangi

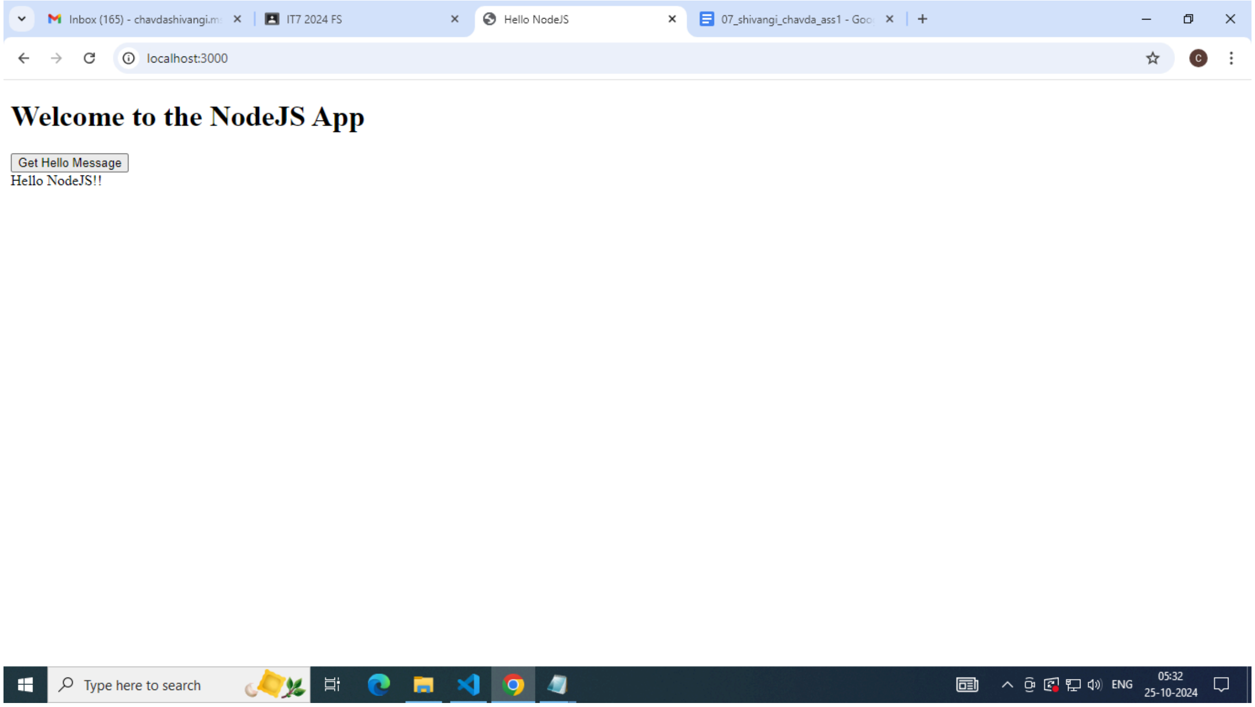


```
Data received: shivangi
```

Q:2

```
// server.js

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

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

// Serve static files from the public directory
app.use(express.static('public'));

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

// Serve the HTML page
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// Start the server
app.listen(PORT, () => {
    console.log(`Server is listening on http://localhost:${PORT}`);
});

//index.hrml

<!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 Hello 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>
```



Welcome to the NodeJS App

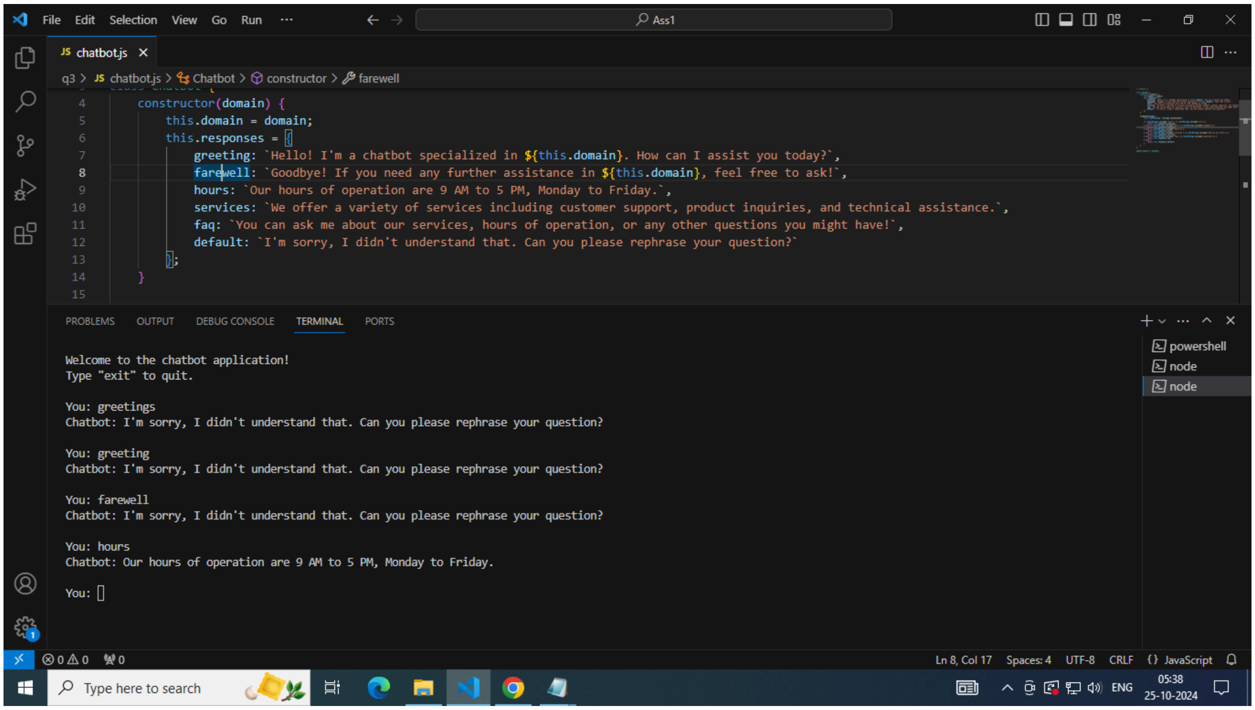
Get Hello Message

Hello NodeJS!!

Q:3

```
// 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;
```

```
// app.js

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

// Initialize the chatbot with a specific domain
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(); // Ask the next question
    });
};

// Start the conversation
askQuestion();
```



File Edit Selection View Go Run ···
Ass1
chatbot.js
q3 > chatbot.js > Chatbot > constructor > farewell
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
node
node
Ln 8, Col 17 Spaces: 4 UTF-8 CRLF {} JavaScript
Type here to search
ENG
05:38
25-10-2024

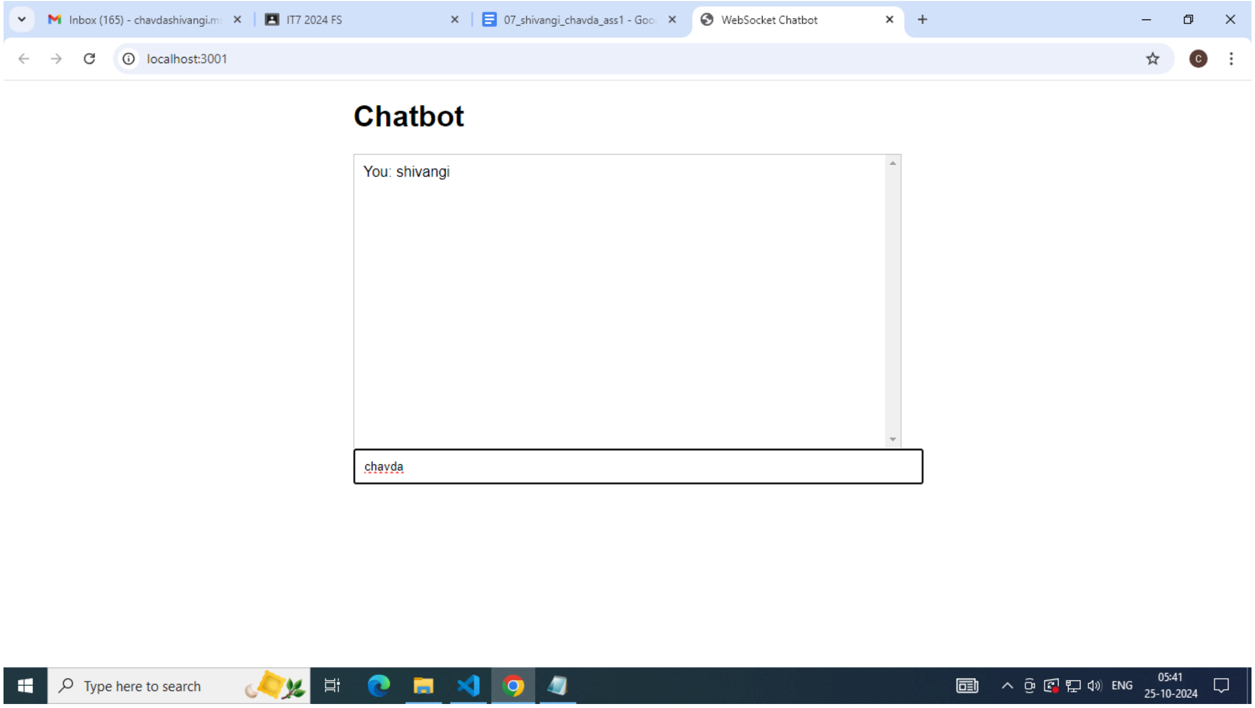
```
    constructor(domain) {
        this.domain = domain;
        this.responses = {
            greeting: `Hello! I'm a chatbot specialized in ${this.domain}. How can I assist you today?`,
            farewell: `Goodbye! If you need any further assistance in ${this.domain}, feel free to ask!`,
            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?`
        };
    }
```

```
Welcome to the chatbot application!
Type "exit" to quit.

You: greetings
Chatbot: I'm sorry, I didn't understand that. Can you please rephrase your question?

You: greeting
Chatbot: I'm sorry, I didn't understand that. Can you please rephrase your question?

You: farewell
Chatbot: I'm sorry, I didn't understand that. Can you please rephrase your question?

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

You: 
```

Q:4

```
// 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');

// Serve static files from the public directory
app.use(express.static('public'));

// Handle WebSocket connections
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');
  });
});

// Start the server
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;

// 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');
```

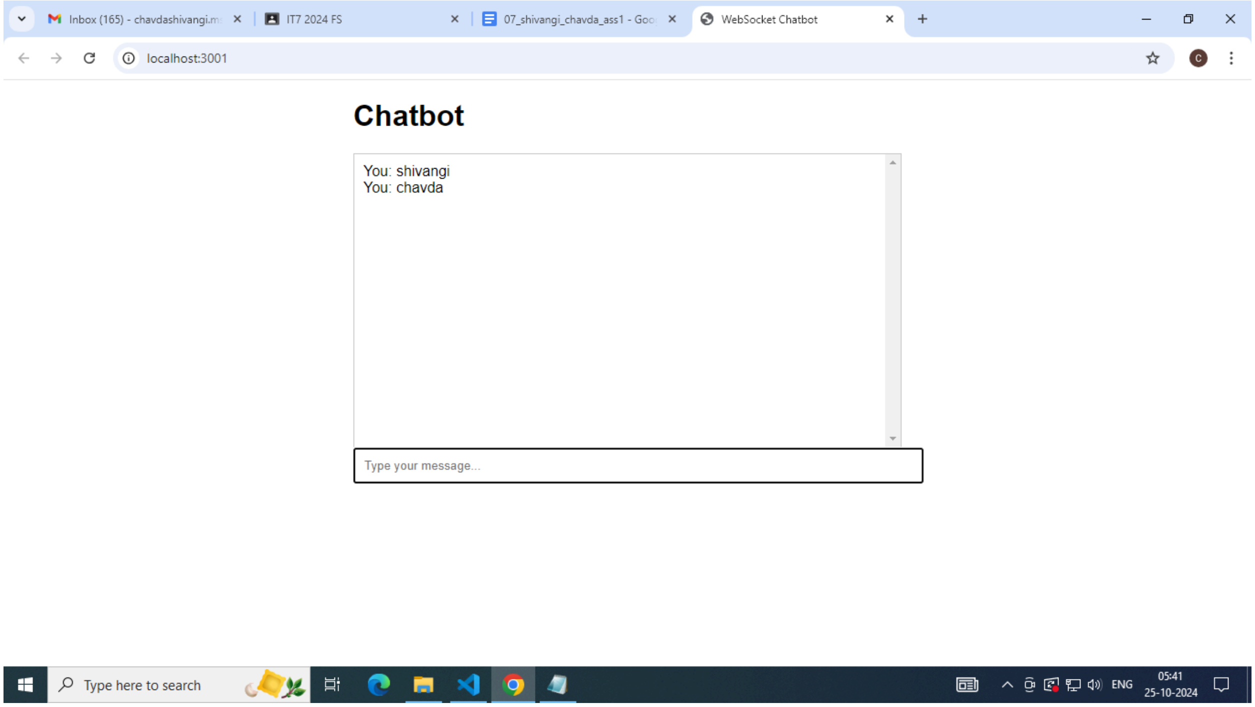
```
      message.textContent = `Chatbot: ${event.data}`;
      messagesDiv.appendChild(message);
      messagesDiv.scrollTop = messagesDiv.scrollHeight; // Scroll to the bottom
    };

    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>
```



Chatbot

You: shivangi

chavda



Chatbot

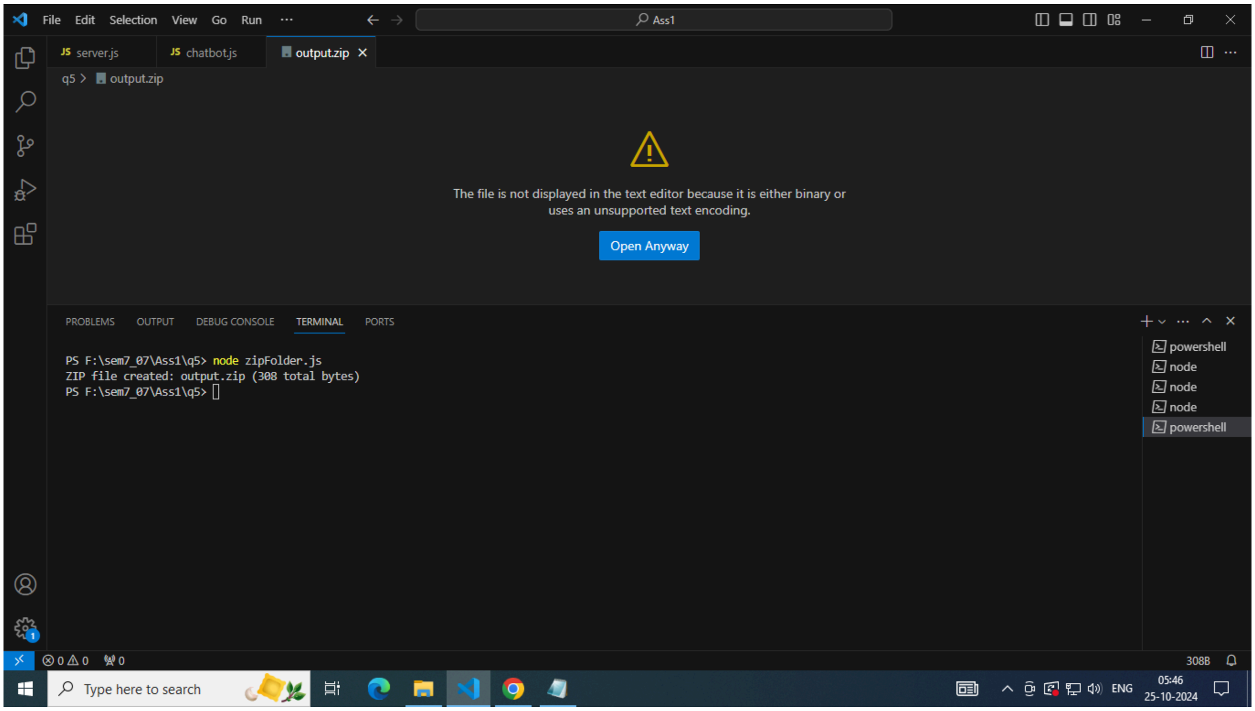
You: shivangi
You: chavda

Type your message...

Q:5

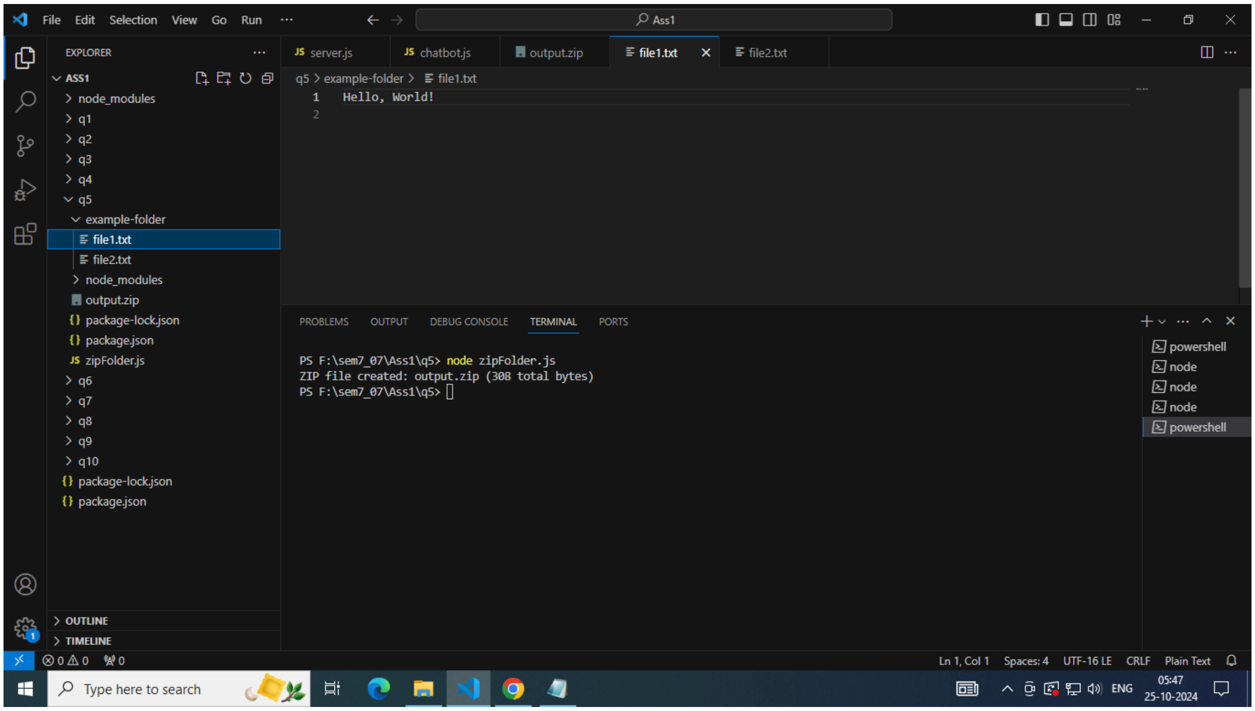
```
// zipFolder.js

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 = 'example-folder'; // Change this to the folder you want to zip
const outputZipPath = 'output.zip'; // Name of the output zip file

zipFolder(folderToZip, outputZipPath);
```



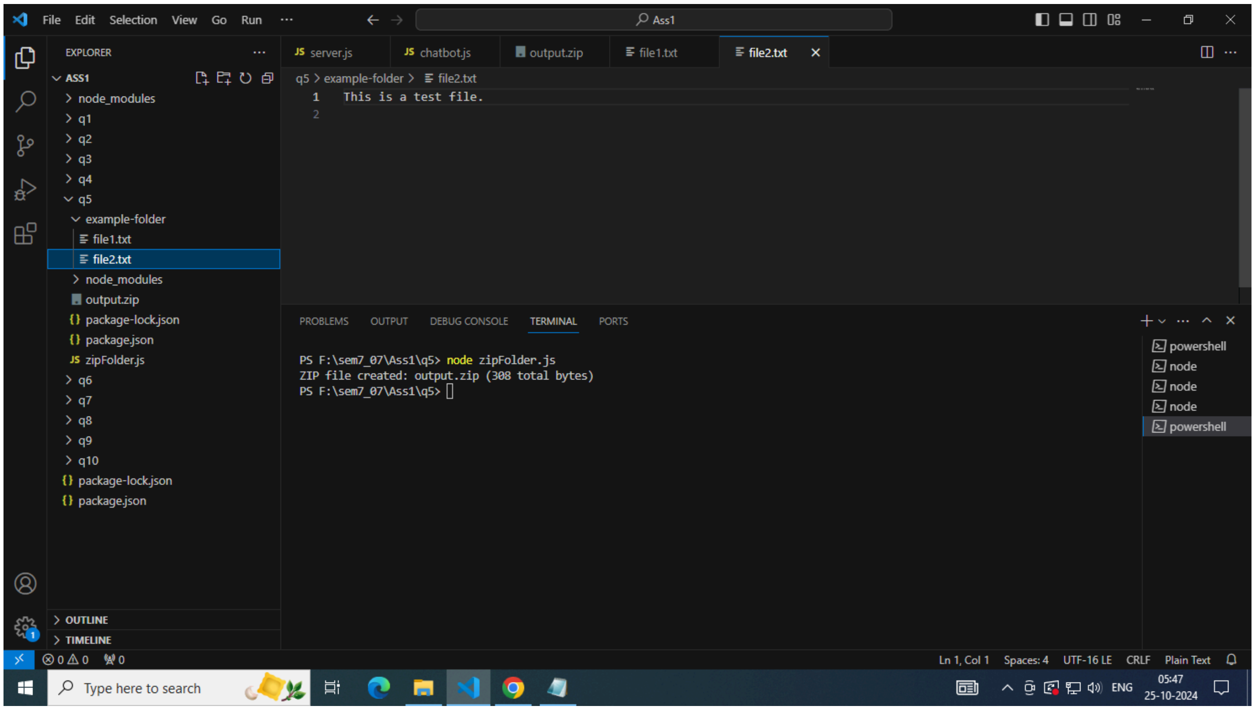
The file is not displayed in the text editor because it is either binary or uses an unsupported text encoding.

```
PS F:\sem7_07\Ass1\q5> node zipFolder.js
ZIP file created: output.zip (308 total bytes)
PS F:\sem7_07\Ass1\q5> 
```



```
Hello, World!
```

```
PS F:\sem7_07\Ass1\q5> node zipFolder.js
ZIP file created: output.zip (308 total bytes)
PS F:\sem7_07\Ass1\q5> 
```

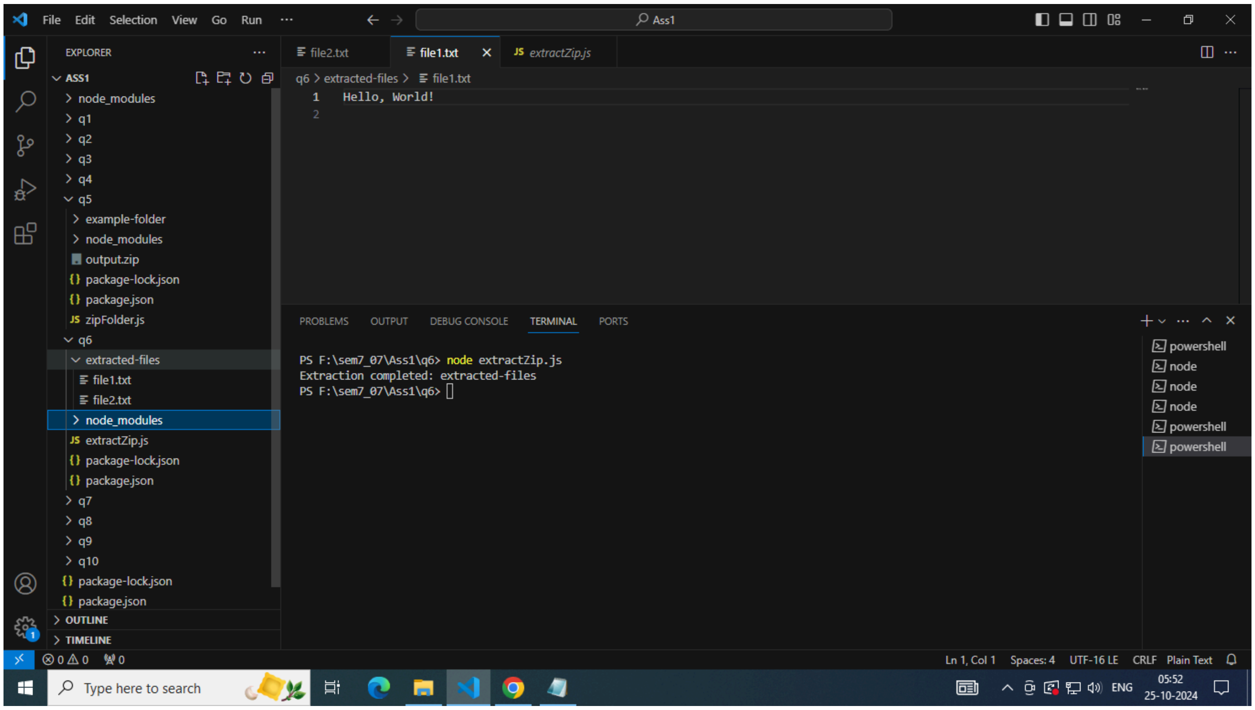


```
This is a test file.
```

```
PS F:\sem7_07\Ass1\q5> node zipFolder.js
ZIP file created: output.zip (308 total bytes)
PS F:\sem7_07\Ass1\q5> 
```

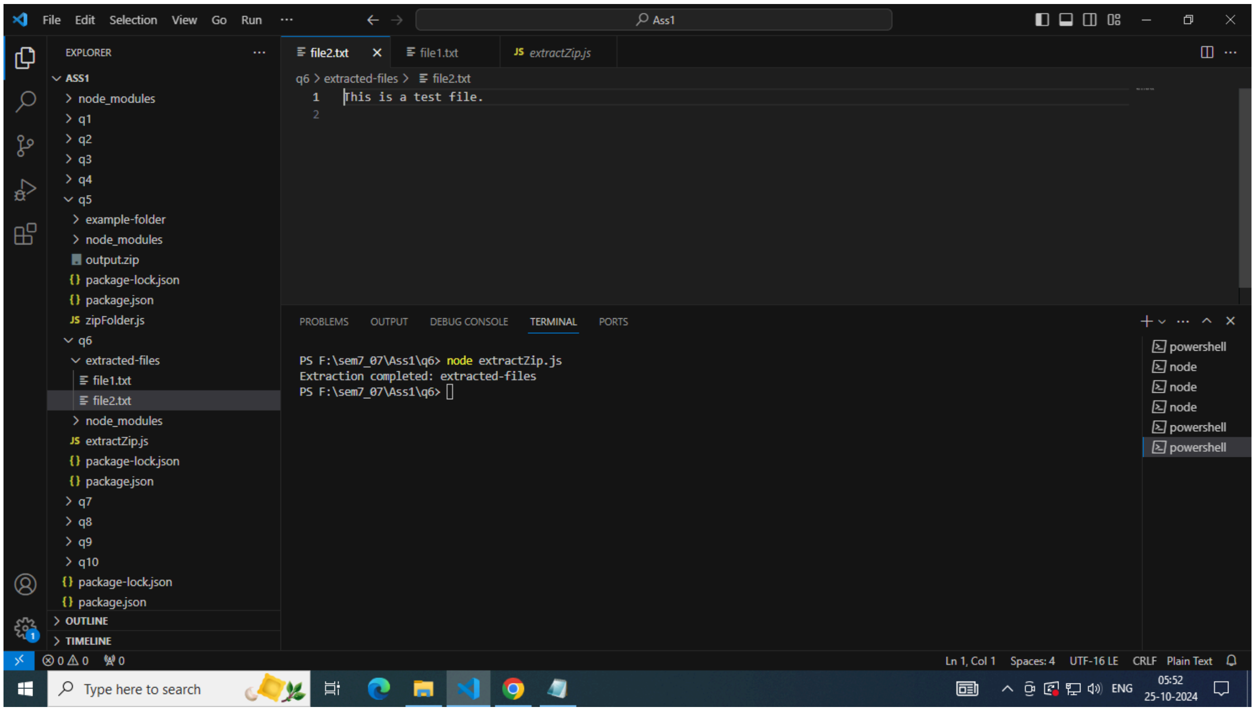
Q:6

```
// extractZip.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 = '../q5/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);
```



```
Hello, World!
```

```
PS F:\sem7_07\Ass1\q6> node extractZip.js
Extraction completed: extracted-files
PS F:\sem7_07\Ass1\q6> 
```

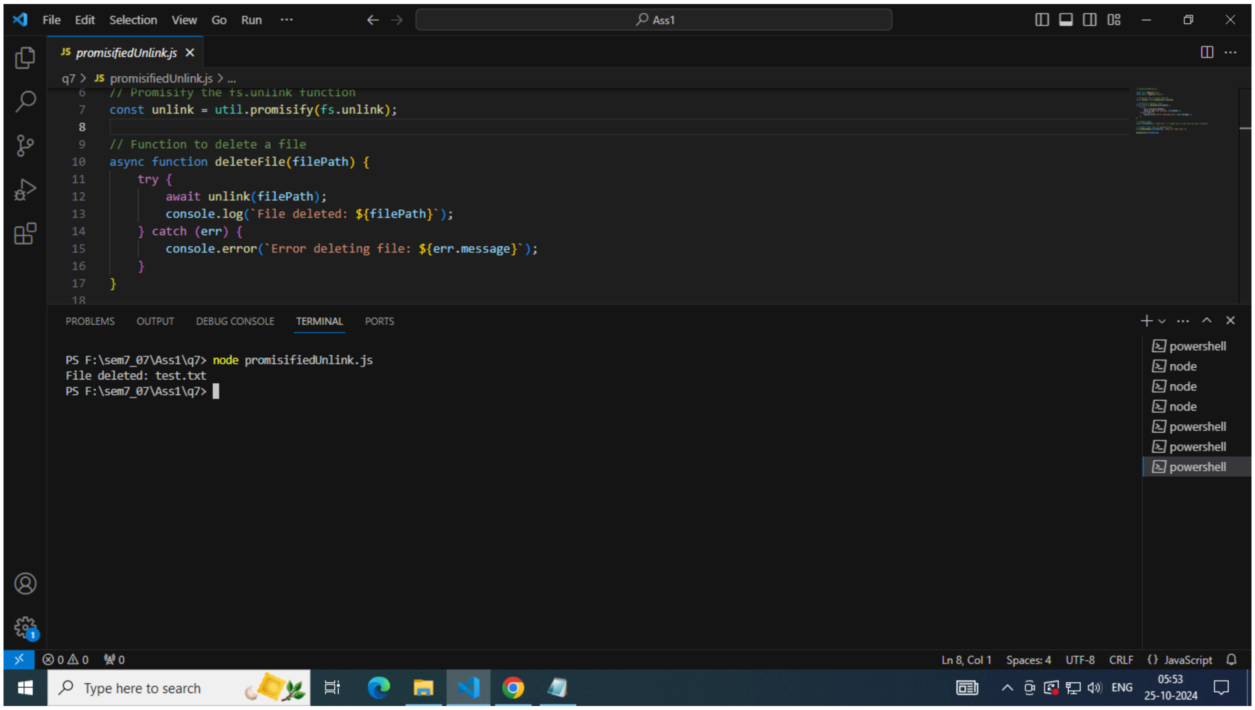


```
This is a test file.
```

```
PS F:\sem7_07\Ass1\q6> node extractZip.js
Extraction completed: extracted-files
PS F:\sem7_07\Ass1\q6> 
```

Q:7

```
// promisifiedUnlink.js

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

// Promisify the fs.unlink function
const unlink = util.promisify(fs.unlink);

// Function to delete a file
async function deleteFile(filePath) {
    try {
        await unlink(filePath);
        console.log(`File deleted: ${filePath}`);
    } catch (err) {
        console.error(`Error deleting file: ${err.message}`);
    }
}

// Example usage
const fileToDelete = 'test.txt'; // Change this to the file you want to delete

// Create a test file for demonstration
fs.writeFileSync(fileToDelete, 'This is a test file.');

deleteFile(fileToDelete);
```



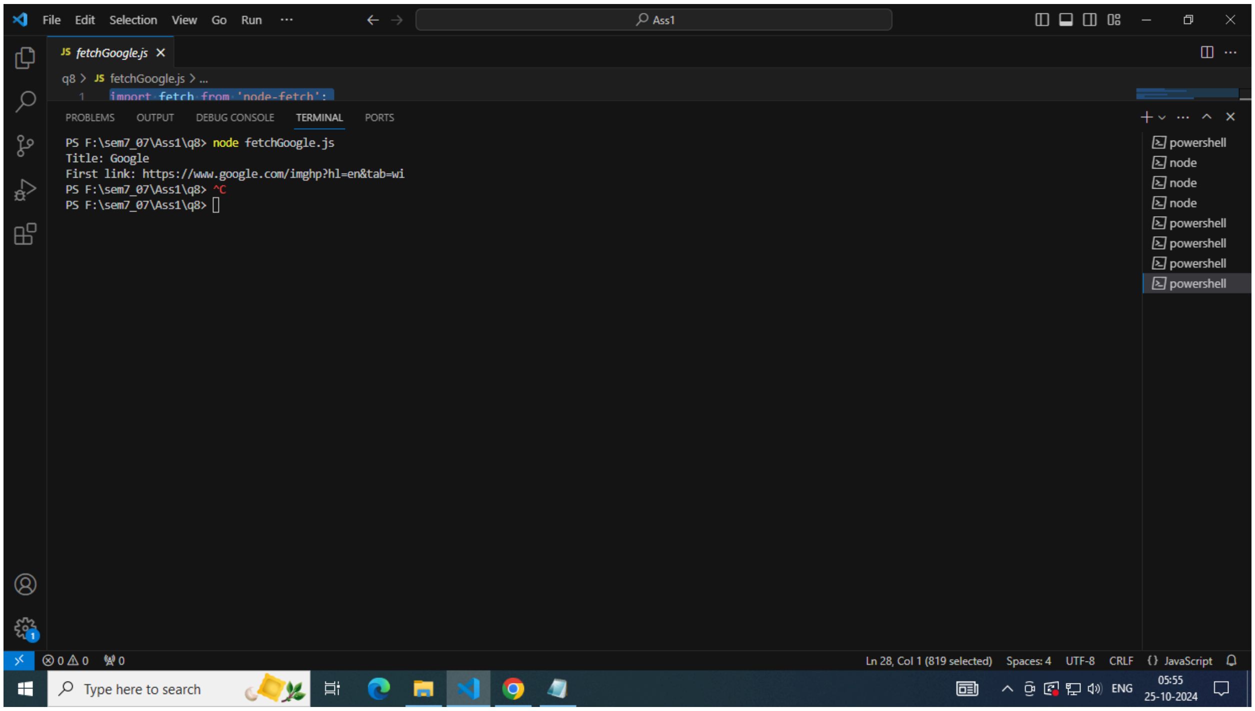
```
// Promisify the fs.unlink function
const unlink = util.promisify(fs.unlink);

// Function to delete a file
async function deleteFile(filePath) {
    try {
        await unlink(filePath);
        console.log(`File deleted: ${filePath}`);
    } catch (err) {
        console.error(`Error deleting file: ${err.message}`);
    }
}
```

```
PS F:\sem7_07\Ass1\q7> node promisifiedUnlink.js
File deleted: test.txt
PS F:\sem7_07\Ass1\q7>
```

Q:8

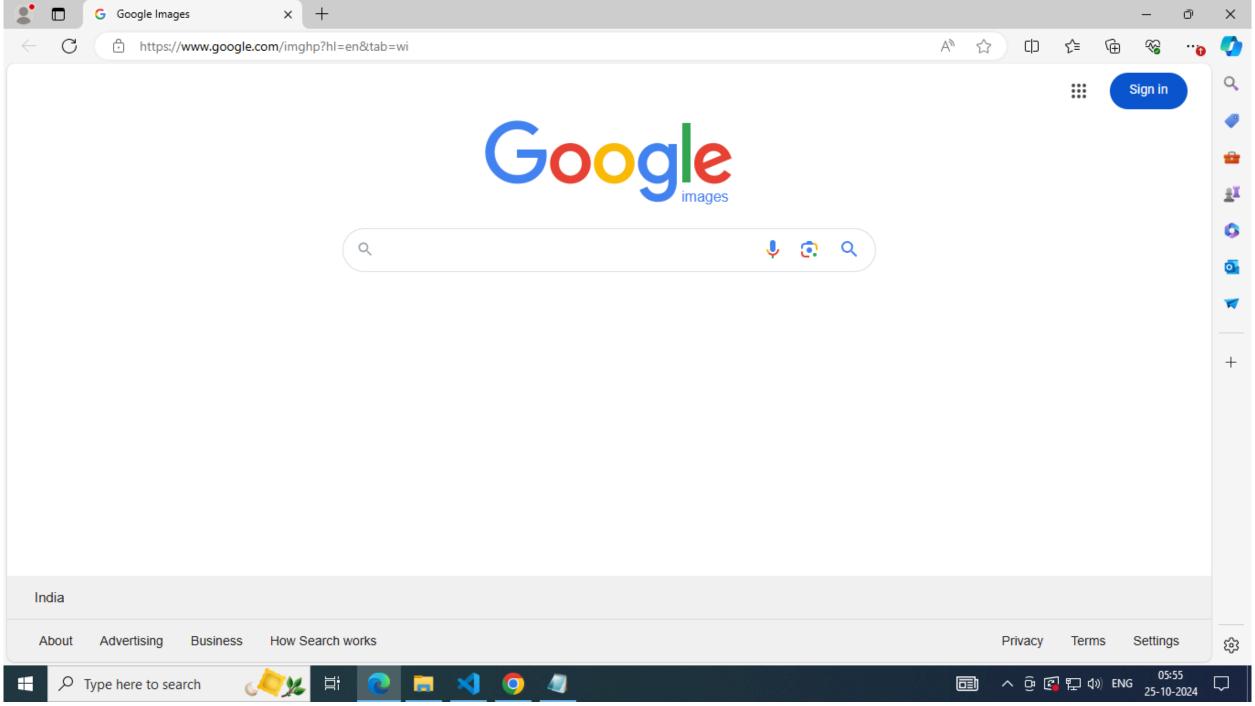
//fetchGoogle.js

```
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();
```



```
import fetch from 'node-fetch';
```

```
PS F:\sem7_07\Ass1\q8> node fetchGoogle.js
Title: Google
First link: https://www.google.com/imghp?hl=en&tab=wi
PS F:\sem7_07\Ass1\q8> ^C
PS F:\sem7_07\Ass1\q8> 
```



Q:9

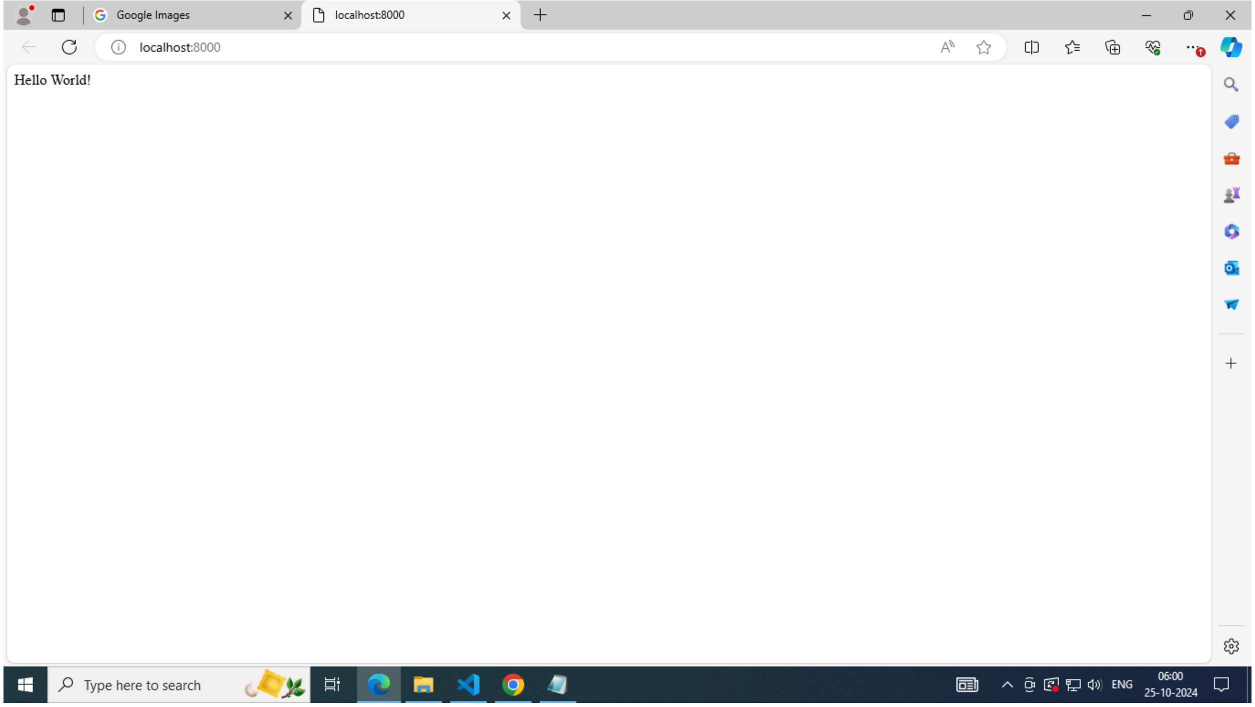
app.js

```
const mysql = require('mysql2/promise');


const dbConfig = {
    host: 'localhost',
    user: 'yourUsername',
    password: 'yourPassword',
    database: 'company'
};

async function connectDB() {
    const connection = await mysql.createConnection(dbConfig);
    console.log('Connected to the database.');
    return connection;
}

async function insertEmployee(connection, name, position, salary) {
    const query = 'INSERT INTO employee (name, position, salary) VALUES (?, ?, ?)';
    await connection.execute(query, [name, position, salary]);
    console.log('Employee record inserted.');
}


async function displayEmployees(connection) {
    const [rows] = await connection.execute('SELECT * FROM employee');
    console.log('Employee Records:');
    console.table(rows);
}


async function main() {
    const connection = await connectDB();

    try {

        await insertEmployee(connection, 'John Doe', 'Developer', 60000);


        await displayEmployees(connection);
    } catch (error) {
        console.error('Error:', error);
    } finally {
        await connection.end();
        console.log('Connection closed.');
    }
}


main();
```

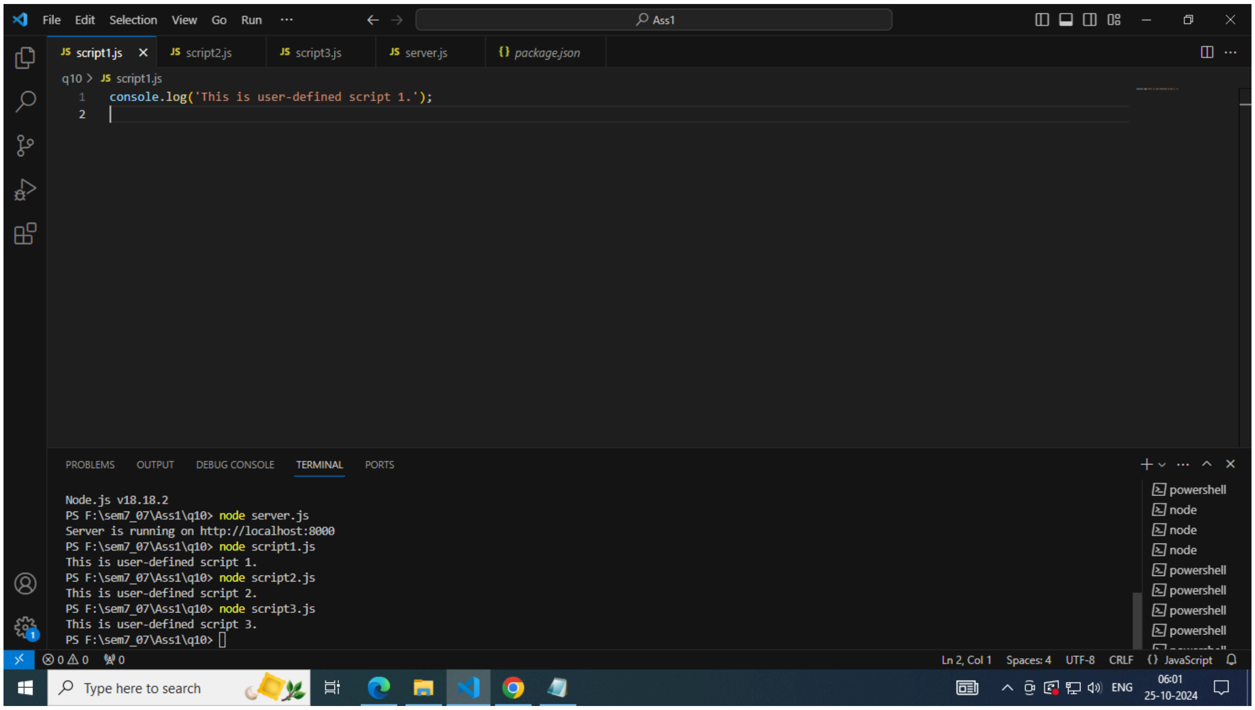
Q:10

```
//script1.js
console.log('This is user-defined script 1.');

//script2.js
console.log('This is user-defined script 2.');

//script3.js
console.log('This is user-defined script 3.');

//server.js
```

```
import express from 'express';

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

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

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



```
Hello World!
```



```
console.log('This is user-defined script 1.');

```

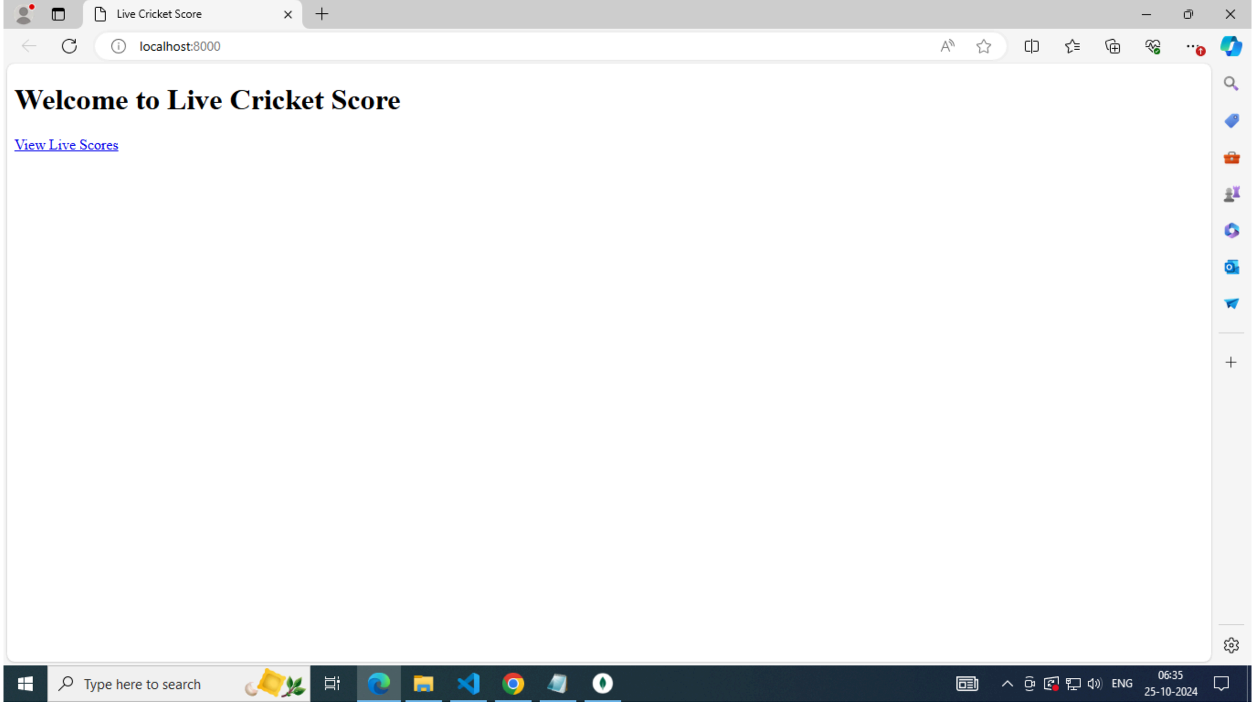
```
Node.js v18.18.2
PS F:\sem7_07\Ass1\q10> node server.js
Server is running on http://localhost:8000
PS F:\sem7_07\Ass1\q10> node script1.js
This is user-defined script 1.
PS F:\sem7_07\Ass1\q10> node script2.js
This is user-defined script 2.
PS F:\sem7_07\Ass1\q10> node script3.js
This is user-defined script 3.
PS F:\sem7_07\Ass1\q10> 
```

Q:11

//server.js

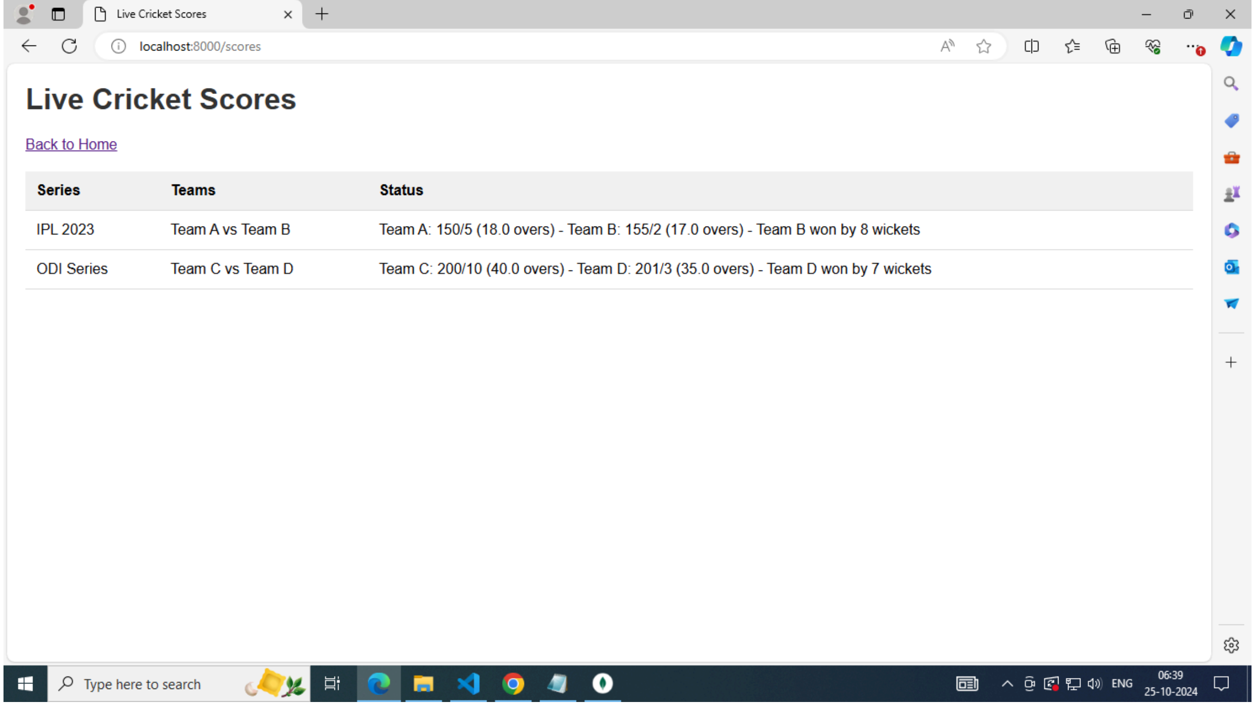
```
// server.js
const express = require('express');
const app = express();
const PORT = process.env.PORT || 8000;

// 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}`);
});

//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>

//scores.js

<!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>
```



Welcome to Live Cricket Score

View Live 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