Name: Bhatt Srushti Daxeshkumar
Roll no: 006
Sem: 7th Sem (BSc-IT)
Subject: Application Development using Full Stack

Practical Assignment: 1

1. Develop a web server with following functionalities:
    a. Serve static resources.
    b. Handle GET request.
    c. Handle POST request.

Code:

File1.html

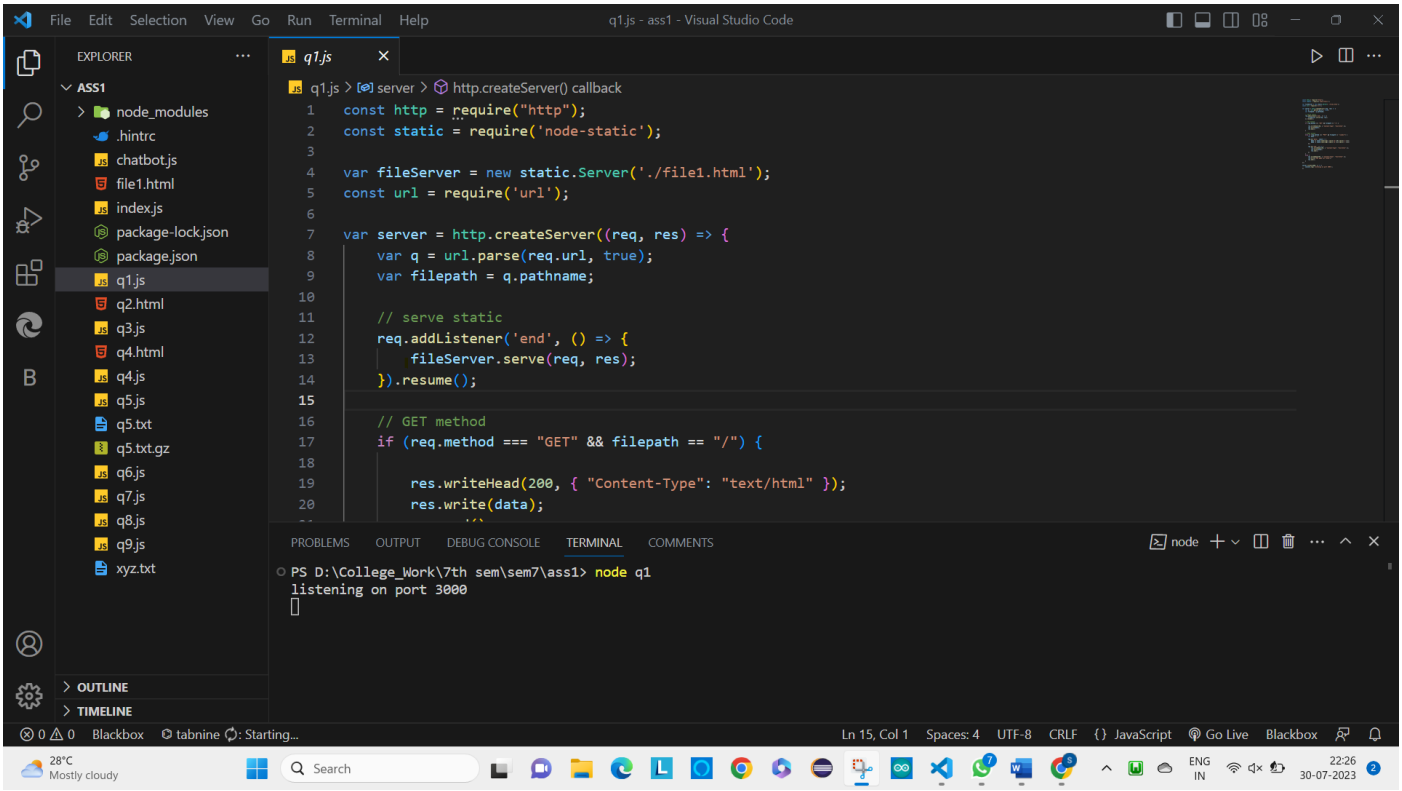
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Document</title>
    </head>
    <body>
        <form method="GET" action="/submit">
            <h1>GET METHOD</h1>
            <label>First Name: </label><input type="text" name="fname"
                placeholder="First Name"><br><br>
            <label>Last Name: </label><input type="text" name="lname"
                placeholder="Last Name"><br><br>
            <input type="submit" value="Submit" name="submit">
        </form>
        <br>
        <br>
        <br>
        <h1>POST METHOD</h1>
        <form method="POST" action="/submit">
            <label>First Name: </label><input type="text" name="fname"
                placeholder="First Name"><br><br>
            <label>Last Name: </label><input type="text" name="lname"
                placeholder="Last Name"><br><br>
            <input type="submit" value="Submit" name="submit">
        </form>
    </body>
</html>
```

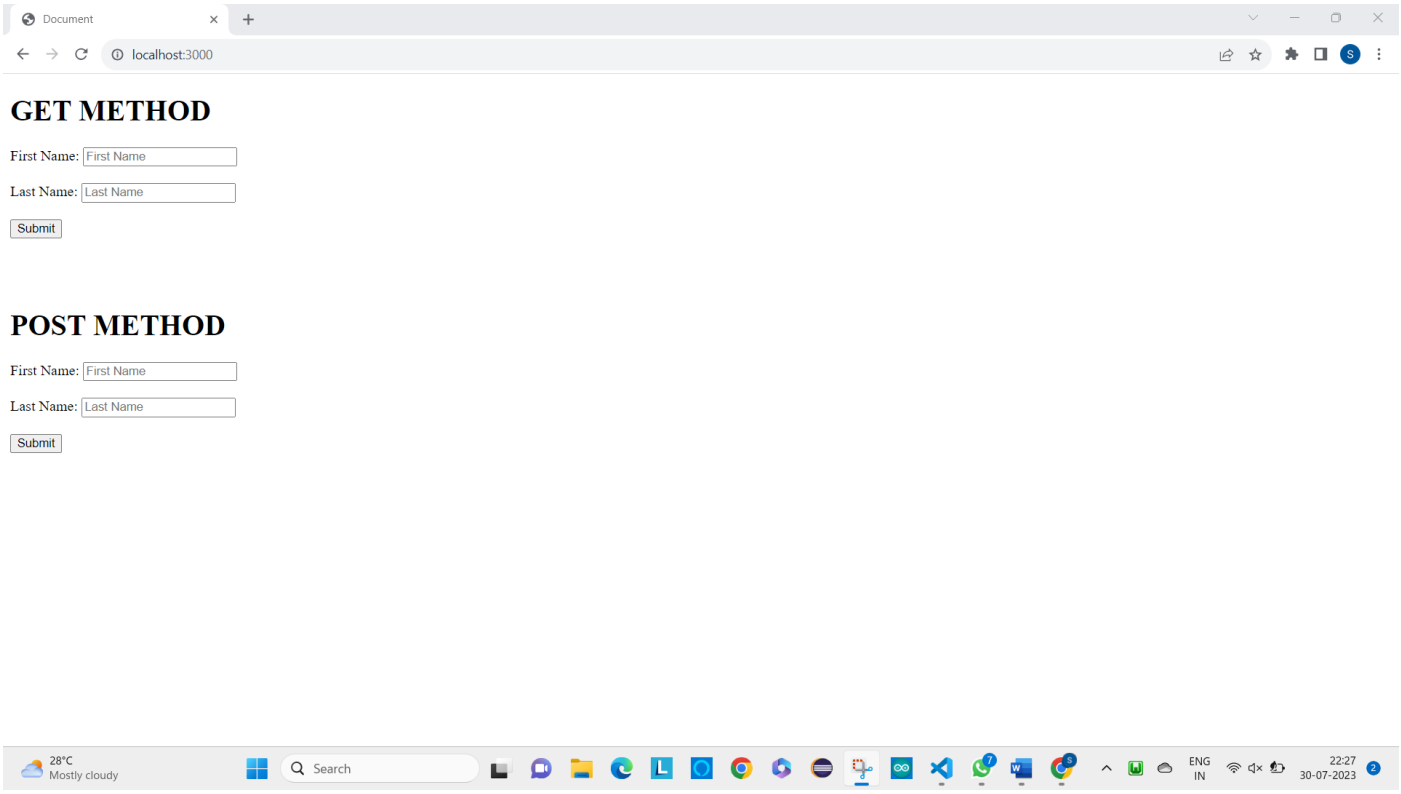
Q1.js

```javascript
const http = require("http");
const static = require('node-static');

var fileServer = new static.Server('./file1.html');
const url = require('url');

var server = http.createServer((req, res) => {
    var q = url.parse(req.url, true);
    var filepath = q.pathname;

    // serve static
```

```
    req.addListener('end', () => {
        fileServer.serve(req, res);
    }).resume();

    // GET method
    if (req.method === "GET" && filepath == "/") {

        res.writeHead(200, { "Content-Type": "text/html" });
        res.write(data);
        res.end();
    }

    // POST method
    else if (req.method === "POST" && filepath == "/submit") {
        let body = '';

        req.on('data', chunk => {
            body += chunk.toString().split('&')[0].split('=')[1];
            body += chunk.toString().split('&')[1].split('=')[1];
        });

        req.on('end', () => {
            res.writeHead(200, { "Content-Type": "text/html" });
            res.write(body);
            res.end();
        })
    }
    else {
        res.writeHead(404, { "Content-Type": "text/html" });
        res.write("404 page not found");
        res.end();
    }
});

server.listen(3000, () => {
    console.log(`listening on port 3000`);
})
```

Output:


q1.js - ass1 - Visual Studio Code
EXPLORER
ASS1
node_modules
.hintrc
chatbot.js
file1.html
index.js
package-lock.json
package.json
q1.js
q2.html
q3.js
q4.html
q4.js
q5.js
q5.txt
q5.txt.gz
q6.js
q7.js
q8.js
q9.js
xyz.txt
q1.js > server > http.createServer() callback
const http = require("http");
const static = require('node-static');
var fileServer = new static.Server('./file1.html');
const url = require('url');
var server = http.createServer((req, res) => {
var q = url.parse(req.url, true);
var filepath = q.pathname;
// serve static
req.addListener('end', () => {
fileServer.serve(req, res);
}).resume();
// GET method
if (req.method === "GET" && filepath == "/") {
res.writeHead(200, { "Content-Type": "text/html" });
res.write(data);
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS
PS D:\College_Work\7th sem\sem7\ass1> node q1
listening on port 3000


Document
localhost:3000
GET METHOD
First Name: First Name
Last Name: Last Name
Submit
POST METHOD
First Name: First Name
Last Name: Last Name
Submit

localhost:3000/submit?fname=Sr

localhost:3000/submit?fname=Srushti&lname=Bhatt&submit=Submit

28°C
Mostly cloudy

Search

ENG
IN

22:28
30-07-2023

localhost:3000/submit

localhost:3000/submit

SrushtiBhatt

28°C
Mostly cloudy

Search

ENG
IN

22:29
30-07-2023

2. Develop nodejs application with following requirements:
    a. Develop a route "/gethello" with GET method. It displays "Hello NodeJS!!" as response.
    b. Make an HTML page and display.
    c. Call "/gethello" route from HTML page using AJAX call. (Any frontend AJAX call API can be used.)

Code:

Q2.html

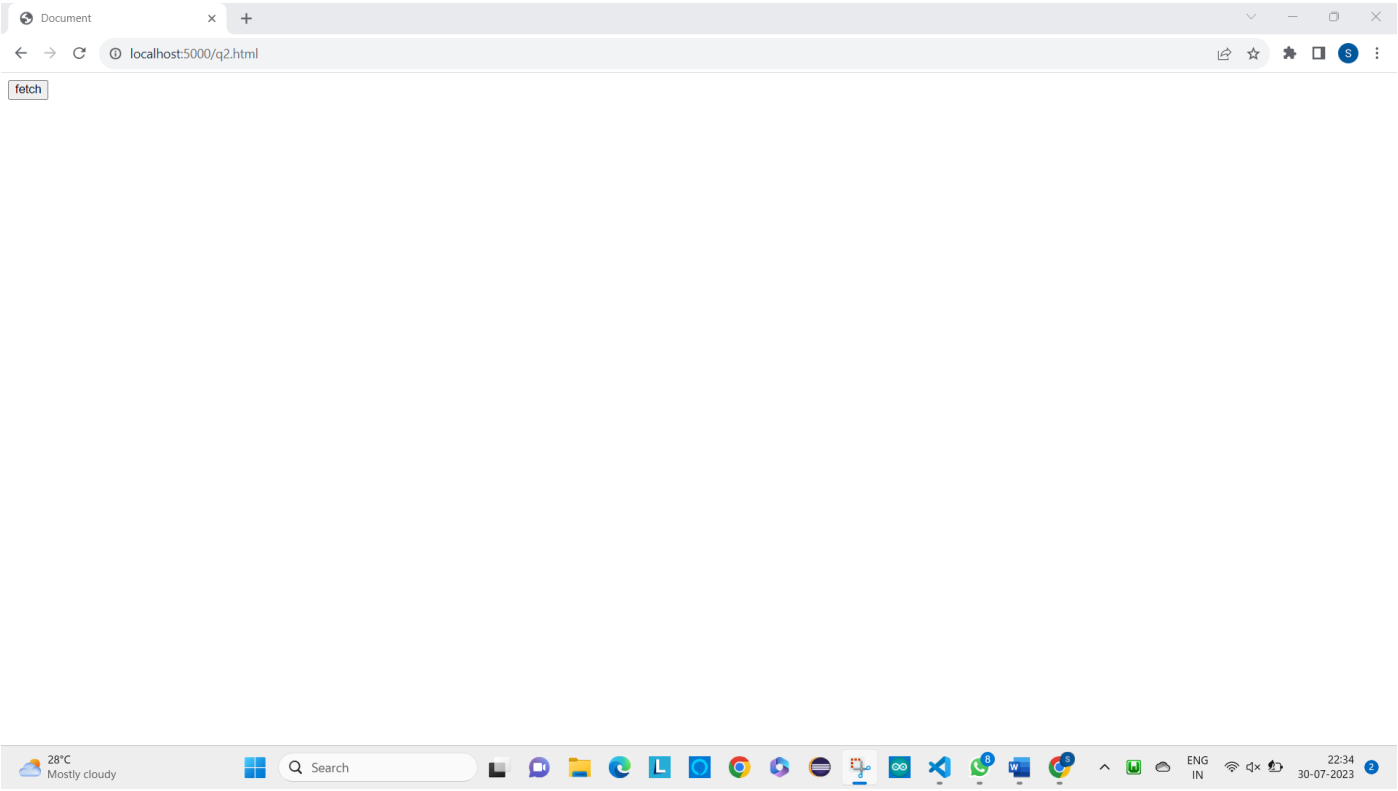
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Document</title>
        <script
            src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></scrip
t>

    </head>
    <body>
        <input type="submit" onclick="f1();" value="fetch">
        <div id="result"> </div>
    </body>
    <script>
        function f1(){
            $.ajax({
                url:'/gethello',
                method:'GET',
                success:function(data){
                    $('#result').text(data);
                }
            })
        }
    </script>
</html>
```

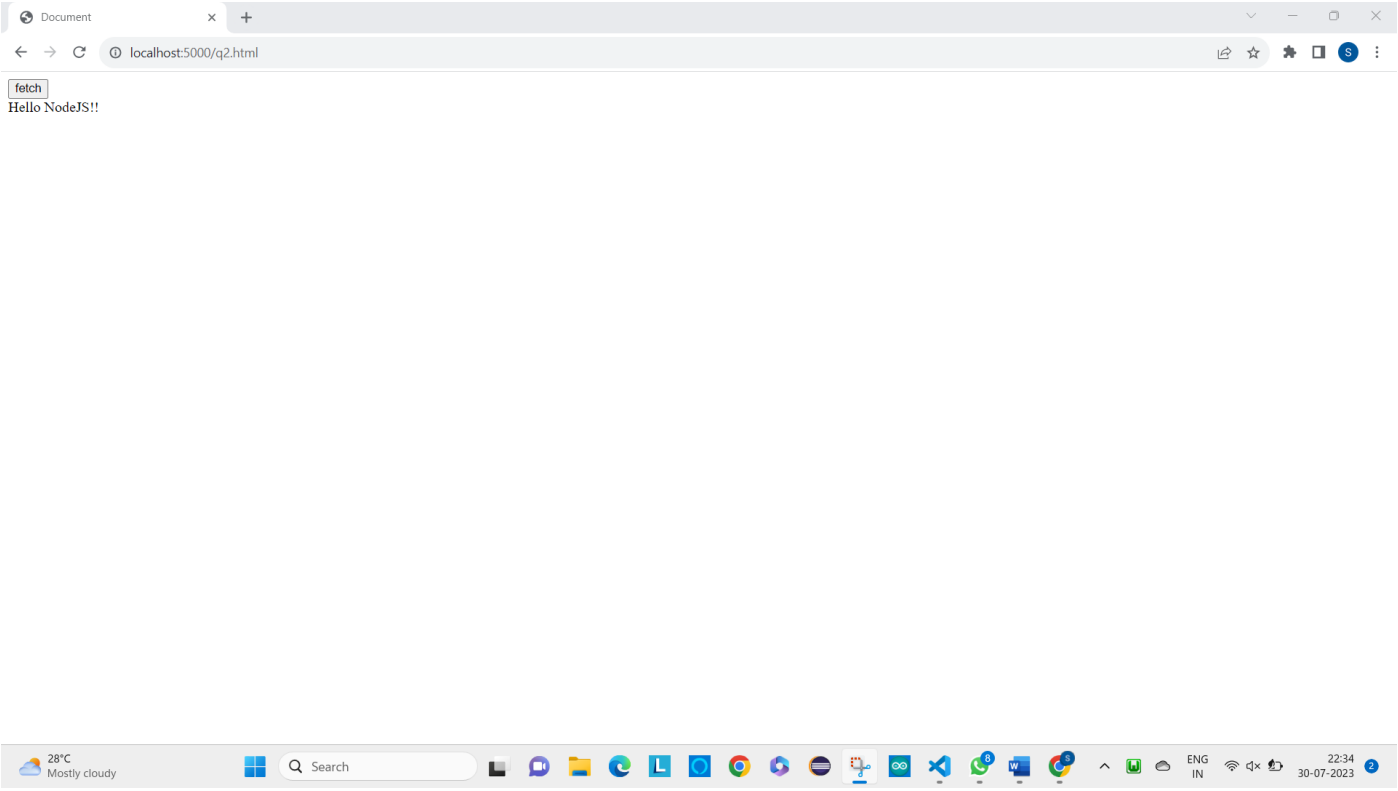
Index.js

```javascript
const http = require("http");
const fs = require('fs');

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

    // GET method
    if (req.method === "GET" && req.url === "/gethello") {
        res.writeHead(200, { "Content-Type": "text/plain" });
        res.write("Hello NodeJS!!");
```

```
        res.end();
    }
    else if (req.method === "GET" && req.url === "/") {
        res.writeHead(404, { "Content-Type": "text/plain" });
        res.write("404 page not found");
        res.end();
    }

    fs.readFile('q2.html', null, function (err, data) {
        if (err) {
            res.writeHead(404);
            res.write('file Not found!!');

        }
        else {
            res.writeHead(200);
            res.write(data);
        }
        res.end();
    })


});

server.listen(5000, () => {
    console.log(`listening on port 5000`);
})
```

Output:





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

Code:

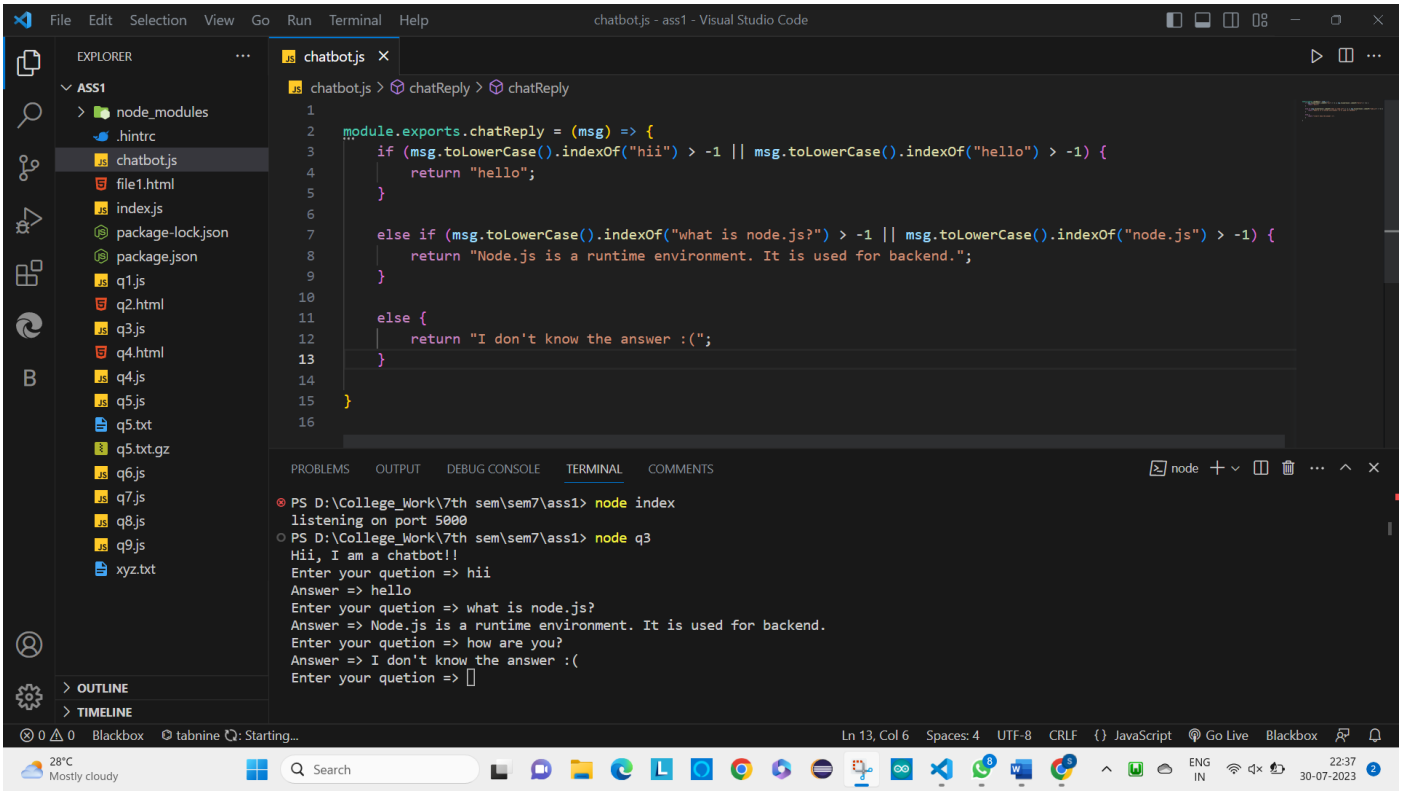
Q3.js

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

const interface = readline.createInterface(process.stdin, process.stdout);
console.log("Hii, I am a chatbot!!");
interface.setPrompt("Enter your quetion => ");
interface.prompt();

interface.on('line', (msg) => {
    console.log("Answer => " + chatreply.chatReply(msg));
    interface.prompt();
})
```

Chatbot.js

```
module.exports.chatReply = (msg) => {
    if (msg.toLowerCase().indexOf("hii") > -1 || msg.toLowerCase().indexOf("hello") > -1)
{
        return "hello";
    }

    else if (msg.toLowerCase().indexOf("what is node.js?") > -1 ||
msg.toLowerCase().indexOf("node.js") > -1) {
        return "Node.js is a runtime environment. It is used for backend.";
    }

    else {
        return "I don't know the answer :(";
    }

}
```

Output:



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

Code:

Q4.html

```html
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <title></title>
    </head>

    <body style="text-align: center;">
        <h1>WebSocket Chat Bot</h1>
        <div id="chat">
            <div id="messages" style=" font-size: 25px;"></div>
            <div style="position: fixed;bottom: 0; width: 100%; padding: 20px;">
                <hr>
                <input type="text" id="inputMessage"
                    placeholder="Type your message here..."
                    style="font-size: xx-large;" />
                <button onclick="sendMessage()"
                    style="font-size: xx-large;"><b>></b></button>
            </div>
        </div>
        <script>
        const ws = new WebSocket('ws://localhost:4589');

        ws.onmessage = (event) => {
            displayMessage('Server: '+event.data);
        };

        function sendMessage() {
            const inputMessage = document.getElementById('inputMessage');
            const message = inputMessage.value;
            inputMessage.value = '';

            displayMessage('You: ' + message);
            ws.send(message);
        }

        function displayMessage(message) {
            const messagesDiv = document.getElementById('messages');
            const messageDiv = document.createElement('div');
            messageDiv.textContent = message;
            messagesDiv.appendChild(messageDiv);
        }
    </script>
    </body>

</html>
```

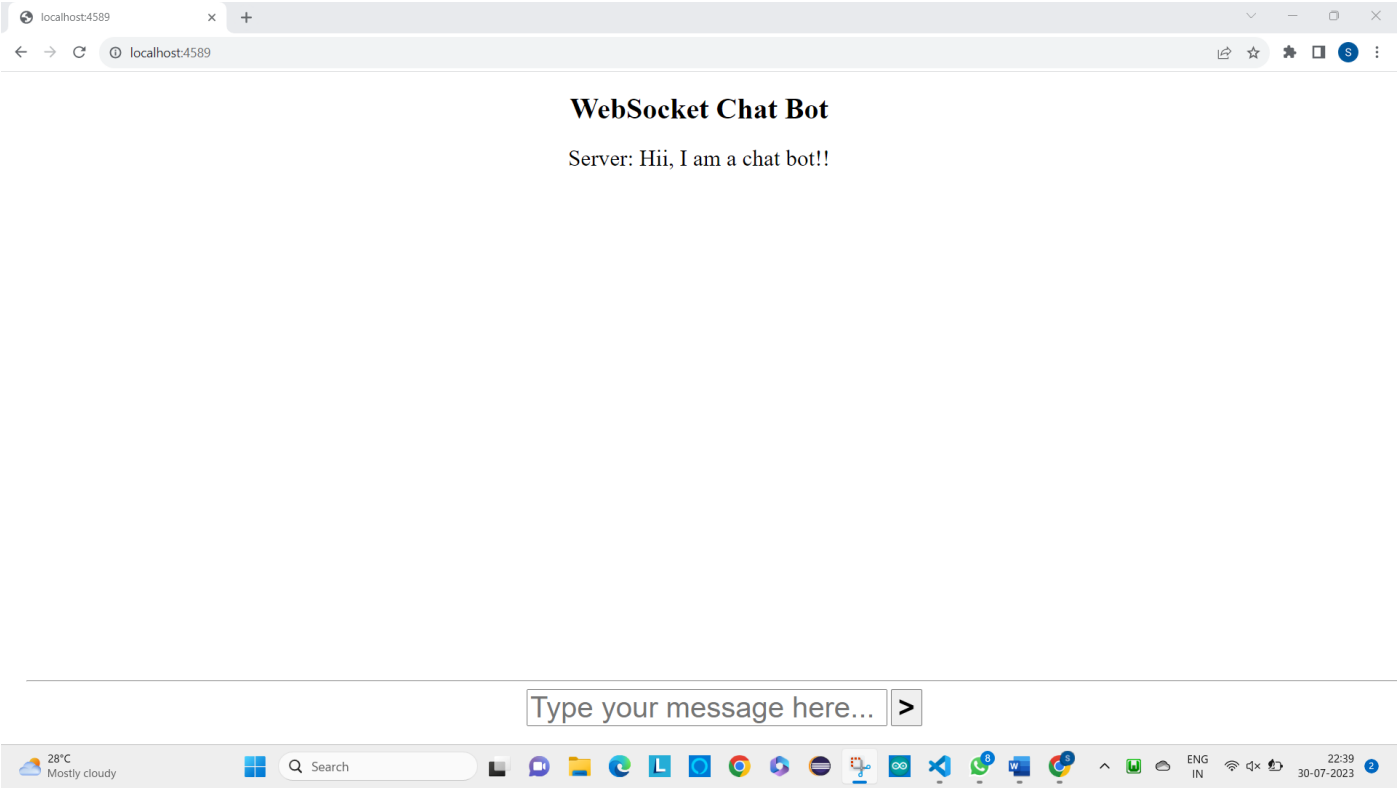
Q4.js

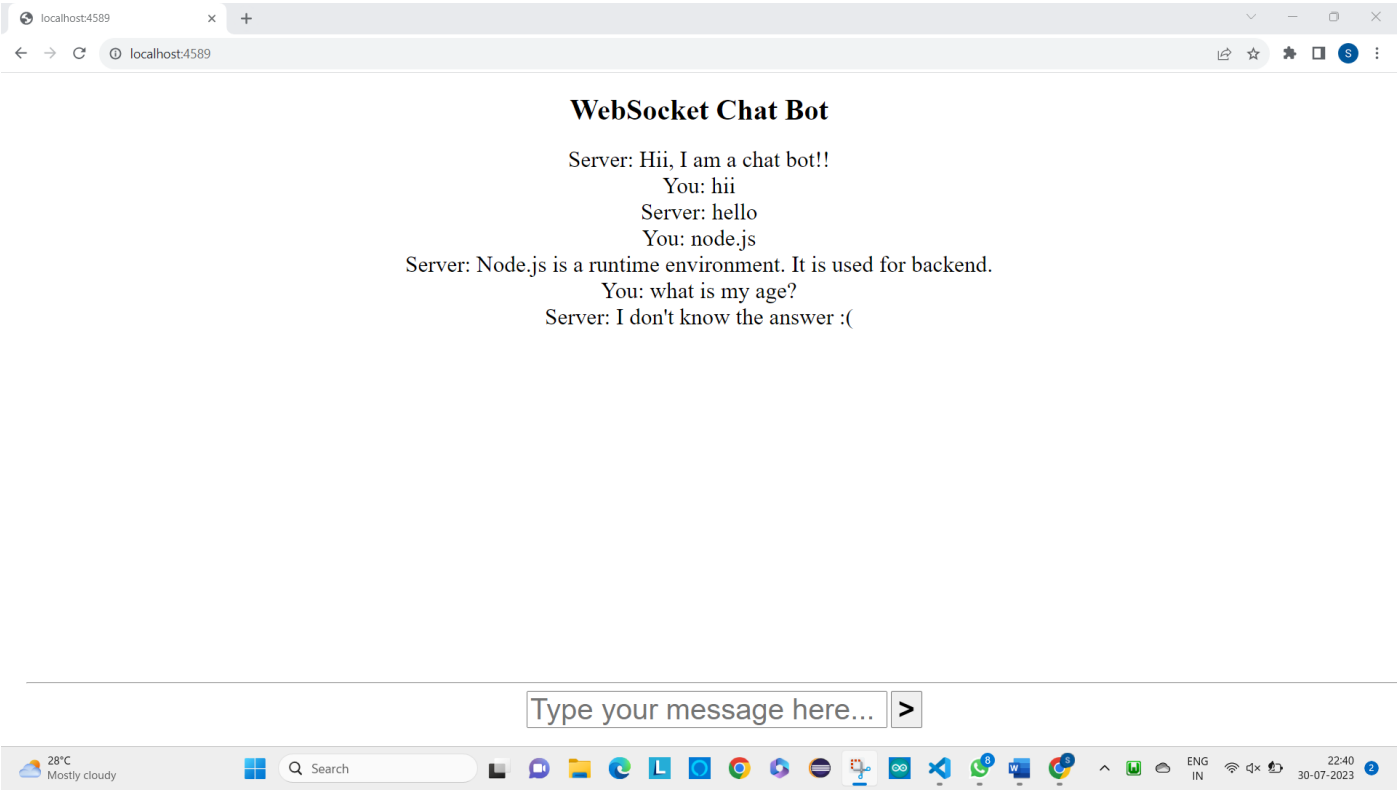
```
const http = require('http');
const st = require('node-static');
const chatBot = require('./chatbot'); // Import chatbot.js module
const WebSocket = require('ws');

const file = new st.Server('./q4.html');

const server = http.createServer((req, res) => {
    req.on('end', () => {
        file.serve(req, res);
    }).resume();
});

server.listen(4589, () => {
    console.log("Server listening on 4589");
});

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

wss.on('connection', (ws) => {
    ws.send("Hii, I am a chat bot!!");

    ws.on('message', (data) => {
        const message = data.toString();
        const reply = chatBot.chatReply(message);
        ws.send(reply);
    });
});
```

Chatbot.js

```
module.exports.chatReply = (msg) => {
    if (msg.toLowerCase().indexOf("hii") > -1 || msg.toLowerCase().indexOf("hello") > -1)
{
        return "hello";
    }

    else if (msg.toLowerCase().indexOf("what is node.js?") > -1 ||
msg.toLowerCase().indexOf("node.js") > -1) {
        return "Node.js is a runtime environment. It is used for backend.";
    }

    else {
        return "I don't know the answer :(";
    }

}
```

Output:


localhost:4589
localhost:4589
WebSocket Chat Bot
Server: Hii, I am a chat bot!!
Type your message here...
>
28°C
Mostly cloudy
Search
ENG
IN
22:39
30-07-2023

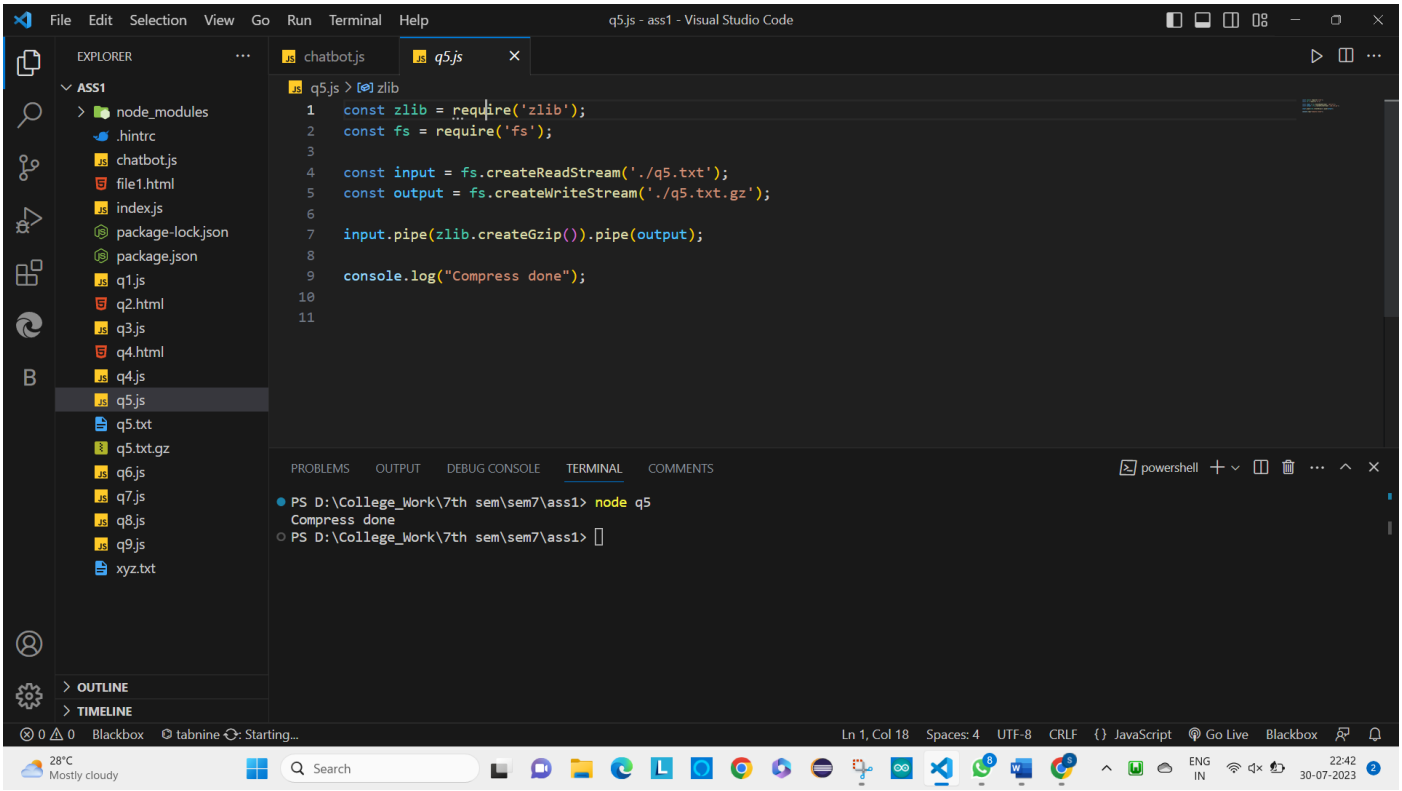

localhost:4589
localhost:4589
WebSocket Chat Bot
Server: Hii, I am a chat bot!!
You: hii
Server: hello
You: node.js
Server: Node.js is a runtime environment. It is used for backend.
You: what is my age?
Server: I don't know the answer :(
Type your message here...
>
28°C
Mostly cloudy
Search
ENG
IN
22:40
30-07-2023

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

Code:

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

const input = fs.createReadStream('./q5.txt');
const output = fs.createWriteStream('./q5.txt.gz');

input.pipe(zlib.createGzip()).pipe(output);

console.log("Compress done");
```

Output:

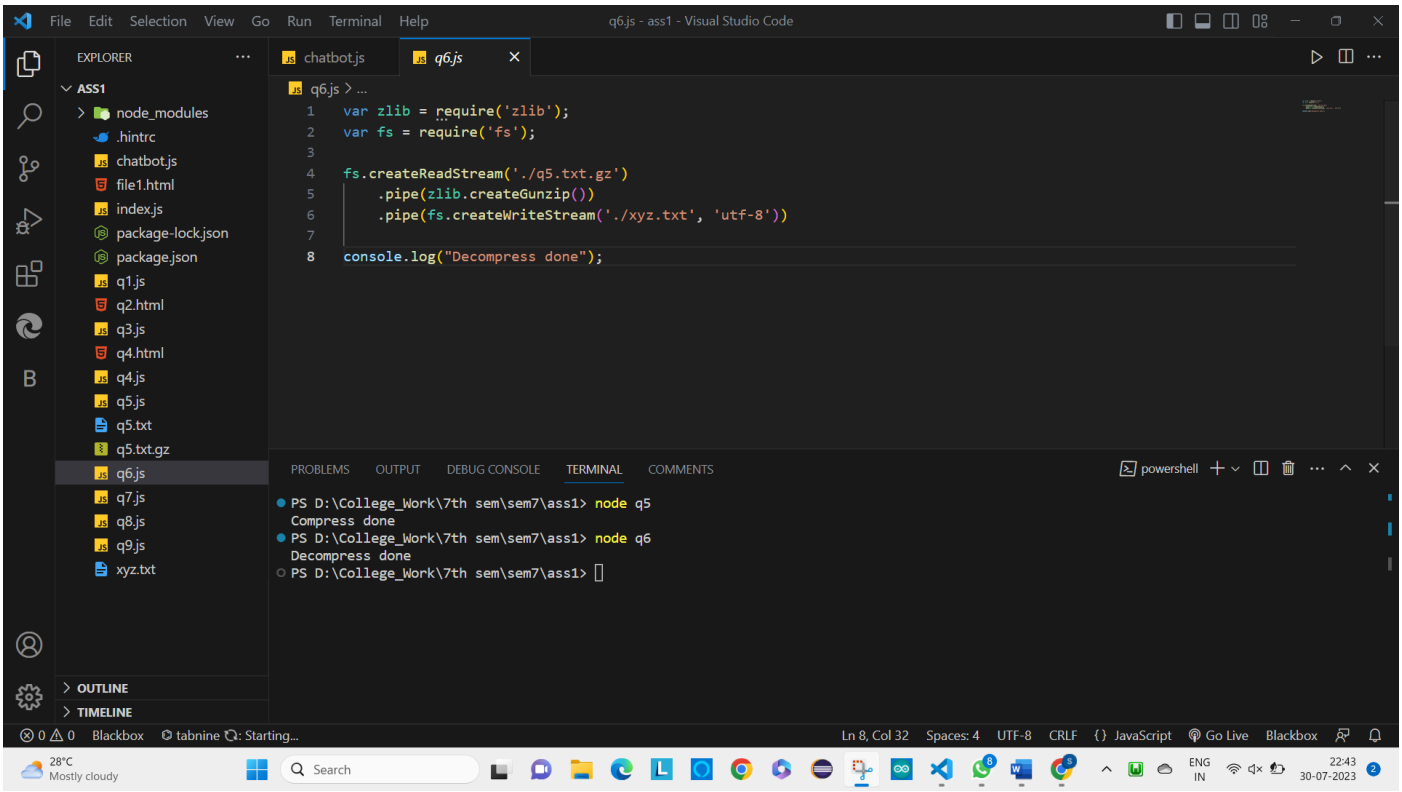


6. Write a program to extract a zip file.

Code:

```
var zlib = require('zlib');
var fs = require('fs');

fs.createReadStream('./q5.txt.gz')
    .pipe(zlib.createGunzip())
    .pipe(fs.createWriteStream('./xyz.txt', 'utf-8'))

console.log("Decompress done");
```

Output:



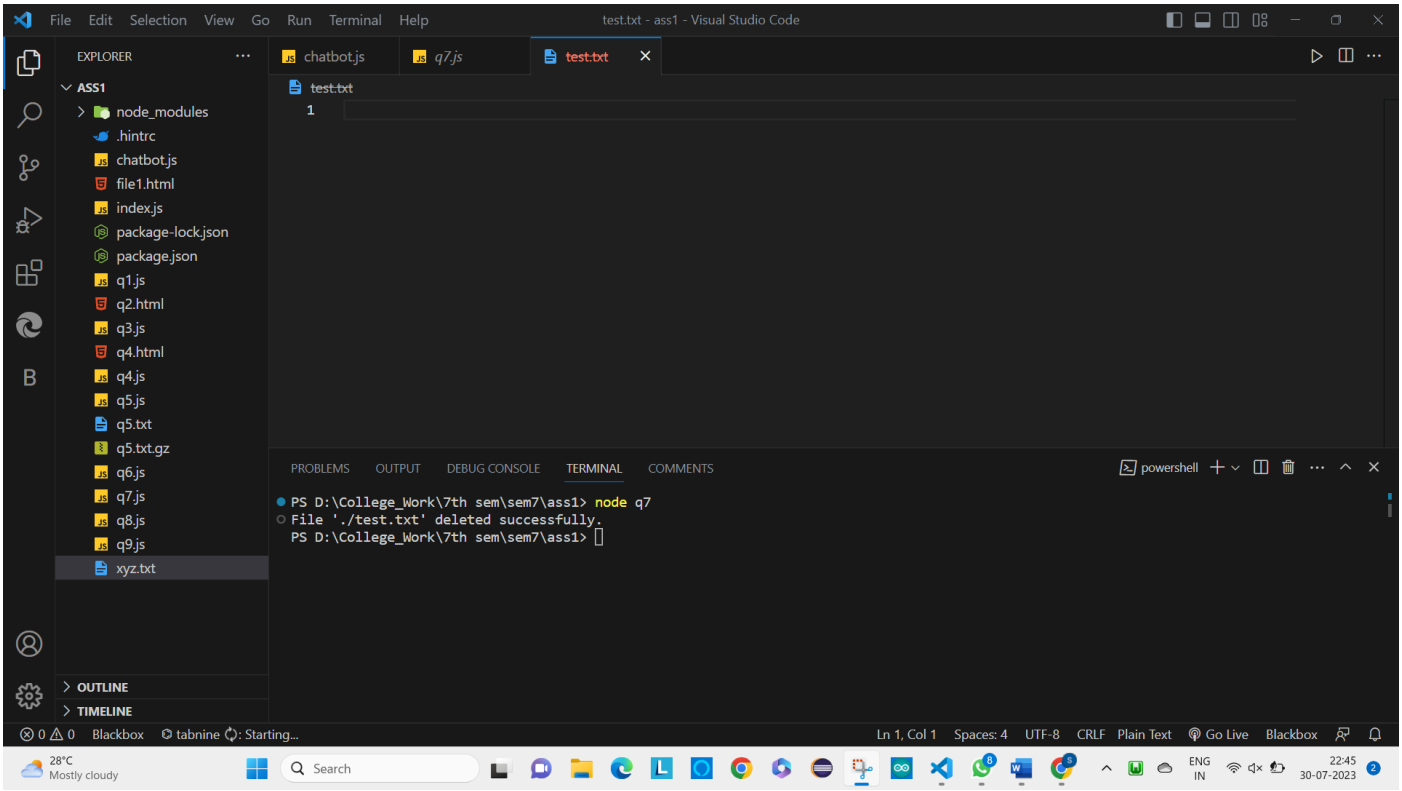
7. Write a program to promisify fs.unlink function and call it.

Code:

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

function promisifiedUnlink(filePath) {
    return new Promise((resolve, reject) => {
        fs.unlink(filePath, (err) => {
            if (err) {
                reject(err);
            } else {
                resolve();
            }
        });
    });
}

async function deleteFile(filePath) {
    try {
        await promisifiedUnlink(filePath);
        console.log(`File '${filePath}' deleted successfully.`);
    } catch (error) {
        console.error(`Error deleting the file '${filePath}': ${error.message}`);
    }
}
```

```
// Call the deleteFile function with the file path
const filePathToDelete = './test.txt';
deleteFile(filePathToDelete);
```

Output:



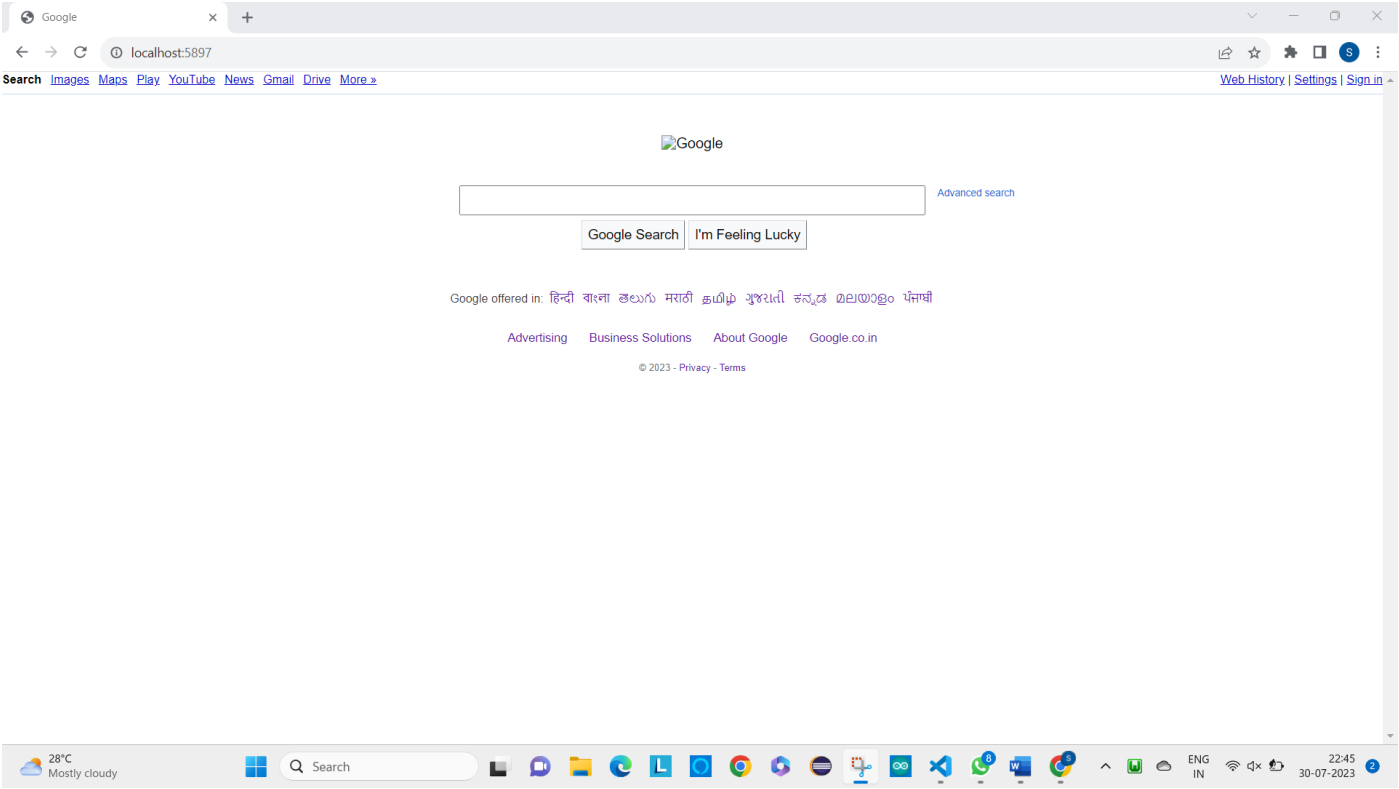
8. Fetch data of google page using note-fetch using async-await model.

Code:

```
const http = require('http');
const server = http.createServer((req, res) => {
    async function fetchGooglePage() {
        try {
            const fetch = await import('node-fetch');
            const response = await fetch.default('https://www.google.com');

            if (!response.ok) {
                throw new Error('Network response was not ok');
            }

            const data = await response.text();
            // console.log(data);
            res.end(data);
        } catch (error) {
            console.error('Error fetching data:', error.message);
        }
    }
```

```
    fetchGooglePage();
})

server.listen(5897, () => {
    console.log("Listing on 5897");
})
```

Output:



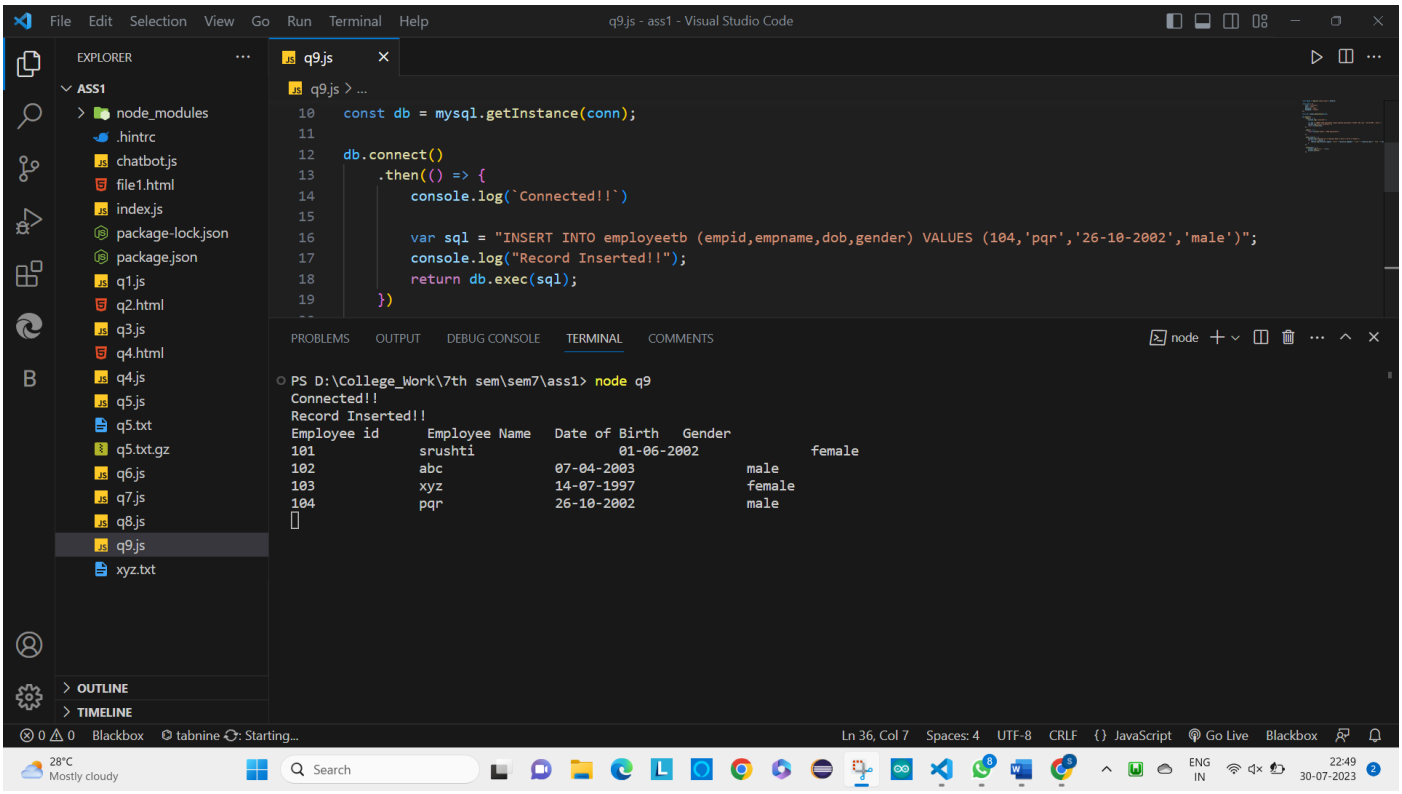
9. Write a program that connect Mysql database, Insert a record in employee table and display all records in employee table using promise based approach.

Code:

```
const mysql = require('nodejs-mysql').default;

const conn = ({
    host: "localhost",
    user: "root",
    password: "root",
    database: "7thsem"
});

const db = mysql.getInstance(conn);
```

```
db.connect()
    .then(() => {
        console.log(`Connected!!`)

        var sql = "INSERT INTO employeetb (empid,empname,dob,gender) VALUES
(302,'abc','25-06-2022','male')";
        console.log("Record Inserted!!");
        return db.exec(sql);
    })

    .then(() => {
        return db.exec("SELECT * FROM employeetb");

    })

    .then((result) => {
        console.log('Employee id \t Employee Name \t Date of Birth \t Gender');
        for (var i in result) {
            console.log(result[i].empid + "\t\t" + result[i].empname + " \t\t " +
result[i].dob + " \t\t " + result[i].gender);
        }
    })

    .catch((err) => {
        console.log("Error: " + err);
        process.exit(0);
    })
```

Output:

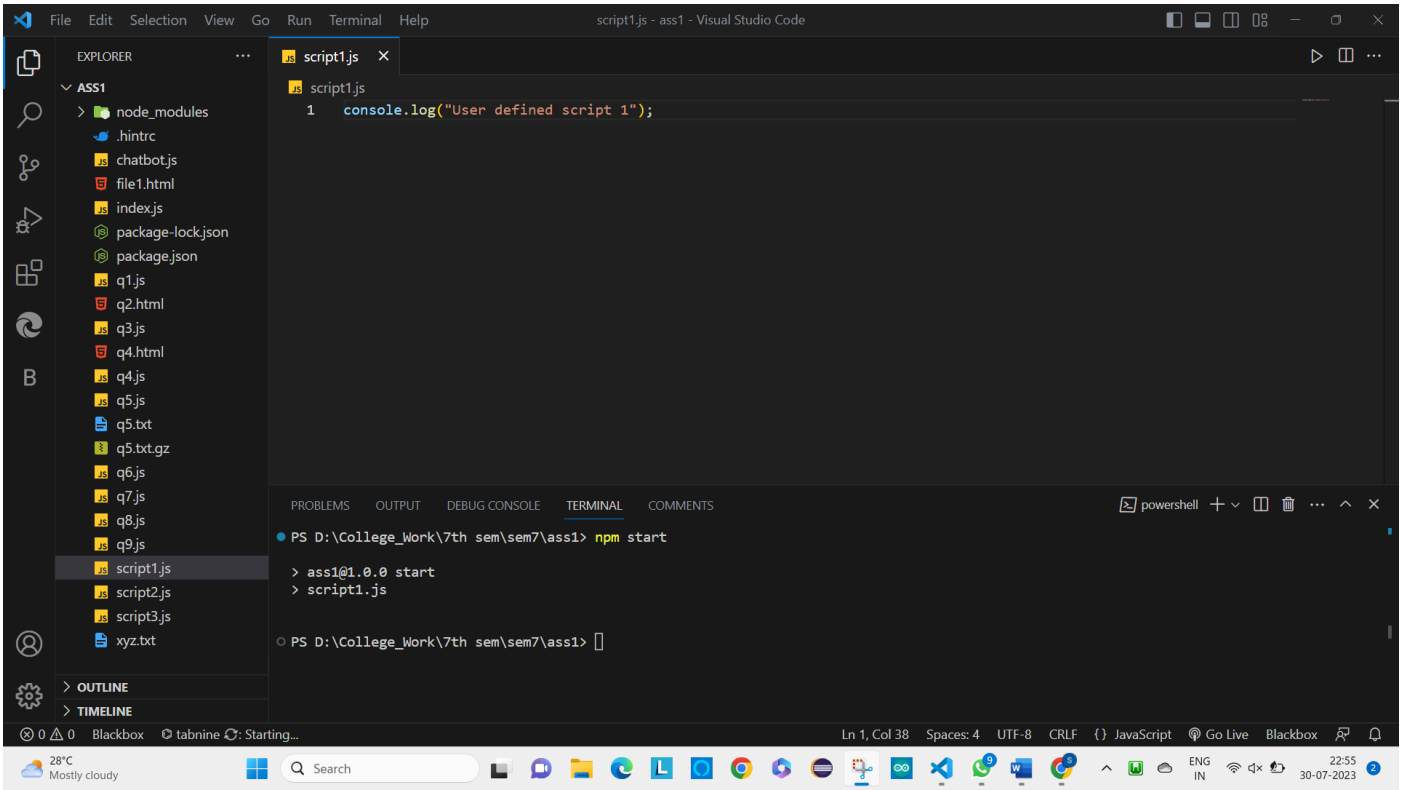


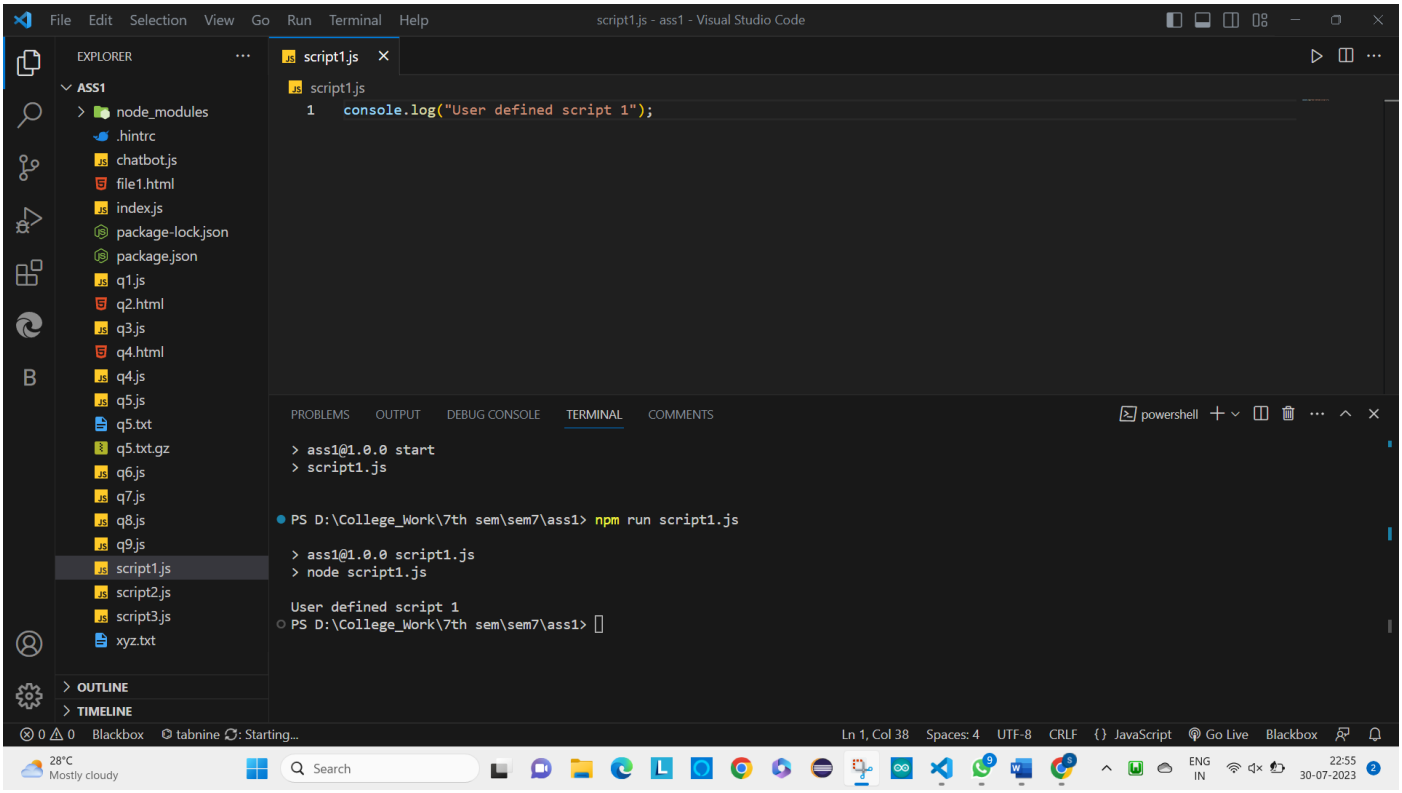
10. Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs application.

Code:

```
{
  "name": "ass1",
  "version": "1.0.0",
  "description": "",
  "main": "chatbot.js",
  "scripts": {
    "start": "script1.js",
    "test": "echo \"Error: no test specified\" && exit 1",
    "script1.js": "node script1.js",
    "script2.js": "node script2.js",
    "script3.js": "node script3.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "mysql": "^2.18.1",
    "node-static": "^0.7.11",
    "nodejs-mysql": "^0.1.3"
  }
}
```

Output:





11. Develop an application to show live cricket score.

Working of Application: