Que(1) Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.

- Handle POST request.

Ans:-

Server.js

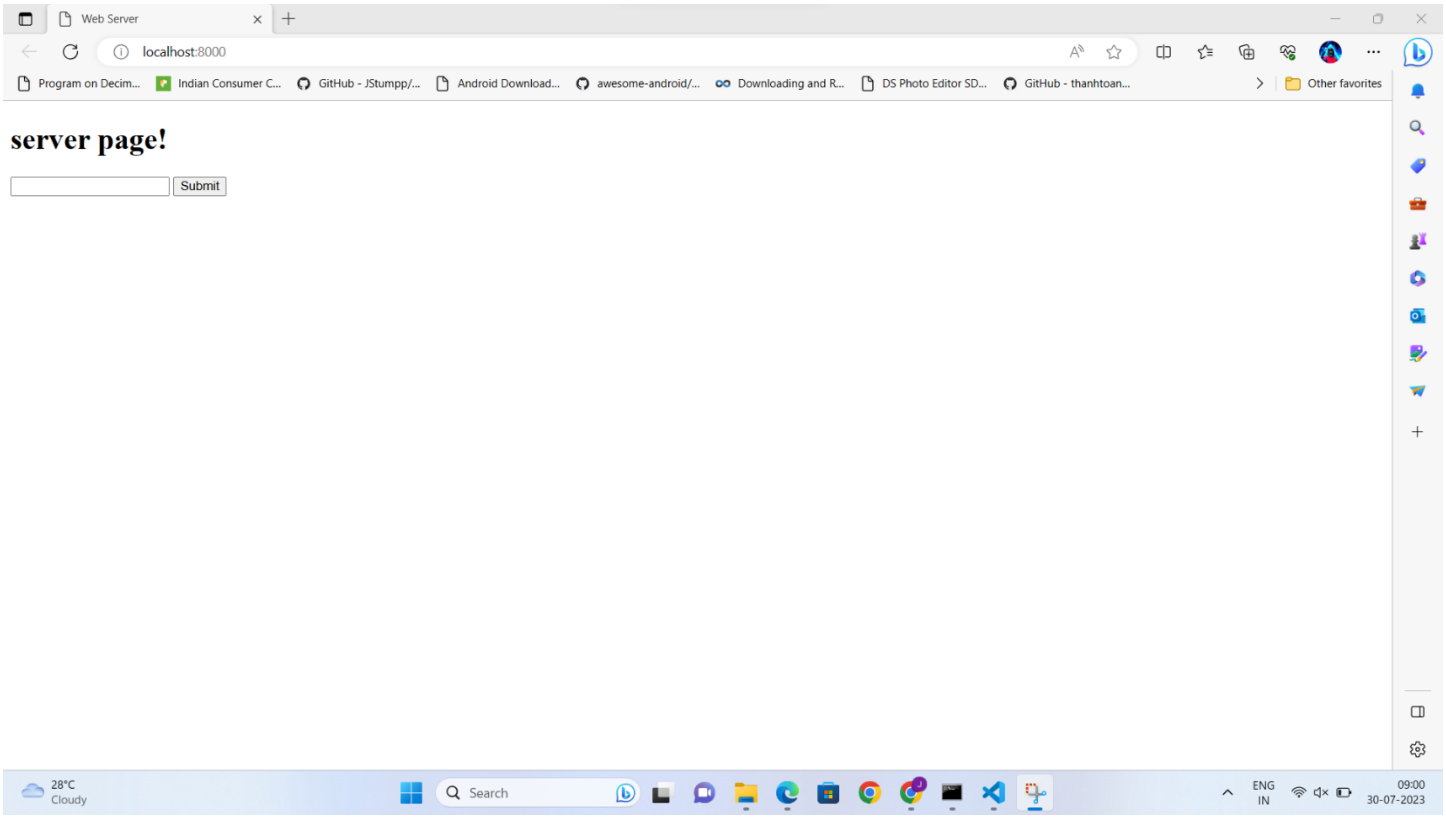
```
const http = require('http');
const fs = require('fs');

const port = 8000;

// Helper function to serve static files
function serveStaticFile(res, filename, contentType) {
  fs.readFile(filename, (err, data) => {
    if (err) {
      res.writeHead(500, { 'Content-Type': 'text/plain' });
      res.end('Internal Server Error');
    } else {
      res.writeHead(200, { 'Content-Type': contentType });
      res.end(data);
    }
  });
}

// Create the server
const server = http.createServer((req, res) => {
  if (req.method === 'GET') {
    if (req.url === '/') {
      // Serve the index.html file
      serveStaticFile(res, './index.html', 'text/html');
    }
  } else if (req.method === 'POST') {
    if (req.url === '/') {
      let body = '';
      req.on('data', (chunk) => {
        body += chunk;
      });

      req.on('end', () => {

        // Do something with the request body
        res.writeHead(200, { 'Content-Type': 'text/plain' });
```

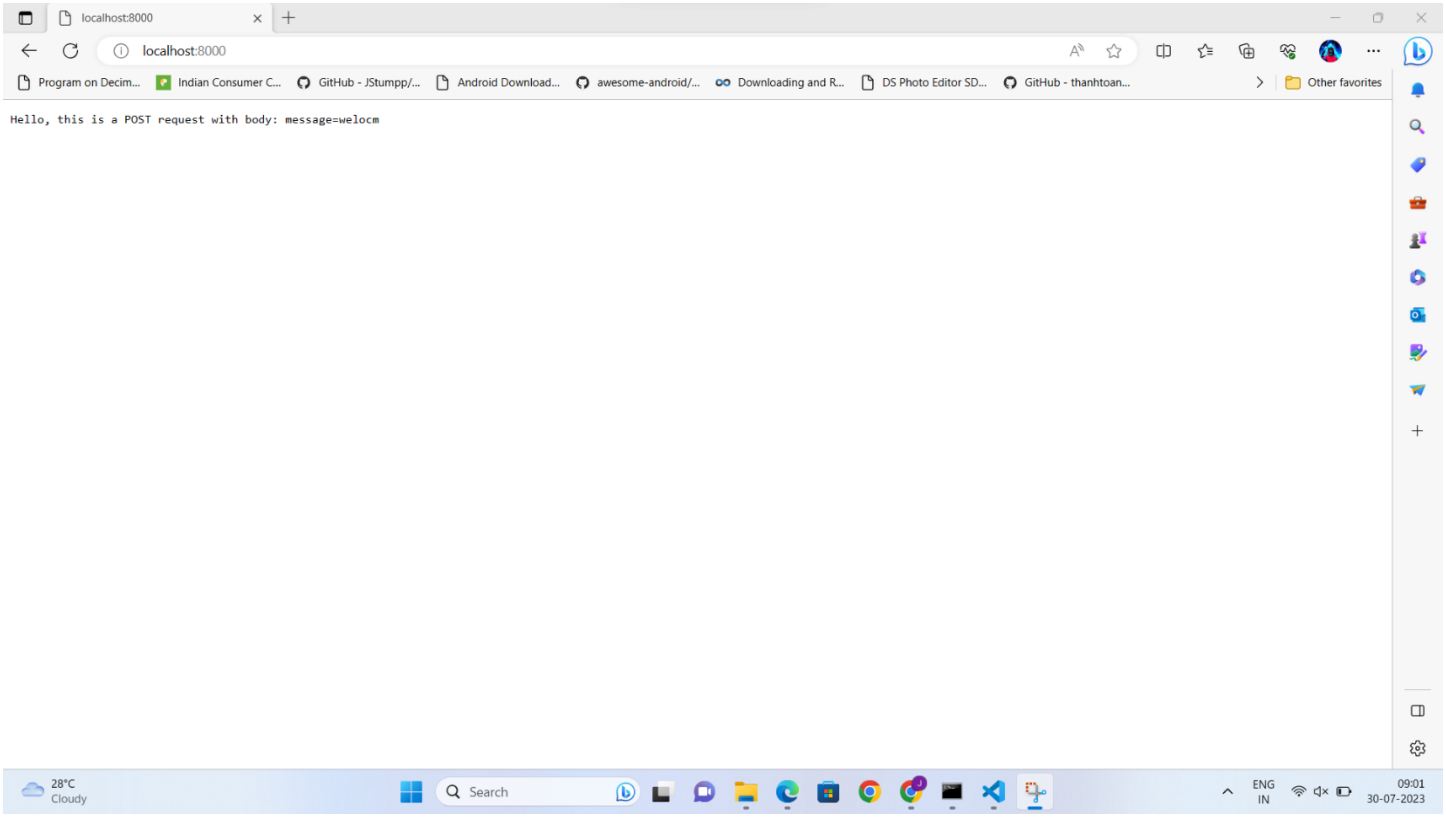
```
        res.end('Hello, this is a POST request with body: ' + body);
      });
    } else {
      // Handle 404 Not Found
      res.writeHead(404, { 'Content-Type': 'text/plain' });
      res.end('404 Not Found');
    }
  } else {
    // Handle other HTTP methods
    res.writeHead(101, { 'Content-Type': 'text/plain' });
    res.end('101 Not Implemented');
  }
});

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

Index.js

```
<!DOCTYPE html>
<html>
<head>
  <title>Web Server</title>
</head>
<body>
  <h1> server page!</h1>
  <!-- <p>This is a static HTML file served by the web server.</p> -->
  <form action="/" method="post">
    <input type="text" name="message" />
    <button type="submit">Submit</button>
  </form>
</body>
</html>
```

Output:-


Web Server
localhost:8000
Program on Decim...
Indian Consumer C...
GitHub - JStumpp/...
Android Download...
awesome-android/...
Downloading and R...
DS Photo Editor SD...
GitHub - thanhtoan...
Other favorites
server page!
Submit
28°C
Cloudy
Search
ENG
IN
09:00
30-07-2023


localhost:8000
localhost:8000
Program on Decim...
Indian Consumer C...
GitHub - JStumpp/...
Android Download...
awesome-android/...
Downloading and R...
DS Photo Editor SD...
GitHub - thanhtoan...
Other favorites
Hello, this is a POST request with body: message=welocm
28°C
Cloudy
Search
ENG
IN
09:01
30-07-2023

Que(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.)

Ans:-

Que2.js

```
const express = require("express")
const app = express()
const fs = require("fs")

app.get("/",(req,res)=>{
    fs.readFile("./index2_1.html","utf8",(err,data)=>{
        res.send(data)
    })
})
app.get("/gethello",(req,res)=>{
    fs.readFile("./index2.html","utf8",(err,data)=>{
        res.send(data)
    })
    // res.send(`<h1>hello nodeJs!!!</h1>`)
})

app.listen(3000)
```

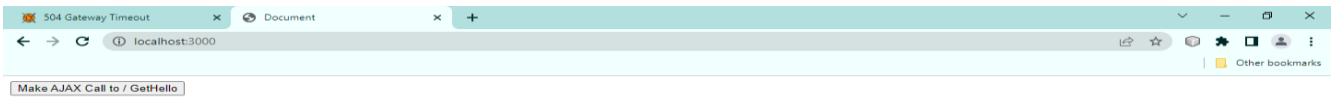
index2.html

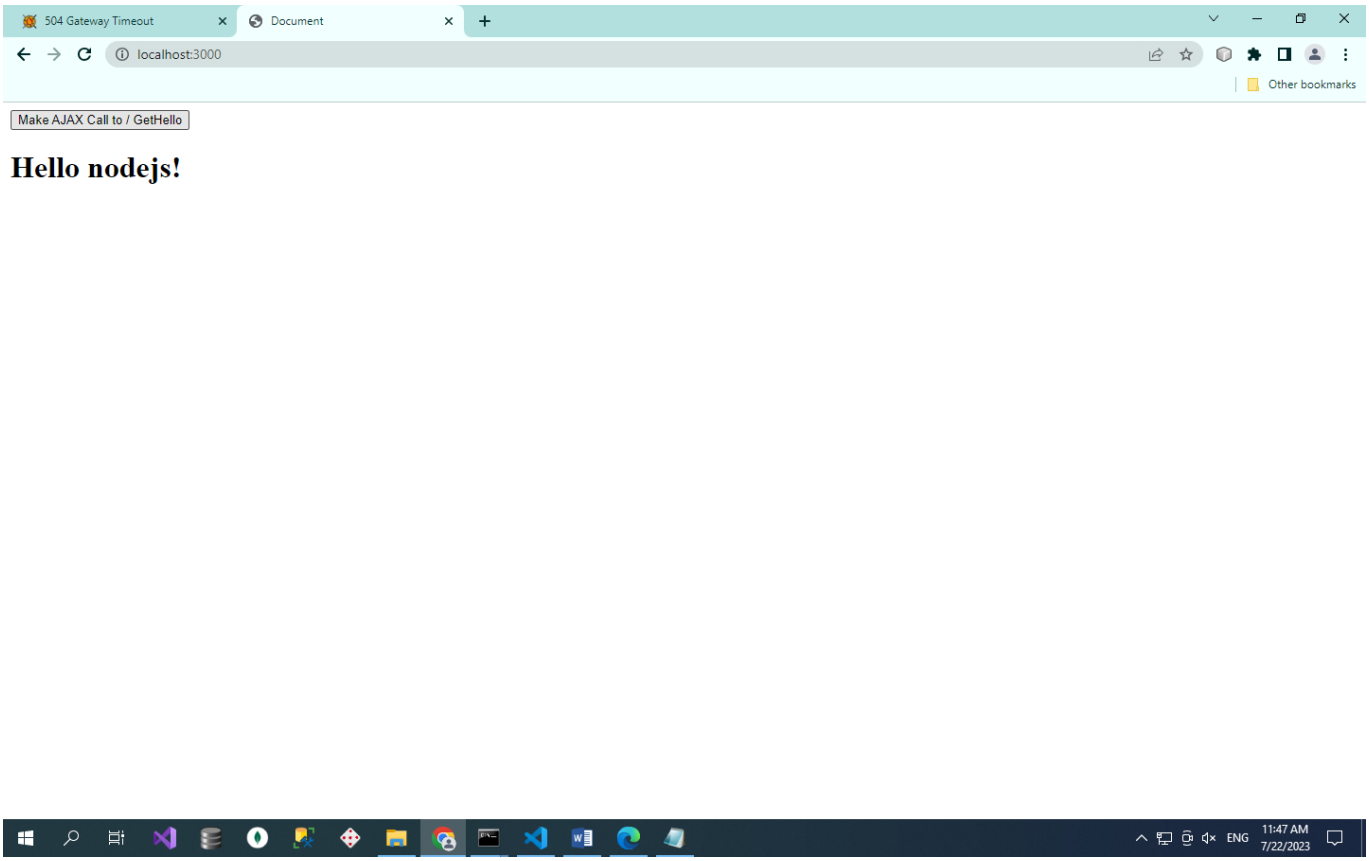
```
<!-- Get Complete Source Code from Pabbly.com -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
   <h1>Hello nodejs!</h1>
</body>
</html>
```

Index2_1.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>

    <button id="ajaxButton">Make AJAX Call to / GetHello</button>
    <div id="ajaxResponse"></div>

 </body>
 <script>

 document.getElementById("ajaxButton").addEventListener("click", () => {
       fetch("/gethello").then(response=>response.text()).then(data => {
          document.getElementById("ajaxResponse").innerHTML = data;
    }).catch(error => {
     console.error(error)
    });
 });
 </script>
</html>
```

Output:-





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

Ans:- chatBot.js

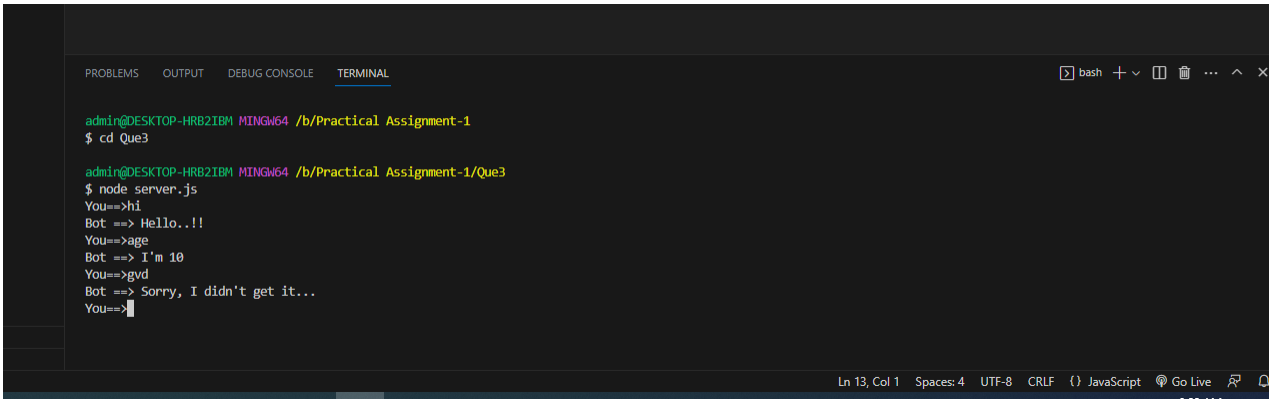
```
module.exports.reply = function (msg) {
    this.Bot_Age = 10;
    this.Bot_Name = "abc";

    msg = msg.toLowerCase();

    if(msg.indexOf("hi")>-1)
    {
        return "Hello..!!";
    }

    else if(msg.indexOf("age") > -1 &&
            msg.indexOf("your"))
        {
            return "I'm " + this.Bot_Age;
        }
        return "Sorry, I didn't get it... ";
```

```
}
```

Server.js

```
var Chatbot = require('./chatBot');
var readline = require('readline');

var r1 = readline.createInterface(process.stdin, process.stdout);
r1.setPrompt("You==>");
r1.prompt();
r1.on('line', function(message) {
    console.log('Bot ==> '+ Chatbot.reply(message));
    r1.prompt();
}).on('close',function(){  //chaining events.
    process.exit(0);
});
```

Output:-



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

Ans:-

Websocket.js

```
const WebSocket = require('ws')
var http = require('http');
var url = require('url');
var Chatbot = require('../Que3/chatbot.js');

var st = require('node-static');

var fileServer = new st.Server('../public');

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

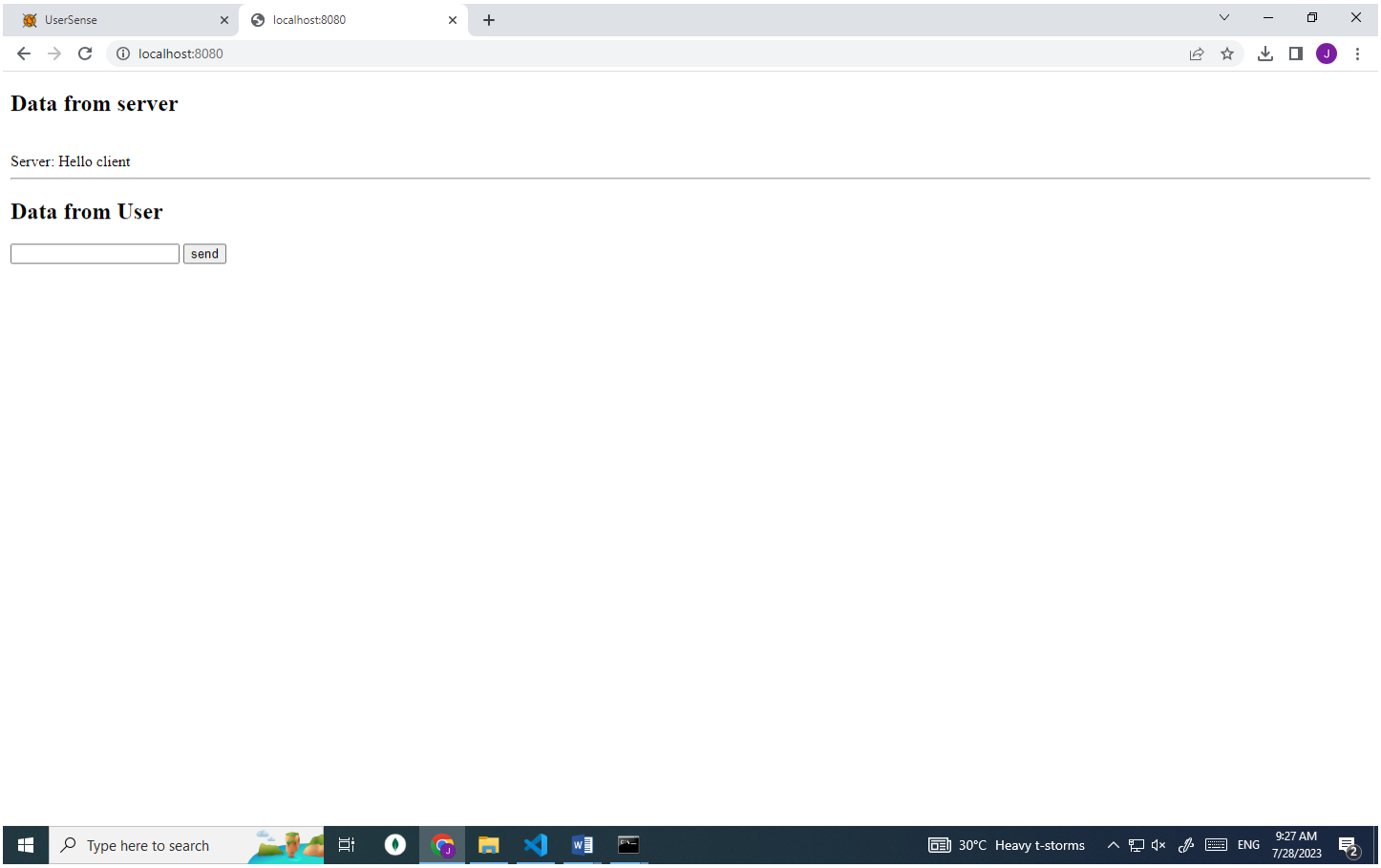
```
    request.on('end', function () {
    var get = url.parse(request.url, true).query;
    fileServer.serve(request, response);
    }).resume();

}).listen(8080, function() {
    console.log((new Date()) +
      ' Server is listening on port 8080');
});

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

wss.on('connection', function(ws) {
  ws.send('Hello client')

  ws.on('message', message => {
    console.log(`Received message => ${message}`)
    // console.log(Chatbot.ChatbotReply(message))
    ws.send(Chatbot.reply(message))
  })

})
```

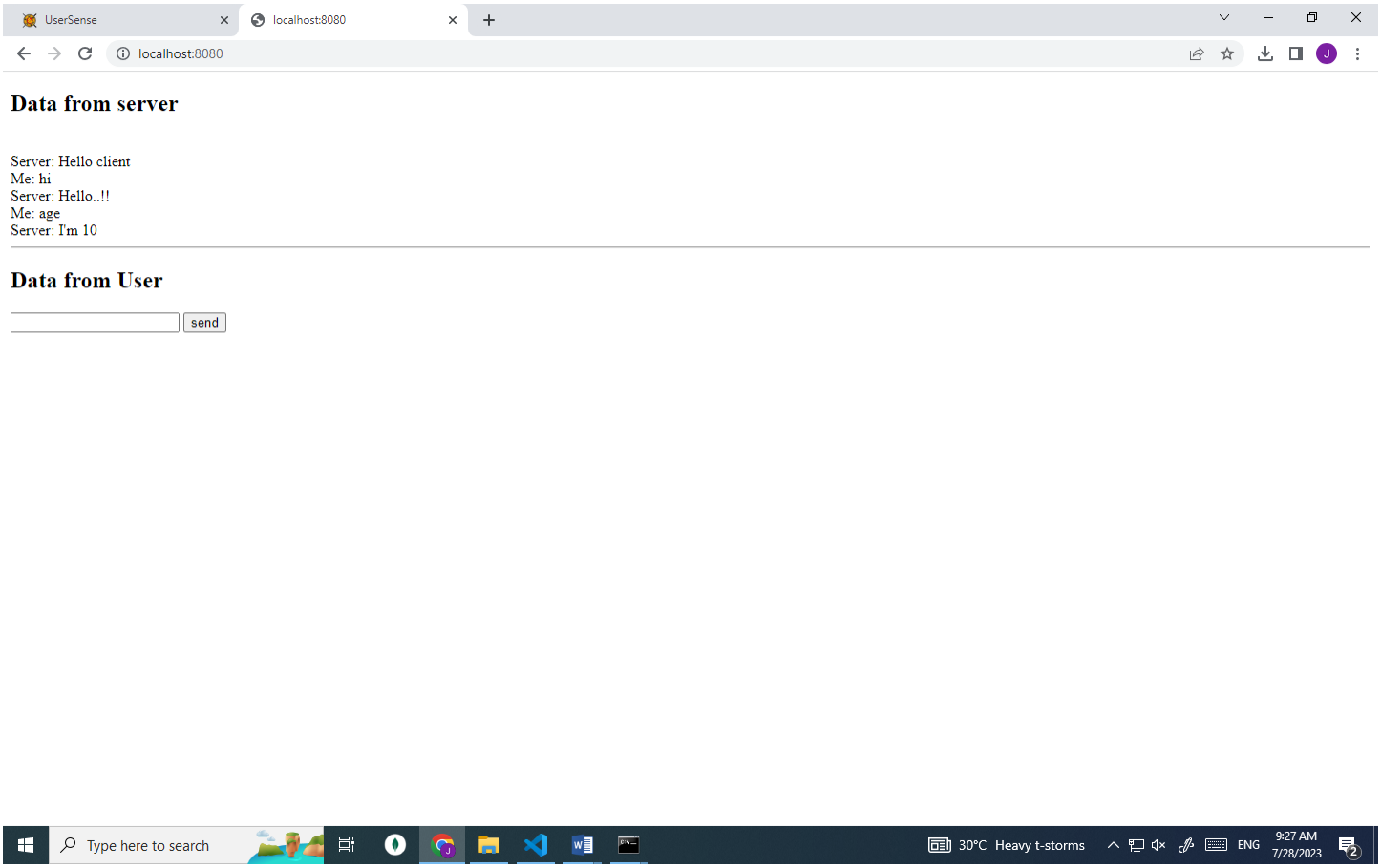
Public file / index html

```
<!DOCTYPE html >
<html>
    <body>
<script language="javascript">
var ws = new WebSocket('ws://localhost:8080');
ws.addEventListener("message", function(e) {
  var msg = e.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);
  document.getElementById('message').value="";
}
</script>
<h2>Data from server</h2>
        <div id="chatlog"></div>
<hr/>
```

```
<h2>Data from User</h2>
  <input type="text" id="message"  />
        <input type="button" id="b1" onclick="sendMessage()" value="send" />

    </body>
</html>
```

OutPut:-





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

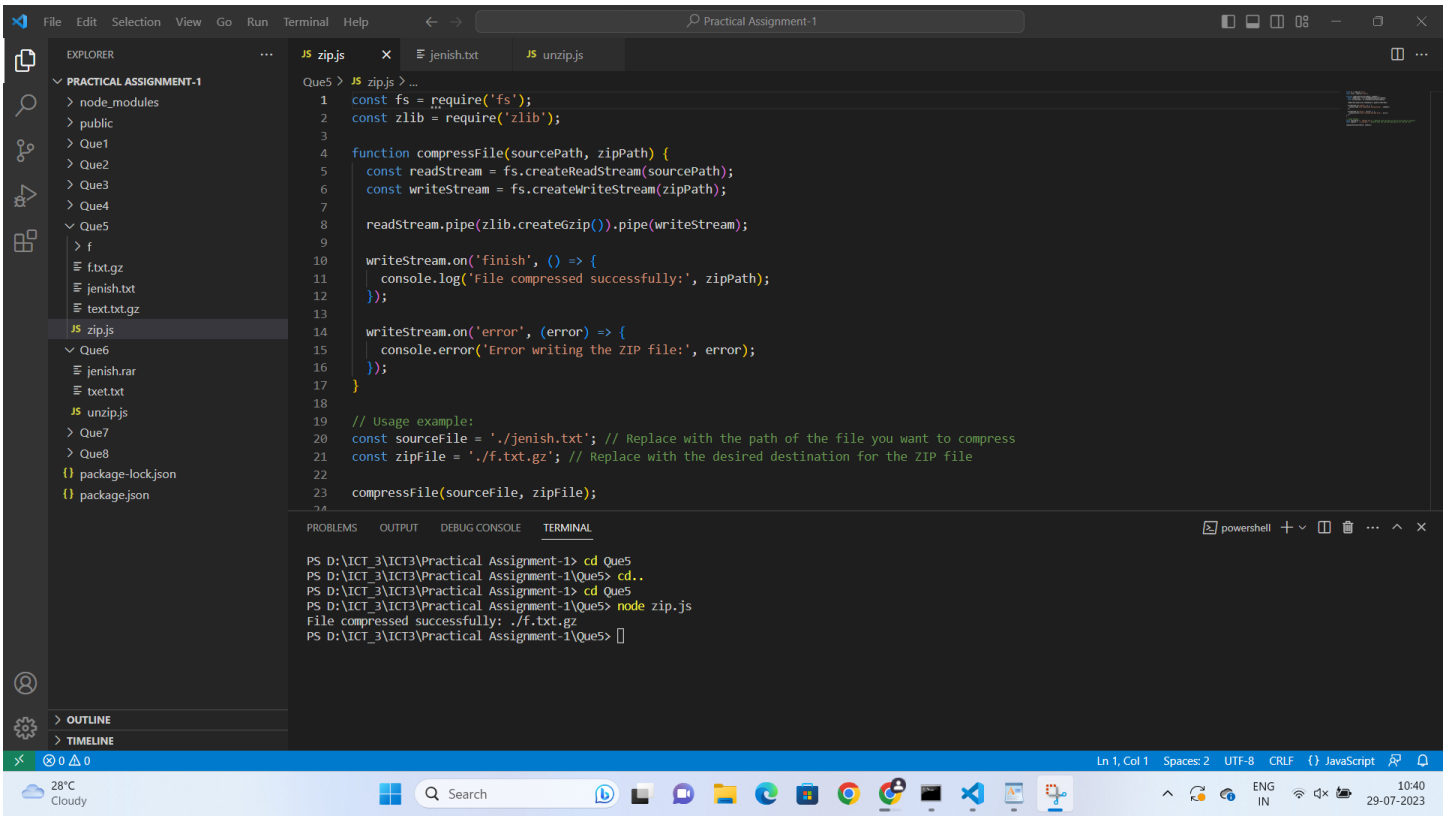
Ans:-

Zip.js

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

function compressFile(sourcePath, zipPath) {
  const readStream = fs.createReadStream(sourcePath);
  const writeStream = fs.createWriteStream(zipPath);

  readStream.pipe(zlib.createGzip()).pipe(writeStream);

  writeStream.on('finish', () => {
    console.log('File compressed successfully:', zipPath);
  });

  writeStream.on('error', (error) => {
    console.error('Error writing the ZIP file:', error);
  });
```

```
}

// Usage example:
const sourceFile = './jenish.txt'; // Replace with the path of the file you want to compress
const zipFile = './f.txt.gz'; // Replace with the desired destination for the ZIP file

compressFile(sourceFile, zipFile);
```

Output:-



Que(6) Write a program to promisify fs.unlink function and call it.

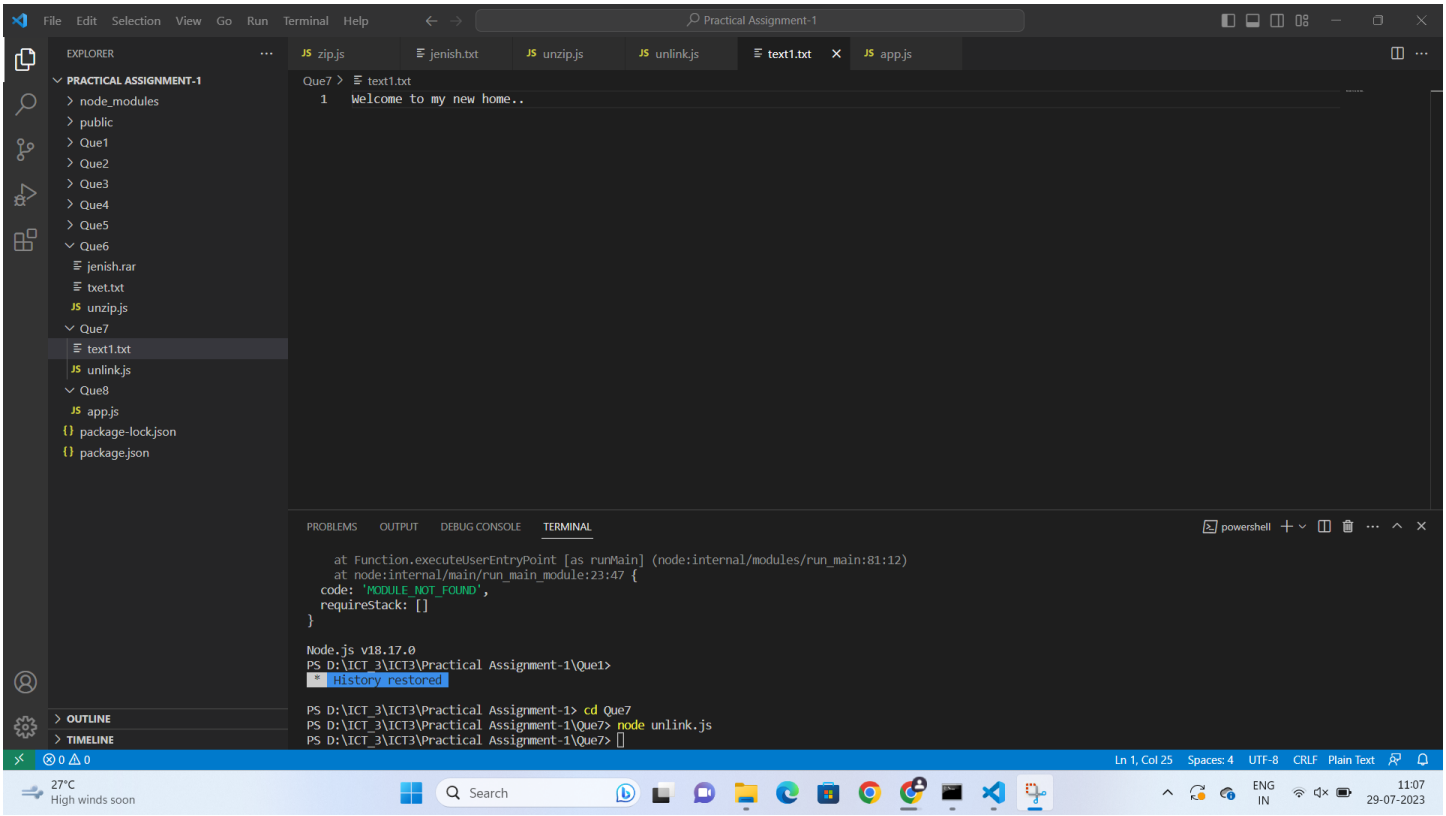
Ans:-

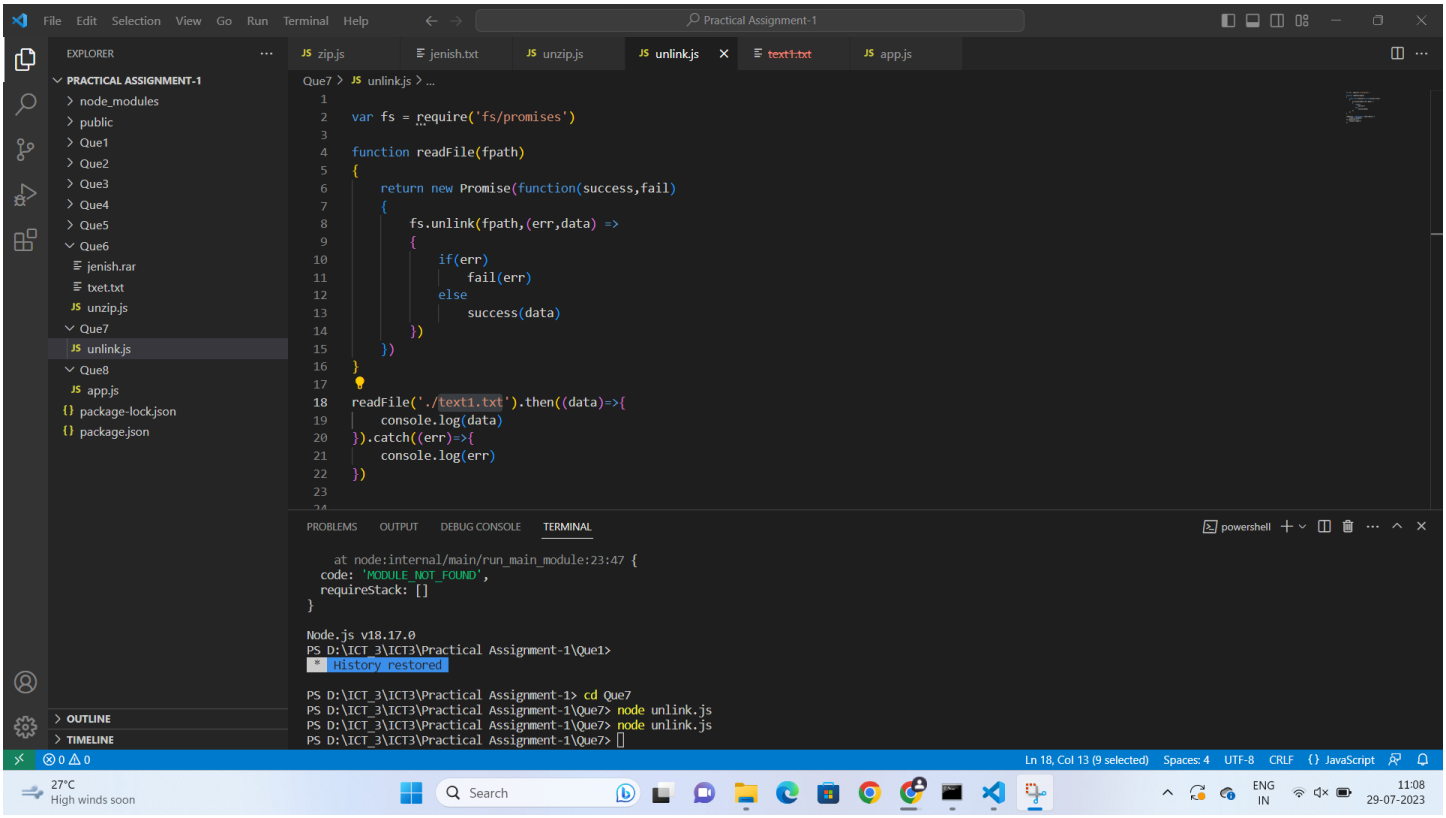
Unlink.js

```
var fs = require('fs/promises')

function readFile(fpath)
{
    return new Promise(function(success,fail)
    {
```

```
        fs.unlink(fpath,(err,data) =>
        {
            if(err)
                fail(err)
            else
                success(data)
        })
    })
}

readFile('./text1.txt').then((data)=>{
    console.log(data)
}).catch((err)=>{
    console.log(err)
})
```

Output:-





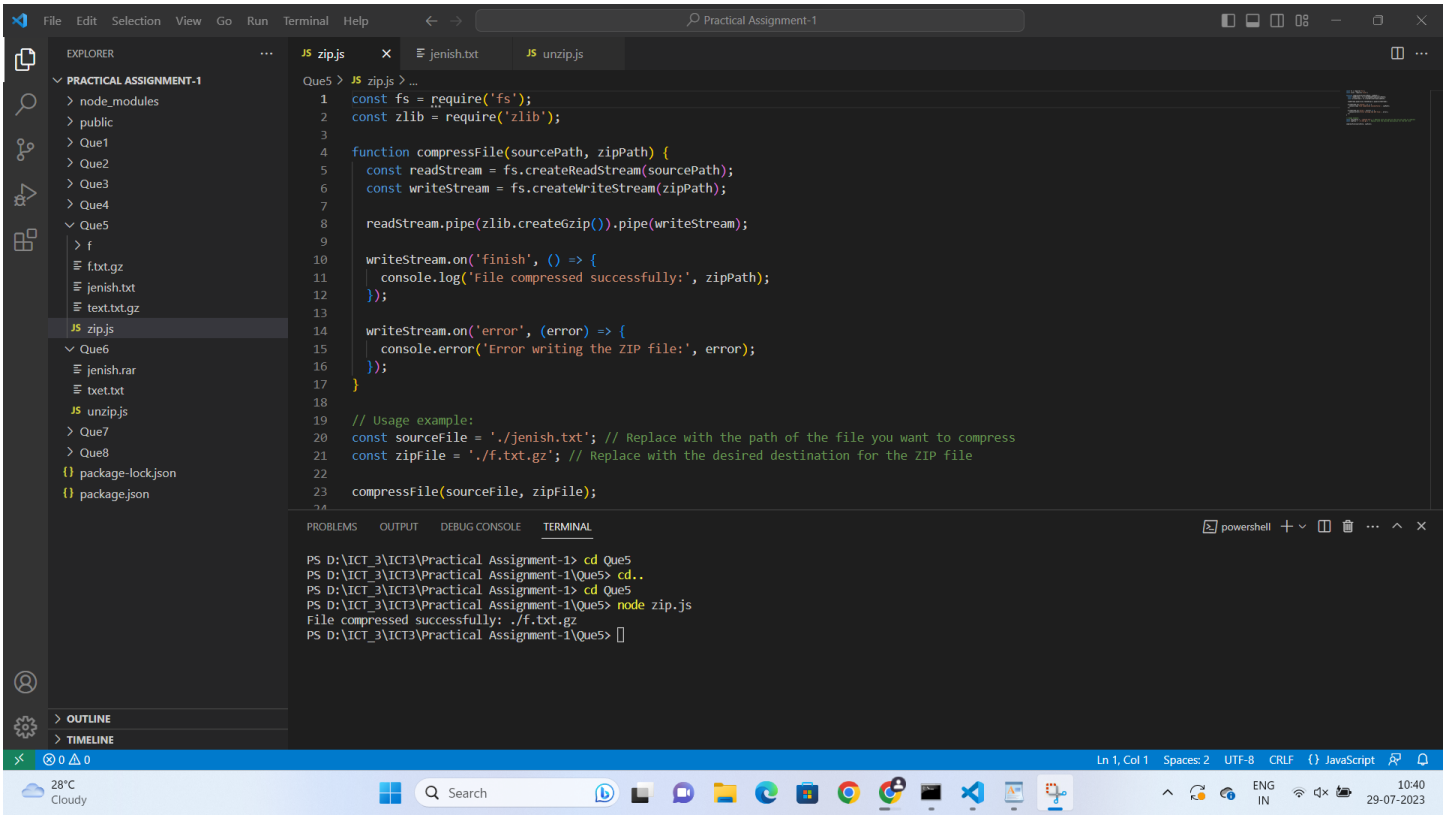
Que(7):- Write a program to extract a zip file.

Ans:-

Unzip.js

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

function decompressZlib(inputFilePath, outputFilePath) {
  const compressedData = fs.readFileSync(inputFilePath);
  zlib.unzip(compressedData, (error, decompressedData) => {
    if (error) {
      console.error('Error decompressing data:', error);
    } else {
      fs.writeFileSync(outputFilePath, decompressedData);
      console.log('Data successfully decompressed and saved to:', outputFilePath);
    }
  });
}

// Usage example:
const compressedFilePath = './f.txt.gz'; // Replace with the path of the zlib-compressed file
const decompressedFilePath = './f.txt.ungz'; // Replace with the desired output file path

decompressZlib(compressedFilePath, decompressedFilePath);
```

OutPut:-



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

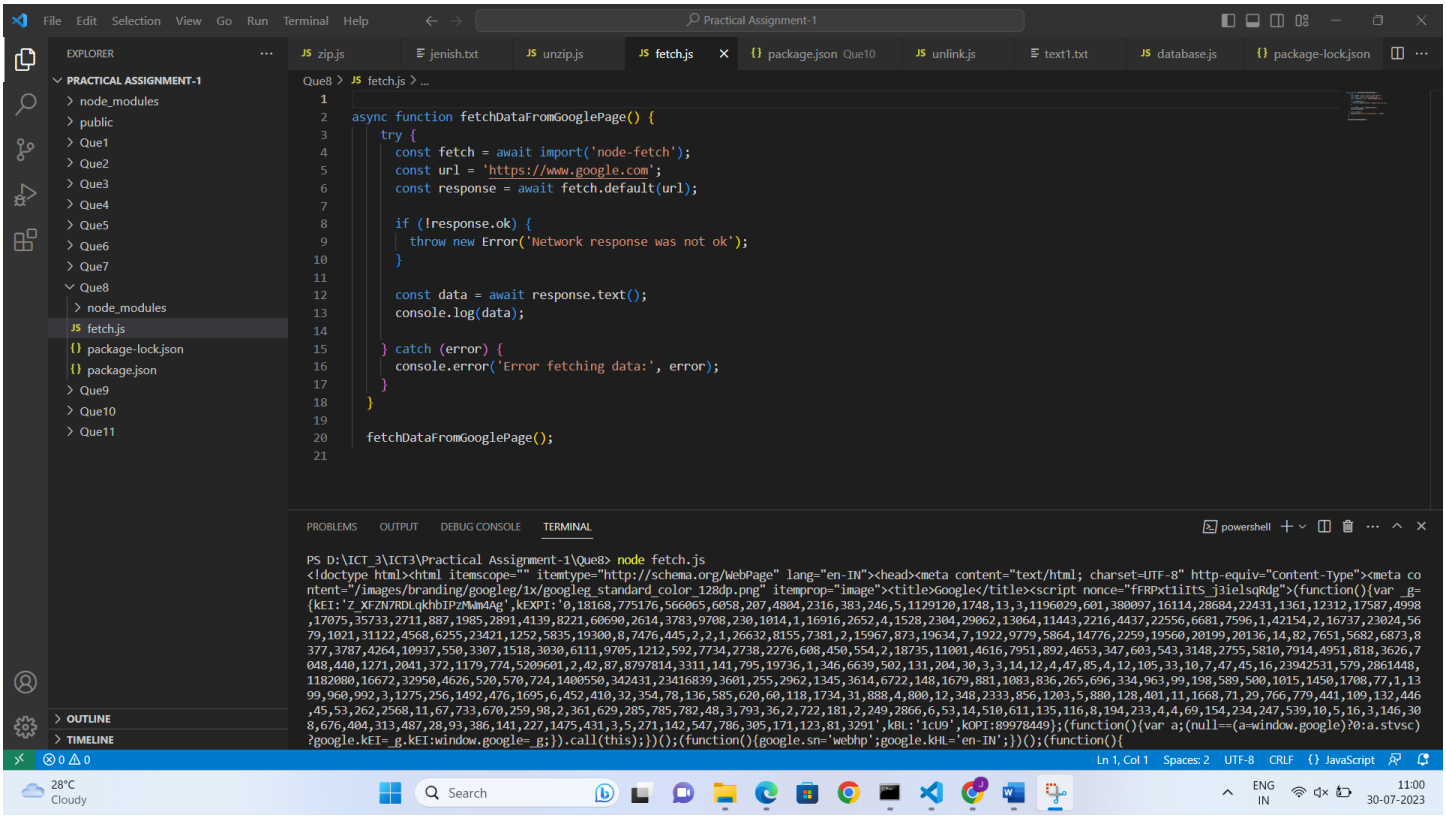
Ans:-

Fetch.js

```
async function fetchDataFromGooglePage() {
    try {
      const fetch = await import('node-fetch');
      const url = 'https://www.google.com';
      const response = await fetch.default(url);

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

      const data = await response.text();
      console.log(data);

    } catch (error) {
      console.error('Error fetching data:', error);
    }
```

```
}

fetchDataFromGooglePage();
```

OutPut:-



Que(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.

Ans:-

Database.js

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


const config = {
    host : "localhost",
    user : "root",
    password : "root",
    database : "employee_db"
}
```

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


db.connect()
  .then(function(){
    console.log("Connected!!");


    var sql = "INSERT INTO employee (username, password, firstname, lastname, email) VALUES
('abc', 'xx', 'abc1', 'a', 'a46884@gmail.com')";

    return db.exec(sql);

}).then(function(res){
    console.log(res);
    return db.exec("SELECT * FROM employee");
}).then(function(result){
    for( var i in result){
        console.log("Username: ", result[i].username + " " + "Password: " +
result[i].password);
        process.exit(0);
    }
}).catch(function(err){
    console.log("ERROR: ", err);
    process.exit(0);
});
```

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

Ans:-

Server.js

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

const PORT = 3000;

const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, world!');
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
```

```
  }
});

server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Script1.js

```
// User-defined script 1
console.log('Running user-defined script 1');
```

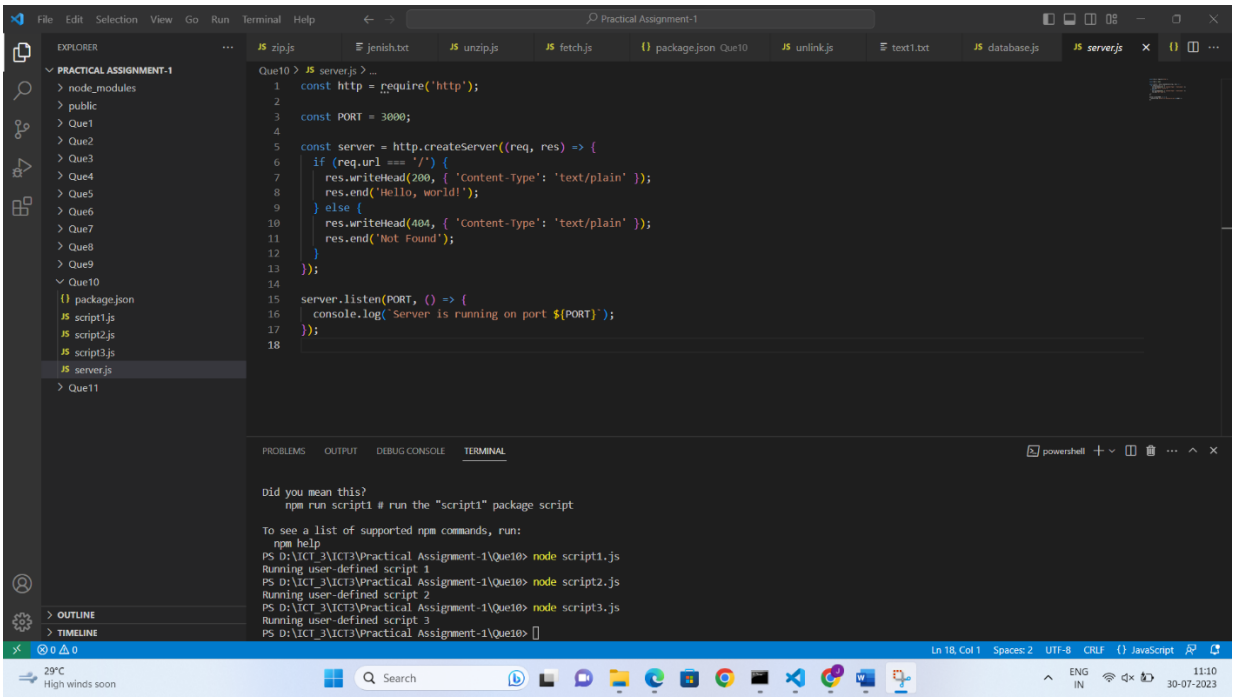
Script2.js

```
// User-defined script 2
console.log('Running user-defined script 2');
```

Script3.js

```
// User-defined script 3
console.log('Running user-defined script 3');
```

Output:-



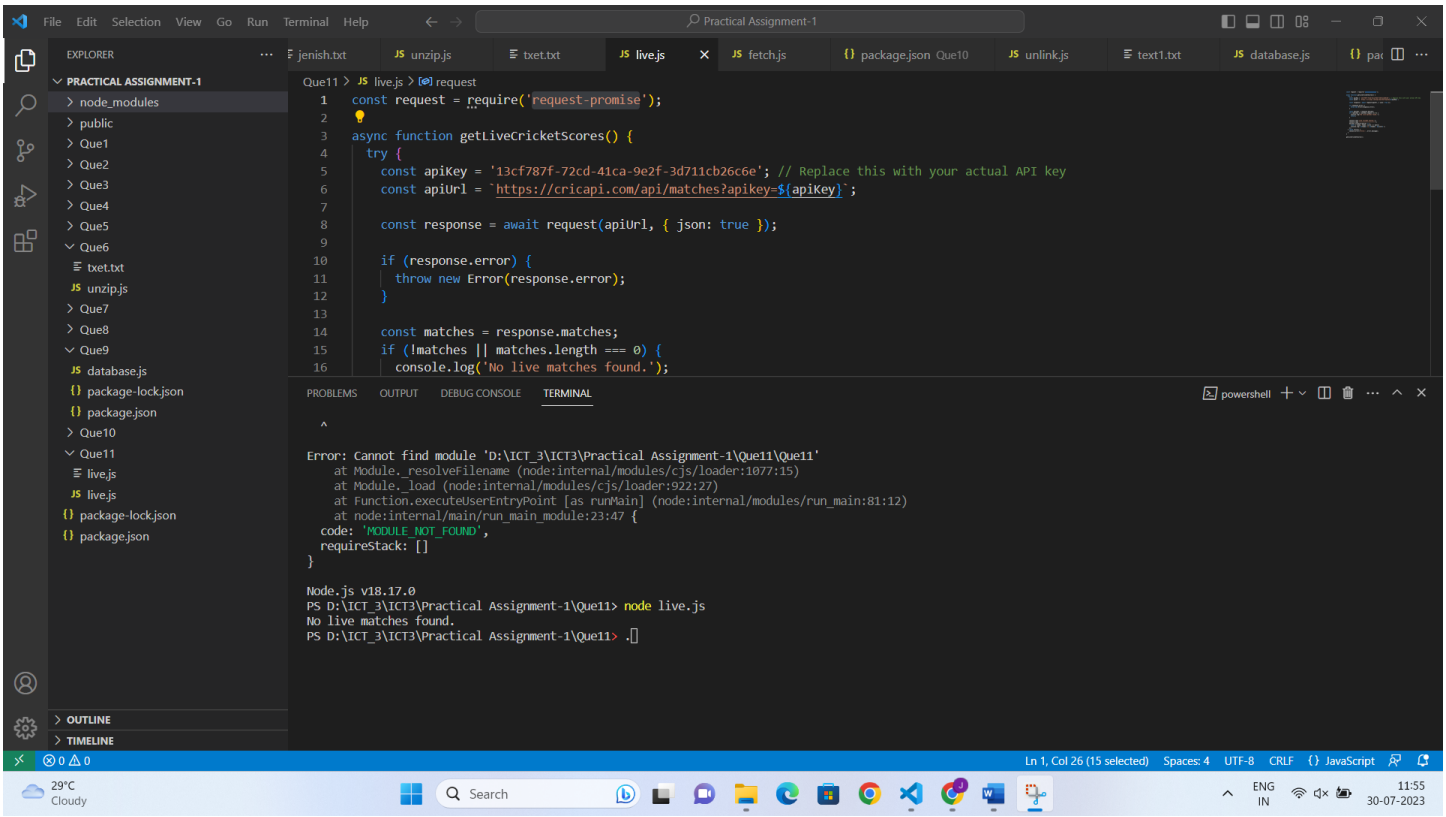
Que(11) Develop an application to show live cricket score.

Ans:-

Live.js

```
const request = require('request-promise');

async function getLiveCricketScores() {
  try {
    const apiKey = '13cf787f-72cd-41ca-9e2f-3d711cb26c6e'; // Replace this with your actual
API key
    const apiUrl = `https://cricapi.com/api/matches?apikey=${apiKey}`;

    const response = await request(apiUrl, { json: true });

    if (response.error) {
      throw new Error(response.error);
    }

    const matches = response.matches;
    if (!matches || matches.length === 0) {
      console.log('No live matches found.');
      return;
    }

    console.log('Live Cricket Scores:');
    console.log('----------------------');
    matches.forEach((match) => {
      const { team1, team2, score } = match;
      console.log(`${team1} vs ${team2}: ${score}`);
    });
  } catch (error) {
    console.error('Error:', error.message);
  }
}

getLiveCricketScores();
```

output:-


File Edit Selection View Go Run Terminal Help
Practical Assignment-1
EXPLORER
PRACTICAL ASSIGNMENT-1
node_modules
public
Que1
Que2
Que3
Que4
Que5
Que6
txet.txt
unzip.js
Que7
Que8
Que9
database.js
package-lock.json
package.json
Que10
Que11
live.js
live.js
package-lock.json
package.json
OUTLINE
TIMELINE
jenish.txt
unzip.js
txet.txt
live.js
fetch.js
package.json Que10
unlink.js
text1.txt
database.js
Que11 > live.js > request
1 const request = require('request-promise');
2
3 async function getLiveCricketScores() {
4 try {
5 const apiKey = '13cf787f-72cd-41ca-9e2f-3d711cb26c6e'; // Replace this with your actual API key
6 const apiUrl = `https://cricapi.com/api/matches?apikey=${apiKey}`;
7
8 const response = await request(apiUrl, { json: true });
9
10 if (response.error) {
11 throw new Error(response.error);
12 }
13
14 const matches = response.matches;
15 if (!matches || matches.length === 0) {
16 console.log('No live matches found.');
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell
^
Error: Cannot find module 'D:\ICT_3\ICT3\Practical Assignment-1\Que11\Que11'
at Module._resolveFilename (node:internal/modules/cjs/loader:1077:15)
at Module._load (node:internal/modules/cjs/loader:922:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
at node:internal/main/run_main_module:23:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
Node.js v18.17.0
PS D:\ICT_3\ICT3\Practical Assignment-1\Que11> node live.js
No live matches found.
PS D:\ICT_3\ICT3\Practical Assignment-1\Que11>
Ln 1, Col 26 (15 selected) Spaces: 4 UTF-8 CRLF JavaScript
29°C
Cloudy
Search
ENG
IN
11:55
30-07-2023