

Practical Assignment -1

Name: Tailor kruti P

Roll No: 76

Semester : 7th

Subject : Application development using Node.js

1. Develop a web server with following functionalities:

- Serve static resources.
- Handle GET request.
- Handle POST request.

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

const server = http.createServer((req, res) => {
  var ul = url.parse(req.url, true);
  if (req.url === "/hi") {
    var path = "./index.html";
    fs.readFile(path, (err, data) => {
      if (err) {
        res.writeHead(404, { 'Content-Type': 'text/html' });
        return res.end("404 page not found");
      }
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.write(data);
      res.end();
    });
  }
  else if (ul.pathname === "/process" && req.method === "GET") {
    {
      res.write("First name : " + ul.query.fname + "\n Last Name : "
+ ul.query.lname);
      res.end();
    }
  }
  else if (ul.pathname === "/process" && req.method === "POST") {
    {
      let body = '';
      req.on('data', chunk => {
        body += chunk.toString();
      });
      req.on('end', () => {
        req.end(body);
      });
    }
  }
});

server.listen(8080);
console.log("Server listening on port 8080");
```

The screenshot shows a Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The editor is open to a file named `server.js` in the `Q1` folder. The code is a simple HTTP server that listens on port 8080. It handles GET requests for `/process` and POST requests for `/data`. A 404 error is shown in the terminal, indicating that the address is already in use.

```
1 const http = require('http');
2 const fs = require('fs');
3 const url = require('url');
4
5 const server = http.createServer((req, res) => {
6   var ul = url.parse(req.url, true);
7   if (req.url === "/hi") {
8     var path = "./index.html";
9     fs.readFile(path, (err, data) => {
10       if (err) {
11         res.writeHead(404, { 'Content-Type': 'text/html' });
12         return res.end("404 page not found");
13       }
14       res.writeHead(200, { 'Content-Type': 'text/html' });
15       res.write(data);
16       res.end();
17     });
18   } else if (ul.pathname === "/process" && req.method === "GET") {
19     res.write("First name : " + ul.query.fname + " Last Name: " + ul.query.lname);
20     res.end();
21   } else if (ul.pathname === "/process" && req.method === "POST") {
22     let body = '';
23     req.on('data', chunk => {
24       body += chunk;
25     });
26   }
27 });
28 server.listen(8080, () => {
29   console.log('Server listening on port 8080');
30 });
```

Terminal output:

```
PS C:\Users\Veru\OneDrive\Documents\node> cd Q1
PS C:\Users\Veru\OneDrive\Documents\node\Q1> node server
Server listening on port 8080
node:events:491
    throw er; // Unhandled 'error' event
    ^
Error: listen EADDRINUSE: address already in use :::8080
```

The screenshot shows the same Visual Studio Code editor with the `server.js` file. The terminal now shows a different error, `4009`, which is a `syscall: 'listen', address: '::', port: 8080` error. This error occurs when the server attempts to listen on port 8080 but fails due to a system-level issue.

```
1 const http = require('http');
2 const fs = require('fs');
3 const url = require('url');
4
5 const server = http.createServer((req, res) => {
6   var ul = url.parse(req.url, true);
7   if (req.url === "/hi") {
8     var path = "./index.html";
9     fs.readFile(path, (err, data) => {
10       if (err) {
11         res.writeHead(404, { 'Content-Type': 'text/html' });
12         return res.end("404 page not found");
13       }
14       res.writeHead(200, { 'Content-Type': 'text/html' });
15       res.write(data);
16       res.end();
17     });
18   } else if (ul.pathname === "/process" && req.method === "GET") {
19     res.write("First name : " + ul.query.fname + " Last Name: " + ul.query.lname);
20     res.end();
21   } else if (ul.pathname === "/process" && req.method === "POST") {
22     let body = '';
23     req.on('data', chunk => {
24       body += chunk;
25     });
26   }
27 });
28 server.listen(8080, () => {
29   console.log('Server listening on port 8080');
30 });
```

Terminal output:

```
at emitErrorNT (node:net:176:18)
at process.processTicksAndRejections (node:internal/process/task_queues:121:2) {
  code: 'EADDRINUSE',
  error: 4009,
  syscall: 'listen',
  address: '::',
  port: 8080
}
```

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.)

```
const http= require('http');
const fs= require('fs');
http.createServer((req,res)=>{
  if(req.method=="GET"){
    if(req.url=="/"){
      res.end("Home page");
    }
    else if(req.url =="/gethello")
    {
      fs.readFile("./hello.html",(err,data)=>{
        if(err)
        {
          return res.send("something went wrong!!");
        }
        else{
          res.writeHead(200,{ 'Content-Type': 'text/html'});
          res.write(data);
          return res.end();
        }
      })
    }
  }
  else if(req.url=="/ajaxcall")
  {
    fs.readFile('./ajaxcall.html',(err,data)=>{
      if(err){
        return res.send("Something went wrong");
      }
      else{
        res.writeHead(200,{ 'Content-Type': 'text/html'});
        res.write(data);
        return res.end();
      }
    })
  }
}).listen(8080,()=>{
  console.log("Server listeing on port 8080");
})
```

Ajaxcall.html

```
<!DOCTYPE html>
<html>
  <body>
    <div id = "page_content"></div>
    <button onClick="LoadData()"> Fetch Page </button>
    <script>
      function LoadData(){
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange =function(){
          if(this.readyState == 4 && this.status ==200){
            document.getElementById("page_content").innerHTML=this
            .responseText;
          }
        };
        xhttp.open("GET", "/gethello",true);
        xhttp.send();
      }
    </script>
  </body>
</html>
```

Hello.html

```
<!DOCTYPE html>
<html>
  <body>
    <h3> Hello Node JS !!!</h3>
  </body>
</html>
```

Output:

The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The main editor window displays a JavaScript file named `server.js` with the following code:

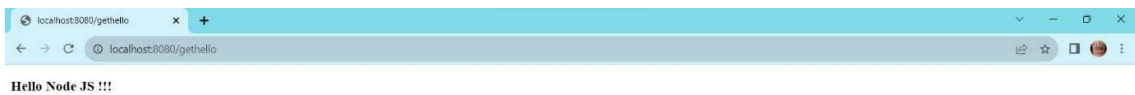
```
1 const http = require('http');
2 const fs = require('fs');
3 http.createServer((req, res) => {
4   if (req.method === 'GET') {
5     res.end('Home page');
6   }
7   else if (req.url === '/gethello') {
8     fs.readFile('./hello.html', (err, data) => {
9       if (err) {
10        return res.send('something went wrong!');
11      }
12      else {
13        res.writeHead(200, { 'Content-Type': 'text/html' });
14        res.write(data);
15        return res.end();
16      }
17    })
18  }
19  else (req.url === '/ajaxcall') {
20    fs.readFile('./ajaxcall.html', (err, data) => {
21      if (err) {
22        return res.send('something went wrong!');
23      }
24    })
25  }
26 })
27 ).listen(8080);
28 console.log('Server listening on port 8080');
```

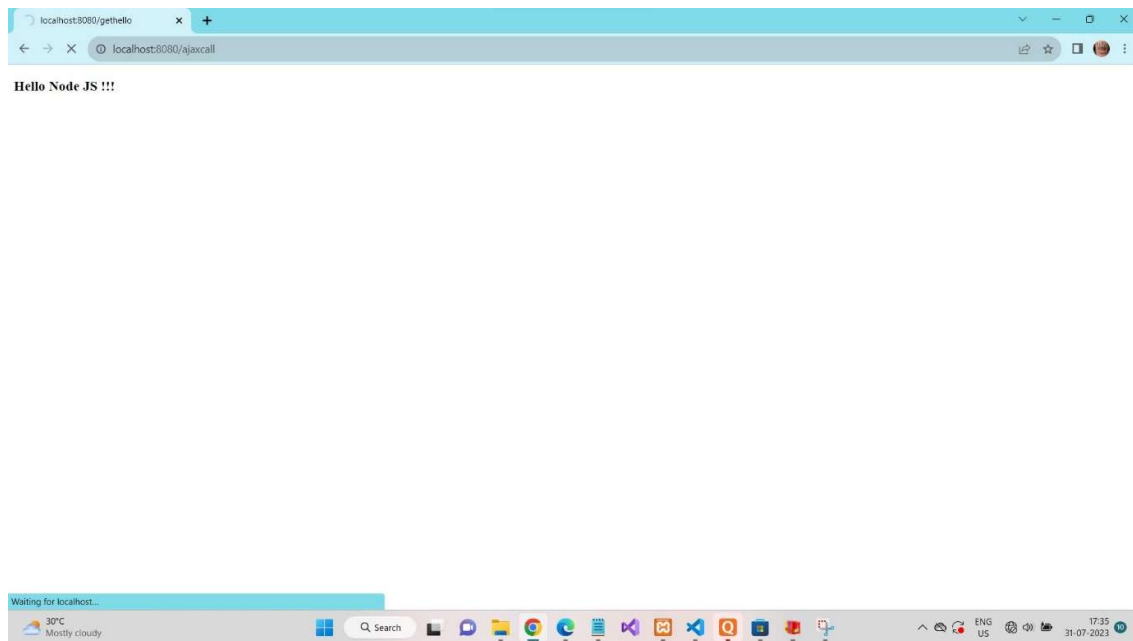
The terminal window at the bottom shows the command `node server` being executed, resulting in the output: `Server listening on port 8080`.

The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The main editor window displays a JavaScript file named `server.js` with the following code:

```
1 const http = require('http');
2 const fs = require('fs');
3 http.createServer((req, res) => {
4   if (req.method === 'GET') {
5     res.end('Home page');
6   }
7   else if (req.url === '/gethello') {
8     fs.readFile('./hello.html', (err, data) => {
9       if (err) {
10        return res.send('something went wrong!');
11      }
12      else {
13        res.writeHead(200, { 'Content-Type': 'text/html' });
14        res.write(data);
15        return res.end();
16      }
17    })
18  }
19  else (req.url === '/ajaxcall') {
20    fs.readFile('./ajaxcall.html', (err, data) => {
21      if (err) {
22        return res.send('something went wrong!');
23      }
24    })
25  }
26 })
27 ).listen(8080);
28 console.log('Server listening on port 8080');
```

The terminal window at the bottom shows the command `node server` being executed, resulting in the output: `Server listening on port 8080`.





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

Chatboat.js

```
module.exports.ChatbotReply = function(message){
  this.Bot_Age = 25;
  this.Bot_Name = "Kruti";
  this.Bot_University = "VNSGU";
  this.Bot_Country = "India";

  message = message.toLowerCase()
  if(message.indexOf("hi")>-1 || message.indexOf("hello")>-1 ||
message.indexOf("Welcome")>-1){
    return "Hi!!";
  }
  else if(message.indexOf("age")>-1 && message.indexOf("Your")){
    return "I'm" + this.Bot_Age;
  }
  else if(message.indexOf("how")>-1 && message.indexOf("are") &&
message.indexOf("you")){
    return "I'm Fine ^_^"
  }
  else if(message.indexOf("where")>-1 && message.indexOf("live") &&
message.indexOf("you")){
    return "i live in " +this.Bot_Country;
  }
}
```



```

    return "Sorry ,i didn't get it :(";
}

```

App.js

```

var Chatbot = require('./chatboat');
var readline = require('readline');

var r1 = readline.createInterface(process.stdin,process.stdout);
r1.setPrompt("You ==>");
r1.prompt();

r1.on('line',function(message){
    console.log('But ==>'+Chatbot.ChatbotReply(message));
    r1.prompt();
}).on('close',function(){
    process.exit(0);
});

```

Output:

The screenshot shows the Visual Studio Code interface with the following components:

- Explorer:** Shows the file structure with folders like 'node_modules', 'output', 'zip', 'files', 'global.js', and 'package-lock.json'.
- Editor:** Displays the code for 'app.js' with line numbers 1 through 13.
- Terminal:** Shows the command prompt output:


```

PS C:\Users\kruti\OneDrive\Documents\node> cd chatboat
PS C:\Users\kruti\OneDrive\Documents\node\chatboat> node app
You ==>hi
But ==>Hi!!
You ==>your age
But ==>i'm 25
You ==>how are you
But ==>i'm fine A A
You ==>where live you
But ==>i live in India
You ==>ok
But ==>sorry ,i didn't get it :(
You ==>

```
- Taskbar:** Shows the Windows taskbar at the bottom with the date 30-07-2023 and time 12:37.

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

Websocket.js

```
const WebSocket = require('ws')
var http = require('http');
var fs = require('fs');

var httpserver = http.createServer(function(req,res){
  console.log(req.url);
  if(req.url=="/"){
    fs.readFile("./index.html",(err,data)=>{
      res.write(data);
      res.end();
    })
  }
}).listen(8080,function(){
  console.log((new Date())+'server listening on port 8080');
});

const wss = new WebSocket.Server({server:httpserver})
wss.on("connection",(clientws)=>{
  clientws.send("Hello Client")
  clientws.on("message",(msg)=>{
    console.log("Received" +msg)
    clientws.send("Received"+msg)
  })
})
})
```

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>
  </body>
</html>
```

```

Data From Server
</h2>
<div id = "chatlog"></div></hr>
<h2>Data from Client</h2>
<input type = "text" id = "message"/>
<input type = "button" id="b1" onclick="sendMessage()" value = "send"/>
</body>
</html>

```

Output:

The screenshot shows a VS Code editor with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project structure with files like 'chatboat.js', 'index.html', and 'websockets.js'. The main editor window displays the code for 'websockets.js', which is a Node.js web server using the 'http' and 'fs' modules. The server listens on port 8080 and serves 'index.html' files. It also uses 'ws' for WebSocket connections. The terminal window shows an error message: 'at emitErrorNT (node:net:176:18) at process.processTicksAndRejections (node:internal/process/task_queues:82:21) { code: 'EADDRINUSE', errno: -4091, syscall: 'listen', address: '127.0.0.1', port: 8080 }'. This indicates that port 8080 is already in use.

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

```

const JSZip = require('jszip');
const fs = require('fs');
const zip = new JSZip();
try{
    const pdfData = fs.readFileSync('mypic.pdf');
    zip.file("PDFFile.pdf",pdfData);
    zip.file("Textfile.txt","Hello NodeJS \n");
    const images =["IMG_20200922_221737_989.jpg","IMG-20200922-WA0025.jpg"];
    const img = zip.folder("images");
    for(const image of images){
        const imageData = fs.readFileSync(image);

```

```

        img.file(image, imageData);
    }
    zip.generateNodeStream({type: 'nodebuffer', streamFiles: true}).pipe(fs.createWriteStream('mypic.zip')).on('finish', function(){
        console.log("mypic.zip written");
    });
} catch(err){
    console.error(err)
}

```

Output:

```

1  const JSZip = require('jszip');
2  const fs = require('fs');
3  const zip = new JSZip();
4  try{
5      const pdfData = fs.readFileSync('mypic.pdf');
6      zip.file("PDFFile.pdf", pdfData);
7      zip.file("textfile.txt", "Hello NodeJS \n");
8      const images = ["IMG_20200922_221737_989.jpg", "IMG_20200922_WA0025.jpg"];
9      const img = zip.folder("images");
10     for(const image of images){
11         const imageData = fs.readFileSync(image);
12         img.file(image, imageData);
13     }
14     zip.generateNodeStream({type: 'nodebuffer', streamFiles: true}).pipe(fs.createWriteStream('mypic.zip')).on('finish', function(){
15         console.log("mypic.zip written");
16     });
17 } catch(err){
18     console.error(err)
19 }

```

PS C:\Users\kruti\OneDrive\Documents\node> cd zip
PS C:\Users\kruti\OneDrive\Documents\node\zip> node app
mypic.zip written
PS C:\Users\kruti\OneDrive\Documents\node\zip>

6. Write a program to extract a zip file.

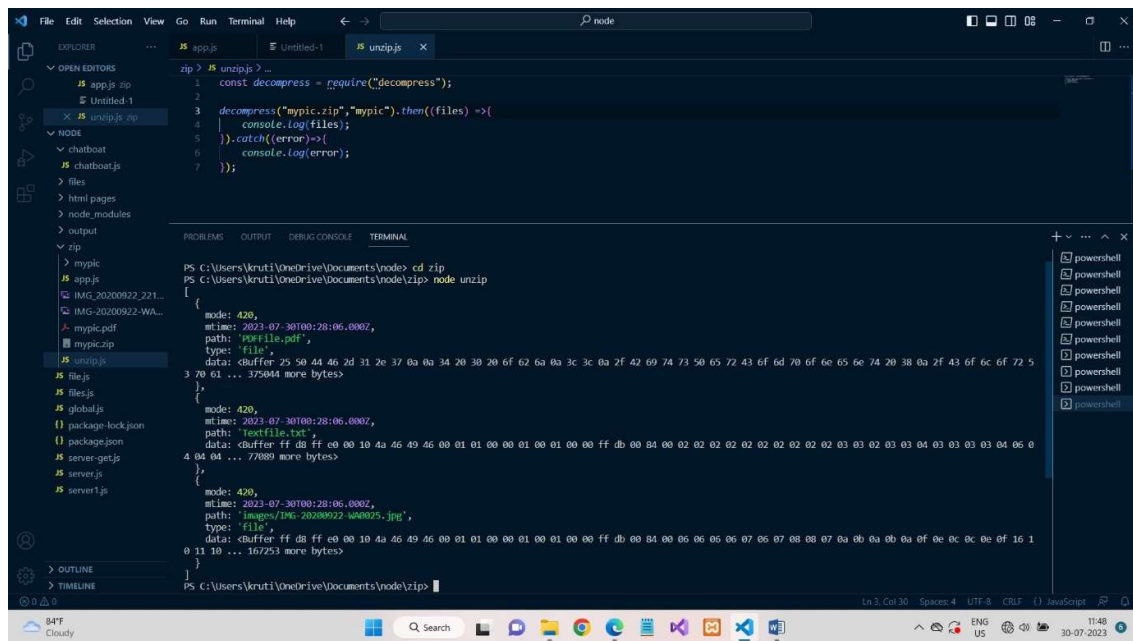
```

const decompress = require("decompress");

decompress("mypic.zip", "mypic").then((files) =>{
    console.log(files);
}).catch((error) =>{
    console.log(error);
});

```

Output:



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

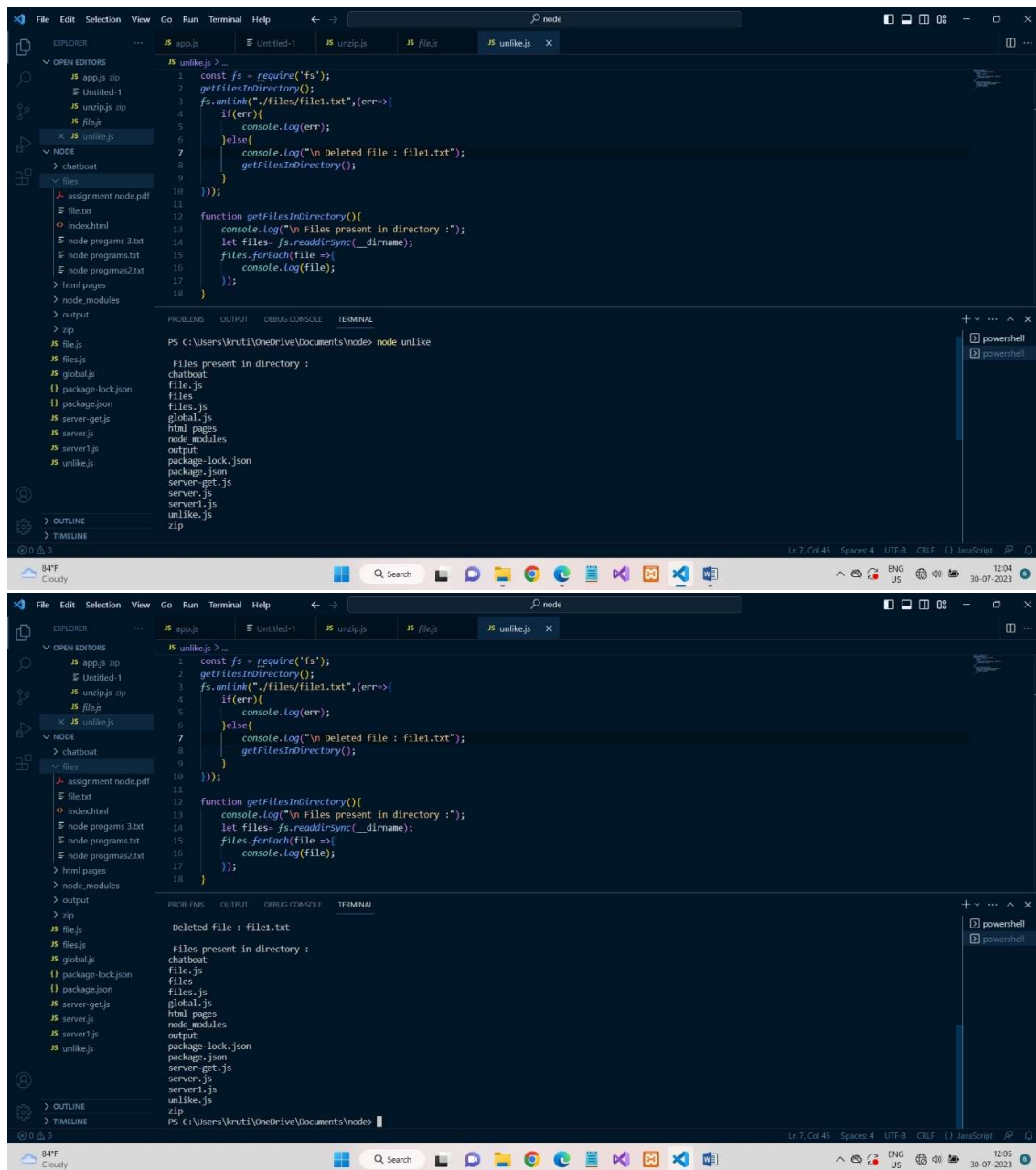
```

const fs = require('fs');
getFilesInDirectory();
fs.unlink("./files/file1.txt", (err=>{
  if(err){
    console.log(err);
  }else{
    console.log("\n Deleted file : file1.txt");
    getFilesInDirectory();
  }
}));

function getFilesInDirectory(){
  console.log("\n Files present in directory :");
  let files= fs.readdirSync(__dirname);
  files.forEach(file =>{
    console.log(file);
  });
}

```

Output:



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

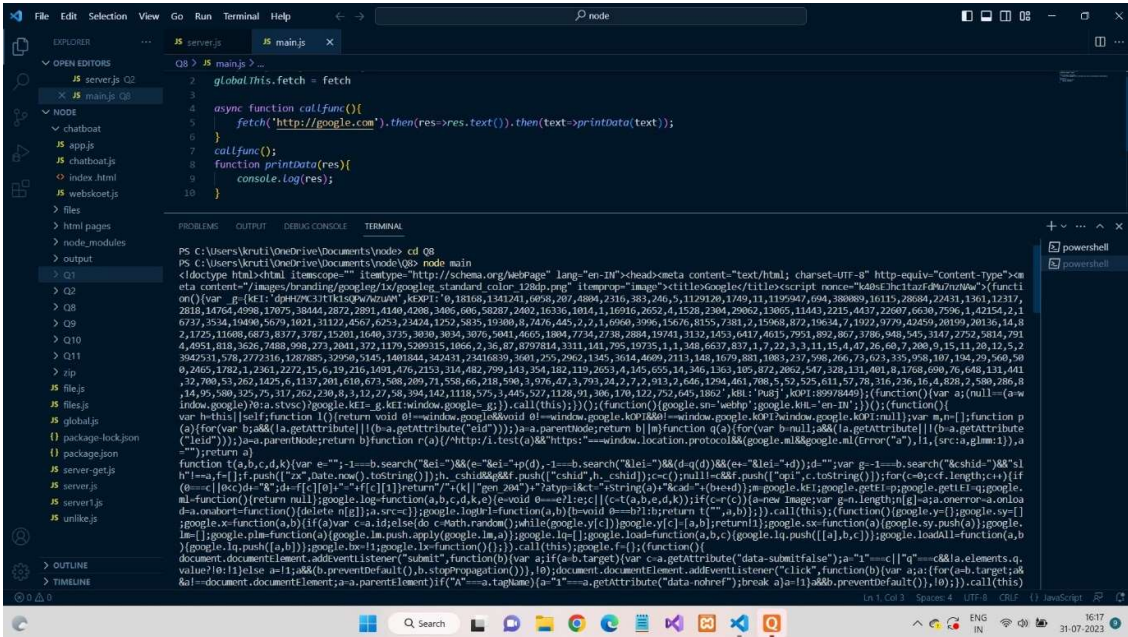
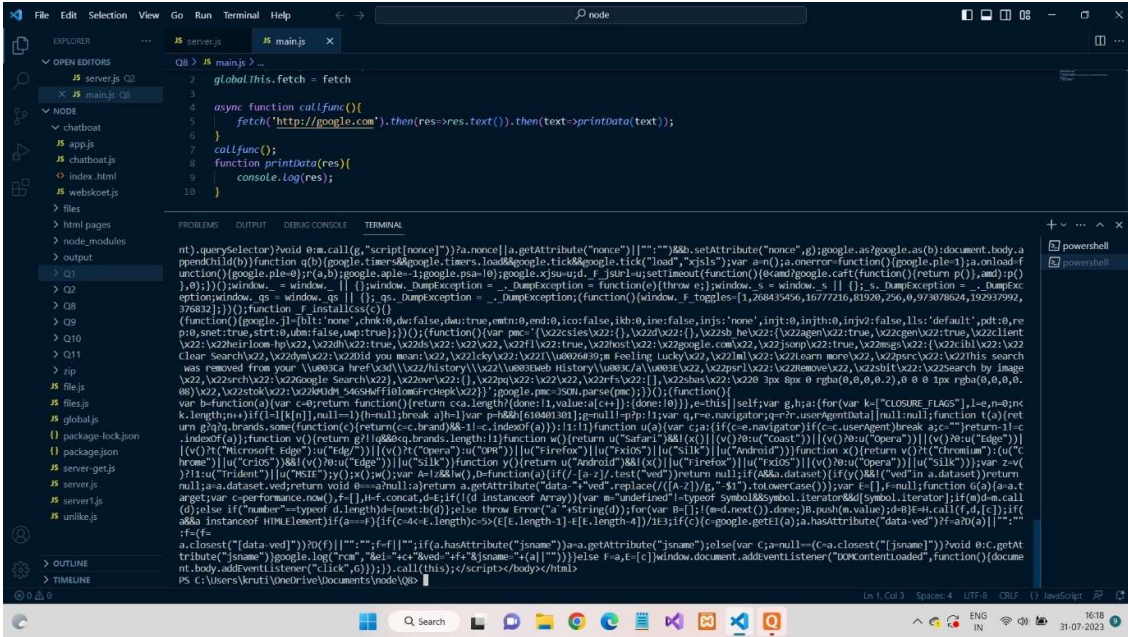
```

//import fetch from 'node-fetch'
globalThis.fetch = fetch

async function callfunc(){
  fetch('http://google.com').then(res=>res.text()).then(text=>printData(text));
}
callfunc();
function printData(res){

```


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.

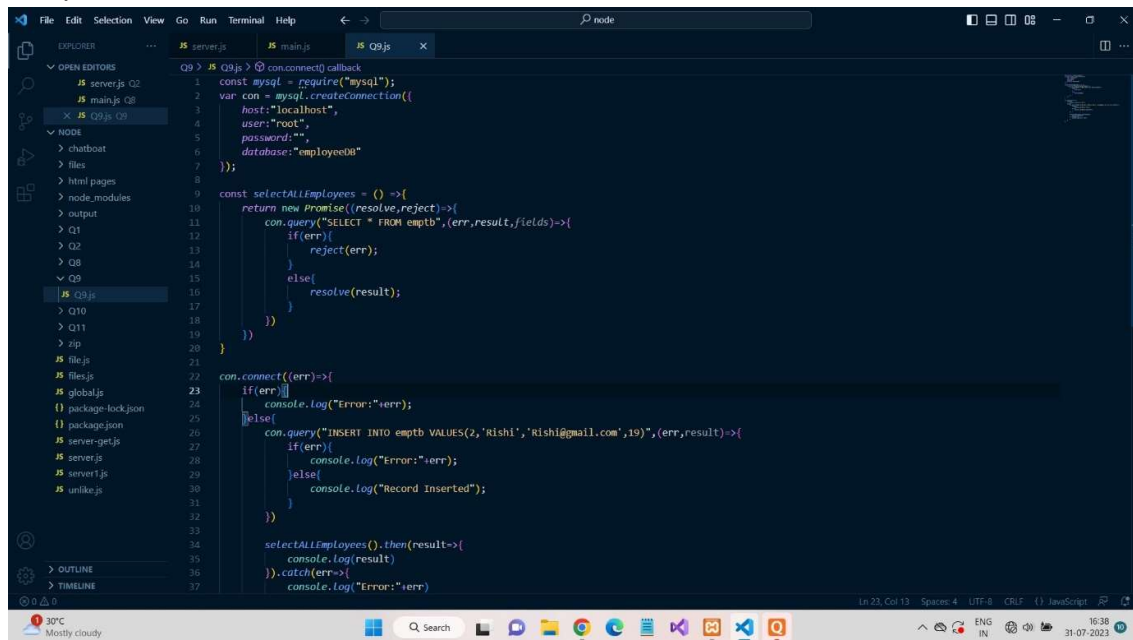
```
const mysql = require("mysql");
var con = mysql.createConnection({
  host:"localhost",
  user:"root",
  password:"",
  database:"employeeDB"
});

const selectALLEmployees = () =>{
  return new Promise((resolve,reject)=>{
    con.query("SELECT * FROM emptb",(err,result,fields)=>{
      if(err){
        reject(err);
      }
      else{
        resolve(result);
      }
    })
  })
}

con.connect((err)=>{
  if(err){
    console.log("Error:"+err);
  }else{
    con.query("INSERT INTO emptb
VALUES(2,'Rishi','Rishi@gmail.com',19)",(err,result)=>{
      if(err){
        console.log("Error:"+err);
      }else{
        console.log("Record Inserted");
      }
    })

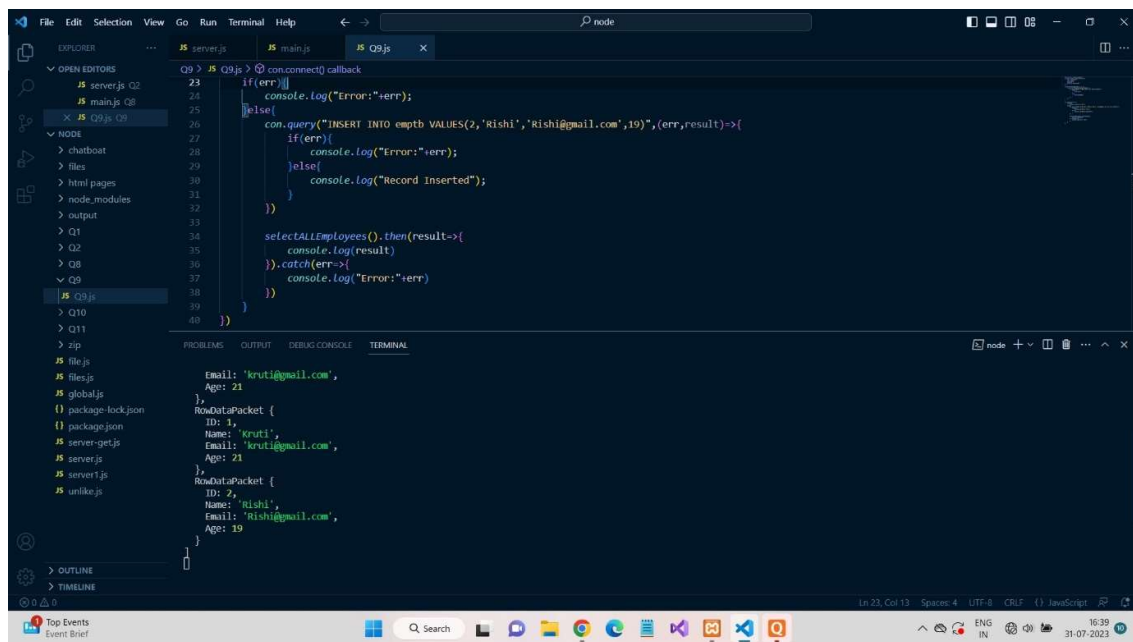
    selectALLEmployees().then(result=>{
      console.log(result)
    }).catch(err=>{
      console.log("Error:"+err)
    })
  }
})
```


Output:



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The code editor displays a JavaScript file named `Q9.js` with the following content:

```
Q9 > JS Q9.js > con.connect() callback
1  const mysql = require("mysql");
2  var con = mysql.createConnection({
3    host: "localhost",
4    user: "root",
5    password: "",
6    database: "employeeDB"
7  });
8
9  const selectAllEmployees = () => {
10   return new Promise((resolve, reject) => {
11     con.query("SELECT * FROM empth", (err, result, fields) => {
12       if (err) {
13         reject(err);
14       } else {
15         resolve(result);
16       }
17     });
18   });
19 }
20
21
22 con.connect((err) => {
23   if (err) {
24     console.log("Error: " + err);
25   } else {
26     con.query("INSERT INTO empth VALUES(2, 'Rishi', 'Rishi@gmail.com', 19)", (err, result) => {
27       if (err) {
28         console.log("Error: " + err);
29       } else {
30         console.log("Record Inserted");
31       }
32     });
33   }
34
35   selectAllEmployees().then(result => {
36     console.log(result);
37   }).catch(err => {
38     console.log("Error: " + err);
39   });
40 })
```



The screenshot shows the same VS Code editor with the `Q9.js` file. The output window at the bottom displays the following logs:

```
Email: 'kruti@gmail.com',
Age: 21
},
RowDataPacket {
  ID: 1,
  Name: 'Kruti',
  Email: 'kruti@gmail.com',
  Age: 21
},
RowDataPacket {
  ID: 2,
  Name: 'Rishi',
  Email: 'Rishi@gmail.com',
  Age: 19
}
```

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

Script 1:

```
console.log("This is the script 1");
```

Script 2:

```
console.log("This is the script 2");
```

Script 3:

```
console.log("This is the script 3");
```

Test:

```
console.log("This is a test script !!");
```

package.json :

```
{
  "name": "node",
  "version": "1.0.0",
  "main": "file.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "start": "node server.js",
    "test1": "test.js",
    "UD script1": "script1.js",
    "UD script2": "script2.js",
    "UD script3": "script3.js",
    "server": "Q10.js"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "decompress": "^4.2.1",
    "jszip": "^3.10.1",
    "mysql": "^2.18.1",
    "node-fetch": "^3.3.2",
    "package-lock.json": "^1.0.0",
    "package.json": "^2.0.1",
    "pakage.json": "^1.0.0",
    "ws": "^8.13.0"
  }
}
```