

### **Assignment- 1**

Name : Iwala Rohan KamleshBhai.

Roll No : 18.

Sem : 7<sup>th</sup>.

Subject : Application Development With Full Stack.(705).

Date : 30-Jul-23.

Github Link :-

1. Develop a web server with following functionalities:

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

Index.html

```
<!doctype html>
<html lang="en">

<head>
  <title>Title</title>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

  <!-- Bootstrap CSS v5.2.1 -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
rel="stylesheet"
  integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">

</head>

<body>
  <div class="container">
    <form action="/submit" method="post">
      <div class="mb-3">
        <label for="email" class="form-label">Email</label>
        <input type="email" class="form-control" name="email"
id="email" placeholder="Entar Email here ...">
      </div>
      <div class="mb-3">
        <label for="password" class="form-label">Password</label>
        <input type="password" class="form-control" name="password"
id="password"
          placeholder="Entar Password here ...">
      </div>
      <button type="submit" class="btn btn-primary">Submit</button>
    </form>
  </div>
```

```

    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"
    integrity="sha384-
oBqDVMmZ9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlhm/jp3"
crossorigin="anonymous">
    </script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.min.js"
    integrity="sha384-
7VPbUDkoPSGFnVtYi0QogXtr74QeVeeIs99Qfg5YCF+TidwNdjvaKZX19NZ/e6oz"
crossorigin="anonymous">
    </script>
</body>

</html>

```

Index.js

```

const http = require("http");
const fs = require("fs");

http.createServer((req, res) => {
    const responseObject = { API: req.url, METHOD: req.method, Body:
(req.body) ? req.body : {} };
    if (req.url === '/' && req.method == 'GET') {
        res.setHeader("Content-Type", "application/json");
        res.write(JSON.stringify(responseObject));
        res.end();
    }
    else if (req.url === '/data' && req.method == 'GET') {
        res.setHeader("Content-Type", "application/json");
        res.write(JSON.stringify(responseObject));
        res.end();
    }
    else if (req.url === '/form' && req.method == 'GET') {
        fs.readFile('./index.html', 'utf8', (err, data) => {
            if (err) {
                res.writeHead(500, { 'Content-Type': 'text/plain' });
                res.end('Internal Server Error');
            } else {
                res.writeHead(200, { 'Content-Type': 'text/html' });
                res.end(data);
            }
        });
    }
});

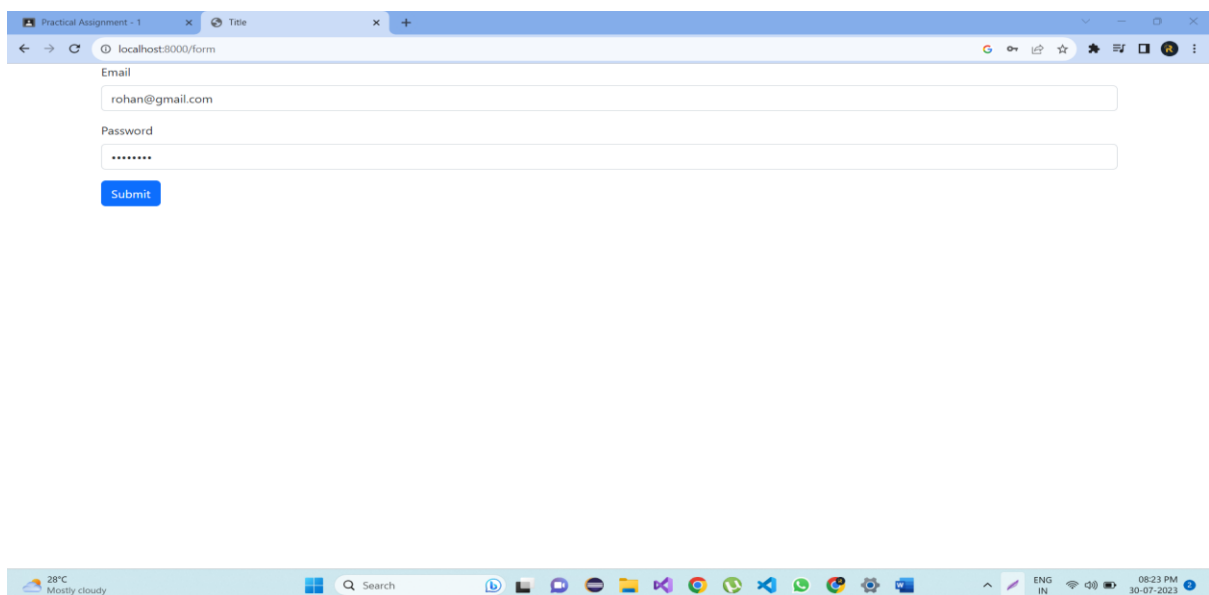
```

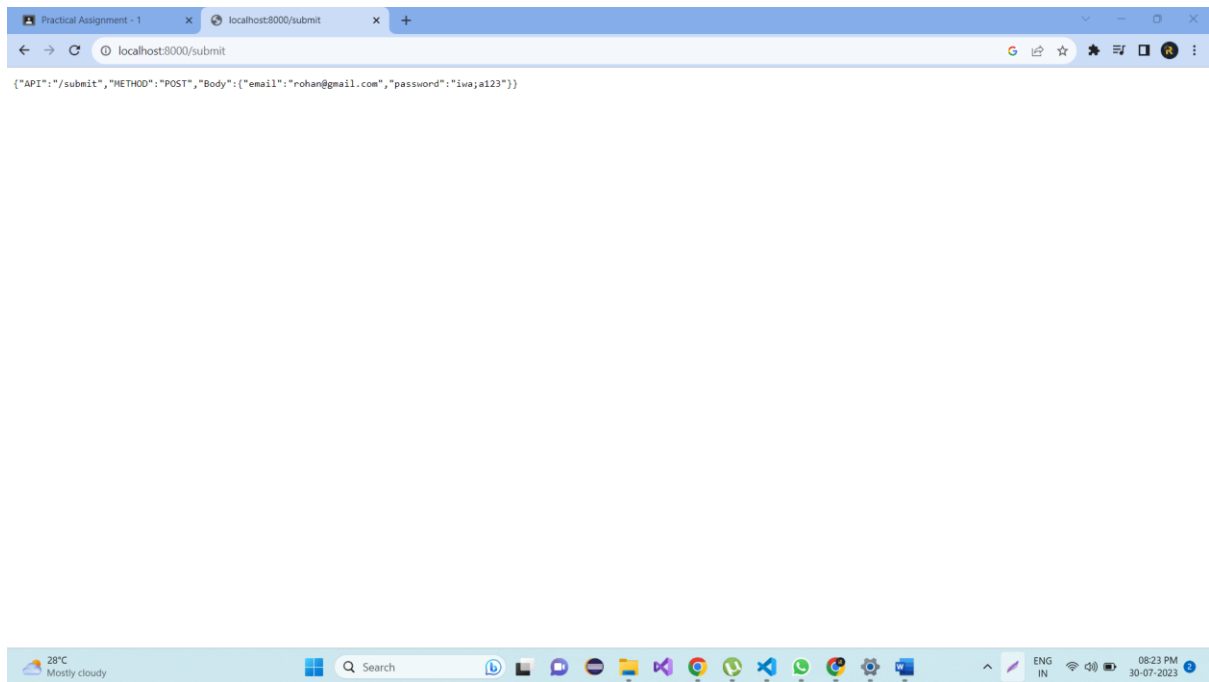
```

}
else if (req.url === '/submit' && req.method === 'POST') {
  let body = '';
  req.on('data', (chunk) => {
    body += chunk.toString();
  });
  req.on('end', () => {
    const formData = new URLSearchParams(body);
    const email = formData.get('email');
    const password = formData.get('password');
    responseObject.Body = {
      email: email,
      password: password
    }
    res.write(JSON.stringify(responseObject));
    res.end();
  });
}
else {
  res.write(JSON.stringify(responseObject));
  res.end();
}
}).listen(8000, () => {
  console.log("server is listening on port 8000");
  console.log("localhost:8000/");
});

```

Output:





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.)

Ajaxcall.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>
  <div id="content-Page">

    </div>
    <button onclick="fetchdata()">Fecth Data</button>
```

```

</body>
<script>
    function fetchdata() {
        var httpRequest = new XMLHttpRequest();
        httpRequest.onreadystatechange = function () {
            if (httpRequest.readyState == 4 && httpRequest.status == 200) {
                document.getElementById("content-Page").innerHTML =
httpRequest.responseText
            }
        };
        httpRequest.open("GET", '/gethello', true);
        httpRequest.send();
    }
</script>

</html>

```

Hello.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>
    <h1>hello Rohan IWala</h1>
</body>

</html>

```

Server.js

```

var http = require('http');
var fs = require('fs');
http.createServer((req, res) => {
    if (req.method === 'GET' && req.url === '/') {
        res.end("hello Page 1");
    }

    if (req.url === '/gethello') {
        fs.readFile('./hello.html', (err, data) => {
            if (err) {
                fs.write("page not found");
            }
        });
    }
});

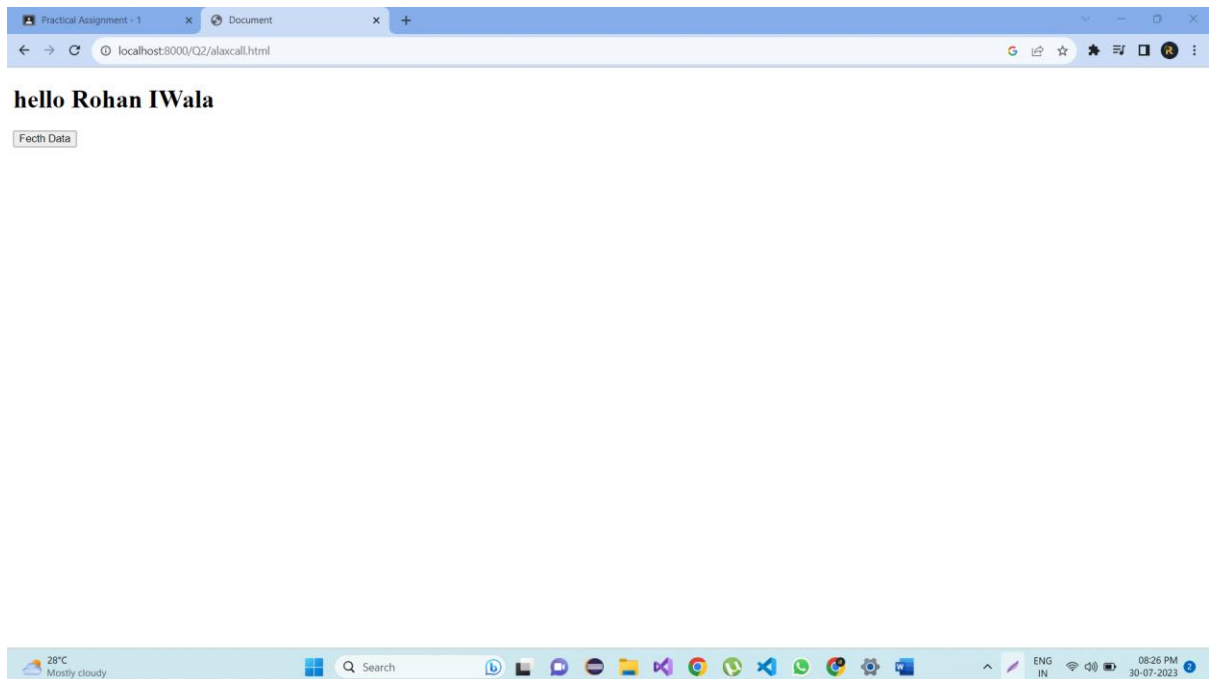
```

```

        fs.end();
    }
    else {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write(data);
        res.end();
    }
})
}
if (req.url === '/Q2/alaxcall.html') {
    fs.readFile('./alaxcall.html', (err, data) => {
        if (err) {
            fs.write("pagenotfound");
            fs.end();
        }
        else {
            res.writeHead(200, { 'Content-Type': 'text/html' });
            res.write(data);
            res.end();
        }
    })
}
// console.log("hello");
}).listen(8000);

```

Output:



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

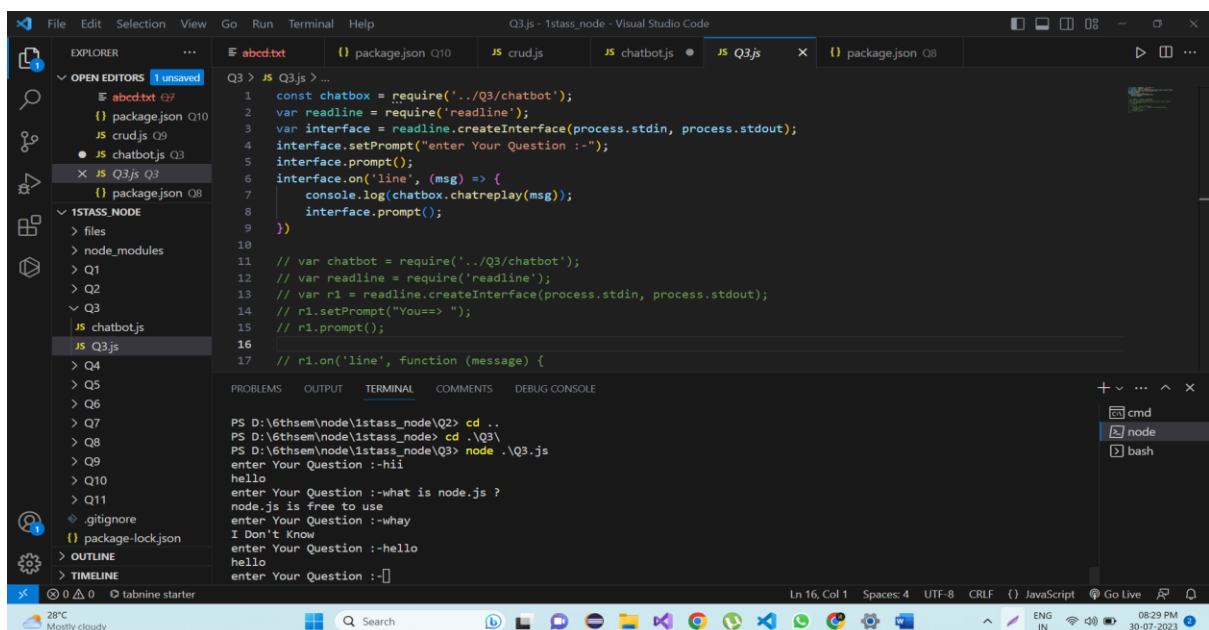
Chatbot.js

```
module.exports.chatreply = (message) => {  
  if (message.toLowerCase().indexOf('hii') > -1 ||  
message.toLowerCase().indexOf('hello') > -1) {  
    return "hello";  
  }  
  else if (message.toLowerCase().indexOf('What is Node.js ?') > -1 ||  
message.toLowerCase().indexOf('node.js') > -1) {  
    return "node.js is free to use";  
  }  
  else {  
    return "I Don't Know";  
  }  
}
```

Q3.js

```
const chatbox = require('../Q3/chatbot');  
var readline = require('readline');  
var interface = readline.createInterface(process.stdin, process.stdout);  
interface.setPrompt("enter Your Question :-");  
interface.prompt();  
interface.on('line', (msg) => {  
  console.log(chatbox.chatreply(msg));  
  interface.prompt();  
})
```

Output:



The screenshot shows the Visual Studio Code interface with the 'Q3.js' file open in the editor. The file contains the code for the chatbot module and the main application logic. The terminal at the bottom shows the execution of the program, demonstrating the chatbot's responses to various inputs.

```
Q3.js  
1 const chatbox = require('../Q3/chatbot');  
2 var readline = require('readline');  
3 var interface = readline.createInterface(process.stdin, process.stdout);  
4 interface.setPrompt("enter Your Question :-");  
5 interface.prompt();  
6 interface.on('line', (msg) => {  
7   console.log(chatbox.chatreply(msg));  
8   interface.prompt();  
9 })  
10  
11 // var chatbot = require('../Q3/chatbot');  
12 // var readline = require('readline');  
13 // var rl = readline.createInterface(process.stdin, process.stdout);  
14 // rl.setPrompt("You==> ");  
15 // rl.prompt();  
16  
17 // rl.on('line', function (message) {  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

Terminal Output:

```
PS D:\6thsem\node\1stass_node> cd ..  
PS D:\6thsem\node\1stass_node> cd .\Q3\  
PS D:\6thsem\node\1stass_node\Q3> node .\Q3.js  
enter Your Question :-hii  
hello  
enter Your Question :-what is node.js ?  
node.js is free to use  
enter Your Question :-why  
I Don't Know  
enter Your Question :-hello  
hello  
enter Your Question :-[]
```



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

Q4.html

```
<!DOCTYPE html>
<html>

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

<body>
  <h1>WebSocket Chat Bot</h1>
  <div id="chat">
    <div id="messages"></div>
    <input type="text" id="inputMessage" placeholder="Type your message
here..." />
    <button onclick="sendMessage()">Send</button>
  </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>
```

## Mainchat.js

```
const http = require('http');
const st = require('node-static');
const chatBot = require('../Q3/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();
});

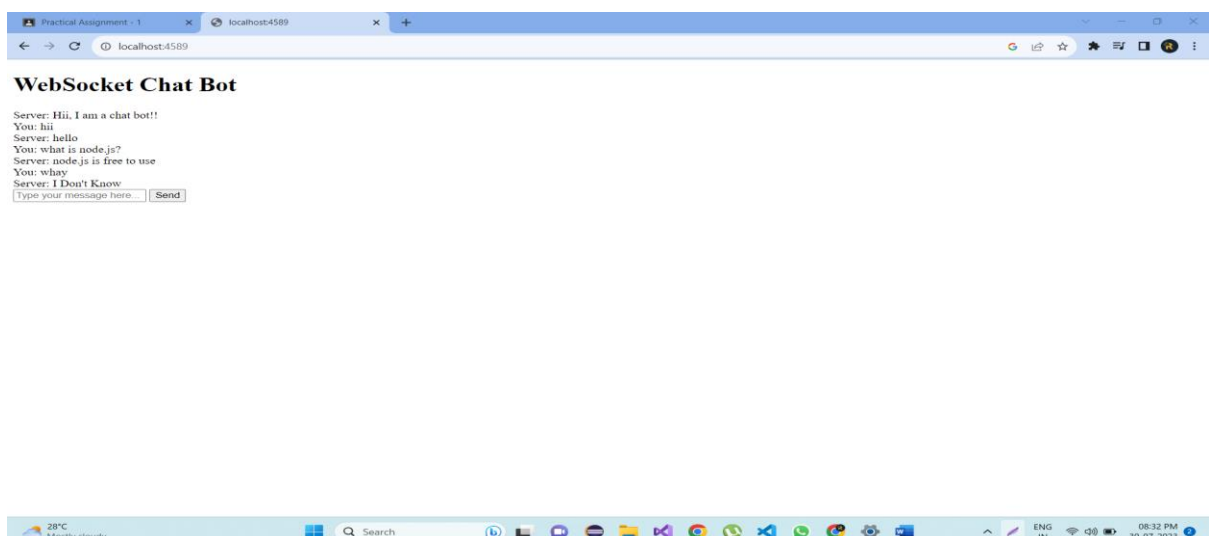
server.listen(4589, () => {
  console.log("Server listening on 4589");
});

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

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

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

## Output:



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

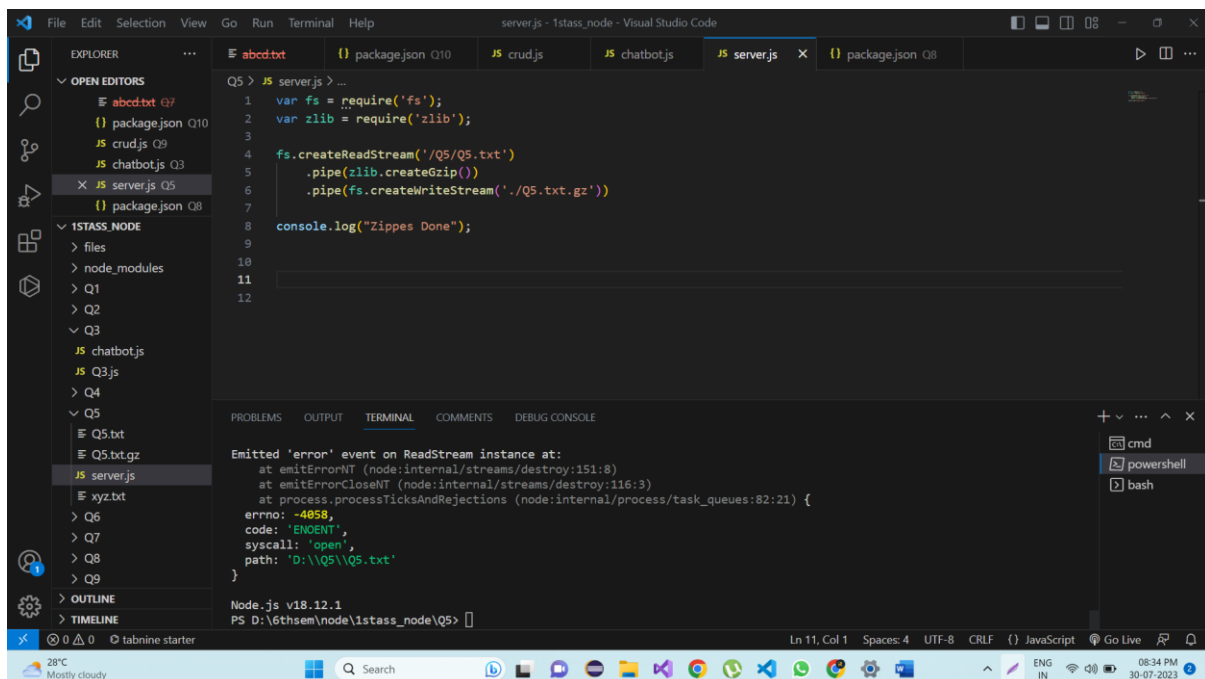
Server.js

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

fs.createReadStream('/Q5/Q5.txt')
  .pipe(zlib.createGzip())
  .pipe(fs.createWriteStream('./Q5.txt.gz'))

console.log("Zippes Done");
```

output:



6. Write a program to extract a zip file.

Decom.js

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

fs.createReadStream('../Q5/Q5.txt.gz')
  .pipe(zlib.createGunzip())
  .pipe(fs.createWriteStream('../Q5/xyz.txt', 'utf-8'))

console.log("dcompress done")
```

output:

```
Q5 > JS serverjs > ...
1 var fs = require('fs');
2 var zlib = require('zlib');
3
4 fs.createReadStream('/Q5/Q5.txt')
5   .pipe(zlib.createGzip())
6   .pipe(fs.createWriteStream('./Q5.txt.gz'))
7
8 console.log("Zippes Done");
9
10
11
12
```

```
PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE
+ v ... ^ x
cmd
powershell
bash

Emitted 'error' event on ReadStream instance at:
  at emitErrorNT (node:internal/streams/destroy:151:8)
  at emitErrorCloseNT (node:internal/streams/destroy:116:3)
  at process.processTicksAndRejections (node:internal/process/task_queues:82:21) {
    errno: -4058,
    code: 'ENOENT',
    syscall: 'open',
    path: 'D:\Q5\Q5.txt'
  }

Node.js v18.12.1
PS D:\6thsem\node\1stass_node\Q5>
```

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

Promiseunlink.js

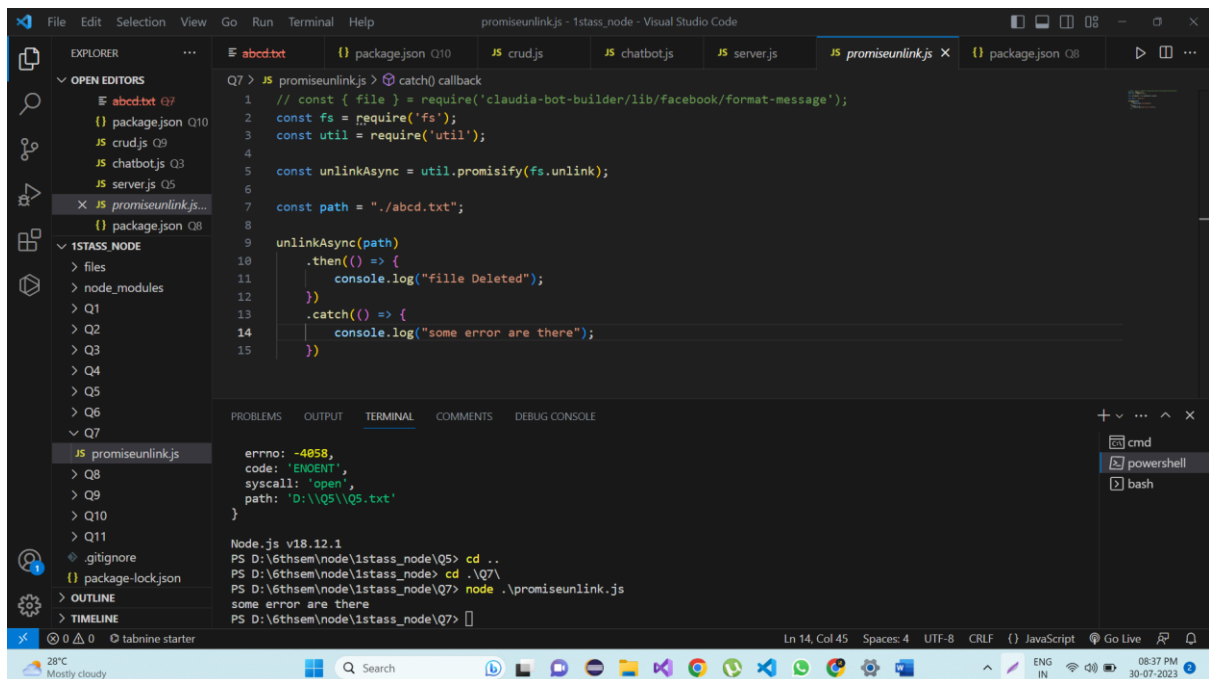
```
// const { file } = require('claudia-bot-builder/lib/facebook/format-
message');
const fs = require('fs');
const util = require('util');

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

const path = "./abcd.txt";

unlinkAsync(path)
  .then(() => {
    console.log("file Deleted");
  })
  .catch(() => {
    console.log("some error are there");
  })
})
```

Output:



The screenshot shows the Visual Studio Code editor with a file named 'promiseunlink.js' open. The code in the file is as follows:

```
1 // const { file } = require('claudia-bot-builder/lib/facebook/format-message');
2 const fs = require('fs');
3 const util = require('util');
4
5 const unlinkAsync = util.promisify(fs.unlink);
6
7 const path = './abcd.txt';
8
9 unlinkAsync(path)
10 .then(() => {
11     console.log("file Deleted");
12 })
13 .catch(() => {
14     console.log("some error are there");
15 })
```

The terminal output shows an error: 'ENOENT' (Error: file or directory does not exist) because the file 'abcd.txt' does not exist. The error message is: 'errno: -4058, code: 'ENOENT', syscall: 'open', path: 'D:\\Q5\\Q5.txt''. The terminal also shows the command 'node .\\promiseunlink.js' being executed.

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

Q8.js

```
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');
            }

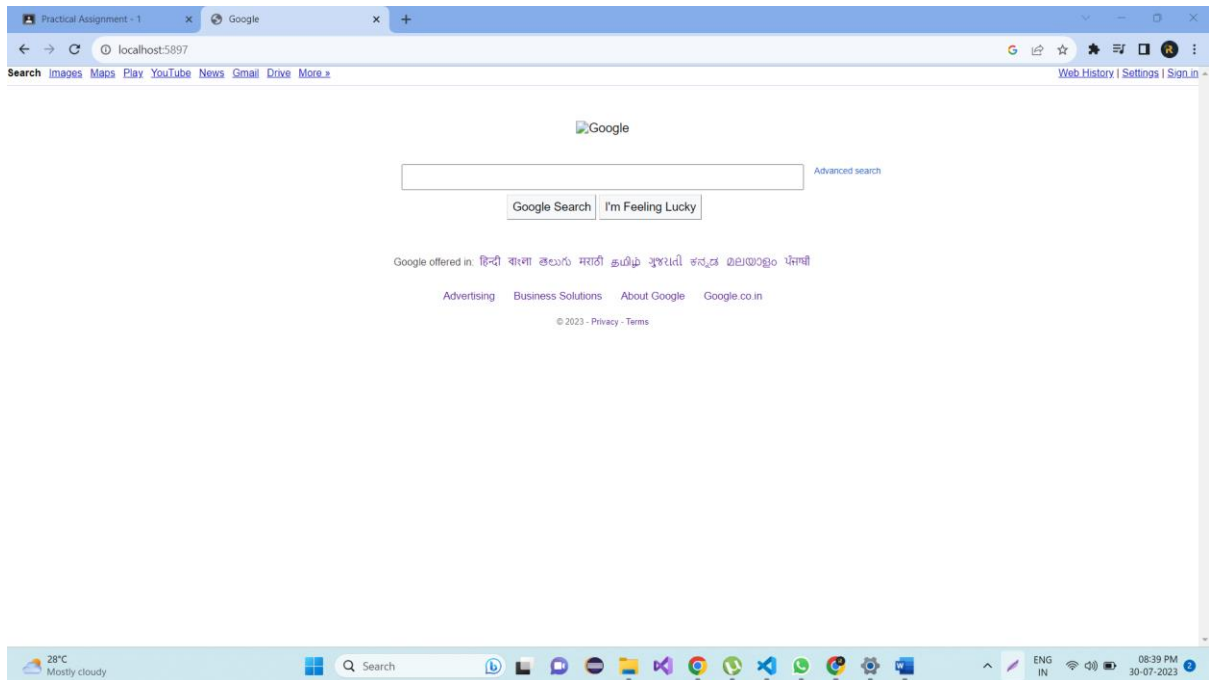
            const data = await response.text();
            // console.log(data);
            res.end(data);
        } catch (error) {
            console.error('Error fetching data:', error.message);
        }
    }

    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.

Crud.js

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

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

const db = mysql.getInstance(conn);

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

    var sql = "INSERT INTO employeetb (empid,empname,joinDate) VALUES (201,'abc','25-06-2022')";
    console.log("Record Inserted!!");
    return db.exec(sql);
  })
```

```

.then((display) => {
    // var sqlDisplay = "SELECT * FROM employeeetb";
    // console.log(display);
    return db.exec("SELECT * FROM employeeetb");
})

.then((result) => {
    console.log('Employee Name \t Date of Join');
    for (var i in result) {
        console.log(result[i].empname + " \t\t " + result[i].joindate);
    }
})

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

```

Output:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists the project files. The main editor window shows the JavaScript code. The TERMINAL panel at the bottom displays the output of running the script:

```

abc undefined
PS D:\6thsem\node\1stass_node\Q9> node .\crud.js
Connected!!
Record Inserted!!
Employee Name Date of Join
abcd 21-aug-2023
abcd 21-aug-2023
abcd 21-aug-2023
abcd 21-aug-2023
abc 25-06-2022
abc

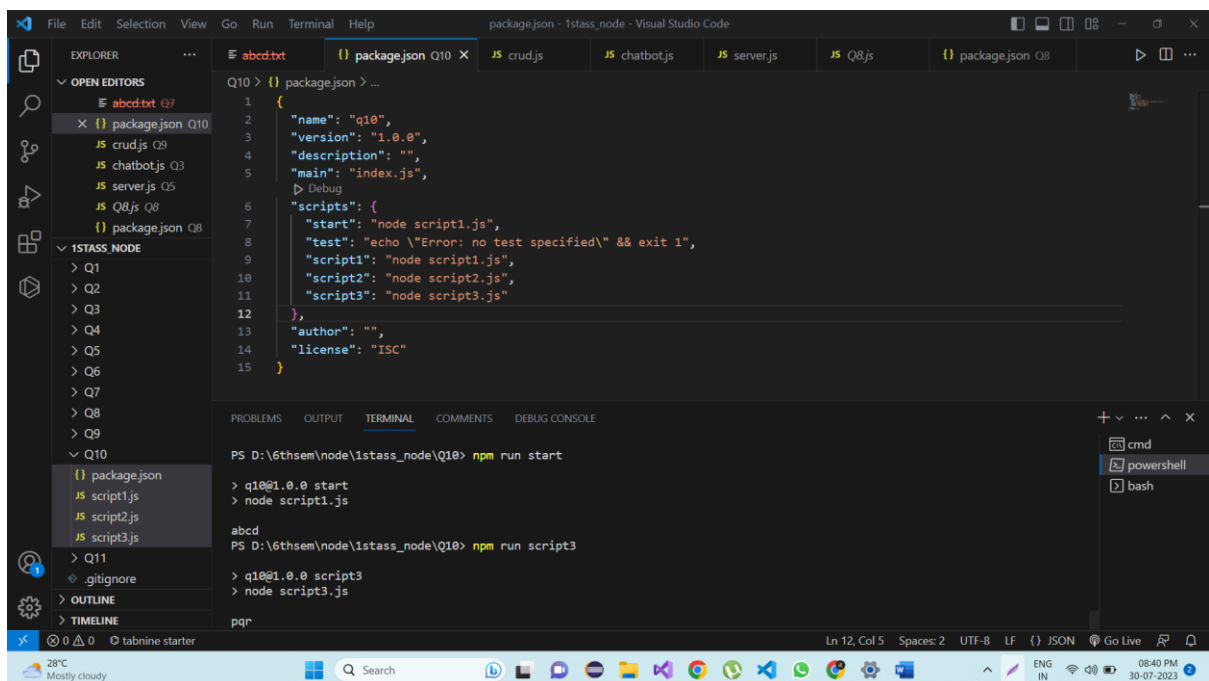
```

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

Pacackage.json

```
{
  "name": "q10",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node script1.js",
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "script1": "node script1.js",
    "script2": "node script2.js",
    "script3": "node script3.js"
  },
  "author": "",
  "license": "ISC"
}
```

Output:



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named '1stass\_node' with a 'package.json' file selected. The main editor displays the content of 'package.json', which matches the code provided in the previous block. Below the editor, the 'TERMINAL' tab is active, showing the command prompt output of running 'npm run start' and 'npm run script3'. The output for 'npm run start' shows 'q10@1.0.0 start' followed by 'node script1.js' and the output 'abcd'. The output for 'npm run script3' shows 'q10@1.0.0 script3' followed by 'node script3.js' and the output 'pqr'.

```
Q10 > {} package.json > ...
1 {
2   "name": "q10",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node script1.js",
8     "test": "echo \\\"Error: no test specified\\\" && exit 1",
9     "script1": "node script1.js",
10    "script2": "node script2.js",
11    "script3": "node script3.js"
12  },
13  "author": "",
14  "license": "ISC"
15 }

PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE

PS D:\6thsem\node\1stass_node\Q10> npm run start

> q10@1.0.0 start
> node script1.js

abcd
PS D:\6thsem\node\1stass_node\Q10> npm run script3

> q10@1.0.0 script3
> node script3.js

pqr
```



11. Develop an application to show live cricket score.

Crick.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="./css/bootstrap.min.css">
  <script src="./js/bootstrap.bundle.js" defer></script>
  <title>Document</title>
</head>

<body>
  <div class="container my-4">
    <h2 class="my-4">Cricket Score</h2>
    <div class="d-flex justify-content-around">
      <div class="row">

        <div class="card text mb-3">
          <div class="card-header" id="card-header1"></div>
          <div class="card-body">
            <h5 class="card-title" id="card-title1"></h5>
            <p class="card-text">
              <div id="batting-team1"></div>
              <div id="match-score1"></div>
            </p>
          </div>
        </div>
        <div class="card text mb-3">
          <div class="card-header" id="card-header2"></div>
          <div class="card-body">
            <h5 class="card-title" id="card-title2"></h5>
            <p class="card-text">
              <div id="batting-team2"></div>
              <div id="match-score2"></div>
            </p>
          </div>
        </div>
        <div class="card mb-3">
          <div class="card-header" id="card-header3"></div>
          <div class="card-body">
            <h5 class="card-title" id="card-title3"></h5>
            <p class="card-text">
```

```

        <div id="batting-team3"></div>
        <div id="match-score3"></div>

    </p>
  </div>
</div>
</div>
</div>
</div>
</div>
</body>
<script>
  const ws = new WebSocket("ws://localhost:3300");
  ws.addEventListener("message", (msg) => {
    let res = JSON.parse(msg.data);
    console.log(res);
    let match = res[1];
    document.getElementById("card-header1").innerText = match.name;
    document.getElementById("card-title1").innerText = match.status;
    document.getElementById("batting-team1").innerText =
match.score[0].inning;
    document.getElementById("match-score1").innerText = match.score[0].r +
 '/' + res[0].score[1].w + ' Over : ' + res[0].score[1].o;

    let match2 = res[2];
    document.getElementById("card-header2").innerText = match2.name;
    document.getElementById("card-title2").innerText = match2.status;
    document.getElementById("batting-team2").innerText =
match2.score[0].inning;
    document.getElementById("match-score2").innerText = match2.score[0].r
+ '/' + res[0].score[1].w + ' Over : ' + res[0].score[1].o;

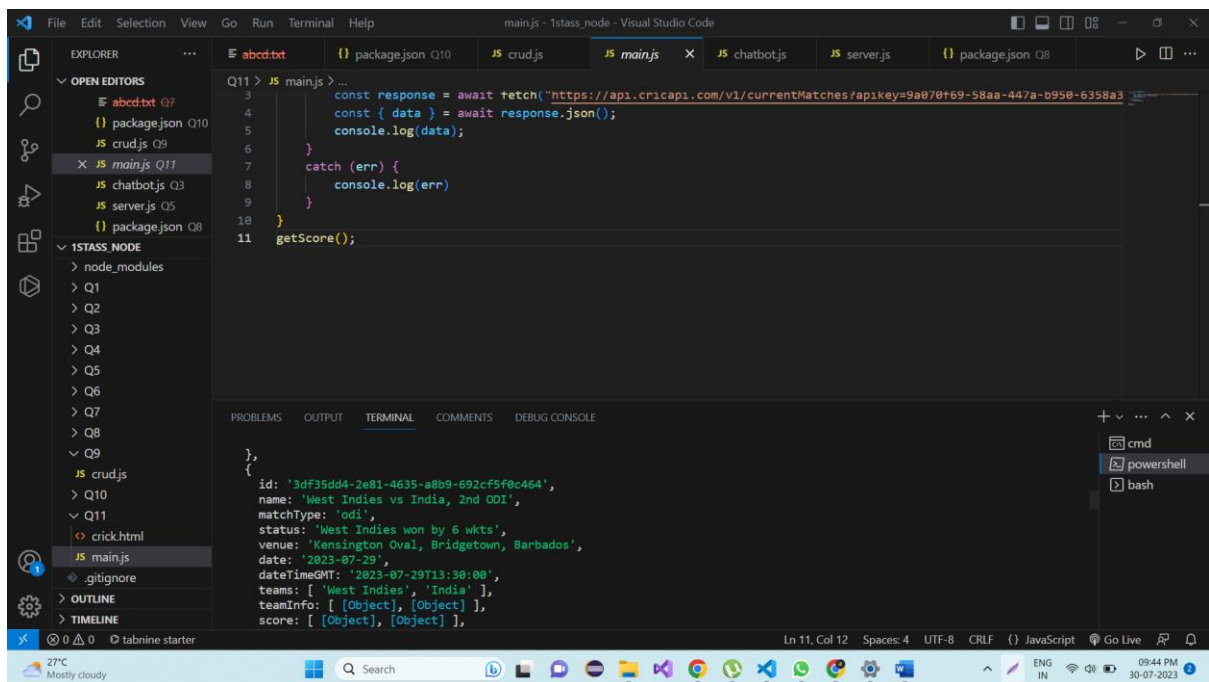
    let match3 = res[3];
    document.getElementById("card-header3").innerText = match3.name;
    document.getElementById("card-title3").innerText = match3.status;
    document.getElementById("batting-team3").innerText =
match3.score[0].inning;
    document.getElementById("match-score3").innerText = match3.score[0].r
+ '/' + res[0].score[1].w + ' Over : ' + res[0].score[1].o;
  })
</script>
</html>

```

Main.js

```
async function getScore() {
  try {
    const response = await
fetch("https://api.cricapi.com/v1/currentMatches?apikey=9a070f69-58aa-447a-
b950-6358a39bf75e&offset=0");
    const { data } = await response.json();
    console.log(data);
  }
  catch (err) {
    console.log(err)
  }
}
getScore();
```

output:



```
Q11> JS main.js > ...
3      const response = await fetch("https://api.cricapi.com/v1/currentMatches?apikey=9a070f69-58aa-447a-b950-6358a3
4      const { data } = await response.json();
5      console.log(data);
6    }
7    catch (err) {
8      console.log(err)
9    }
10  }
11  getScore();

},
{
  id: '3df35dd4-2e01-4635-a0b9-692cf5f0c464',
  name: 'West Indies vs India, 2nd ODI',
  matchType: 'odi',
  status: 'West Indies won by 6 wkts',
  venue: 'Kensington Oval, Bridgetown, Barbados',
  date: '2023-07-29',
  dateTimeGMT: '2023-07-29T13:30:00',
  teams: [ 'West Indies', 'India' ],
  teamInfo: [ [Object], [Object] ],
  score: [ [Object], [Object] ],
```