

Name : Dhodiya Krupali R.

Roll No : 11

MSC - ICT ( Sem-3 )

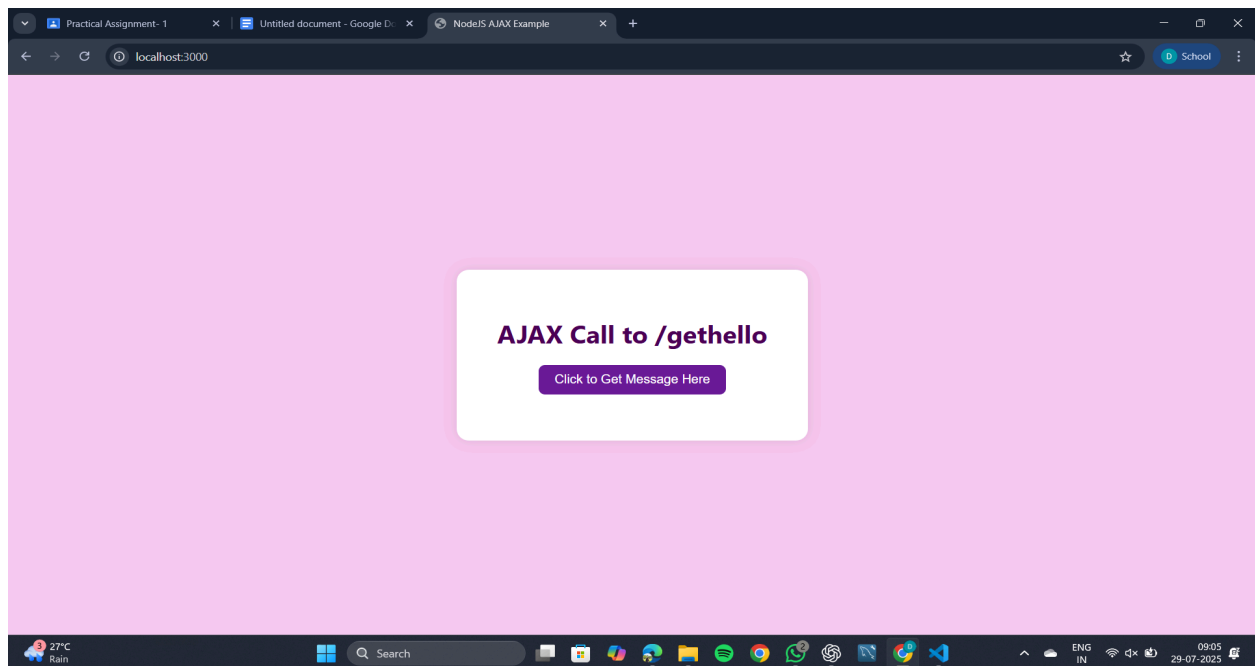
# Practical Assignment- 1

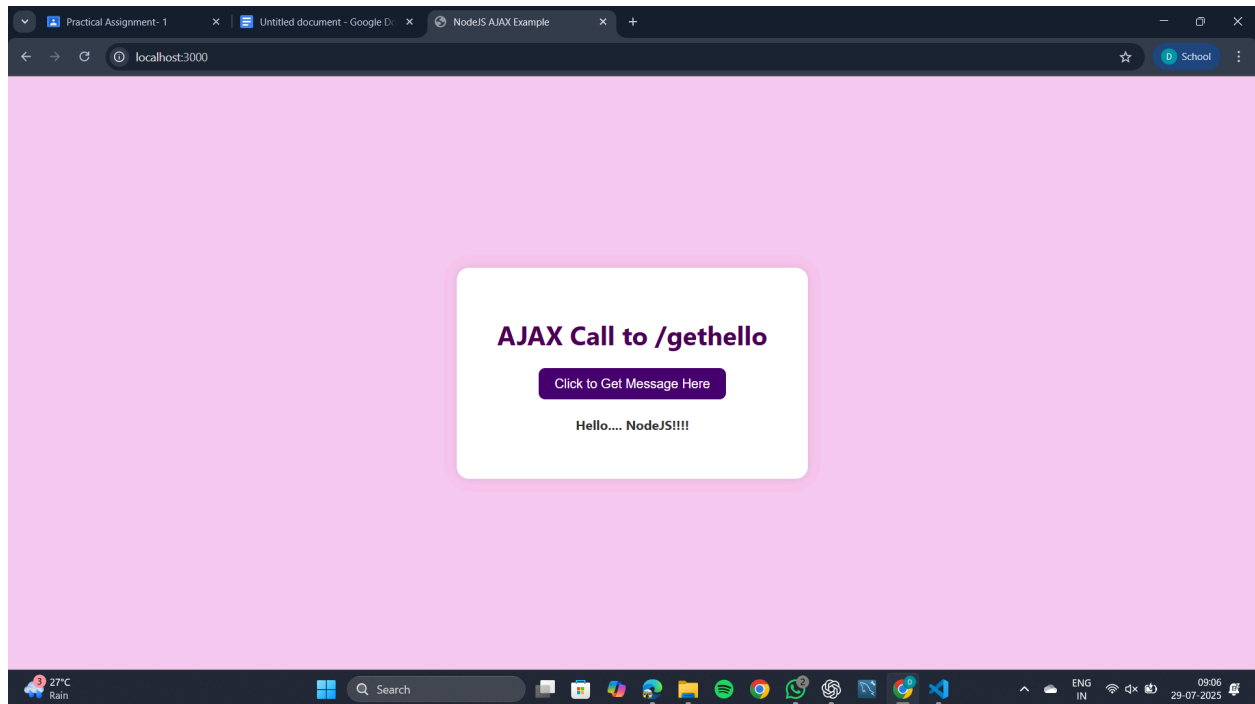
GitHub Link : [https://github.com/Krupali119/11\\_Krupali\\_dhodiya\\_701\\_A1](https://github.com/Krupali119/11_Krupali_dhodiya_701_A1)

Q.1. 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.)

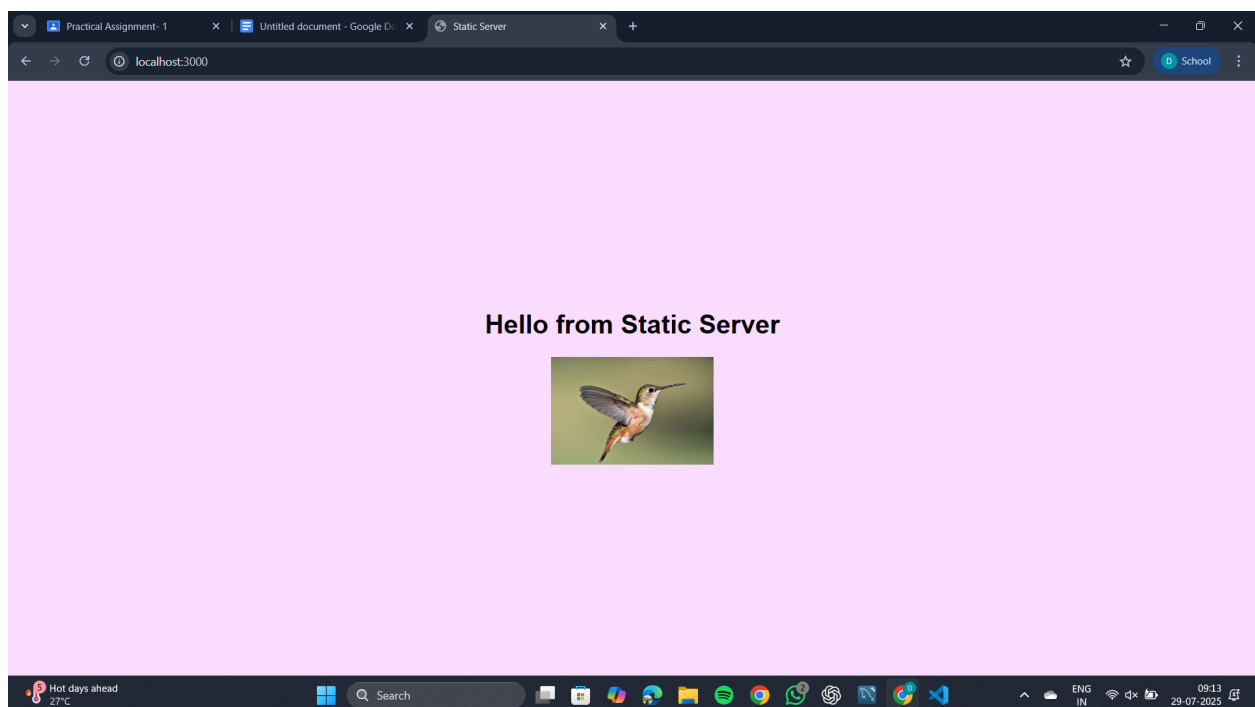
Output :





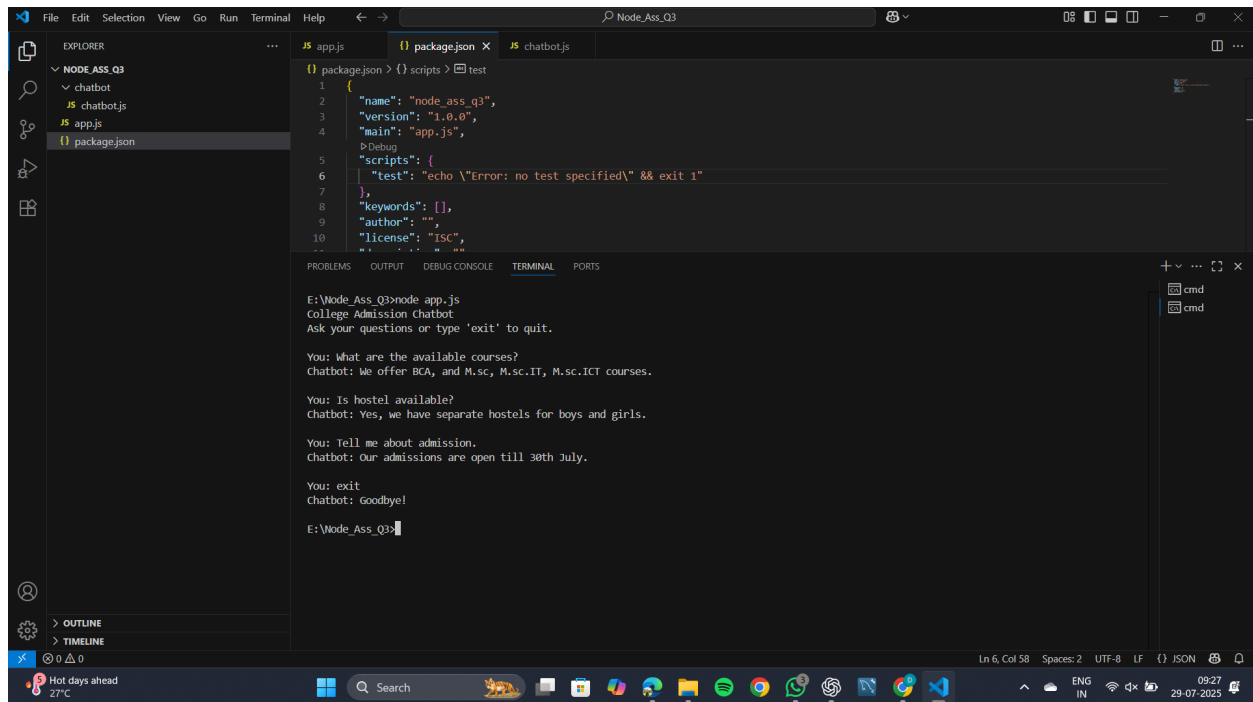
**Q.2. Develop a web server which serves static resources.**

**Output :**



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

Output :



```
{} package.json > {} scripts > test
1
2   "name": "node_ass_q3",
3   "version": "1.0.0",
4   "main": "app.js",
5   "scripts": {
6     "test": "echo \\\"Error: no test specified\\\" && exit 1"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  ...
12  ...

E:\Node_Ass_Q3>node app.js
College Admission Chatbot
Ask your questions or type 'exit' to quit.

You: What are the available courses?
Chatbot: We offer BCA, and M.sc, M.sc.IT, M.sc.ICT courses.

You: Is hostel available?
Chatbot: Yes, we have separate hostels for boys and girls.

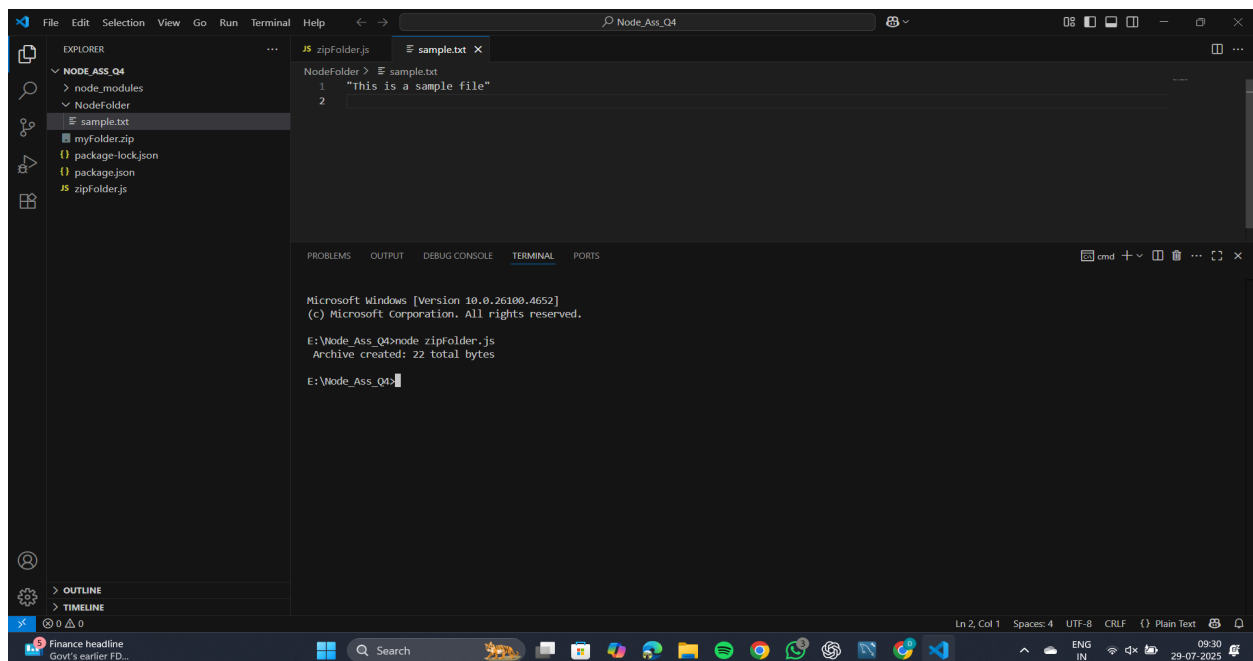
You: Tell me about admission.
Chatbot: Our admissions are open till 30th July.

You: exit
Chatbot: Goodbye!

E:\Node_Ass_Q3>
```

### Q.4. Write a program to create a compressed zip file for a folder.

Output :



```
{} zipFolder.js > {} sample.txt
NodeFolder > {} sample.txt
1   "this is a sample file"
2

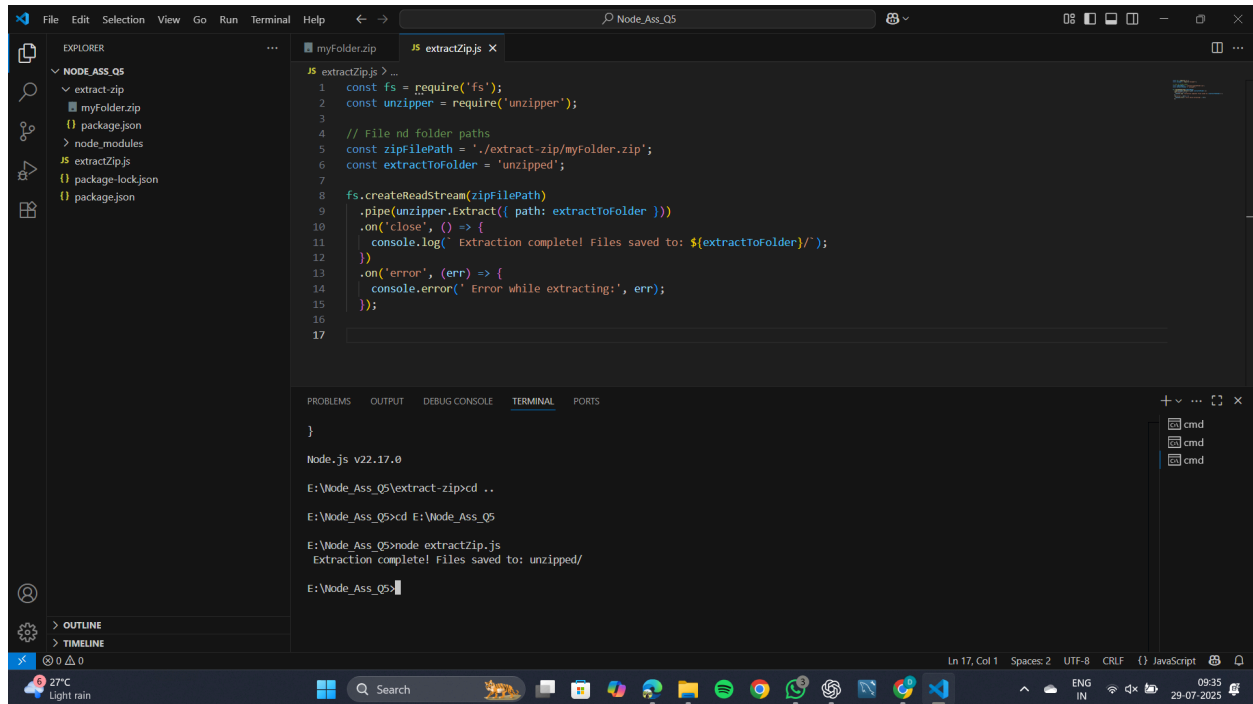
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

E:\Node_Ass_Q4>node zipFolder.js
Archive created: 22 total bytes

E:\Node_Ass_Q4>
```

Q.5. Write a program to extract a zip file.

Output :



The screenshot shows a VS Code editor with a file explorer on the left showing a project named 'NODE\_ASS\_Q5' containing 'extract-zip' and 'myFolder.zip'. The main editor displays a JavaScript file named 'extractZip.js' with the following code:

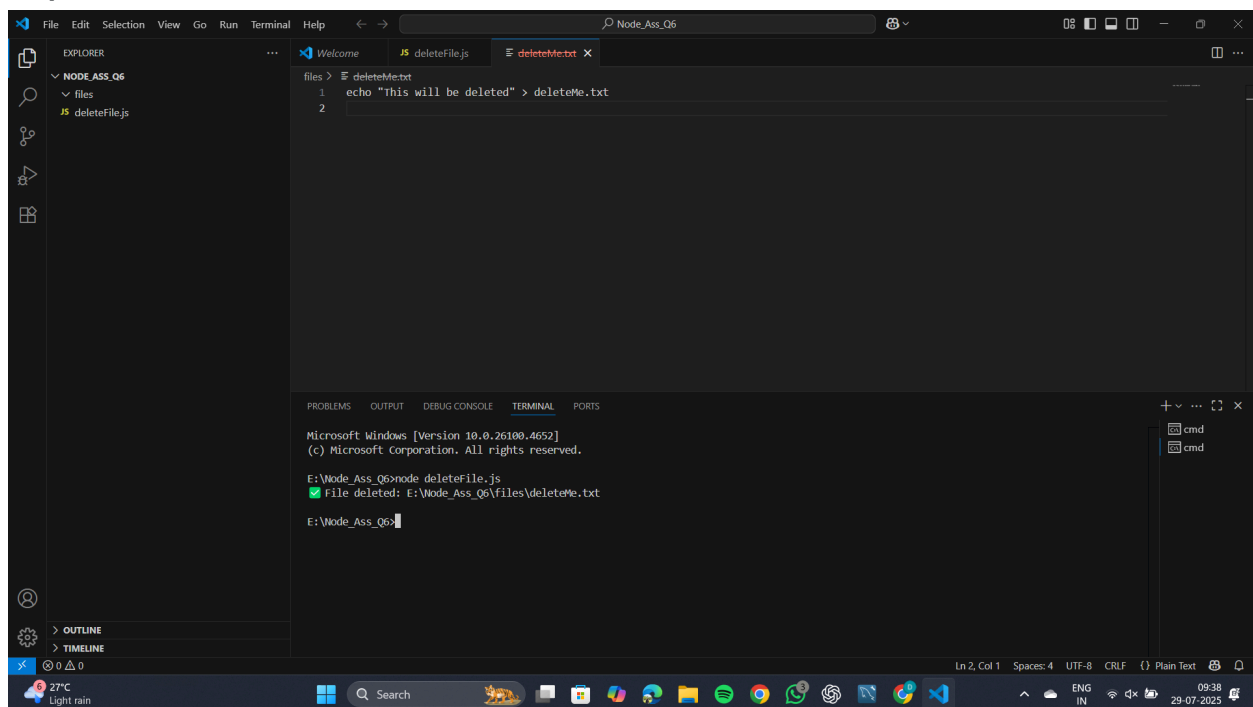
```
1 const fs = require('fs');
2 const unzipper = require('unzipper');
3
4 // File nd folder paths
5 const zipFilePath = './extract-zip/myFolder.zip';
6 const extractToFolder = 'unzipped';
7
8 fs.createReadStream(zipFilePath)
9   .pipe(unzipper.Extract({ path: extractToFolder }))
10  .on('close', () => {
11    console.log("Extraction complete! Files saved to: ${extractToFolder}");
12  })
13  .on('error', (err) => {
14    console.error("Error while extracting:", err);
15  });
16
17
```

The terminal at the bottom shows the execution of the program:

```
Node.js v22.17.0
E:\Node_Ass_Q5\extract-zip>cd ..
E:\Node_Ass_Q5>cd E:\Node_Ass_Q5
E:\Node_Ass_Q5>node extractZip.js
Extraction complete! Files saved to: unzipped/
E:\Node_Ass_Q5>
```

Q.6. Write a program to promisify fs.unlink function and call it.

Output :



The screenshot shows a VS Code editor with a file explorer on the left showing a project named 'NODE\_ASS\_Q6' containing 'files' and 'deleteFile.js'. The main editor displays a JavaScript file named 'deleteFile.js' with the following code:

```
1 echo "This will be deleted" > deleteMe.txt
2
```

The terminal at the bottom shows the execution of the program:

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

E:\Node_Ass_Q6>node deleteFile.js
File deleted: E:\Node_Ass_Q6\files\deleteMe.txt
E:\Node_Ass_Q6>
```

## Q.7. Fetch data of google page using node-fetch using async-await model.

The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project named 'NODE\_ASS\_Q7' with a subdirectory 'que\_7' containing 'fetchGoogle.js', 'package-lock.json', and 'package.json'. The 'fetchGoogle.js' file is open in the editor. The code in the file is as follows:

```
1 const fetch = require('node-fetch');
2
3 async function fetchGoogleHomePage() {
4   try {
5     const response = await fetch('https://www.google.com');
6     const data = await response.text();
7     console.log("Fetched data from Google's homepage:\n");
8     console.log(data.substring(0, 1000));
9   } catch (error) {
10    console.error("Error fetching google page:", error.message);
11  }
12 }
13
```

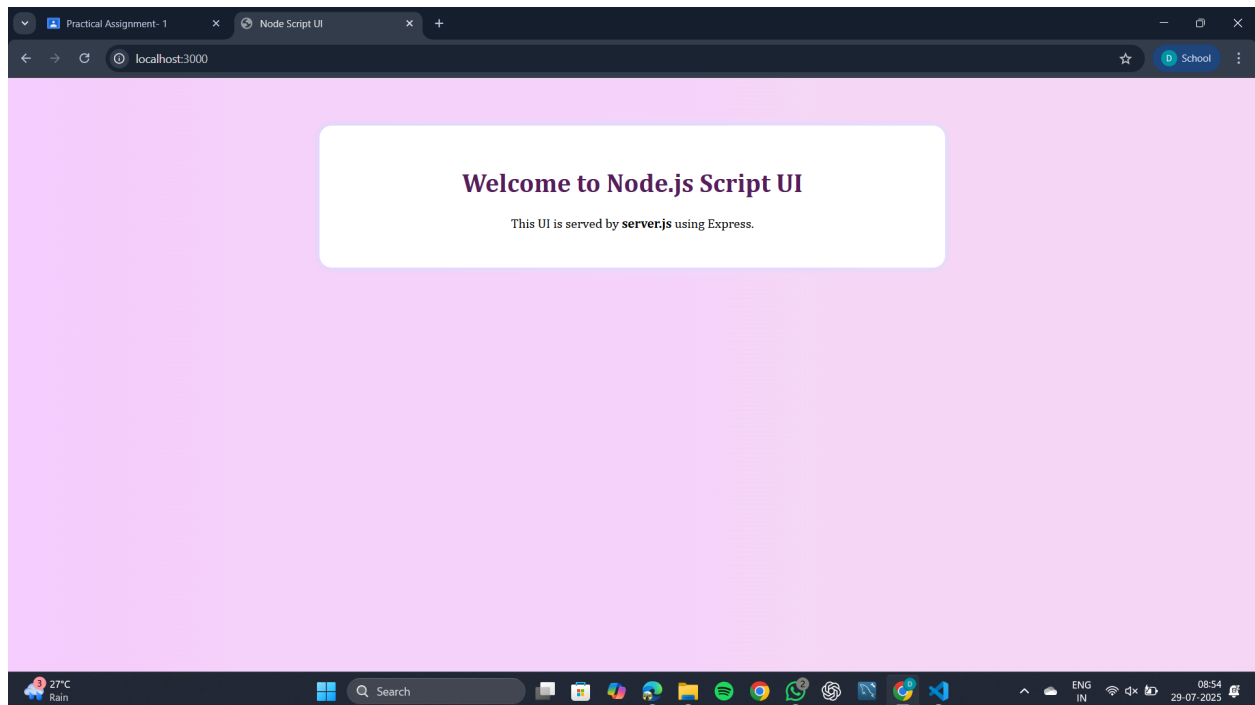
The terminal output shows the command 'node fetchGoogle.js' being executed, which results in the following output:

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

E:\Node_Ass_Q7>cd que_7

E:\Node_Ass_Q7\que_7>node fetchGoogle.js
Fetched data from Google's homepage:
<doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IN"><head><meta content="text/html; charset=UTF-8" http-equiv="
content-Type"><meta content="/images/branding/google/1x/google_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce=
"G_9knl1ihX0z5KvdiPmKw">(function(){var _g={keyI:"5FC1a1f0fpmP5QUPte77AAQ",keyPI:"0,202854,2,3497397,1131,538661,14111,64702,94324,266577,21981
0,6601,2,63631,11106,5219187,11389,36812642,25366949,14110,8944,56228,6753,23878,9140,4599,328,6225,1116,36710,26339,6748,8300,8206,3292,4134,3
0379,28334,54212,352,2265,1834,14781,3118,3,2749,3856,3858,5773,27612,3050,2,1667,11805,2801,460,2990,35,3420,2693,134,37,661,2013,7946,12107,5
683,3604,13805,3967,2731,6430,9471,646,617,3605,4,4,4811,935,3,1470,3436,438,1737,4835,1,3467,2,213,691,2,728,2083,463,7,1378,600,1033,1578,437
7,3,138,341,982,3,8065,692,2646,103,4,1,321,5099,670,31,3,109,2117,1842,4,421,1561,4142,1447,3,2,2,2,1331,359,1,4556,4,111,1481,2,2,298,1,4,12
```

## Q.8. Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs Application.



The screenshot shows a VS Code editor window with the Explorer sidebar on the left displaying the file structure of 'NODE\_ASS\_Q8'. The main editor area shows the 'index.html' file. The Terminal panel at the bottom displays the execution of a Node.js application. The application starts by running 'server.js', which serves the 'index.html' file on 'http://localhost:3000'. It then runs a test script 'test.js' which reports 'Test script is running successfully!'. Following this, three custom tasks are executed: 'task1' (generating a report), 'task2' (cleaning temp files), and 'task3' (backing up the database). The status bar at the bottom indicates the current line is 16, column 7, in a UTF-8 file with CRLF line endings.

```
que_8 > public > index.html > html > head > style
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>

> node server.js

Server running on http://localhost:3000
Terminate batch job (Y/N)? y

E:\Node_Ass_Q8\que_8>npm run test

> que_8@1.0.0 test
> node test.js

Test script is running successfully!

E:\Node_Ass_Q8\que_8>npm run task1

> que_8@1.0.0 task1
> node custom-task1.js

Custom Task 1 executed: Generating report...

E:\Node_Ass_Q8\que_8>npm run task2

> que_8@1.0.0 task2
> node custom-task2.js

Custom Task 2 executed: Cleaning temp files...

E:\Node_Ass_Q8\que_8>npm run task3

> que_8@1.0.0 task3
> node custom-task3.js

Custom Task 3 executed: Backing up database...

E:\Node_Ass_Q8\que_8>
```

## Q.9 A program which calls useful functions in fs module.

The screenshot shows a VS Code editor window with the Explorer sidebar on the left displaying the file structure of 'NODE\_ASS\_Q9'. The main editor area shows the 'fsOperation.js' file. The program uses the 'fs' module to create a folder named 'myFolder', write a file 'example.txt' with the content 'Hello, this is the first line.', and then log the folder and file details. The Terminal panel at the bottom displays the output of the program, showing the folder creation, file writing, and the file content. The status bar at the bottom indicates the current line is 3, column 1, in a JavaScript file with CRLF line endings.

```
1 const fs = require('fs');
2 const path = require('path');
3
4 const folderName = 'myFolder';
5 if (!fs.existsSync(folderName)) {
6   fs.mkdirSync(folderName);
7   console.log('Folder created:', folderName);
8 }
9
10 const filePath = path.join(folderName, 'example.txt');
11 fs.writeFileSync(filePath, 'Hello, this is the first line.\n');
12 console.log('File created and content written.');
```

```

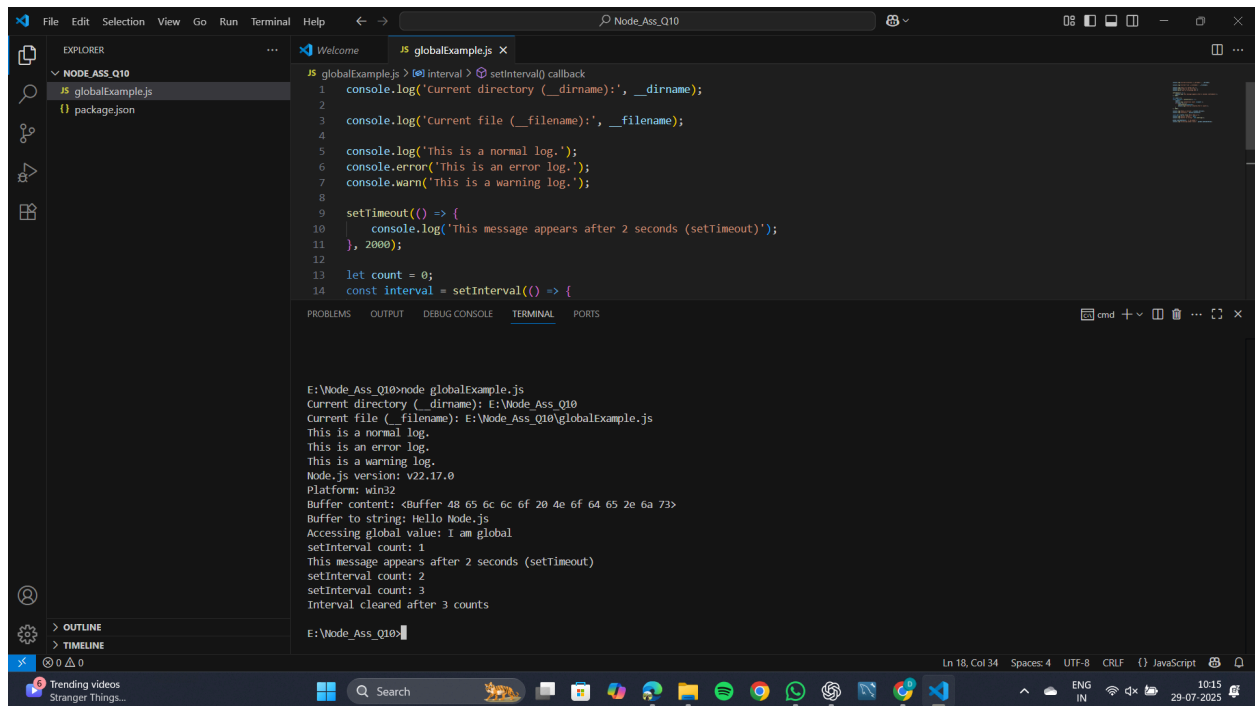
"license": "ISC",
"description": ""
}

E:\Node_Ass_Q9>node fsOperation.js
Folder created: myFolder
File created and content written.
Data appended to the file.
File content:
Hello, this is the first line.
This is an appended line.

File renamed to renamed.txt
Files in folder: [ 'renamed.txt' ]
File deleted.
Folder deleted.

E:\Node_Ass_Q9>
```

Q.10. A program which uses global objects in nodejs.



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

```
1 console.log('Current directory (__dirname):', __dirname);
2
3 console.log('Current file (__filename):', __filename);
4
5 console.log('This is a normal log.');
```

```
6 console.error('This is an error log.');
```

```
7 console.warn('This is a warning log.');
```

```
8
9 setTimeout(() => {
10   console.log('This message appears after 2 seconds (setTimeout)');
```

```
11 }, 2000);
```

```
12
13 let count = 0;
```

```
14 const interval = setInterval(() => {
```

The terminal output shows the execution of the script:

```
E:\Node_Ass_Q10>node globalExample.js
Current directory (__dirname): E:\Node_Ass_Q10
Current file (__filename): E:\Node_Ass_Q10\globalExample.js
This is a normal log.
This is an error log.
This is a warning log.
Node.js version: v22.17.0
Platform: win32
Buffer content: <Buffer 48 65 6c 6c 6f 20 4e 6f 64 65 20 6a 73>
Buffer to string: Hello Node.js
Accessing global value: I am global
setInterval count: 1
This message appears after 2 seconds (setTimeout)
setInterval count: 2
setInterval count: 3
Interval cleared after 3 counts
E:\Node_Ass_Q10>
```

Q.11. Develop a useful package and publish it on [npmjs.com](https://www.npmjs.com)

[node\\_ass\\_q11 - npm](https://www.npmjs.com/package/node_ass_q11)

