

Name : Harikrushn D. Mistry

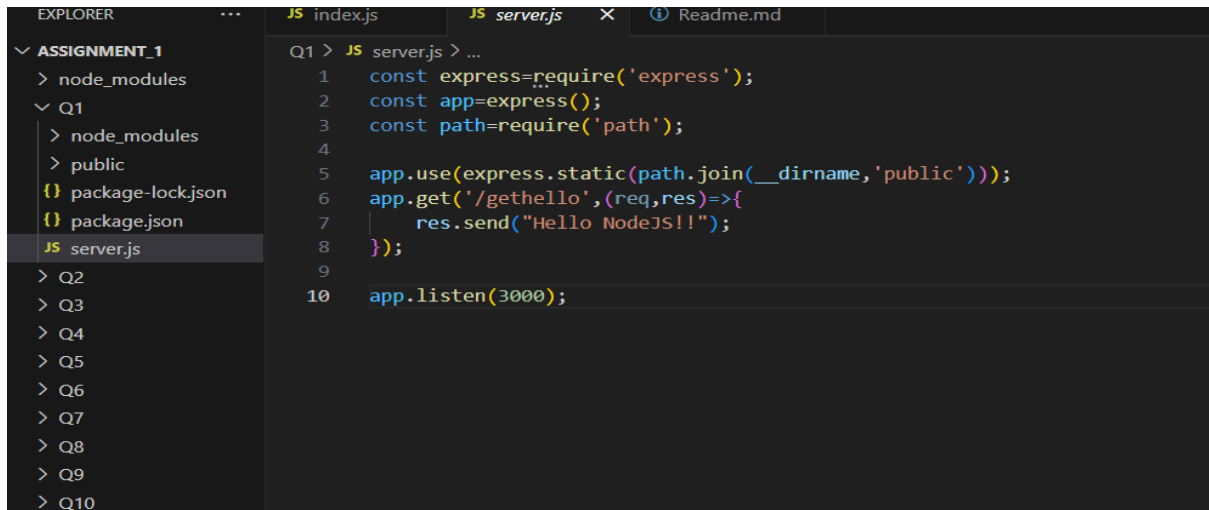
Roll No. : 61

Division : A

Subject : Nodejs

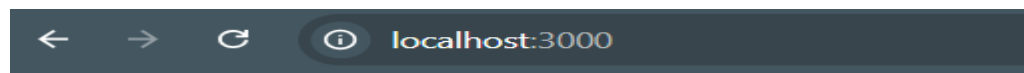
Assignment : Pratical – 1

Q1.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree under 'ASSIGNMENT_1' with folders 'node_modules' and 'Q1'. Inside 'Q1', there are 'node_modules', 'public', 'package-lock.json', 'package.json', and 'server.js'. The 'server.js' file is selected. The main editor area shows the code for 'server.js' with line numbers 1 through 10. The code uses Express.js to create a simple web server that serves static files from the 'public' directory and responds to GET requests at the '/gethello' endpoint with the message 'Hello NodeJS!!'. The server listens on port 3000.

```
Q1 > JS server.js > ...
1  const express=require('express');
2  const app=express();
3  const path=require('path');
4
5  app.use(express.static(path.join(__dirname,'public')));
6  app.get('/gethello',(req,res)=>{
7      res.send("Hello NodeJS!!");
8  });
9
10 app.listen(3000);
```



NodeJS AJAX Example

Get Hello



NodeJS AJAX Example

Get Hello

Hello NodeJS!!

Q2

The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure with folders Q1 through Q11. The file `server.js` is selected in the Q2 folder. The main editor area displays the code for `server.js`, which is a simple Express.js server. The bottom panel shows the Terminal window with the output of running the server using nodemon.

EXPLORER

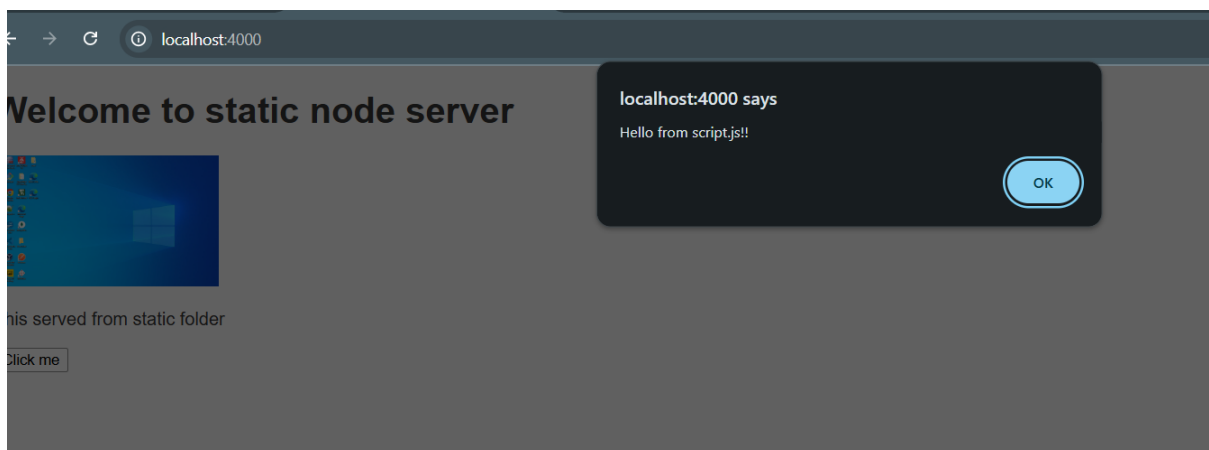
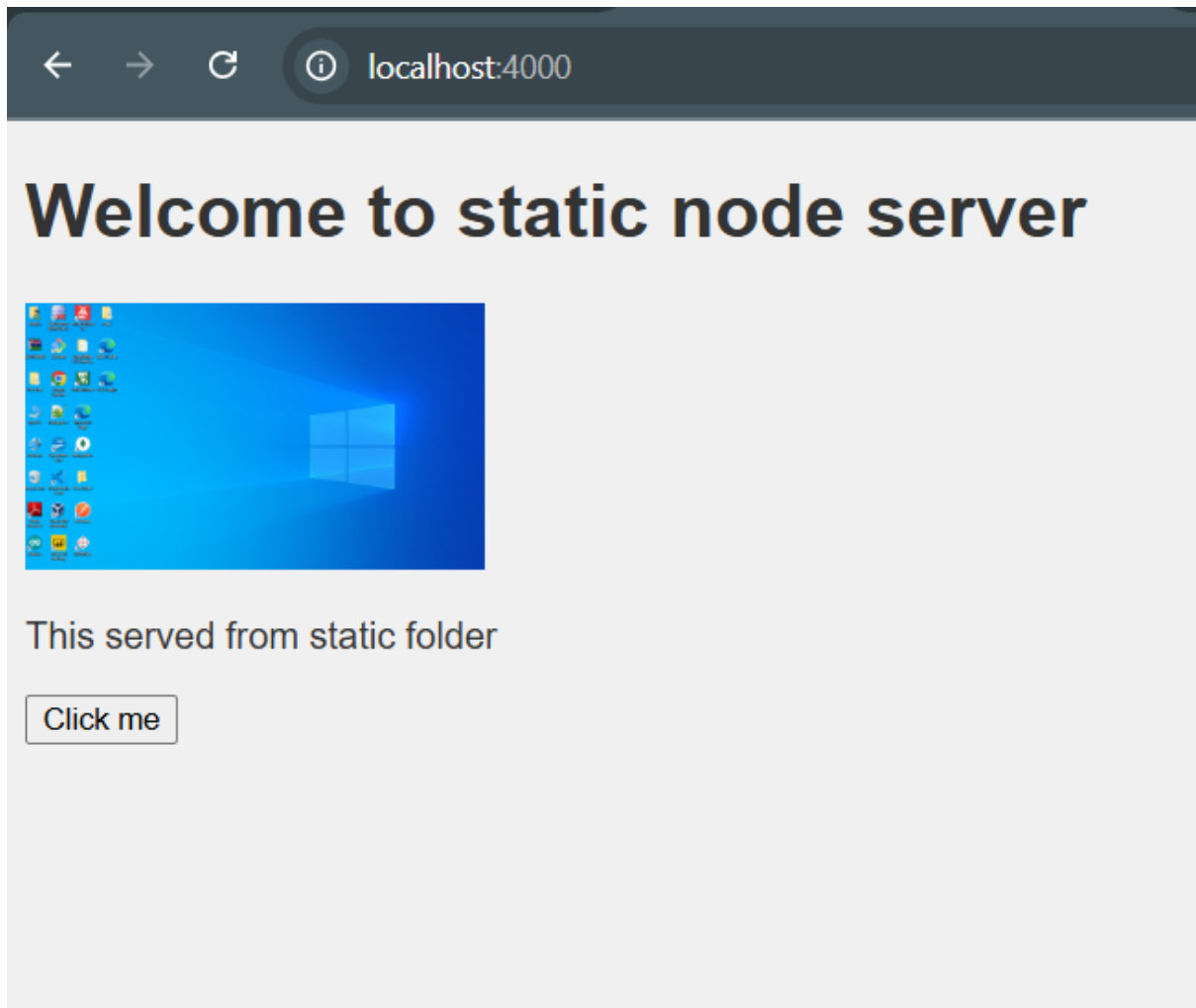
- ASSIG...
- > node_modules
- > Q1
- > Q2
 - > node_modules
 - > public
 - { } package-lock.json
 - { } package.json
 - JS server.js**
 - > Q3
 - > Q4
 - > Q5
 - > Q6
 - > Q7
 - > Q8
 - > Q9
 - > Q10
 - > Q11

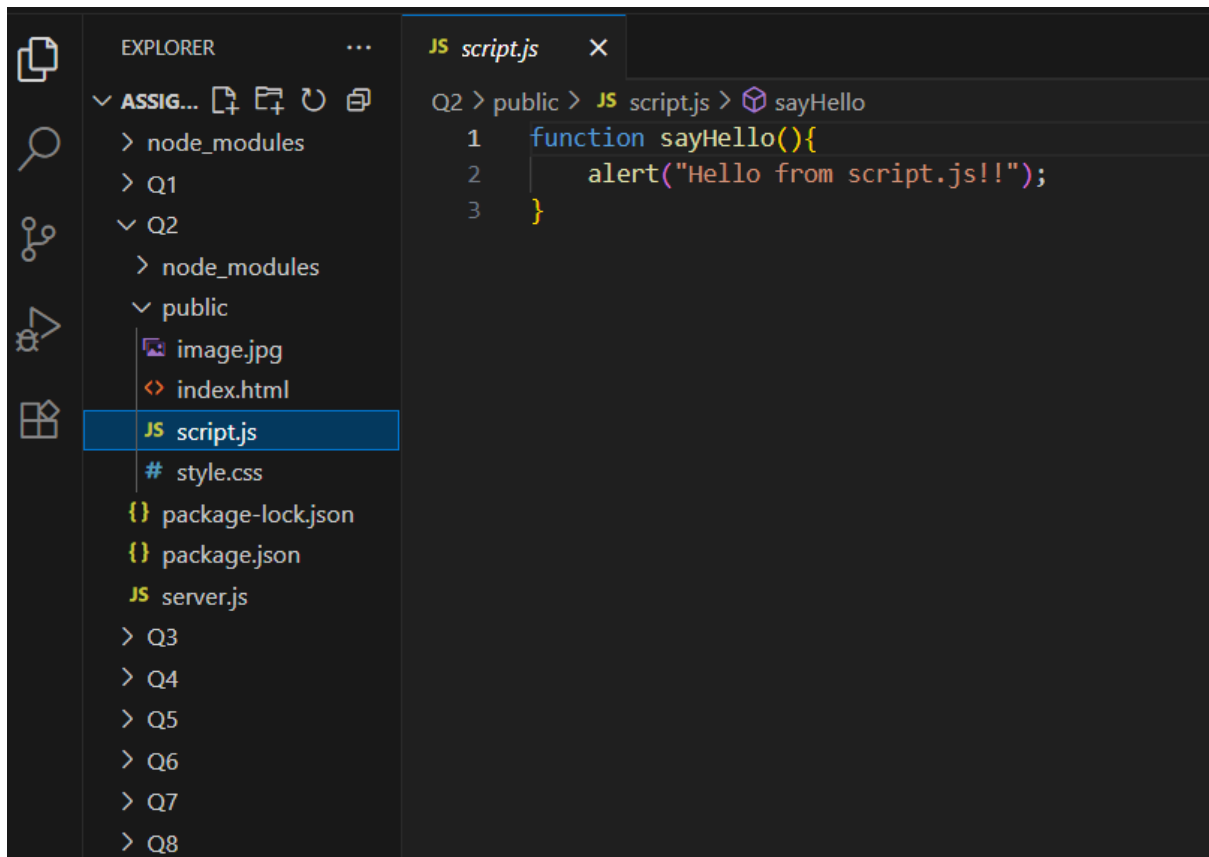
JS server.js

```
Q2 > JS server.js > ...
1  const express=require('express');
2  const app=express();
3  const path=require('path');
4  app.use(express.static(path.join(__dirname,'public')));
5
6  app.listen(4000);
```

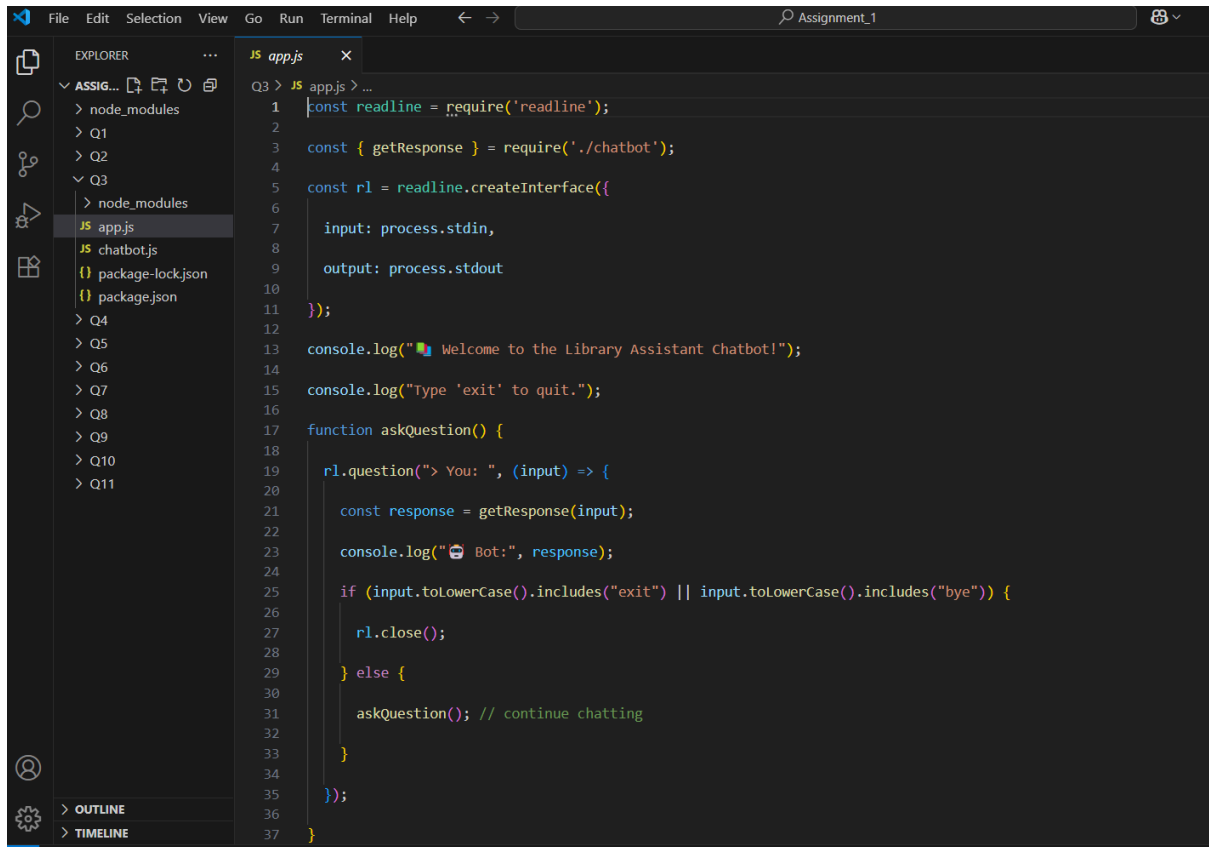
TERMINAL

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
• PS D:\Hari\sem7\Assignment_1\Q1> cd ..
• PS D:\Hari\sem7\Assignment_1> cd Q2
○ PS D:\Hari\sem7\Assignment_1\Q2> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
```





Q3

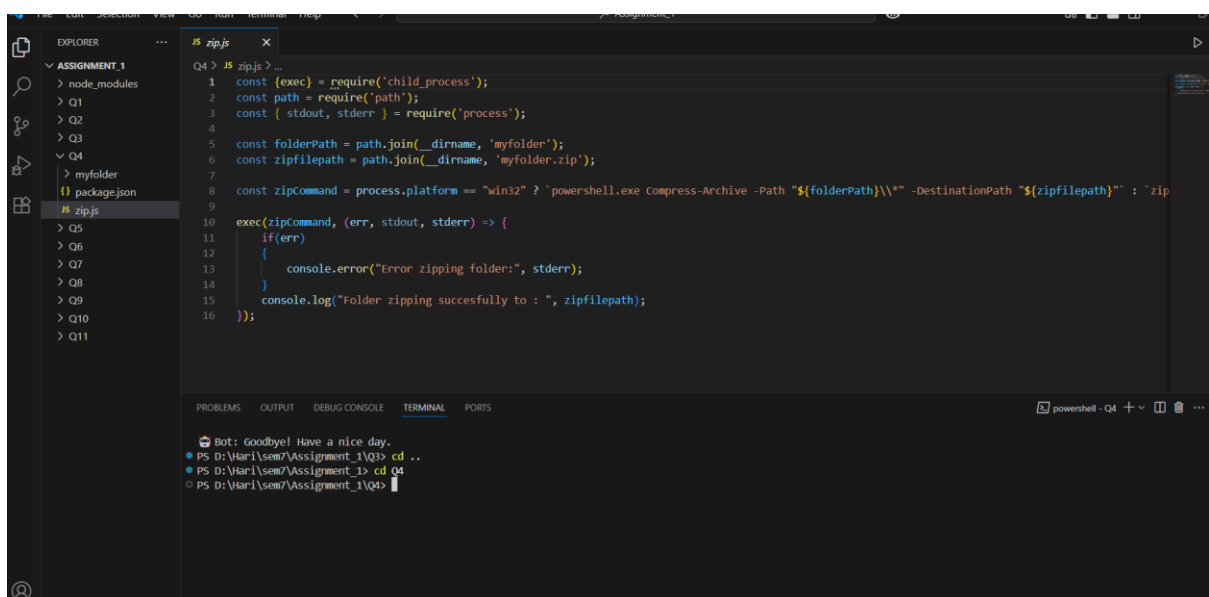


The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left shows a project structure with a folder named 'ASSIG...' containing subfolders 'node_modules', 'Q1', 'Q2', 'Q3', and 'Q4' through 'Q11'. The 'Q3' folder is expanded, showing files 'app.js' and 'chatbot.js'. The 'chatbot.js' file is selected and its content is displayed in the main editor window. The code is a JavaScript function 'getResponse' that takes 'userInput' as an argument and returns a string based on the input. The function uses 'toLowerCase()' and 'includes()' methods. The input is converted to lowercase and then checked for specific keywords: 'book', 'available', 'timing', 'membership', 'bye', and 'exit'. If none of these are found, a default response is returned. The function is then exported as 'module.exports'.

```
1
2 function getResponse(userInput) {
3
4     const input = userInput.toLowerCase();
5
6     if (input.includes("book")) {
7
8         return "We have books on many topics. Please specify the subject.";
9     }
10    } else if (input.includes("available")) {
11
12        return "Yes, the book is available in the general section.";
13    }
14    } else if (input.includes("timing")) {
15
16        return "Library is open from 9 AM to 6 PM on weekdays.";
17    }
18    } else if (input.includes("membership")) {
19
20        return "To get a library membership, please fill the form at the front desk.";
21    }
22    } else if (input.includes("bye") || input.includes("exit")) {
23
24        return "Goodbye! Have a nice day.";
25    }
26    } else {
27
28        return "Sorry, I didn't understand that. Can you rephrase?";
29    }
30 }
31
32
33
34 module.exports = { getResponse };
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● PS D:\Hari\sem7\Assignment_1> cd Q3
○ PS D:\Hari\sem7\Assignment_1\Q3> nodemon app.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
👋 Welcome to the Library Assistant Chatbot!
Type 'exit' to quit.
> You: book
book
🤖 Bot: We have books on many topics. Please specify the subject.
> You: available
available
🤖 Bot: Yes, the book is available in the general section.
> You: timing
timing
timing
🤖 Bot: Library is open from 9 AM to 6 PM on weekdays.
> You: membership
membership
🤖 Bot: To get a library membership, please fill the form at the front desk.
membership
🤖 Bot: To get a library membership, please fill the form at the front desk.
🤖 Bot: To get a library membership, please fill the form at the front desk.
> You: bye
bye
🤖 Bot: Goodbye! Have a nice day.
█
```

Q4



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with 'ASSIGNMENT_1' containing 'node_modules', 'Q1', 'Q2', 'Q3', 'Q4', 'myfolder', and 'package.json'. The 'Q4' folder is expanded, showing 'zip.js'. The terminal shows the execution of 'zip.js' and its output.

```
Q4 > JS zip.js > ...
1  const {exec} = require('child_process');
2  const path = require('path');
3  const { stdout, stderr } = require('process');
4
5  const folderPath = path.join(__dirname, 'myfolder');
6  const zipfilePath = path.join(__dirname, 'myfolder.zip');
7
8  const zipCommand = process.platform === "win32" ? `powershell.exe Compress-Archive -Path "${folderPath}\\*" -DestinationPath "${zipfilePath}"` : `zip
9
10 exec(zipCommand, (err, stdout, stderr) => {
11   if(err)
12   {
13     console.error("Error zipping folder:", stderr);
14   }
15   console.log("Folder zipping successfully to : ", zipfilePath);
16 });
```

The terminal output shows the bot's responses from the previous session and the current command prompt:

```
🤖 Bot: Goodbye! Have a nice day.
● PS D:\Hari\sem7\Assignment_1\Q3> cd ..
● PS D:\Hari\sem7\Assignment_1> cd Q4
○ PS D:\Hari\sem7\Assignment_1\Q4> █
```

```

🤖 Bot: Goodbye! Have a nice day.
PS D:\Hari\sem7\Assignment_1\Q3> cd ..
PS D:\Hari\sem7\Assignment_1> cd Q4
PS D:\Hari\sem7\Assignment_1\Q4> nodemon zip.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node zip.js`
Folder zipping succesfully to : D:\Hari\sem7\Assignment_1\Q4\myfolder.zip
[nodemon] clean exit - waiting for changes before restart

```

Q5

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders Q1 through Q11. Q5 is expanded, showing 'extracted_folder' and 'myfolder.zip'. The file 'JS unzip.js' is selected. The main editor displays the code for 'unzip.js', which uses the 'child_process' module to run a PowerShell command for extracting a zip file. The terminal at the bottom shows the command history and the output of running 'nodemon unzip.js', which successfully extracts the zip file to the 'extracted_folder'.

```

EXPLORER
ASSIGNMENT_1
  node_modules
  Q1
  Q2
  Q3
  Q4
  Q5
    extracted_folder
    myfolder.zip
    JS unzip.js
  Q6
  Q7
  Q8
  Q9
  Q10
  Q11

JS unzip.js
1
2 const { exec } = require('child_process');
3 const path = require('path');
4
5 // Path to your zip file and destination folder
6 const zipFilePath = path.join(__dirname, 'myfolder.zip');
7 const extractTo = path.join(__dirname, 'extracted_folder');
8
9 const unzipCommand = process.platform === 'win32'
10   ? `powershell.exe Expand-Archive -Path "${zipFilePath}" -DestinationPath "${extractTo}"`
11   : `unzip -o "${zipFilePath}" -d "${extractTo}"`;
12
13 exec(unzipCommand, (err, stdout, stderr) => {
14   if (err) {
15     console.error('Extraction error:', stderr);
16     return;
17   }
18   console.log('Zip extracted successfully to:', extractTo);
19 });
20

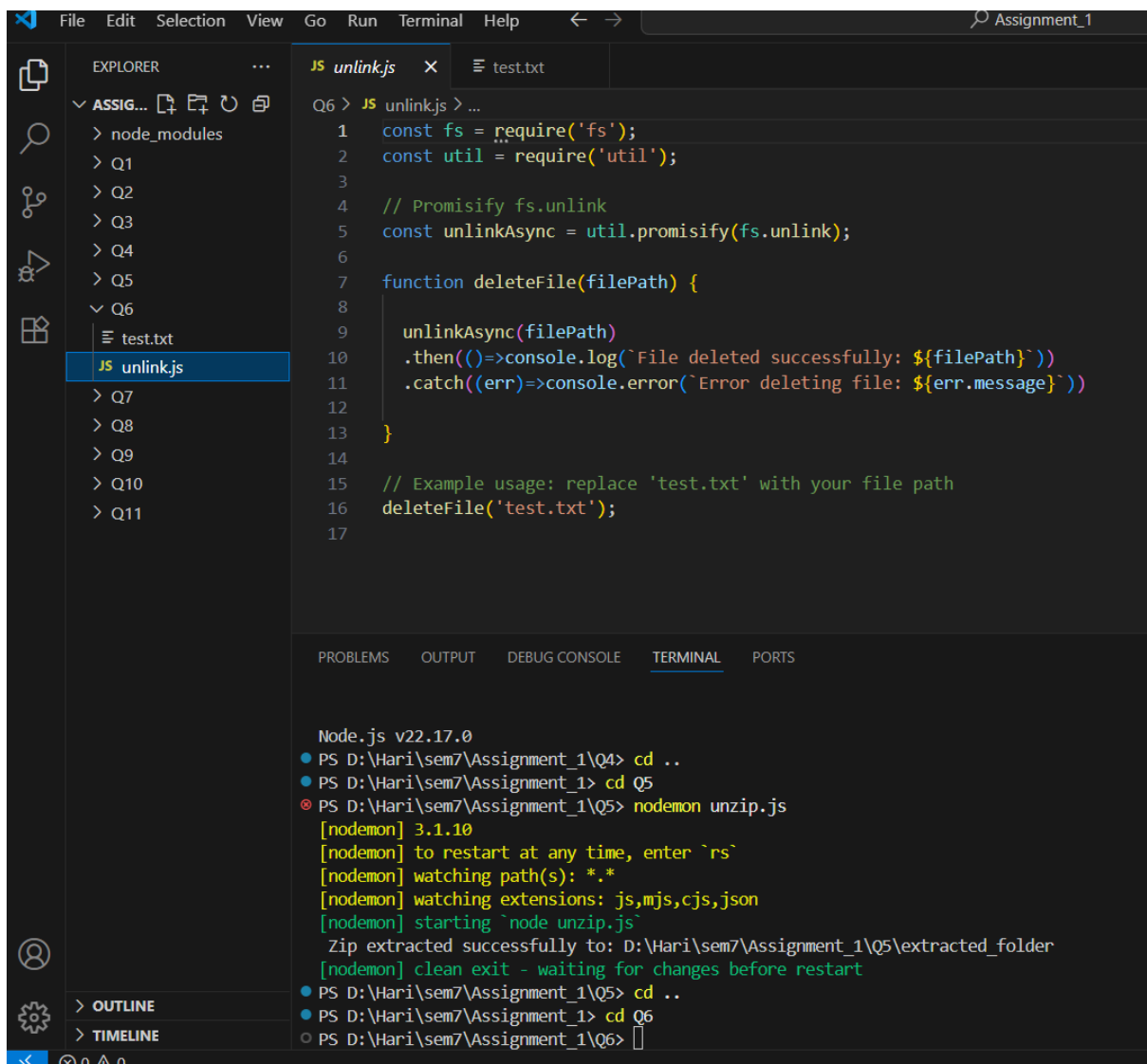
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

requireStack: []
}

Node.js v22.17.0
PS D:\Hari\sem7\Assignment_1\Q4> cd ..
PS D:\Hari\sem7\Assignment_1> cd Q5
PS D:\Hari\sem7\Assignment_1\Q5> nodemon unzip.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node unzip.js`
Zip extracted successfully to: D:\Hari\sem7\Assignment_1\Q5\extracted_folder
[nodemon] clean exit - waiting for changes before restart

```


Q6



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders Q1 through Q11 and a file test.txt. The file unlink.js is selected. The main editor displays the code for unlink.js, which uses the fs module and util.promisify to create an async function deleteFile. The terminal at the bottom shows a series of commands: navigating to the Q4 directory, running 'cd ..', then 'cd Q5', and finally 'nodemon unzip.js'. The terminal output shows that nodemon successfully started and ran 'unzip.js', extracting a zip file to the 'extracted_folder' in the Q5 directory. The status bar at the bottom indicates the current file is 0 lines long.

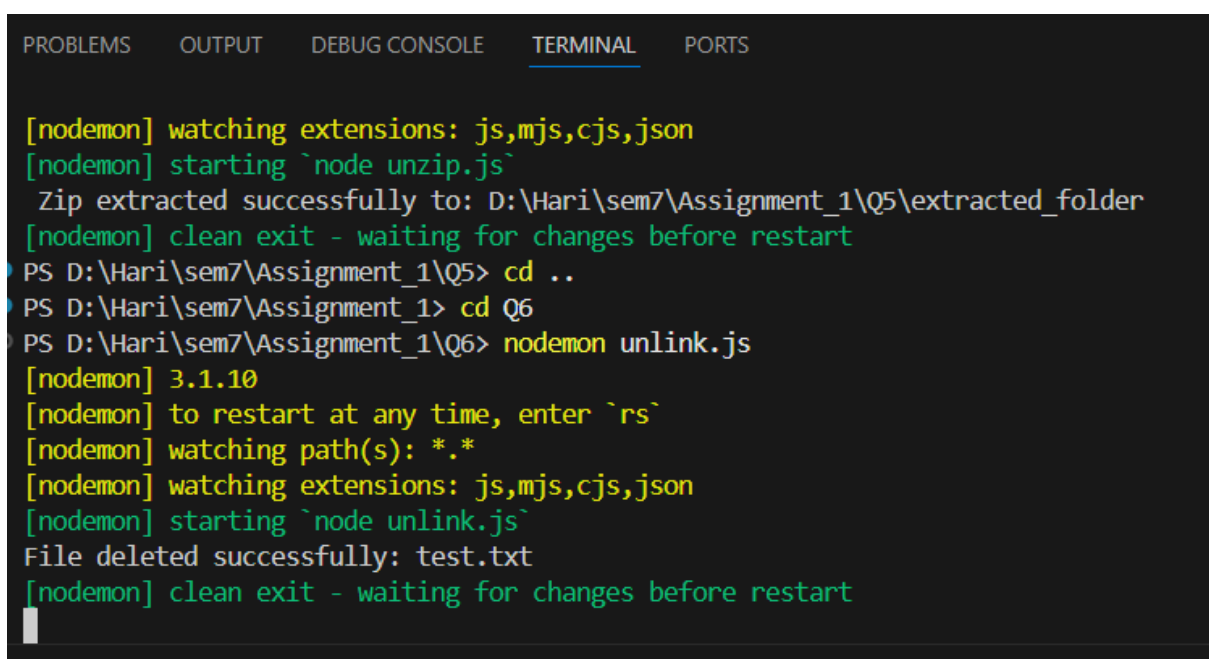
```
1 const fs = require('fs');
2 const util = require('util');
3
4 // Promisify fs.unlink
5 const unlinkAsync = util.promisify(fs.unlink);
6
7 function deleteFile(filePath) {
8
9     unlinkAsync(filePath)
10     .then(()=>console.log(`File deleted successfully: ${filePath}`))
11     .catch((err)=>console.error(`Error deleting file: ${err.message}`))
12 }
13
14
15 // Example usage: replace 'test.txt' with your file path
16 deleteFile('test.txt');
17
```

Node.js v22.17.0

- PS D:\Hari\sem7\Assignment_1\Q4> cd ..
- PS D:\Hari\sem7\Assignment_1> cd Q5
- PS D:\Hari\sem7\Assignment_1\Q5> nodemon unzip.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node unzip.js`
Zip extracted successfully to: D:\Hari\sem7\Assignment_1\Q5\extracted_folder
[nodemon] clean exit - waiting for changes before restart

- PS D:\Hari\sem7\Assignment_1\Q5> cd ..
- PS D:\Hari\sem7\Assignment_1> cd Q6
- PS D:\Hari\sem7\Assignment_1\Q6>



This is a close-up of the terminal window from the previous screenshot. It shows the execution of 'nodemon unlink.js' in the Q6 directory. The output shows that nodemon successfully started and ran 'unlink.js', which deleted the file 'test.txt'. The terminal output is as follows:

```
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node unzip.js`
Zip extracted successfully to: D:\Hari\sem7\Assignment_1\Q5\extracted_folder
[nodemon] clean exit - waiting for changes before restart
PS D:\Hari\sem7\Assignment_1\Q5> cd ..
PS D:\Hari\sem7\Assignment_1> cd Q6
PS D:\Hari\sem7\Assignment_1\Q6> nodemon unlink.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node unlink.js`
File deleted successfully: test.txt
[nodemon] clean exit - waiting for changes before restart
```

Q7

The screenshot shows the VS Code interface with two panels. The left panel is the Explorer view, showing a file tree with a folder named 'ASSIGNMENT_1' containing subfolders 'node_modules' and 'Q1' through 'Q11'. The right panel is the Editor view, showing the code for 'fetch.js'. The code defines a 'fetchGoogle' function that uses 'node-fetch' to make an HTTP request to 'https://www.google.com'. The function is asynchronous and uses 'try-catch' blocks to handle errors. The code is as follows:

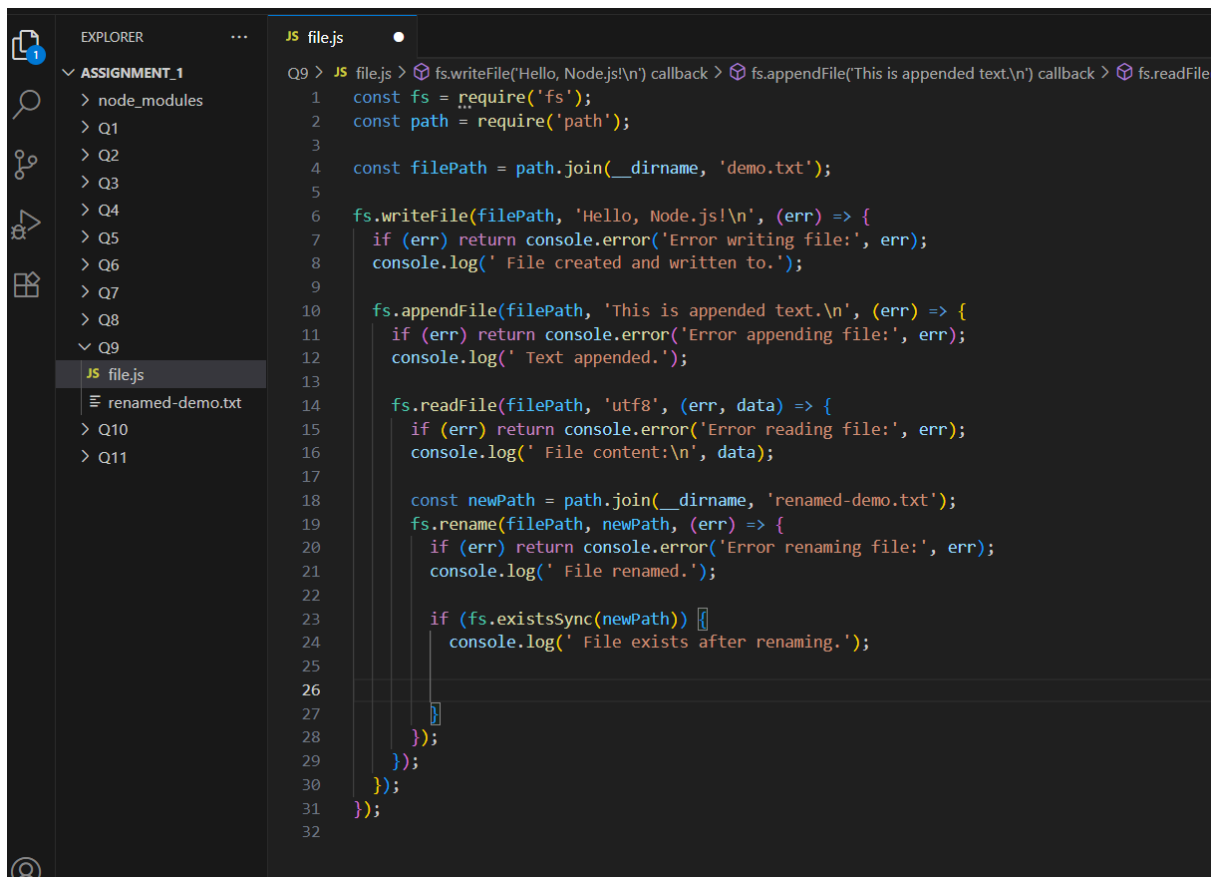
```
Q7 > JS fetch.js > fetchGoogle
1  const fetch = require('node-fetch');
2
3  async function fetchGoogle() {
4    try {
5      const response = await fetch('https://www.google.com');
6      if (!response.ok) {
7        throw new Error(`HTTP error! status: ${response.status}`);
8      }
9      const body = await response.text();
10     console.log('Fetched Google homepage HTML length:', body.length);
11
12     console.log(body);
13   } catch (error) {
14     console.error('Fetch error:', error);
15   }
16 }
17
18 fetchGoogle();
19
```

[illegible]

Q8

```
PS D:\Hari\sem7\Assignment_1\Q8> nodemon server.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node "server.js" server.js`
test
[nodemon] clean exit - waiting for changes before restart
```

Q9



```
Q9 > JS file.js > fs.writeFile('Hello, Node.js!\n') callback > fs.appendFile('This is appended text.\n') callback > fs.readFile
1  const fs = require('fs');
2  const path = require('path');
3
4  const filePath = path.join(__dirname, 'demo.txt');
5
6  fs.writeFile(filePath, 'Hello, Node.js!\n', (err) => {
7    if (err) return console.error('Error writing file:', err);
8    console.log(' File created and written to.');
```

```

[nodemon] to restart at any time, enter `rs`
PS D:\Hari\sem7\Assignment_1\Q9> nodemon file.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node file.js`
[nodemon] starting `node file.js`
File created and written to.
File created and written to.
Text appended.
File content:
Hello, Node.js!
File content:
Hello, Node.js!
This is appended text.
This is appended text.

File renamed.
File exists after renaming.
File exists after renaming.
[nodemon] clean exit - waiting for changes before restart

```

Q10

ASSIGNMENT_1

> env-checker

> node_modules

> Q1

> Q2

> Q3

> Q4

> Q5

> Q6

> Q7

> Q8

> Q9

> Q10

JS globalVariable.js

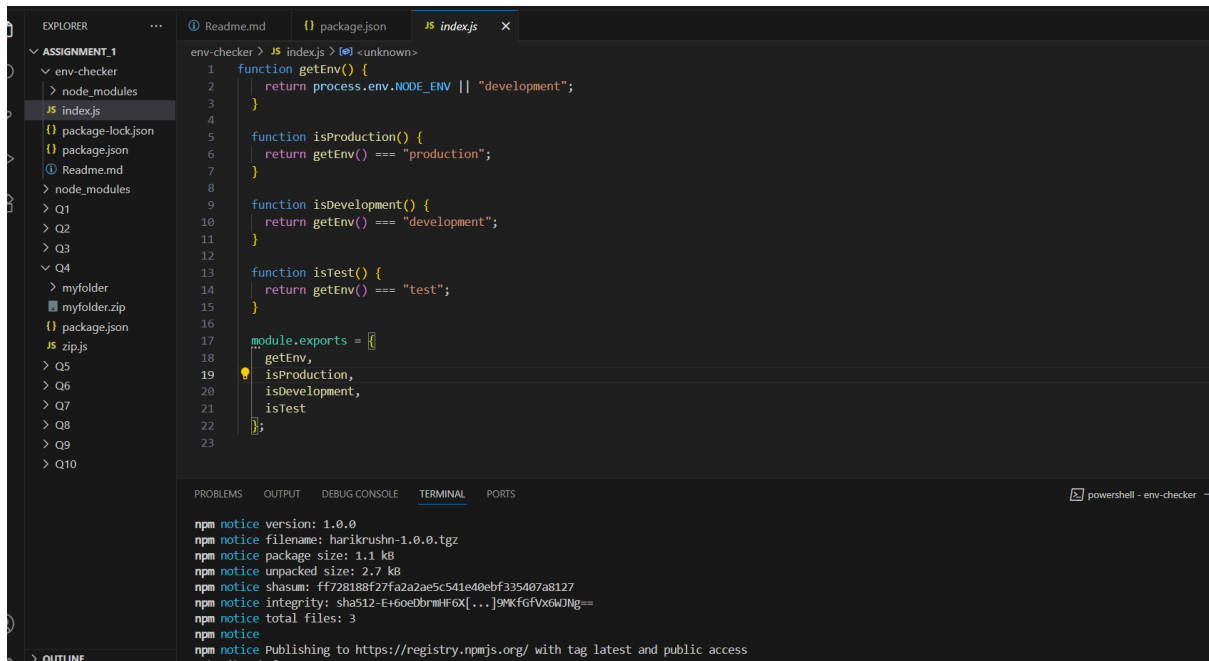
Q10 > JS globalVariable.js > ...

```

7   setTimeout(() => {
9     }, 1000);
10
11  let count = 0;
12  const intervalId = setInterval(() => {
13    count++;
14    console.log(`setInterval: count = ${count}`);
15    if (count === 3) {
16      clearInterval(intervalId);
17      console.log(`Interval cleared`);
18    }
19  }, 1000);
20
21  console.log(`Process PID:`, process.pid);
22  console.log(`Platform:`, process.platform);
23  console.log(`Node version:`, process.version);
24
25  console.log(`NODE_ENV:`, process.env.NODE_ENV || 'not set');
26

```

Q.11 env-checker



The screenshot shows a VS Code editor with the following structure:

- EXPLORER
 - ASSIGNMENT_1
 - env-checker
 - node_modules
 - index.js
 - package-lock.json
 - package.json
 - Readme.md
 - Q1
 - Q2
 - Q3
 - Q4
 - myfolder
 - myfolder.zip
 - package.json
 - zip.js
 - Q5
 - Q6
 - Q7
 - Q8
 - Q9
 - Q10

The main editor shows the `index.js` file with the following code:

```
1 function getEnv() {  
2   return process.env.NODE_ENV || "development";  
3 }  
4  
5 function isProduction() {  
6   return getEnv() === "production";  
7 }  
8  
9 function isDevelopment() {  
10  return getEnv() === "development";  
11 }  
12  
13 function isTest() {  
14  return getEnv() === "test";  
15 }  
16  
17 module.exports = {  
18   getEnv,  
19   isProduction,  
20   isDevelopment,  
21   isTest  
22 };  
23
```

The terminal at the bottom shows the following output:

```
npm notice version: 1.0.0  
npm notice filename: harikrushn-1.0.0.tgz  
npm notice package size: 1.1 kB  
npm notice unpacked size: 2.7 kB  
npm notice shasum: ff728188f27fa2a2ae5c541e40ebf335407a8127  
npm notice integrity: sha512-E+6oeDbrm#F6X[...]9MkfGVx6HJNG==  
npm notice total files: 3  
npm notice  
npm notice Publishing to https://registry.npmjs.org/ with tag latest and public access  
+ harikrushn@1.0.0
```



The screenshot shows the npm profile page for the user **harikrushn2703**. The page includes the following elements:

- Navigation links: Pro, Teams, Pricing, Documentation.
- Search bar: Search packages.
- Profile information:
 - Avatar: A blue and black pixelated character.
 - Username: harikrushn2703.
 - Edit Profile button.
- Package information:
 - 1 Package (indicated by a purple bar).
 - 0 Organizations (indicated by a green bar).
 - Package list: harikrushn
 - Description: Simple Node.js utility to detect the current environment.
 - Published: harikrushn2703 published 1.0.0 • a few seconds ago.
 - + Add Private Packages button.



Support

Help

Company

About

Terms & Policies

Policies