

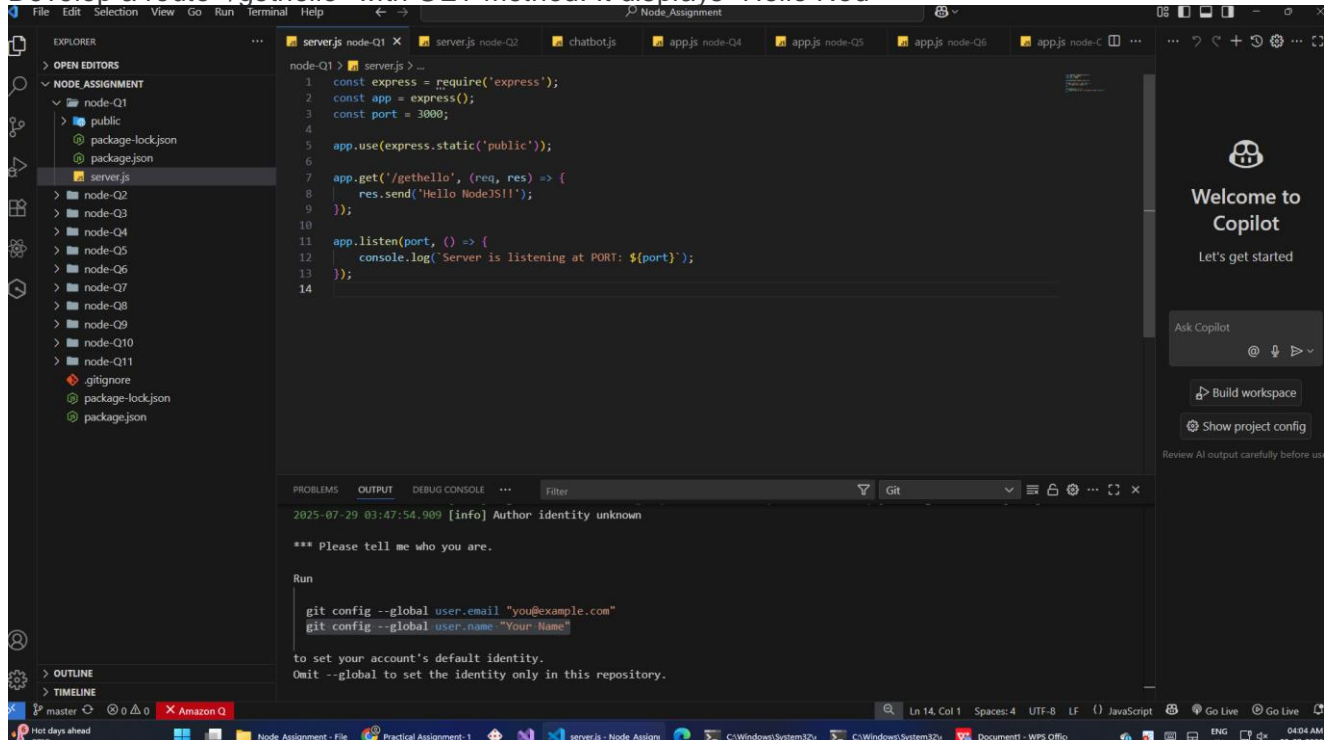
Name : Patel Khush Alpesh

Roll No : 45

Subject : Node js

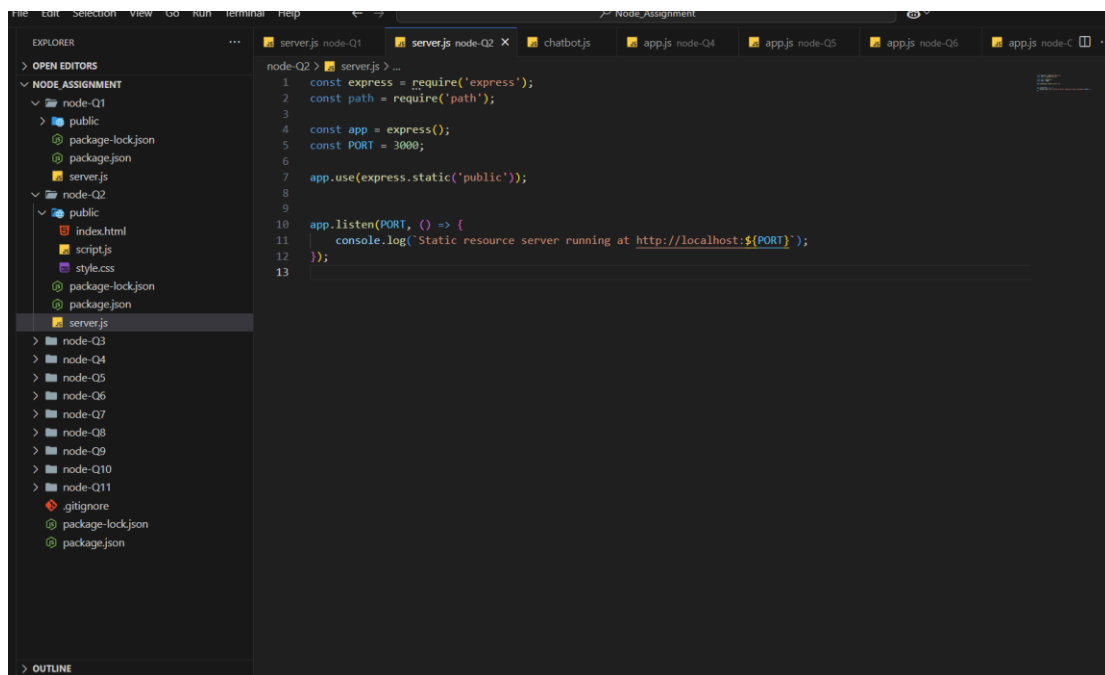
Practical Assignment - 1

- Develop a route `"/gethello"` with GET method. It displays "Hello Nod



eJS!!" as response.

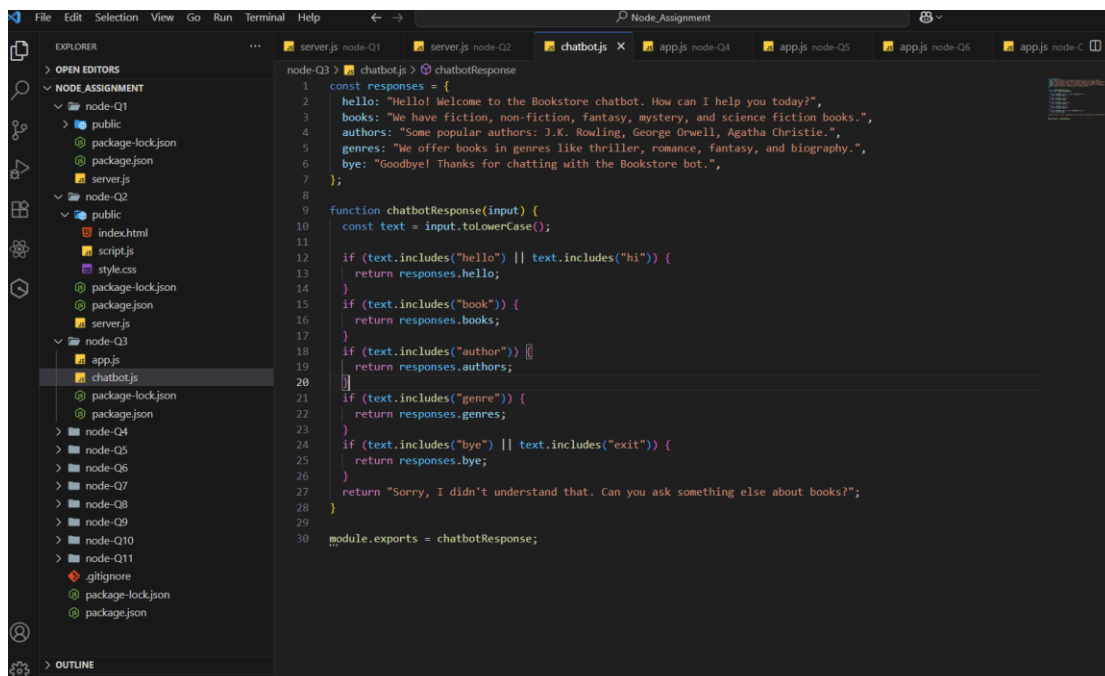
Develop a web server which serves static resources.



The screenshot shows a VS Code editor with a project named 'Node_Assignment'. The Explorer sidebar on the left shows a file tree with folders 'node-Q1' through 'node-Q11'. The 'node-Q2' folder is expanded, showing files 'index.html', 'script.js', 'style.css', 'package-lock.json', and 'server.js'. The 'server.js' file is selected and its content is displayed in the editor. The code uses Express.js to create a static resource server.

```
1 const express = require('express');
2 const path = require('path');
3
4 const app = express();
5 const PORT = 3000;
6
7 app.use(express.static('public'));
8
9
10 app.listen(PORT, () => {
11   console.log('Static resource server running at http://localhost:${PORT}');
12 });
13
```

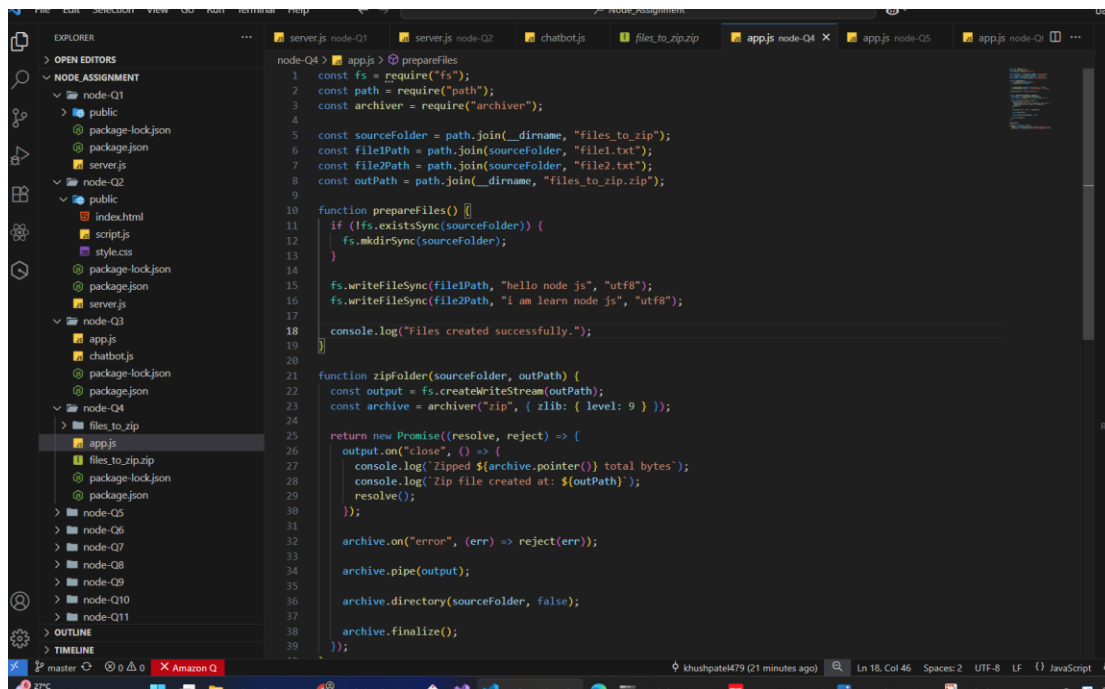
Develop a module for domain specific chatbot and use it in a command line application.



The screenshot shows a VS Code editor with the same 'Node_Assignment' project. The Explorer sidebar shows the 'node-Q3' folder expanded, containing 'app.js', 'chatbot.js', 'package-lock.json', and 'package.json'. The 'chatbot.js' file is selected and its content is displayed in the editor. The code defines a 'chatbotResponse' function that handles various user inputs related to a bookstore chatbot.

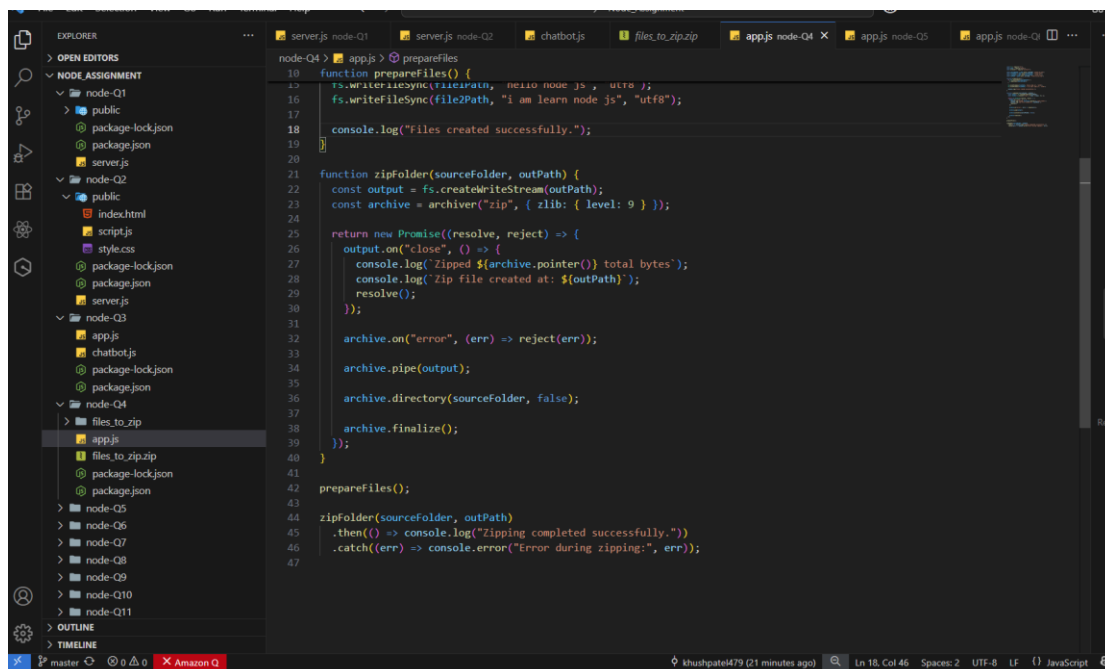
```
1 const responses = {
2   hello: "Hello! Welcome to the Bookstore chatbot. How can I help you today?",
3   books: "We have fiction, non-fiction, fantasy, mystery, and science fiction books.",
4   authors: "Some popular authors: J.K. Rowling, George Orwell, Agatha Christie.",
5   genres: "We offer books in genres like thriller, romance, fantasy, and biography.",
6   bye: "Goodbye! Thanks for chatting with the Bookstore bot.",
7 };
8
9 function chatbotResponse(input) {
10   const text = input.toLowerCase();
11
12   if (text.includes("hello") || text.includes("hi")) {
13     return responses.hello;
14   }
15   if (text.includes("book")) {
16     return responses.books;
17   }
18   if (text.includes("author")) {
19     return responses.authors;
20   }
21   if (text.includes("genre")) {
22     return responses.genres;
23   }
24   if (text.includes("bye") || text.includes("exit")) {
25     return responses.bye;
26   }
27   return "Sorry, I didn't understand that. Can you ask something else about books?";
28 }
29
30 module.exports = chatbotResponse;
```

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



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders for node-Q1 through node-Q11, and a file named 'files_to_zip.zip'. The code editor shows the following JavaScript code:

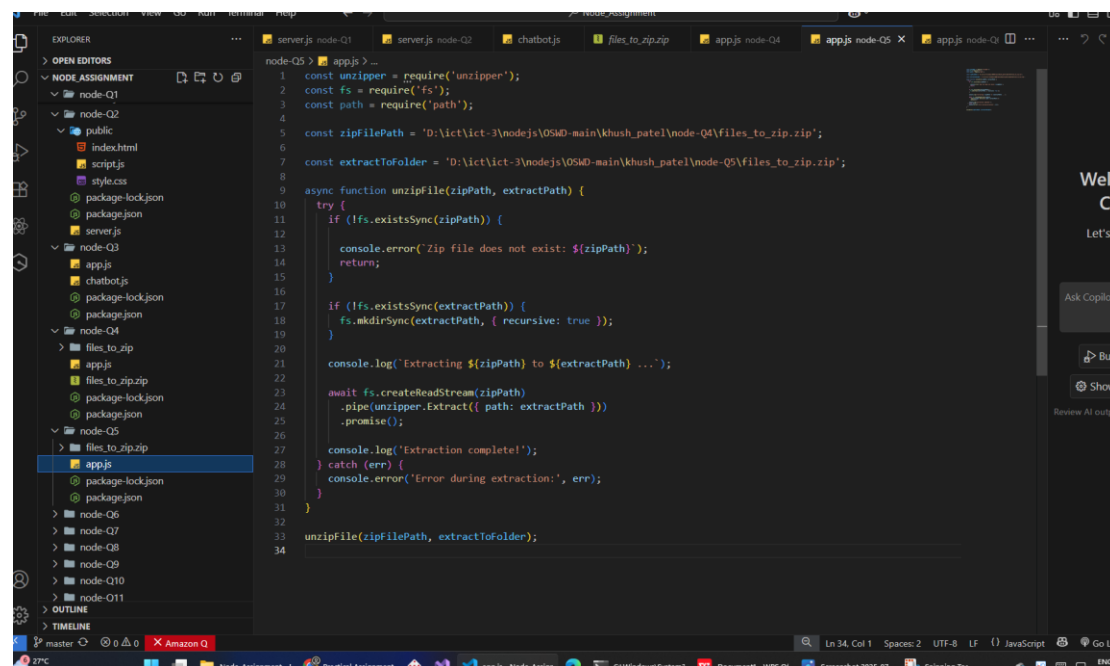
```
node-Q4 > appjs > prepareFiles
1  const fs = require("fs");
2  const path = require("path");
3  const archiver = require("archiver");
4
5  const sourceFolder = path.join(__dirname, "files_to_zip");
6  const file1Path = path.join(sourceFolder, "file1.txt");
7  const file2Path = path.join(sourceFolder, "file2.txt");
8  const outputPath = path.join(__dirname, "files_to_zip.zip");
9
10 function prepareFiles() {
11   if (fs.existsSync(sourceFolder)) {
12     fs.mkdirSync(sourceFolder);
13   }
14
15   fs.writeFileSync(file1Path, "hello node js", "utf8");
16   fs.writeFileSync(file2Path, "i am learn node js", "utf8");
17
18   console.log("Files created successfully.");
19 }
20
21 function zipfolder(sourceFolder, outputPath) {
22   const output = fs.createWriteStream(outputPath);
23   const archive = archiver("zip", { zlib: { level: 9 } });
24
25   return new Promise((resolve, reject) => {
26     output.on("close", () => {
27       console.log(`Zipped ${archive.pointer()} total bytes`);
28       console.log(`Zip file created at: ${outputPath}`);
29       resolve();
30     });
31
32     archive.on("error", (err) => reject(err));
33
34     archive.pipe(output);
35
36     archive.directory(sourceFolder, false);
37
38     archive.finalize();
39   });
40 }
```



The screenshot shows the same VS Code editor with the completed code. The code is now as follows:

```
node-Q4 > appjs > prepareFiles
10 function prepareFiles() {
11   fs.writeFileSync(file1Path, "hello node js", "utf8");
12   fs.writeFileSync(file2Path, "i am learn node js", "utf8");
13
14   console.log("Files created successfully.");
15 }
16
17 function zipfolder(sourceFolder, outputPath) {
18   const output = fs.createWriteStream(outputPath);
19   const archive = archiver("zip", { zlib: { level: 9 } });
20
21   return new Promise((resolve, reject) => {
22     output.on("close", () => {
23       console.log(`Zipped ${archive.pointer()} total bytes`);
24       console.log(`Zip file created at: ${outputPath}`);
25       resolve();
26     });
27
28     archive.on("error", (err) => reject(err));
29
30     archive.pipe(output);
31
32     archive.directory(sourceFolder, false);
33
34     archive.finalize();
35   });
36 }
37
38 prepareFiles();
39
40 zipfolder(sourceFolder, outputPath)
41   .then(() => console.log("Zipping completed successfully."))
42   .catch(err => console.error("Error during zipping:", err));
43 }
```

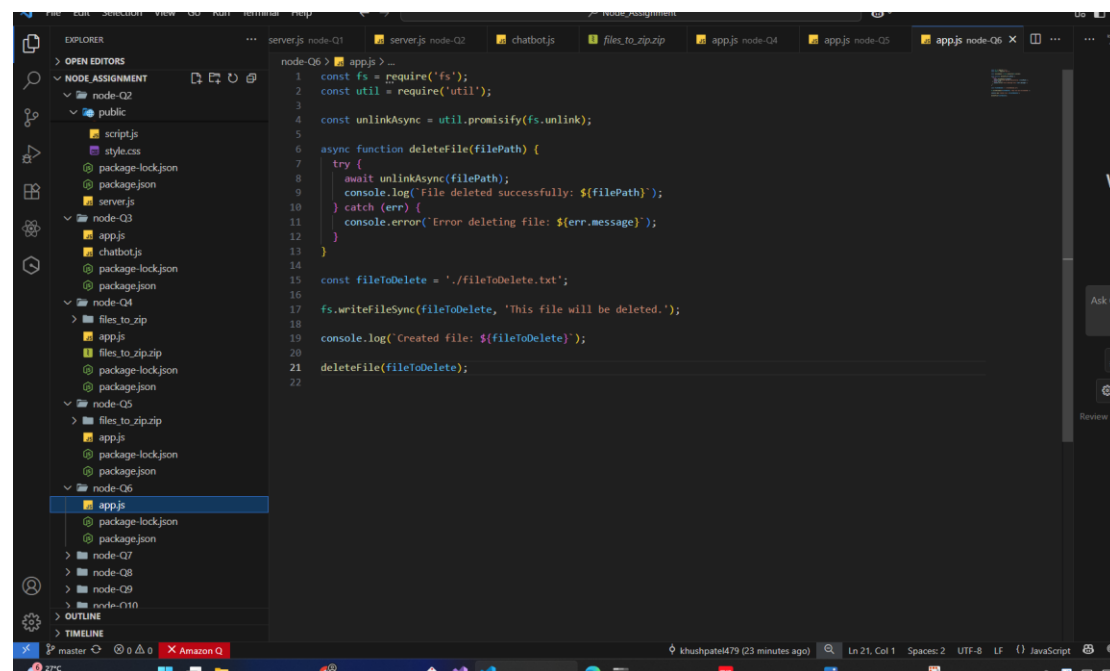
Write a program to extract a zip file.



The screenshot shows a VS Code editor with a project structure on the left and a JavaScript file named `app.js` open in the editor. The code defines a function `unzipFile` that takes a zip file path and an extract path as arguments. It checks if the zip file exists, creates the extract folder if it doesn't, and then uses `fs.createReadStream` and `unzipper.Extract` to extract the files. The code is as follows:

```
node-Q5 > app.js > ...
1 const unzipper = require('unzipper');
2 const fs = require('fs');
3 const path = require('path');
4
5 const zipFilePath = 'D:\\ict-3\\nodejs\\OSMD-main\\khush_patel\\node-Q4\\files_to_zip.zip';
6
7 const extractToFolder = 'D:\\ict-3\\nodejs\\OSMD-main\\khush_patel\\node-Q5\\files_to_zip';
8
9 async function unzipFile(zipPath, extractPath) {
10   try {
11     if (!fs.existsSync(zipPath)) {
12       console.error('Zip file does not exist: ${zipPath}');
13       return;
14     }
15
16     if (!fs.existsSync(extractPath)) {
17       fs.mkdirSync(extractPath, { recursive: true });
18     }
19
20     console.log('Extracting ${zipPath} to ${extractPath} ...');
21
22     await fs.createReadStream(zipPath)
23       .pipe(unzipper.Extract({ path: extractPath }))
24       .promise();
25
26     console.log('Extraction complete!');
27   } catch (err) {
28     console.error('Error during extraction:', err);
29   }
30 }
31
32 unzipFile(zipFilePath, extractToFolder);
33
34
```

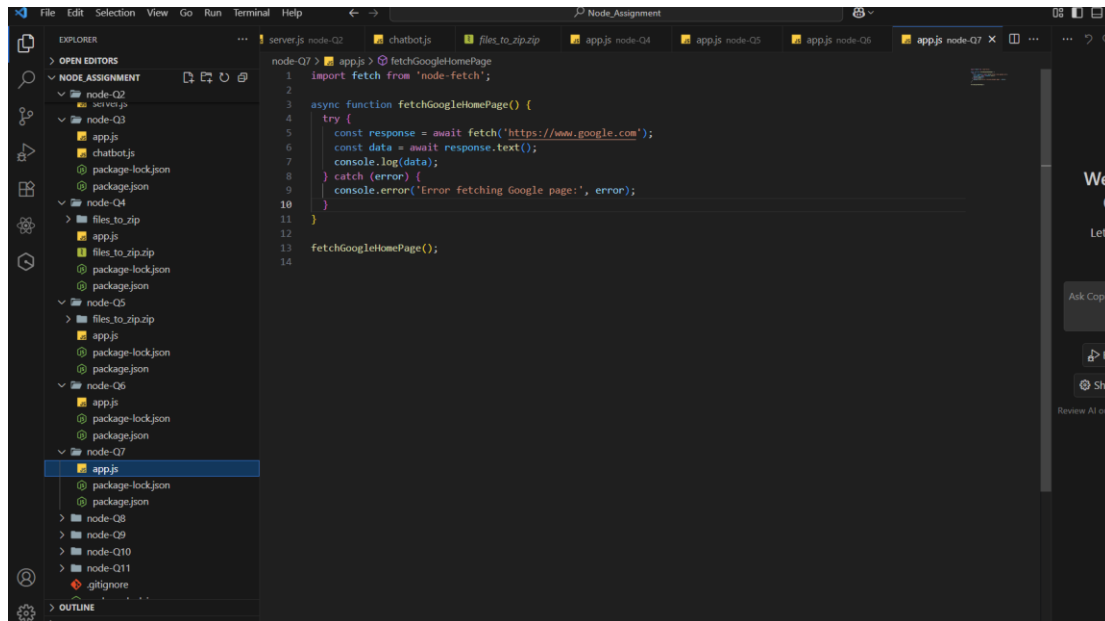
Write a program to promisify fs.unlink function and call it.



The screenshot shows a VS Code editor with a project structure on the left and a JavaScript file named `app.js` open in the editor. The code defines a function `deleteFile` that takes a file path as an argument. It uses `util.promisify` to promisify the `fs.unlink` function. The code is as follows:

```
node-Q6 > app.js > ...
1 const fs = require('fs');
2 const util = require('util');
3
4 const unlinkAsync = util.promisify(fs.unlink);
5
6 async function deleteFile(filePath) {
7   try {
8     await unlinkAsync(filePath);
9     console.log('File deleted successfully: ${filePath}');
10   } catch (err) {
11     console.error('Error deleting file: ${err.message}');
12   }
13 }
14
15 const fileToDelete = './fileToDelete.txt';
16
17 fs.writeFileSync(fileToDelete, 'This file will be deleted.');
```

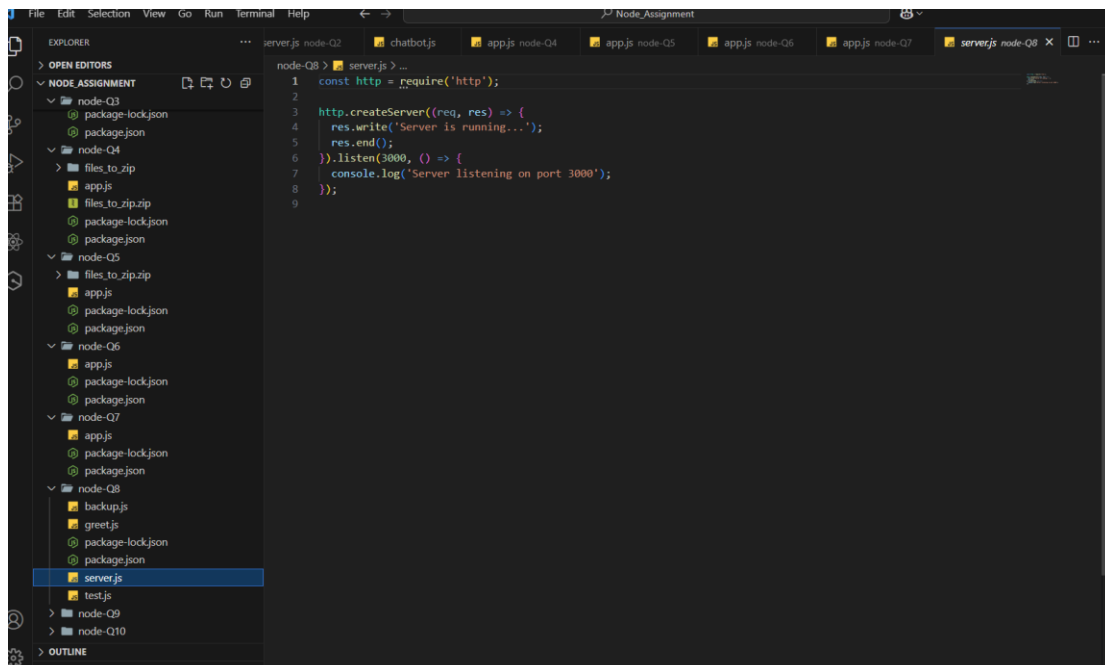
Fetch data of google page using node-fetch using async-await model.



The screenshot shows a VS Code editor with a project named 'Node_Assignment'. The Explorer sidebar on the left shows a file tree with folders 'node-Q2' through 'node-Q11' and files like 'app.js', 'package-lock.json', and 'package.json'. The active editor is 'app.js' in the 'node-Q7' folder. The code in 'app.js' uses 'node-fetch' to fetch data from 'https://www.google.com/' using the async-await model.

```
1 import fetch from '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(data);
8   } catch (error) {
9     console.error('Error fetching Google page:', error);
10  }
11 }
12
13 fetchGoogleHomePage();
14
```

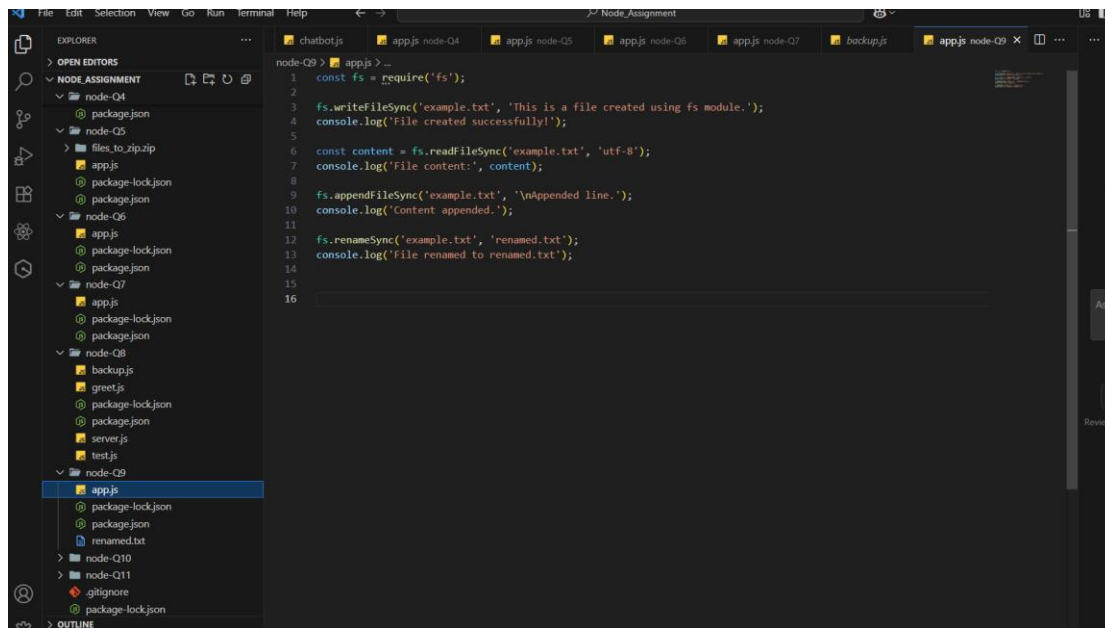
Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs application.



The screenshot shows a VS Code editor with a project named 'Node_Assignment'. The Explorer sidebar on the left shows a file tree with folders 'node-Q3' through 'node-Q10' and files like 'server.js', 'test.js', 'package-lock.json', and 'package.json'. The active editor is 'server.js' in the 'node-Q8' folder. The code in 'server.js' creates a simple HTTP server listening on port 3000.

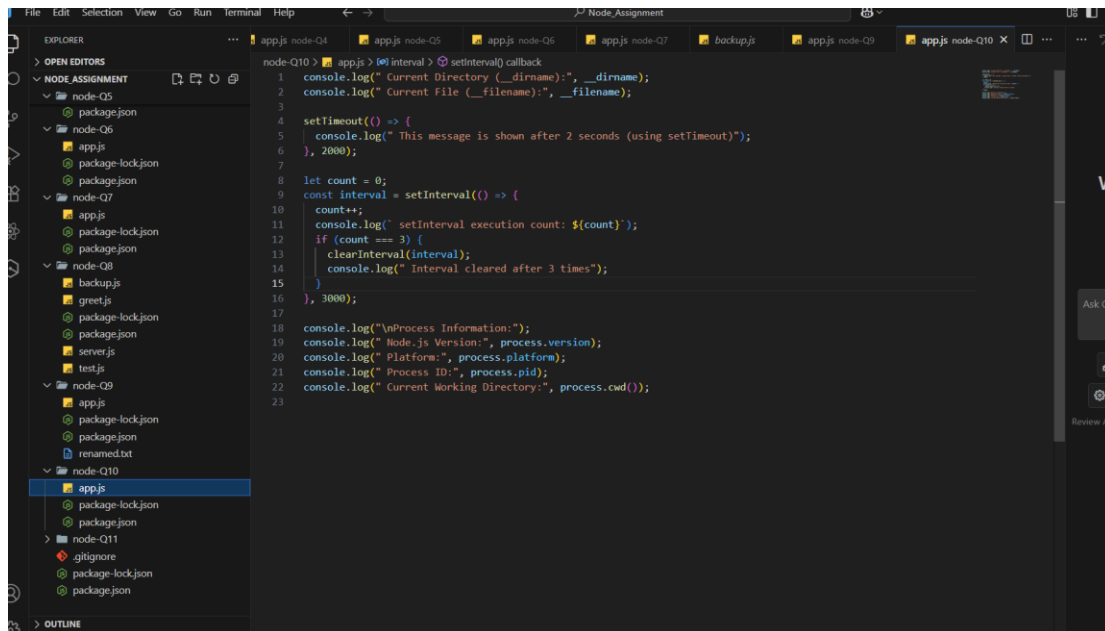
```
1 const http = require('http');
2
3 http.createServer((req, res) => {
4   res.writeHead(200, { 'Content-Type': 'text/plain' });
5   res.end('Hello World\n');
6 }).listen(3000, () => {
7   console.log('Server listening on port 3000');
8 });
9
```

A program which calls useful functions in fs module.



```
node-Q9 > app.js > ...
1  const fs = require('fs');
2
3  fs.writeFileSync('example.txt', 'This is a file created using fs module.');
```

A program which uses global objects in nodejs.



```
node-Q10 > app.js > @interval > setInterval() callback
1  console.log(" Current Directory (__dirname):", __dirname);
2  console.log(" Current File (__filename):", __filename);
3
4  setTimeout(() => {
5    console.log(" This message is shown after 2 seconds (using setTimeout)");
6  }, 2000);
7
8  let count = 0;
9  const interval = setInterval(() => {
10   count++;
11   console.log(" setInterval execution count: ${count}");
12   if (count === 3) {
13     clearInterval(interval);
14     console.log(" Interval cleared after 3 times");
15   }
16 }, 3000);
17
18 console.log("\nProcess Information:");
19 console.log(" Node.js Version:", process.version);
20 console.log(" Platform:", process.platform);
21 console.log(" Process ID:", process.pid);
22 console.log(" Current Working Directory:", process.cwd());
23
```