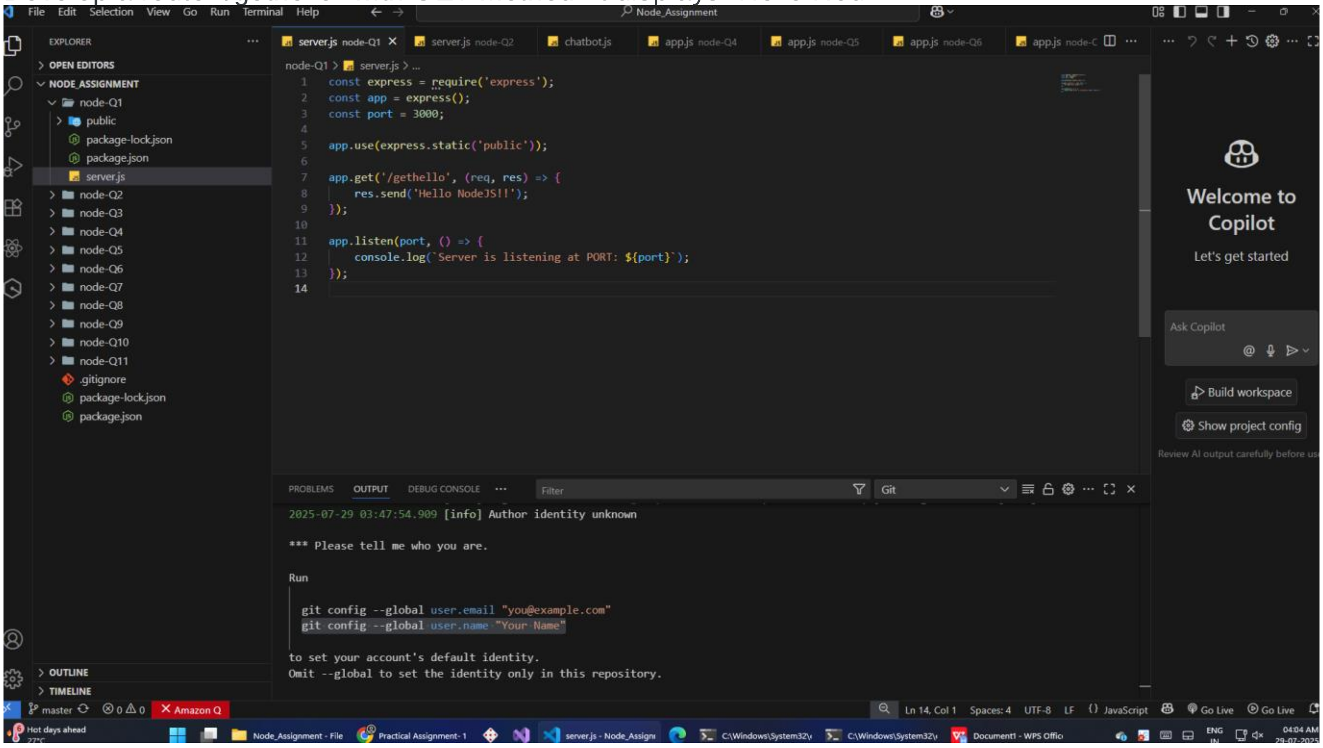
Name : Patel Khush Alpesh

Roll No : 45

Subject : Node js

Practical Assignment - 1

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

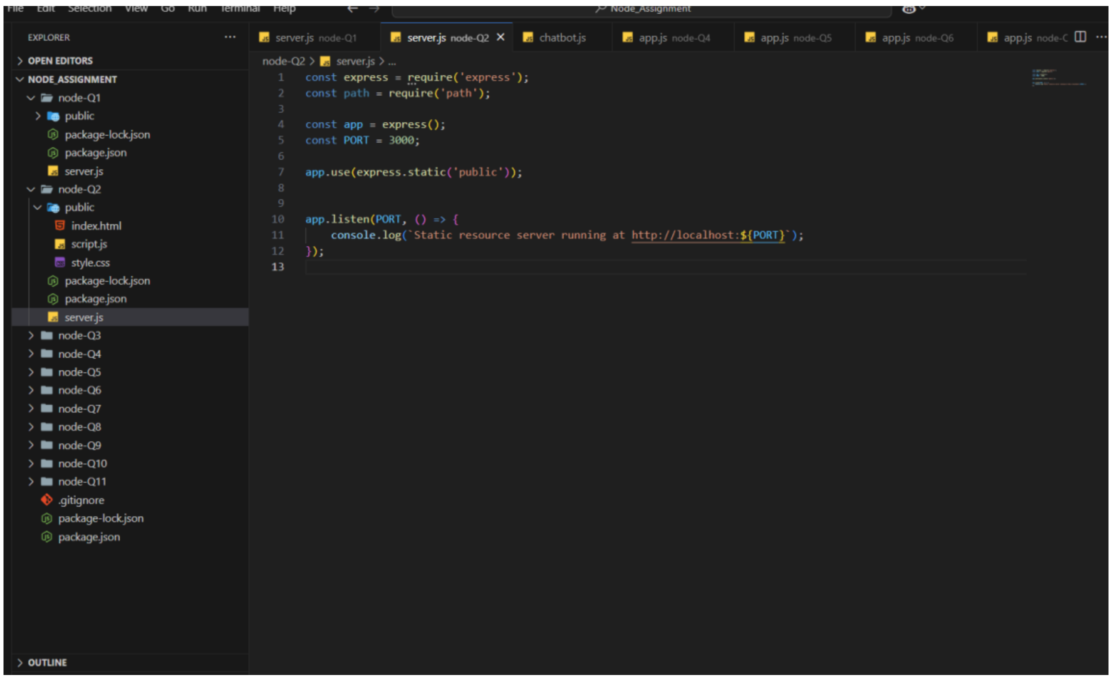


```javascript
const express = require('express');
const app = express();
const port = 3000;

app.use(express.static('public'));

app.get('/gethello', (req, res) => {
    res.send('Hello NodeJS!!');
});

app.listen(port, () => {
    console.log(`Server is listening at PORT: ${port}`);
});
```

```
2025-07-29 03:47:54.909 [info] Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

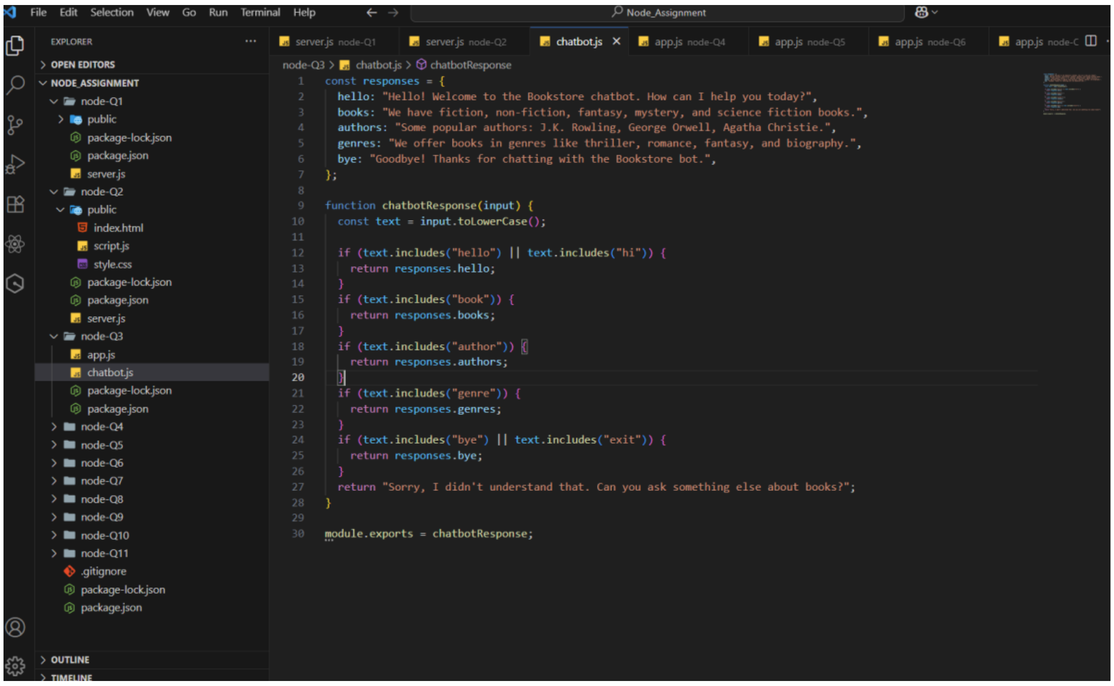
eJS!!" as response.

Develop a web server which serves static resources.



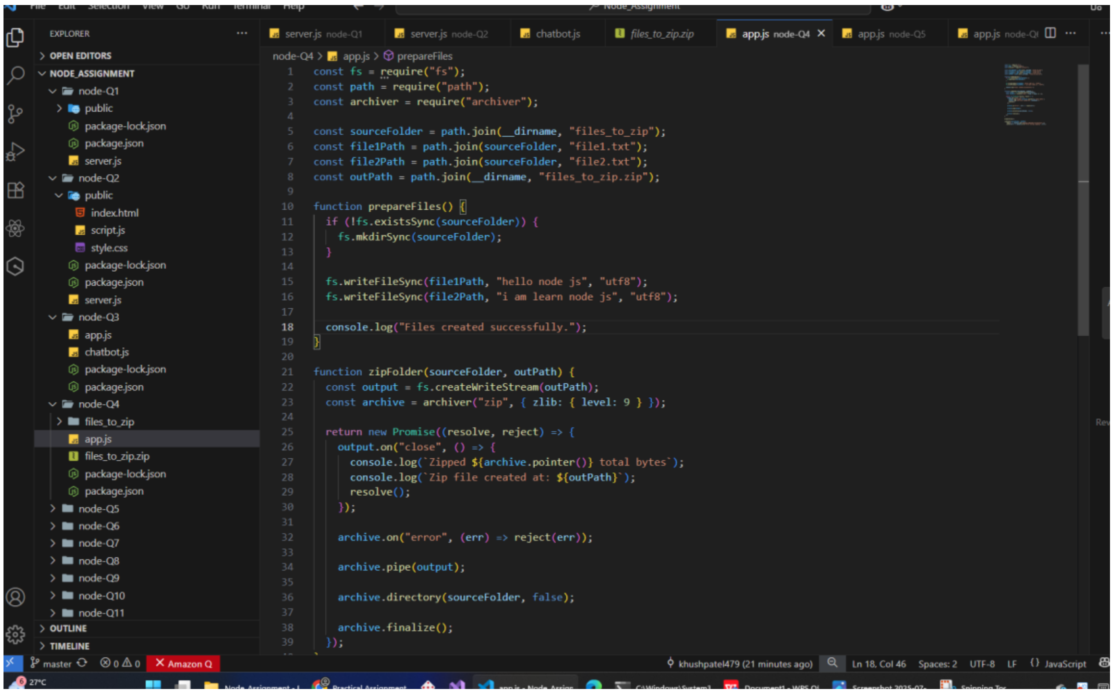
```
const express = require('express');
const path = require('path');

const app = express();
const PORT = 3000;

app.use(express.static('public'));


app.listen(PORT, () => {
    console.log(`Static resource server running at http://localhost:${PORT}`);
});
```

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

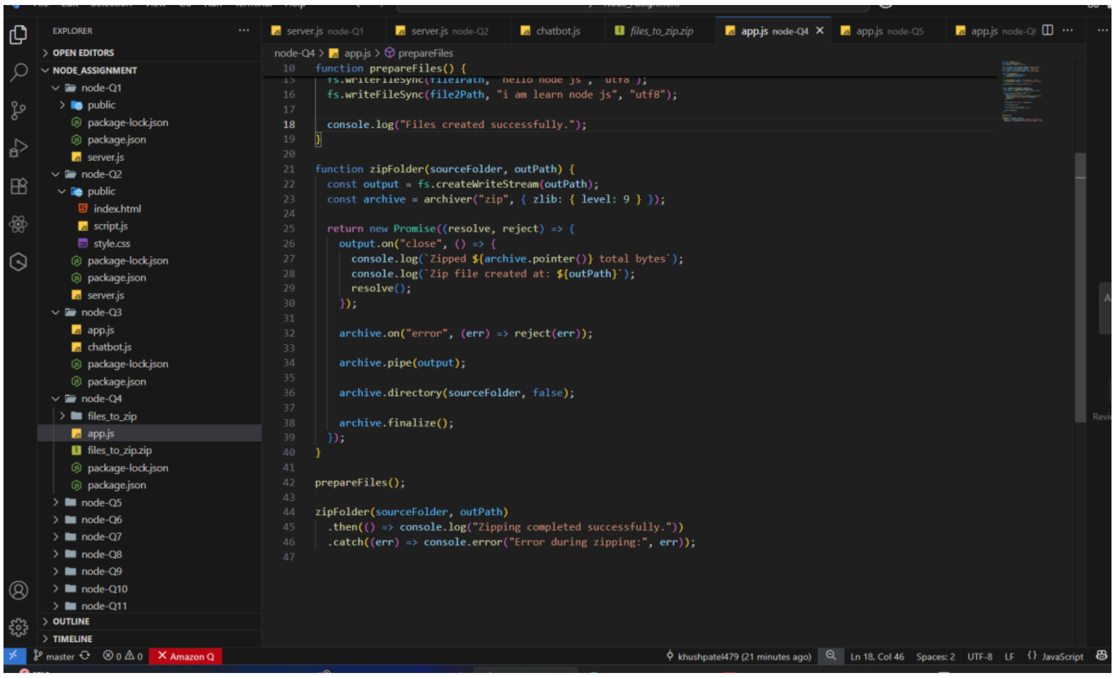


```
const responses = {
  hello: "Hello! Welcome to the Bookstore chatbot. How can I help you today?",
  books: "We have fiction, non-fiction, fantasy, mystery, and science fiction books.",
  authors: "Some popular authors: J.K. Rowling, George Orwell, Agatha Christie.",
  genres: "We offer books in genres like thriller, romance, fantasy, and biography.",
  bye: "Goodbye! Thanks for chatting with the Bookstore bot.",
};

function chatbotResponse(input) {
  const text = input.toLowerCase();

  if (text.includes("hello") || text.includes("hi")) {
    return responses.hello;
  }
  if (text.includes("book")) {
    return responses.books;
  }
  if (text.includes("author")) {
    return responses.authors;
  }
  if (text.includes("genre")) {
    return responses.genres;
  }
  if (text.includes("bye") || text.includes("exit")) {
    return responses.bye;
  }
  return "Sorry, I didn't understand that. Can you ask something else about books?";
}

module.exports = chatbotResponse;
```

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

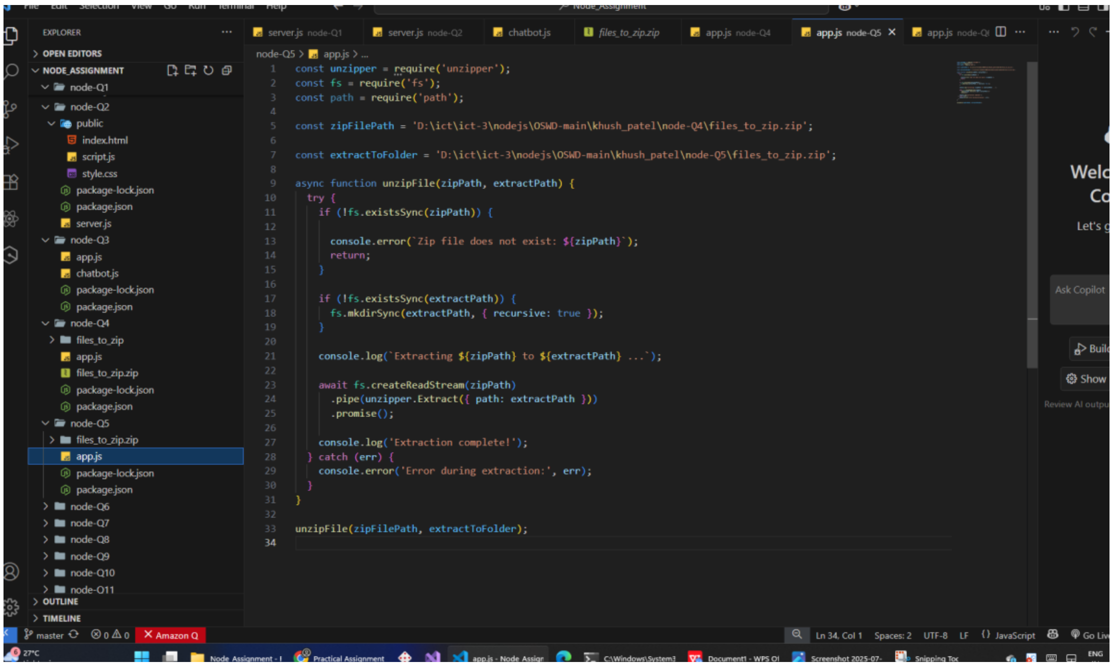


```javascript
const fs = require("fs");
const path = require("path");
const archiver = require("archiver");

const sourceFolder = path.join(__dirname, "files_to_zip");
const file1Path = path.join(sourceFolder, "file1.txt");
const file2Path = path.join(sourceFolder, "file2.txt");
const outPath = path.join(__dirname, "files_to_zip.zip");

function prepareFiles() {
  if (!fs.existsSync(sourceFolder)) {
    fs.mkdirSync(sourceFolder);
  }

  fs.writeFileSync(file1Path, "hello node js", "utf8");
  fs.writeFileSync(file2Path, "i am learn node js", "utf8");

  console.log("Files created successfully.");
}

function zipFolder(sourceFolder, outPath) {
  const output = fs.createWriteStream(outPath);
  const archive = archiver("zip", { zlib: { level: 9 } });

  return new Promise((resolve, reject) => {
    output.on("close", () => {
      console.log(`Zipped ${archive.pointer()} total bytes`);
      console.log(`Zip file created at: ${outPath}`);
      resolve();
    });

    archive.on("error", (err) => reject(err));

    archive.pipe(output);

    archive.directory(sourceFolder, false);

    archive.finalize();
  });
```



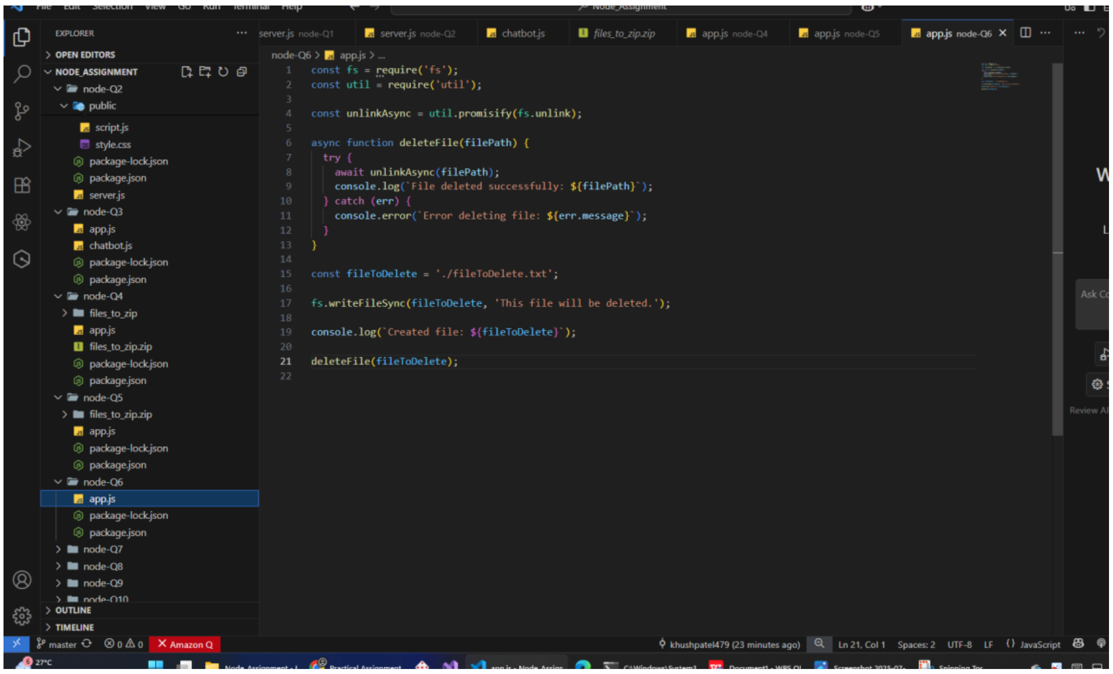
```javascript
function prepareFiles() {
  fs.writeFileSync(file2Path, "i am learn node js", "utf8");

  console.log("Files created successfully.");
}

function zipFolder(sourceFolder, outPath) {
  const output = fs.createWriteStream(outPath);
  const archive = archiver("zip", { zlib: { level: 9 } });

  return new Promise((resolve, reject) => {
    output.on("close", () => {
      console.log(`Zipped ${archive.pointer()} total bytes`);
      console.log(`Zip file created at: ${outPath}`);
      resolve();
    });

    archive.on("error", (err) => reject(err));

    archive.pipe(output);

    archive.directory(sourceFolder, false);

    archive.finalize();
  });
}

prepareFiles();

zipFolder(sourceFolder, outPath)
  .then(() => console.log("Zipping completed successfully."))
  .catch((err) => console.error("Error during zipping:", err));
```

Write a program to extract a zip file.



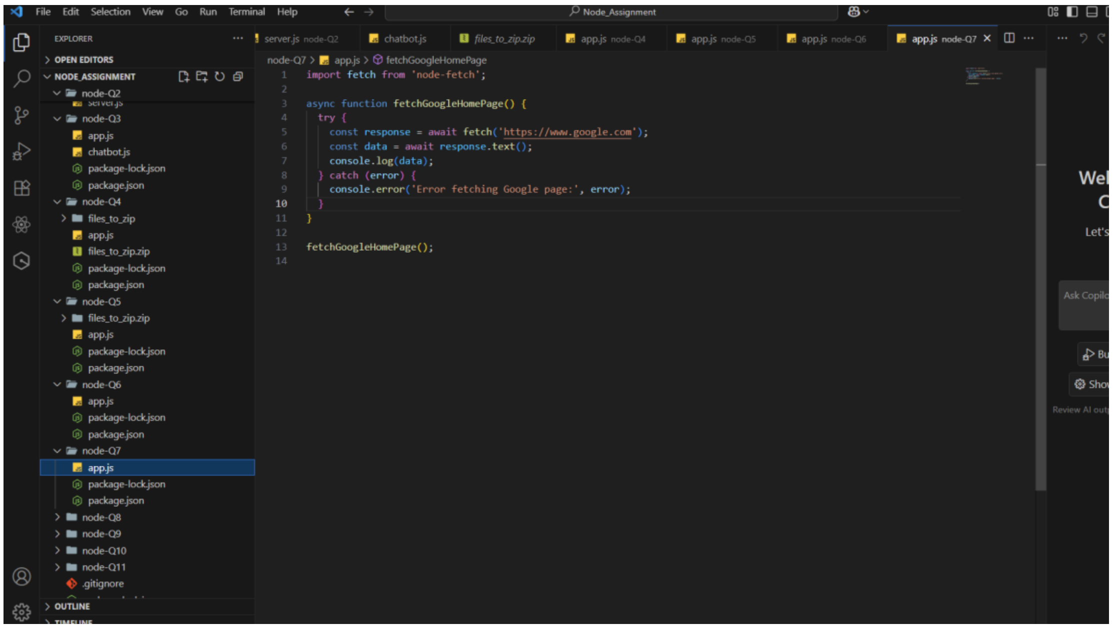
```javascript
const unzipper = require('unzipper');
const fs = require('fs');
const path = require('path');

const zipFilePath = 'D:\ict\ict-3\nodejs\OSWD-main\khush_patel\node-Q4\files_to_zip.zip';

const extractToFolder = 'D:\ict\ict-3\nodejs\OSWD-main\khush_patel\node-Q5\files_to_zip.zip';

async function unzipFile(zipPath, extractPath) {
  try {
    if (!fs.existsSync(zipPath)) {

      console.error(`Zip file does not exist: ${zipPath}`);
      return;
    }

    if (!fs.existsSync(extractPath)) {
      fs.mkdirSync(extractPath, { recursive: true });
    }

    console.log(`Extracting ${zipPath} to ${extractPath} ...`);

    await fs.createReadStream(zipPath)
      .pipe(unzipper.Extract({ path: extractPath }))
      .promise();

    console.log('Extraction complete!');
  } catch (err) {
    console.error('Error during extraction:', err);
  }
}

unzipFile(zipFilePath, extractToFolder);
```

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



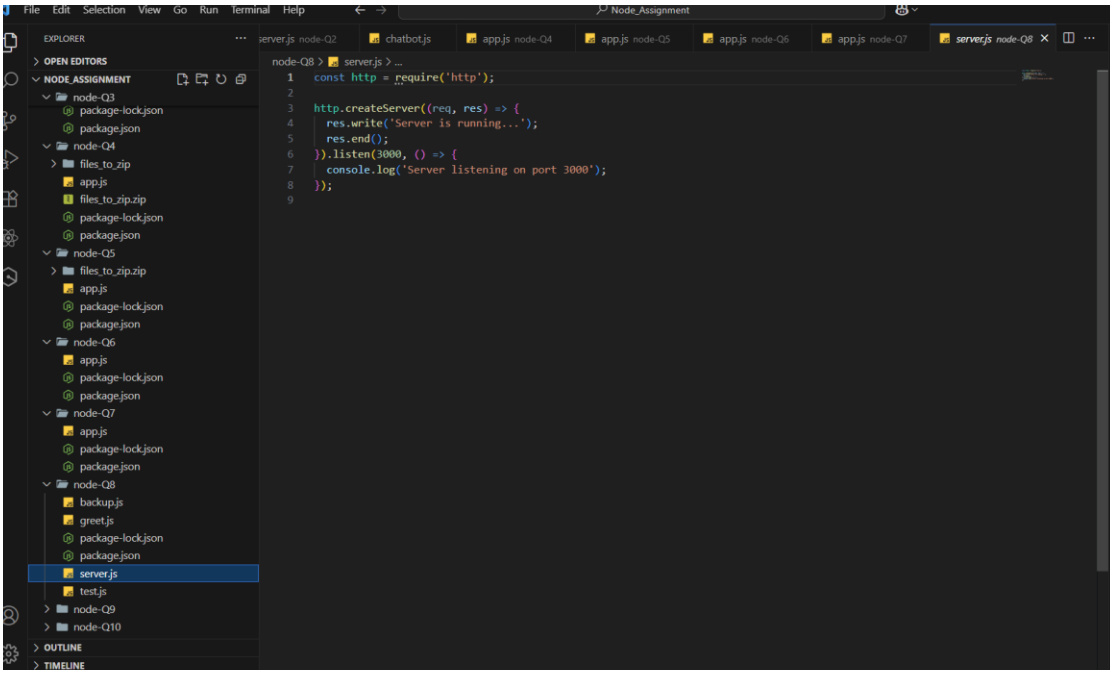
```javascript
const fs = require('fs');
const util = require('util');

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

async function deleteFile(filePath) {
  try {
    await unlinkAsync(filePath);
    console.log(`File deleted successfully: ${filePath}`);
  } catch (err) {
    console.error(`Error deleting file: ${err.message}`);
  }
}

const fileToDelete = './fileToDelete.txt';

fs.writeFileSync(fileToDelete, 'This file will be deleted.');

console.log(`Created file: ${fileToDelete}`);

deleteFile(fileToDelete);
```

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



```javascript
import fetch from 'node-fetch';

async function fetchGoogleHomePage() {
  try {
    const response = await fetch('https://www.google.com');
    const data = await response.text();
    console.log(data);
  } catch (error) {
    console.error('Error fetching Google page:', error);
  }
}

fetchGoogleHomePage();
```

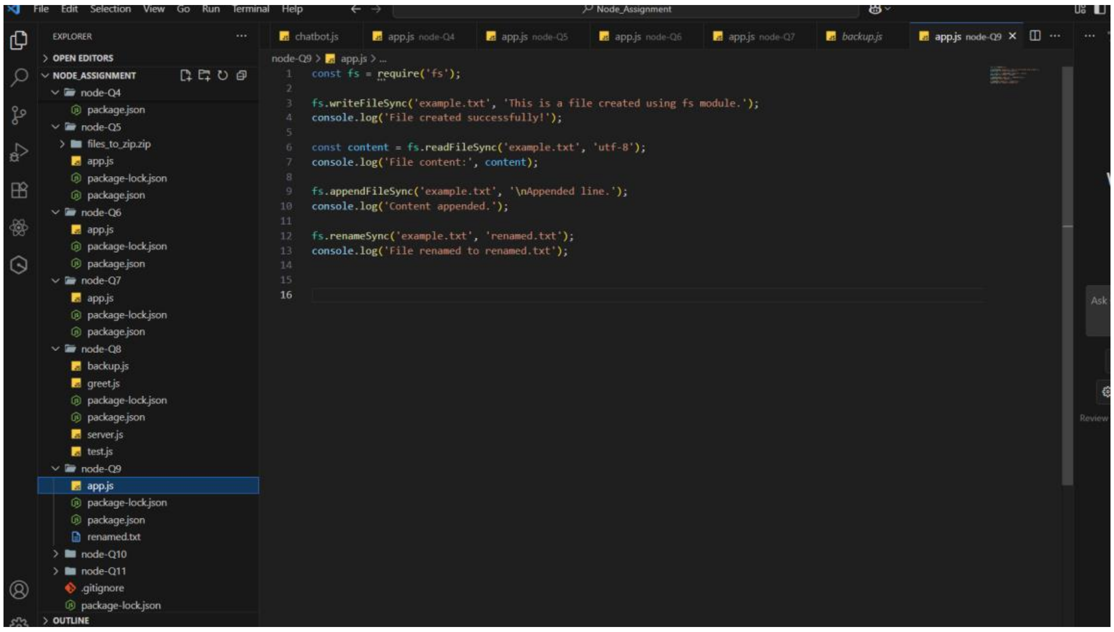
Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs
application.



```javascript
const http = require('http');

http.createServer((req, res) => {
  res.write('Server is running...');
  res.end();
}).listen(3000, () => {
  console.log('Server listening on port 3000');
});
```

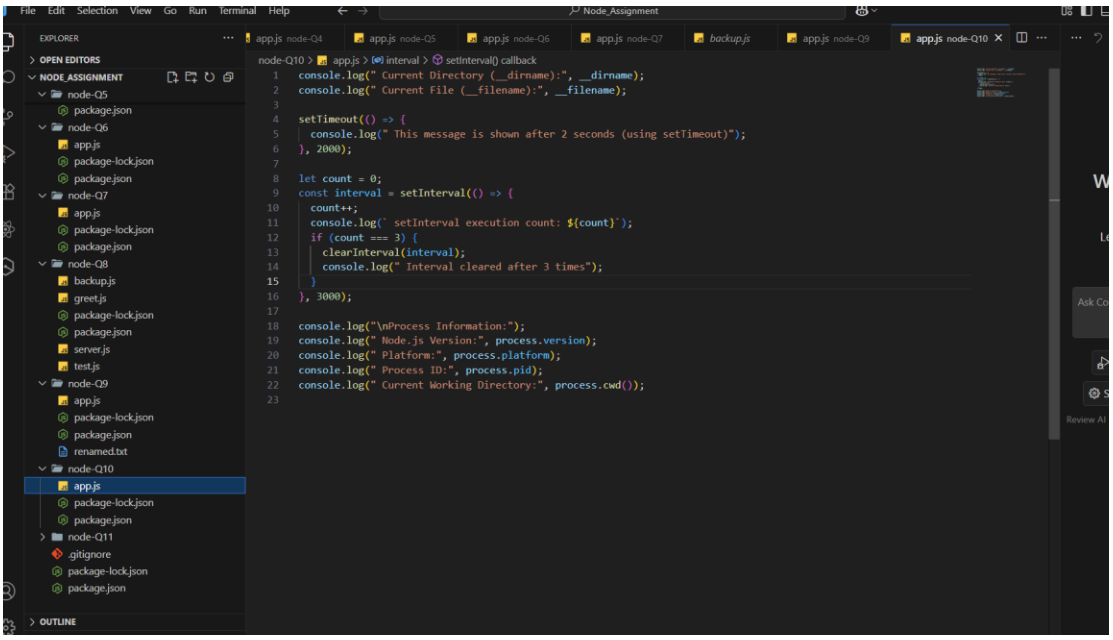
\

A program which calls useful functions in fs modile.



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

fs.writeFileSync('example.txt', 'This is a file created using fs module.');
console.log('File created successfully!');

const content = fs.readFileSync('example.txt', 'utf-8');
console.log('File content:', content);

fs.appendFileSync('example.txt', '\nAppended line.');
console.log('Content appended.');

fs.renameSync('example.txt', 'renamed.txt');
console.log('File renamed to renamed.txt');


```

A program which uses global objects in nodejs.



```
console.log(" Current Directory (__dirname):", __dirname);
console.log(" Current File (__filename):", __filename);

setTimeout(() => {
  console.log(" This message is shown after 2 seconds (using setTimeout)");
}, 2000);

let count = 0;
const interval = setInterval(() => {
  count++;
  console.log(` setInterval execution count: ${count}`);
  if (count === 3) {
    clearInterval(interval);
    console.log(" Interval cleared after 3 times");
  }
}, 3000);

console.log("\nProcess Information:");
console.log(" Node.js Version:", process.version);
console.log(" Platform:", process.platform);
console.log(" Process ID:", process.pid);
console.log(" Current Working Directory:", process.cwd());

```