Name : prapti shah
Roll no : 65
sub : Full stack
Practical assignment : 1
git hub link : https://github.com/Praptishah2712/practical_ass1.git
----
Q1.

index.html :

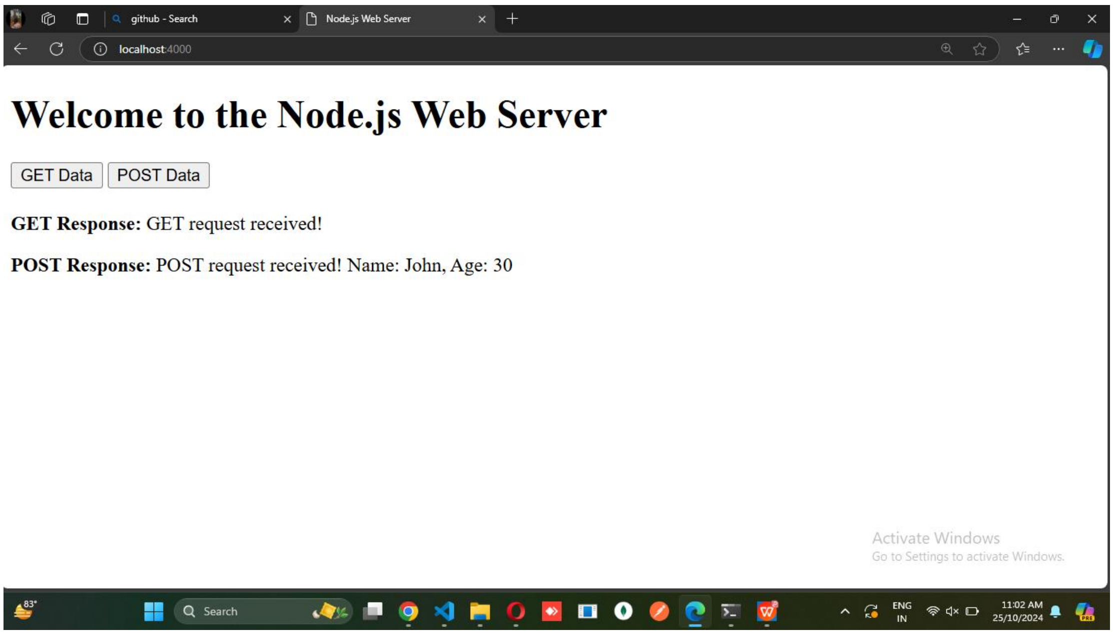
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Node.js Web Server</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>Welcome to the Node.js Web Server</h1>
<!-- Button to trigger GET request -->
<button onclick="fetchData()">GET Data</button>
<!-- Button to trigger POST request -->
<button onclick="submitData()">POST Data</button>
<!-- Section to display the data received from the server -->
<div id="display-data" style="margin-top: 20px;">
<!-- Data from the server will be inserted here -->
</div>
<script>
// Function to fetch data from the GET requestfunction fetchData() {
fetch('/get-data')
.then(response => response.json())
.then(data => {
displayData('GET', data.message);
});
}
// Function to send data with a POST request
function submitData() {
fetch('/submit-data', {
method: 'POST',
headers: {
'Content-Type': 'application/json'
},
body: JSON.stringify({ name: 'John', age: 30 })
})
.then(response => response.json())
.then(data => {
displayData('POST', data.message + ' Name: ' + data.data.name + ', Age: ' + data.data.age);
});
}
// Function to display data on the UI
```

```
function displayData(requestType, message) {
// Get the div where the data will be displayed
const displayDiv = document.getElementById('display-data');
// Create a new paragraph element for the response
const newParagraph = document.createElement('p');
newParagraph.innerHTML = `<strong>${requestType} Response:</strong> ${message}`;//
Append the new paragraph to the div
displayDiv.appendChild(newParagraph);
}
</script>
</body>
</html>
```

Server.js :

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const url = require('url');
// Function to serve static files
function serveStaticFile(res, filePath, contentType, responseCode = 200) {
fs.readFile(filePath, (err, data) => {
if (err) {
res.writeHead(500, { 'Content-Type': 'text/plain' });
res.end('500 - Internal Error');
} else {
res.writeHead(responseCode, { 'Content-Type': contentType });
res.end(data);
}
});
}
// Create the server
const server = http.createServer((req, res) => {
// Parse the URL
const parsedUrl = url.parse(req.url, true);const pathname = parsedUrl.pathname;
// Serve static resources from 'public' folder
if (pathname === '/' || pathname === '/index.html') {
serveStaticFile(res, './public/index.html', 'text/html');
} else if (pathname.match(/\.css$/)) {
serveStaticFile(res, './public' + pathname, 'text/css');
} else if (pathname.match(/\.js$/)) {
serveStaticFile(res, './public' + pathname, 'application/javascript');
} else if (pathname.match(/\.png$/)) {
serveStaticFile(res, './public' + pathname, 'image/png');
} else if (pathname.match(/\.jpg$/)) {
serveStaticFile(res, './public' + pathname, 'image/jpeg');
}
// Handle GET request
else if (pathname === '/get-data' && req.method === 'GET') {
// For example: returning some sample data
res.writeHead(200, { 'Content-Type': 'application/json' });
res.end(JSON.stringify({ message: 'GET request received!' }));
}
// Handle POST request
else if (pathname === '/submit-data' && req.method === 'POST') {
let body = '';
req.on('data', chunk => {
body += chunk;
});
req.on('end', () => {
// Process the POST data
const postData = JSON.parse(body);res.writeHead(200, { 'Content-Type': 'application/json' });
res.end(JSON.stringify({ message: 'POST request received!', data: postData }));
});
}
// Handle 404 - Not Found
else {
res.writeHead(404, { 'Content-Type': 'text/plain' });
res.end('404 - Not Found');
}
});
// Start the server
const PORT = 4000;
server.listen(PORT, () => {
console.log(`Server running on http://localhost:${PORT}`);
});
```

Output:



Welcome to the Node.js Web Server

GET Response: GET request received!

POST Response: POST request received! Name: John, Age: 30

Q2.

app.js :

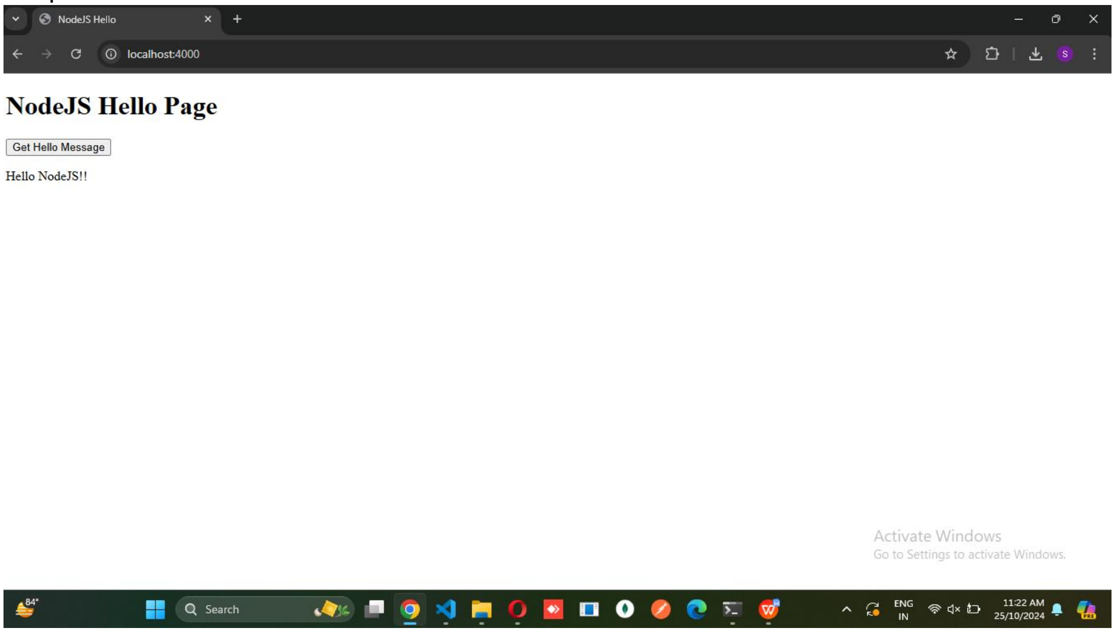
```
const http = require('http');
const fs = require('fs');
const path = require('path');
const server = http.createServer((req, res) => {
if (req.method === 'GET' && req.url === '/') {
// Serve the HTML page
fs.readFile(path.join(__dirname, 'index.html'), (err, data) => {
if (err) {
res.writeHead(500, { 'Content-Type': 'text/plain' });
res.end('Server Error');
return;
}
res.writeHead(200, { 'Content-Type': 'text/html' });
res.end(data);
});
} else if (req.method === 'GET' && req.url === '/gethello') {
// Handle the /gethello route
res.writeHead(200, { 'Content-Type': 'text/plain' });
res.end('Hello NodeJS!!');
} else {
// Handle 404 Not Found
res.writeHead(404, { 'Content-Type': 'text/plain' });
res.end('Not Found');
}
});
const PORT = process.env.PORT || 4000;server.listen(PORT, () => {
console.log(`Server is running on port ${PORT}`);
});
```

index.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>NodeJS Hello</title>
</head>
<body>
<h1>NodeJS Hello Page</h1>
<button id="getHelloBtn">Get Hello Message</button>
<p id="helloMessage"></p>
<script>
document.getElementById('getHelloBtn').addEventListener('click', function() {
fetch('/gethello')
.then(response => response.text())
.then(data => {
```

```
document.getElementById('helloMessage').innerText = data;
})
.catch(error => console.error('Error:', error));
});
</script>
</body>
</html>
```

Output:



NodeJS Hello Page

```
Hello NodeJS!!
```

Q3.

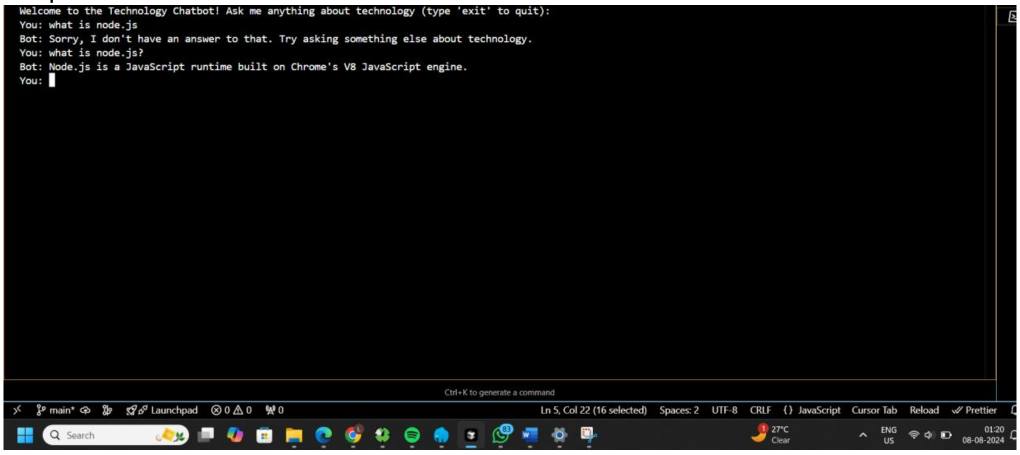
App.js :

```
// app.js
const readline = require('readline');
const chatbot = require('./chatbot');
// Create an interface for reading input from the terminal
const rl = readline.createInterface({
input: process.stdin,
output: process.stdout,
});
console.log("Welcome to the Technology Chatbot! Ask me anything about technology (type
'exit' to
quit):");
// Function to ask questions
function ask() {
rl.question('You: ', (input) => {
if (input.toLowerCase() === 'exit') {console.log("Goodbye!");
rl.close();
return;
}
// Get the response from the chatbot module
const response = chatbot.askQuestion(input);
console.log(`Bot: ${response}`);
// Continue asking questions
ask();
});
}
// Start asking questions
ask();
```

chatbot.js

```
// chatbot.js
// A simple object containing domain-specific responses
const responses = {
"what is node.js?": "Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.",
"what is express?": "Express is a fast, unopinionated, minimalist web framework for
Node.js.",
"what is a chatbot?": "A chatbot is an artificial intelligence (AI) program that simulates
interactive
human conversation.",
"how does async work in javascript?": "Asynchronous programming in JavaScript is achieved
using
callbacks, promises, and async/await.",
};
// The chatbot function
function askQuestion(question) {const normalizedQuestion = question.toLowerCase().trim();
if (responses[normalizedQuestion]) {
return responses[normalizedQuestion];
} else {
return "Sorry, I don't have an answer to that. Try asking something else about technology.";
}
}
// Export the module so it can be used in other files
module.exports = {
askQuestion,
};
```

Output:



```
Welcome to the Technology Chatbot! Ask me anything about technology (type 'exit' to quit):
You: what is node.js
Bot: Sorry, I don't have an answer to that. Try asking something else about technology.
You: what is node.js?
Bot: Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.
You: 
```

Q4

chatbot.js

```
// chatbot.js
// A simple object containing domain-specific responsesconst responses = {
"what is node.js?": "Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.",
"what is express?": "Express is a fast, unopinionated, minimalist web framework for
Node.js.",
"what is a chatbot?": "A chatbot is an artificial intelligence (AI) program that simulates
interactive
human conversation.",
"how does async work in javascript?": "Asynchronous programming in JavaScript is achieved
using
callbacks, promises, and async/await.",
};
// The chatbot function
function askQuestion(question) {
const normalizedQuestion = question.toLowerCase().trim();
if (responses[normalizedQuestion]) {
return responses[normalizedQuestion];
} else {
return "Sorry, I don't have an answer to that. Try asking something else about technology.";
}
}
// Export the module so it can be used in other files
module.exports = {
askQuestion,
};
```

Index.html

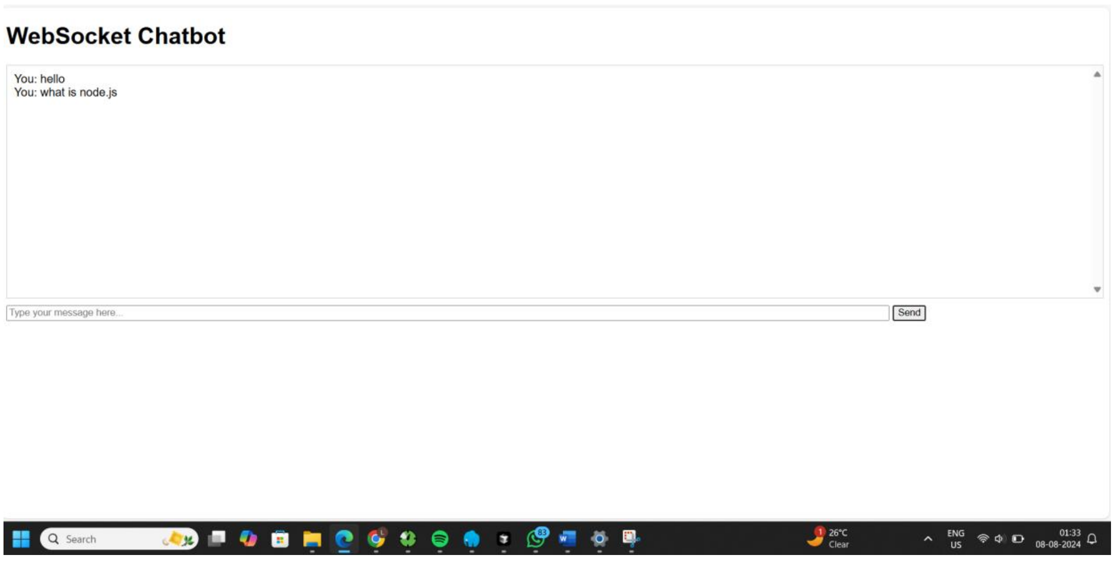
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>WebSocket Chatbot</title><style>
body {
font-family: Arial, sans-serif;
}
#chat {
border: 1px solid #ccc;
height: 300px;
overflow-y: scroll;
padding: 10px;
margin-bottom: 10px;
}
#input {
width: 80%;
}
</style>
</head>
```

```
<body>
<h1>WebSocket Chatbot</h1>
<div id="chat"></div>
<input type="text" id="input" placeholder="Type your message here..."/>
<button id="sendBtn">Send</button>
<script>
const chat = document.getElementById('chat');
const input = document.getElementById('input');
const sendBtn = document.getElementById('sendBtn');
// Connect to the WebSocket server
const socket = new WebSocket('ws://localhost:8080');
// Listen for messages from the serversocket.addEventListener('message', (event) => {
const message = document.createElement('div');
message.textContent = 'Bot: ' + event.data;
chat.appendChild(message);
chat.scrollTop = chat.scrollHeight; // Scroll to the bottom
});
// Send a message to the server when the button is clicked
sendBtn.addEventListener('click', () => {
const userMessage = input.value;
if (userMessage) {
const message = document.createElement('div');
message.textContent = 'You: ' + userMessage;
chat.appendChild(message);
socket.send(userMessage);
input.value = ''; // Clear the input field
}
});
// Optional: Send a message when the user presses Enter
input.addEventListener('keypress', (event) => {
if (event.key === 'Enter') {
sendBtn.click();
}
});
</script>
</body>
</html>
```

Server.js

```
const WebSocket = require('ws');const chatbot = require('./chatbot');
// Create a WebSocket server
const wss = new WebSocket.Server({ port: 8080 });
wss.on('connection', (ws) => {
console.log('New client connected');
// Listen for messages from clients
ws.on('message', (message) => {
console.log(`Received message: ${message}`);
// Get the response from the chatbot module
const response = chatbot.askQuestion(message);
// Send the response back to the client
ws.send(response);
});
ws.on('close', () => {
console.log('Client disconnected');
});
});
console.log('WebSocket server is running on ws://localhost:8080');
```

output:



WebSocket Chatbot

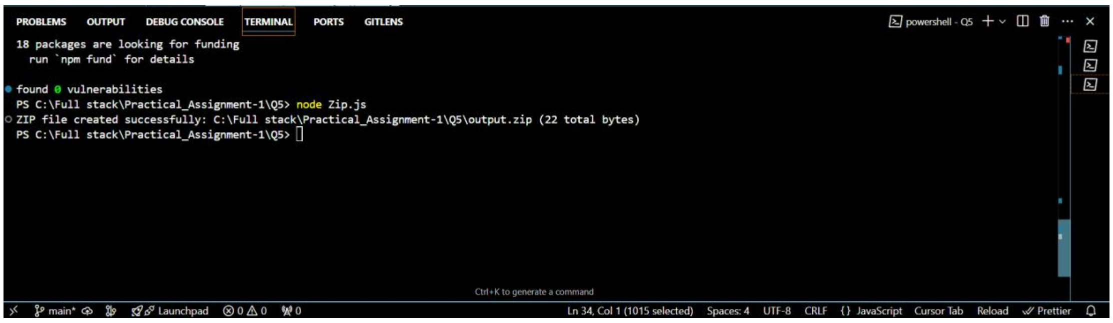
You: hello
You: what is node.js

Q5

Zip.js :

```
const fs = require('fs');
const path = require('path');
const archiver = require('archiver');
// Function to create a ZIP file from a folder
function zipFolder(sourceFolder, outPath) {
const output = fs.createWriteStream(outPath);
const archive = archiver('zip', {
zlib: { level: 9 }, // Sets the compression level
});
output.on('close', () => {
console.log(`ZIP file created successfully: ${outPath} (${archive.pointer()} total bytes)`);
});
archive.on('error', (err) => {
throw err;});
archive.pipe(output);
// Append files from the source folder to the archive
archive.directory(sourceFolder, false); // 'false' means do not include the root folder

// Finalize the archive (i.e., finish the ZIP file)
archive.finalize();
}
// Usage example

const folderToZip = path.join(__dirname, 'your-folder'); // Change 'your-folder' to your

folder name

const zipFilePath = path.join(__dirname, 'output.zip');
zipFolder(folderToZip, zipFilePath);
```

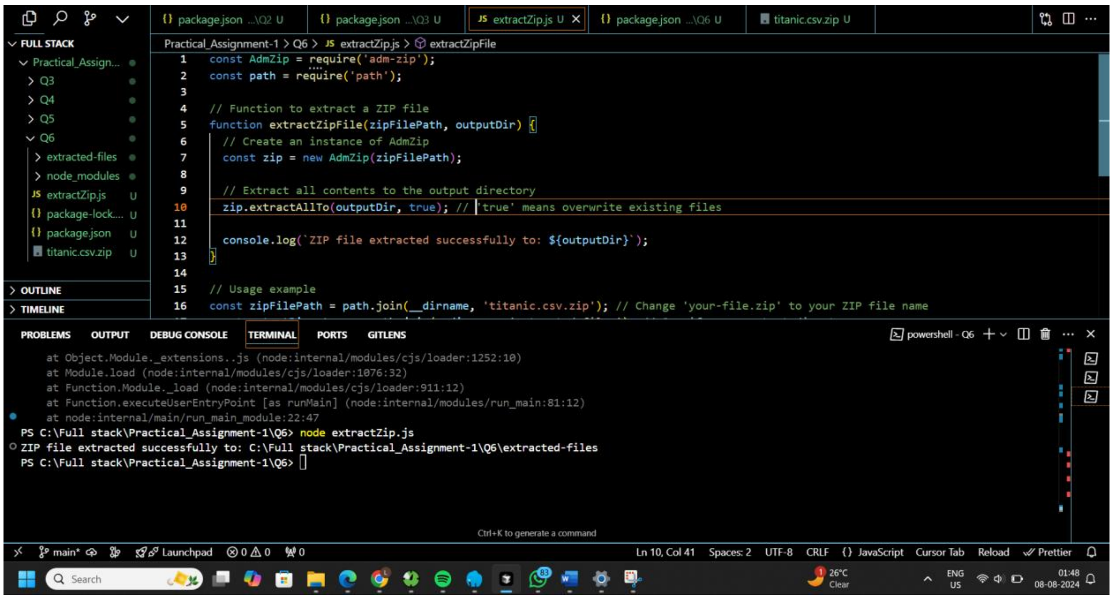
Output:



```
18 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Full stack\Practical_Assignment-1\Q5> node Zip.js
ZIP file created successfully: C:\Full stack\Practical_Assignment-1\Q5\output.zip (22 total bytes)
PS C:\Full stack\Practical_Assignment-1\Q5> 
```

Q6.
extract.js

```
const AdmZip = require('adm-zip');
const path = require('path');
// Function to extract a ZIP filefunction extractZipFile(zipFilePath, outputDir) {
// Create an instance of AdmZip
const zip = new AdmZip(zipFilePath);
// Extract all contents to the output directory
zip.extractAllTo(outputDir, true); // 'true' means overwrite existing files
console.log(`ZIP file extracted successfully to: ${outputDir}`);
}
// Usage example
const zipFilePath = path.join(__dirname, 'titanic.csv.zip'); // Change 'your-file.zip' to your ZIP
file
name
const outputDirectory = path.join(__dirname, 'extracted-files'); // Specify your output
directory
extractZipFile(zipFilePath, outputDirectory)
```



Practical_Assignment-1 > Q6 > extractZip.js > extractZipFile

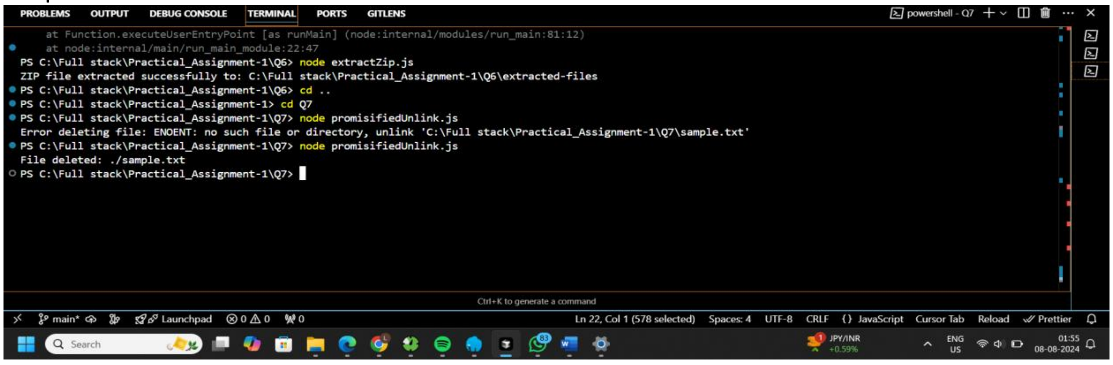
```
const AdmZip = require('adm-zip');
const path = require('path');

// Function to extract a ZIP file
function extractZipFile(zipFilePath, outputDir) {
  // Create an instance of AdmZip
  const zip = new AdmZip(zipFilePath);

  // Extract all contents to the output directory
  zip.extractAllTo(outputDir, true); // 'true' means overwrite existing files

  console.log(`ZIP file extracted successfully to: ${outputDir}`);
}

// Usage example
const zipFilePath = path.join(__dirname, 'titanic.csv.zip'); // Change 'your-file.zip' to your ZIP file name
```

```
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1252:10)
    at Module.load (node:internal/modules/cjs/loader:1076:32)
    at Function.Module._load (node:internal/modules/cjs/loader:911:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:22:47
PS C:\Full stack\Practical_Assignment-1\Q6> node extractZip.js
ZIP file extracted successfully to: C:\Full stack\Practical_Assignment-1\Q6\extracted-files
PS C:\Full stack\Practical_Assignment-1\Q6>
```

Q7

```
const fs = require('fs');
const util = require('util');
// Promisify the fs.unlink function
const unlinkAsync = util.promisify(fs.unlink);
// Function to delete a file using async/await
async function deleteFile(filePath) {
try {
await unlinkAsync(filePath);
console.log(`File deleted: ${filePath}`);
} catch (error) {
console.error(`Error deleting file: ${error.message}`);
}
}
// Usage example
const filePath = './sample.txt'; // Replace with the path to the file you want to delete
// Call the delete function
deleteFile(filePath);
```

output :



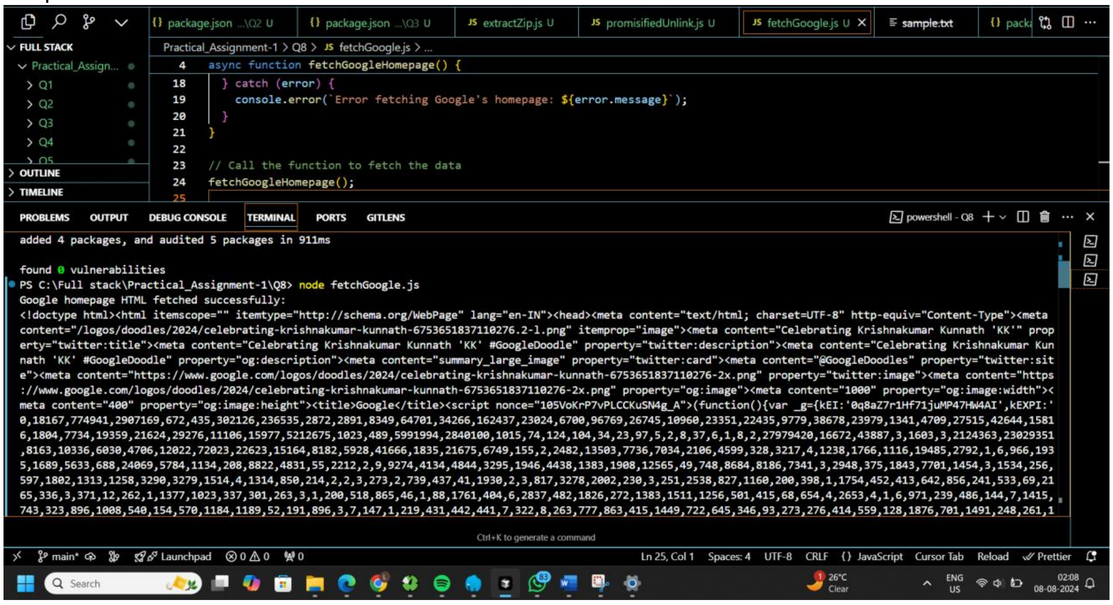
```
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:22:47
PS C:\Full stack\Practical_Assignment-1\Q6> node extractZip.js
ZIP file extracted successfully to: C:\Full stack\Practical_Assignment-1\Q6\extracted-files
PS C:\Full stack\Practical_Assignment-1\Q6> cd ..
PS C:\Full stack\Practical_Assignment-1> cd Q7
PS C:\Full stack\Practical_Assignment-1\Q7> node promisifiedUnlink.js
Error deleting file: ENOENT: no such file or directory, unlink 'C:\Full stack\Practical_Assignment-1\Q7\sample.txt'
PS C:\Full stack\Practical_Assignment-1\Q7> node promisifiedUnlink.js
File deleted: ./sample.txt
PS C:\Full stack\Practical_Assignment-1\Q7>
```

Q8.

fetchGoogle.js :

```
const fetch = require('node-fetch');
// Async function to fetch data from Google's homepage
async function fetchGoogleHomepage() {
try {
const response = await fetch('https://www.google.com');
// Check if the response status is OK (status code 200-299)
if (!response.ok) {
throw new Error(`Network response was not ok. Status code: ${response.status}`);
}
// Get the HTML content from the response
const html = await response.text();
console.log('Google homepage HTML fetched successfully:');
console.log(html);
} catch (error) {
console.error(`Error fetching Google's homepage: ${error.message}`);
}
}
// Call the function to fetch the data
fetchGoogleHomepage();
```

Output:



```
async function fetchGoogleHomepage() {
    } catch (error) {
      console.error(`Error fetching Google's homepage: ${error.message}`);
    }
}

// Call the function to fetch the data
fetchGoogleHomepage();
```

```
added 4 packages, and audited 5 packages in 911ms

found 0 vulnerabilities
PS C:\Full stack\Practical_Assignment-1\Q8> node fetchGoogle.js
Google homepage HTML fetched successfully:
<!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="/logos/doodles/2024/celebrating-krishnakumar-kunnath-6753651837110276.2-1.png" itemprop="image"><meta content="Celebrating Krishnakumar Kunnath 'KK'" prop
erty="twitter:title"><meta content="Celebrating Krishnakumar Kunnath 'KK' #GoogleDoodle" property="twitter:description"><meta content="Celebrating Krishnakumar Kun
nath 'KK' #GoogleDoodle" property="og:description"><meta content="summary_large_image" property="twitter:card"><meta content="@GoogleDoodles" property="twitter:sit
e"><meta content="https://www.google.com/logos/doodles/2024/celebrating-krishnakumar-kunnath-6753651837110276-2x.png" property="twitter:image"><meta content="https
://www.google.com/logos/doodles/2024/celebrating-krishnakumar-kunnath-6753651837110276-2x.png" property="og:image"><meta content="1000" property="og:image:width"><
meta content="400" property="og:image:height"><title>Google</title><script nonce="105VoKrP7vPLCCKuSN4g_A">(function(){var _g={kEI:'0q8aZ7r1Hf71juMP47HW4AI',kEXPI:'
0,18167,774941,2907169,672,435,302126,236535,2872,2891,8349,64701,34266,162437,23024,6700,96769,26745,10960,23351,22435,9779,38678,23979,1341,4709,27515,42644,1581
6,1804,7734,19359,21624,29276,11106,15977,5212675,1023,489,5991994,2840100,1015,74,124,104,34,23,97,5,2,8,37,6,1,8,2,27979420,16672,43887,3,1603,3,2124363,23029351
,8163,10336,6030,4706,12022,72023,22623,15164,8182,5928,41666,1835,21675,6749,155,2,2482,13503,7736,7034,2106,4599,328,3217,4,1238,1766,1116,19485,2792,1,6,966,193
5,1689,5633,688,24069,5784,1134,208,8822,4831,55,2212,2,9,9274,4134,4844,3295,1946,4438,1383,1908,12565,49,748,8684,8186,7341,3,2948,375,1843,7701,1454,3,1534,256,
597,1802,1313,1258,3290,3279,1514,4,1314,850,214,2,2,3,273,2,739,437,41,1930,2,3,817,3278,2002,230,3,251,2538,827,1160,200,398,1,1754,452,413,642,856,241,533,69,21
65,336,3,371,12,262,1,1377,1023,337,301,263,3,1,200,518,865,46,1,88,1761,404,6,2837,482,1826,272,1383,1511,1256,501,415,68,654,4,2653,4,1,6,971,239,486,144,7,1415,
743,323,896,1008,540,154,570,1184,1189,52,191,896,3,7,147,1,219,431,442,441,7,322,8,263,777,863,415,1449,722,645,346,93,273,276,414,559,128,1876,701,1491,248,261,1
```

Q9.
employee.js

```
const mysql = require('mysql2/promise');
// Function to connect to the MySQL database and perform operations
async function manageEmployees() {
try {
// Create a connection to the database
const connection = await mysql.createConnection({
host: '127.0.0.1',
user: 'root', // Replace with your MySQL username
password: '', // Replace with your MySQL password
database: 'testdb' // Replace with your database name
});
console.log('Connected to the MySQL database.');
// Insert a new record into the employee tableconst insertQuery = `INSERT INTO employee
(name, position, salary) VALUES (?, ?, ?)`;
const [insertResult] = await connection.execute(insertQuery, ['John Doe', 'Software
Engineer',
75000]);
console.log('Record inserted:', insertResult);
// Select and display all records from the employee table
const [rows] = await connection.execute('SELECT * FROM employee');
console.log('All employee records:');
console.table(rows);
// Close the database connection
await connection.end();
console.log('Connection closed.');
} catch (error) {
console.error('Error connecting to MySQL or executing query:', error);
}
}
// Call the function to manage employee data
manageEmployees();
```

Q11.
index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Live Cricket Score</title>
<style>
body {
font-family: Arial, sans-serif;
text-align: center;
margin: 50px;
}
#score {
```

```
font-size: 24px;
margin-top: 20px;
}
</style>
</head>
<body>
<h1>Live Cricket Score</h1>
<div id="score">Fetching score...</div>
<script>
async function fetchLiveScore() {
try {
const response = await fetch('/live-score');
const data = await response.json();
// Update the score on the page
document.getElementById('score').innerText = `Score: ${data.score}`;} catch (error) {
document.getElementById('score').innerText = 'Error fetching live score';
}
}
// Fetch the live score every 30 seconds
fetchLiveScore();
setInterval(fetchLiveScore, 30000);
</script>
</body>
</html>
```

Server.js

```
const express = require('express');
const axios = require('axios');
const app = express();
const PORT = 3000;
// Your RapidAPI host and key (replace these with your actual API key and host from
RapidAPI)
const API_HOST = 'free-cricket-live-score1.p.rapidapi.com'; // Example API host
const API_KEY = 'eb3767b9ecmshe0d7acdf8126ab7p12b807jsn1cfed355062d'; // Replace
with your actual API key
// Set up the route to fetch the live cricket score
app.get('/live-score', async (req, res) => {
try {
const response = await axios.get(`https://${API_HOST}/match`, {
params: { id: '66754aac6c794634996ed39c' }, // Replace 'match_id' with the actual match ID
you
want to track
headers: {
'x-rapidapi-host': 'free-cricket-live-score1.p.rapidapi.com',
'x-rapidapi-key': 'eb3767b9ecmshe0d7acdf8126ab7p12b807jsn1cfed355062d'
}});
const matchData = response.data;
res.json(matchData);
} catch (error) {
console.error('Error fetching live cricket score:', error);
res.status(500).send('Error fetching live cricket score');
}
});
// Serve an HTML page to display the score
app.get('/', (req, res) => {
res.sendFile(__dirname + '/index.html');
});
// Start the server
app.listen(PORT, () => {
console.log(`Server running on http://localhost:${PORT}`);
});
```

server.js

```
// server.js
const http = require('http');
const server = http.createServer((req, res) => {
res.statusCode = 200;
res.setHeader('Content-Type', 'text/plain');res.end('Hello, Node.js Server!\n');
});
const PORT = 3000;
server.listen(PORT, () => {
console.log(`Server running at http://localhost:${PORT}/`);
});
```

Package.json

```
{
"name": "node-app",
"version": "1.0.0",
"description": "",
"main": "server.js",
"scripts": {
"start": "node server.js",
"test": "echo \"Running tests...\" && exit 0",
"user-script1": "echo 'This is user-defined script 1!'",
"user-script2": "echo 'This is user-defined script 2!'",
"user-script3": "echo 'This is user-defined script 3!'",
"dev": "nodemon server.js"
},
"devDependencies": {
"nodemon": "^2.0.22"
},
"author": "",
"license": "ISC"
}
```