

Que(1) Develop a web server with following functionalities:

- Serve static resources.
- Handle GET request.
- Handle POST request.

Ans:-

Server.js

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

const port = 8000;

// Helper function to serve static files
function serveStaticFile(res, filename, contentType) {
  fs.readFile(filename, (err, data) => {
    if (err) {
      res.writeHead(500, { 'Content-Type': 'text/plain' });
      res.end('Internal Server Error');
    } else {
      res.writeHead(200, { 'Content-Type': contentType });
      res.end(data);
    }
  });
}

// Create the server
const server = http.createServer((req, res) => {
  if (req.method === 'GET') {
    if (req.url === '/') {
      // Serve the index.html file
      serveStaticFile(res, './index.html', 'text/html');
    }
  } else if (req.method === 'POST') {
    if (req.url === '/') {
      let body = '';
      req.on('data', (chunk) => {
        body += chunk;
      });

      req.on('end', () => {
        // Do something with the request body
        res.writeHead(200, { 'Content-Type': 'text/plain' });
      });
    }
  }
});
```

```

        res.end('Hello, this is a POST request with body: ' + body);
    });
} else {
    // Handle 404 Not Found
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('404 Not Found');
}
} else {
    // Handle other HTTP methods
    res.writeHead(101, { 'Content-Type': 'text/plain' });
    res.end('101 Not Implemented');
}
});

// Start the server
server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});

```

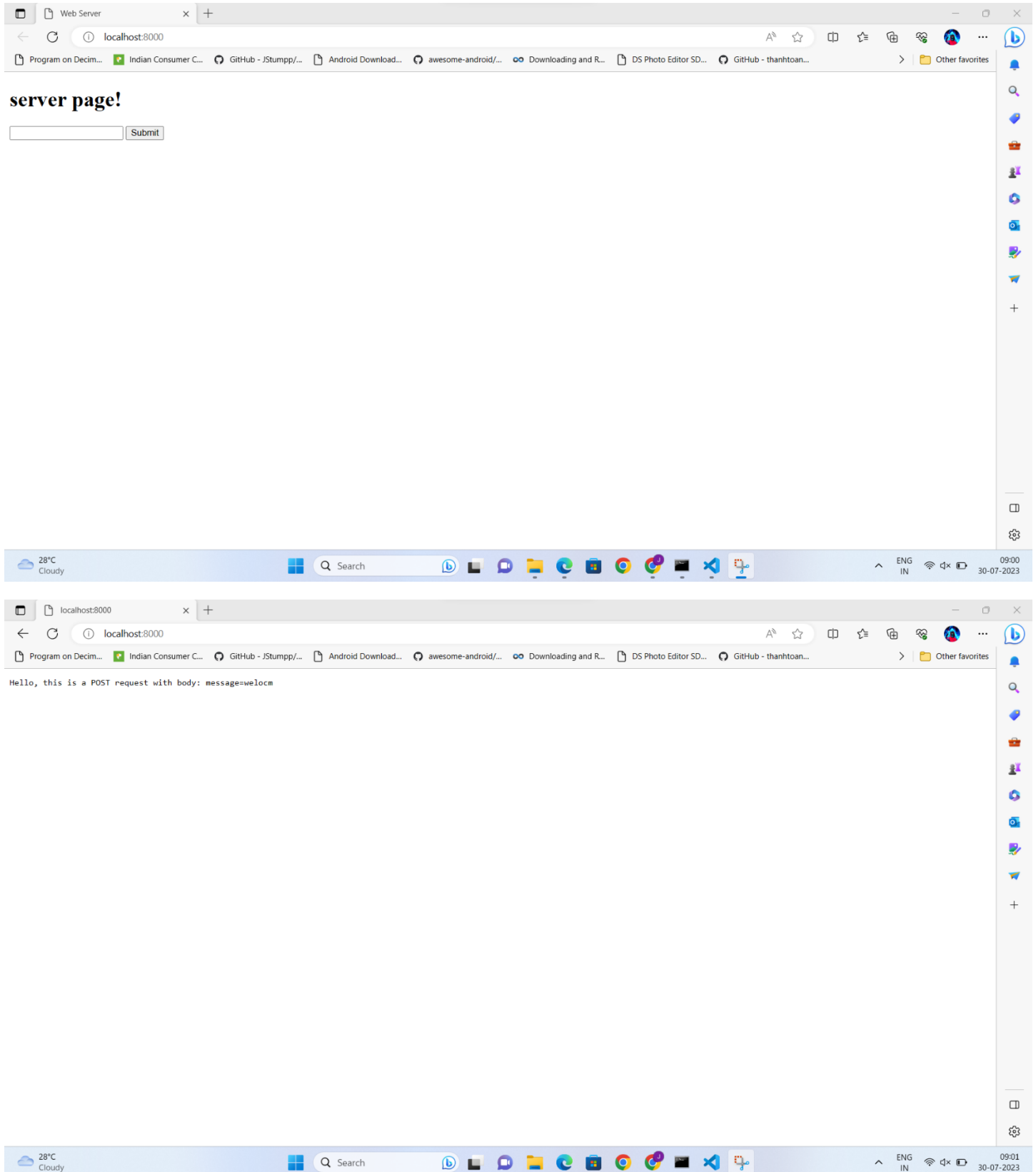
Index.js

```

<!DOCTYPE html>
<html>
<head>
    <title>Web Server</title>
</head>
<body>
    <h1> server page!</h1>
    <!-- <p>This is a static HTML file served by the web server.</p> -->
    <form action="/" method="post">
        <input type="text" name="message" />
        <button type="submit">Submit</button>
    </form>
</body>
</html>

```

Output:-



Que(2) 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.)

Ans:-

Que2.js

```
const express = require("express")
const app = express()
const fs = require("fs")

app.get("/", (req, res) => {
  fs.readFile("./index2_1.html", "utf8", (err, data) => {
    res.send(data)
  })
})

app.get("/gethello", (req, res) => {
  fs.readFile("./index2.html", "utf8", (err, data) => {
    res.send(data)
  })
  // res.send(`<h1>hello nodeJs!!!</h1>`)
})

app.listen(3000)
```

index2.html

```
<!-- Get Complete Source Code from Pabbly.com -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <h1>Hello nodejs!</h1>
</body>
</html>
```

Index2_1.html

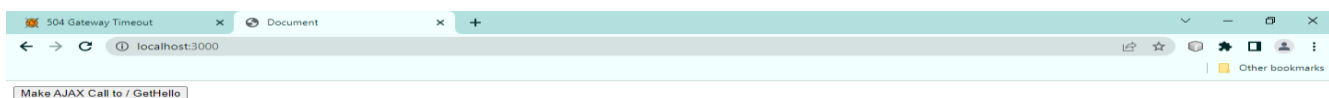
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

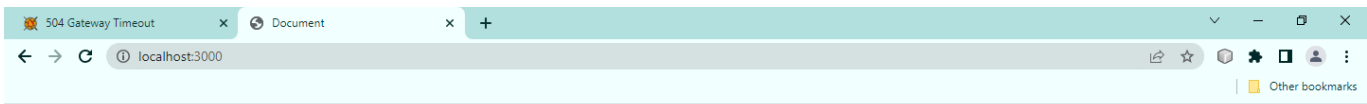
  <button id="ajaxButton">Make AJAX Call to / GetHello</button>
  <div id="ajaxResponse"></div>

</body>
<script>

document.getElementById("ajaxButton").addEventListener("click", () => {
  fetch("/gethello").then(response=>response.text()).then(data => {
    document.getElementById("ajaxResponse").innerHTML = data;
  }).catch(error => {
    console.error(error)
  });
});
</script>
</html>
```

Output:-





Hello nodejs!



Que(3) Develop a module for domain specific chatbot and use it in a command line application.

Ans:- chatBot.js

```
module.exports.reply = function (msg) {  
  this.Bot_Age = 10;  
  this.Bot_Name = "abc";  
  
  msg = msg.toLowerCase();  
  
  if(msg.indexOf("hi")>-1)  
  {  
    return "Hello..!!";  
  }  
  
  else if(msg.indexOf("age") > -1 &&  
    msg.indexOf("your"))  
  {  
    return "I'm " + this.Bot_Age;  
  }  
  return "Sorry, I didn't get it... ";  
}
```

```
}

```

Server.js

```
var Chatbot = require('./chatBot');
var readline = require('readline');

var r1 = readline.createInterface(process.stdin, process.stdout);
r1.setPrompt("You==>");
r1.prompt();
r1.on('line', function(message) {
    console.log('Bot ==> ' + Chatbot.reply(message));
    r1.prompt();
}).on('close',function(){ //chaining events.
    process.exit(0);
});
```

Output:-



```
admin@DESKTOP-HRB2IBM MINGW64 /b/Practical Assignment-1
$ cd Que3

admin@DESKTOP-HRB2IBM MINGW64 /b/Practical Assignment-1/Que3
$ node server.js
You==>hi
Bot ==> Hello..!!
You==>age
Bot ==> I'm 10
You==>gvd
Bot ==> Sorry, I didn't get it...
You==>
```

Que(4) Use above chatbot module in web based chatting of websocket.

Ans:-

Websocket.js

```
const WebSocket = require('ws')
var http = require('http');
var url = require('url');
var Chatbot = require('../Que3/chatbot.js');

var st = require('node-static');

var fileServer = new st.Server('../public');

var httpserver = http.createServer(function(request, response)
{
```

```

    request.on('end', function () {
      var get = url.parse(request.url, true).query;
      fileServer.serve(request, response);
    }).resume();

  }).listen(8080, function() {
    console.log((new Date()) +
      ' Server is listening on port 8080');
  });

//WebSocket.Server({server: httpserver})
const wss = new WebSocket.Server({ server: httpserver });

wss.on('connection', function(ws) {
  ws.send('Hello client')

  ws.on('message', message => {
    console.log(`Received message => ${message}`)
    // console.log(Chatbot.ChatbotReply(message))
    ws.send(Chatbot.reply(message))
  })
})
})

```

Public file / index.html

```

<!DOCTYPE html >
<html>
  <body>
<script language="javascript">
var ws = new WebSocket('ws://localhost:8080');
ws.addEventListener("message", function(e) {
  var msg = e.data;
  document.getElementById('chatlog').innerHTML+= '<br>Server: ' + msg;
});
function sendMessage(){
  var message = document.getElementById('message').value;
  document.getElementById('chatlog').innerHTML+= '<br> Me: ' + message;
  ws.send(message);
  document.getElementById('message').value="";
}
</script>
<h2>Data from server</h2>
  <div id="chatlog"></div>
</hr>

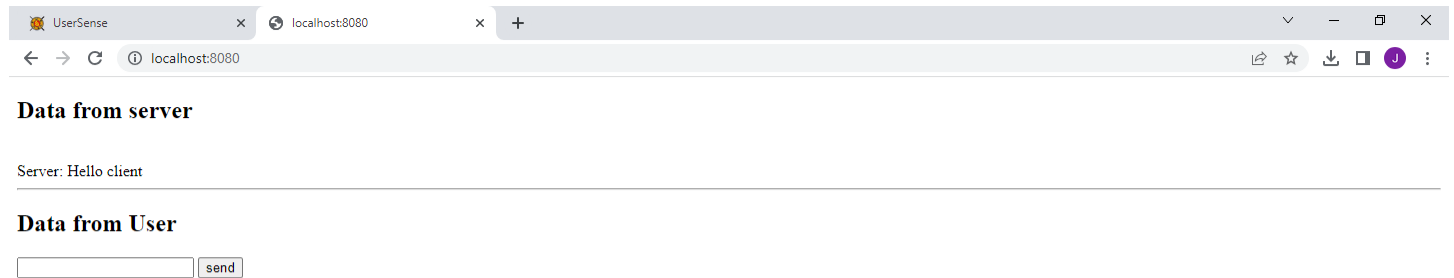
```

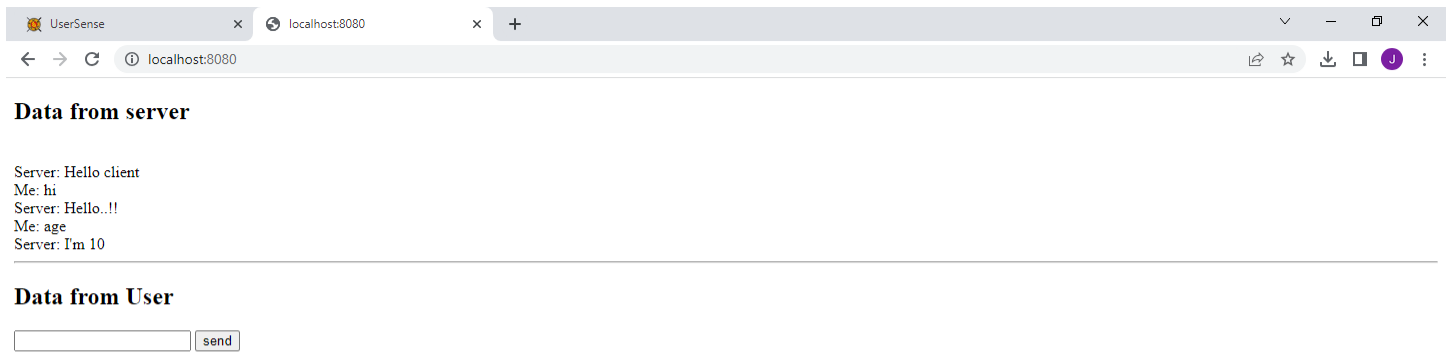


```
<h2>Data from User</h2>
  <input type="text" id="message" />
    <input type="button" id="b1" onclick="sendMessage()" value="send" />

  </body>
</html>
```

OutPut:-





Que(5) Write a program to create a compressed zip file for a folder.

Ans:-

Zip.js

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

function compressFile(sourcePath, zipPath) {
  const readStream = fs.createReadStream(sourcePath);
  const writeStream = fs.createWriteStream(zipPath);

  readStream.pipe(zlib.createGzip()).pipe(writeStream);

  writeStream.on('finish', () => {
    console.log('File compressed successfully:', zipPath);
  });

  writeStream.on('error', (error) => {
    console.error('Error writing the ZIP file:', error);
  });
}
```

```

}

// Usage example:
const sourceFile = './jenish.txt'; // Replace with the path of the file you want to compress
const zipFile = './f.txt.gz'; // Replace with the desired destination for the ZIP file

compressFile(sourceFile, zipFile);

```

Output:-

```

// zip.js
1 const fs = require('fs');
2 const zlib = require('zlib');
3
4 function compressFile(sourcePath, zipPath) {
5   const readStream = fs.createReadStream(sourcePath);
6   const writeStream = fs.createWriteStream(zipPath);
7
8   readStream.pipe(zlib.createGzip()).pipe(writeStream);
9
10  writeStream.on('finish', () => {
11    console.log('File compressed successfully:', zipPath);
12  });
13
14  writeStream.on('error', (error) => {
15    console.error('Error writing the ZIP file:', error);
16  });
17 }
18
19 // Usage example:
20 const sourceFile = './jenish.txt'; // Replace with the path of the file you want to compress
21 const zipFile = './f.txt.gz'; // Replace with the desired destination for the ZIP file
22
23 compressFile(sourceFile, zipFile);
24

```

```

PS D:\ICT_3\ICT3\Practical Assignment-1> cd Que5
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5> cd..
PS D:\ICT_3\ICT3\Practical Assignment-1> cd Que5
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5> node zip.js
File compressed successfully: ./f.txt.gz
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5>

```

Que(6) Write a program to promisify fs.unlink function and call it.

Ans:-

Unlink.js

```

var fs = require('fs/promises')

function readFile(fpath)
{
  return new Promise(function(success,fail)
  {

```

```

    fs.unlink(fpath,(err,data) =>
    {
        if(err)
            fail(err)
        else
            success(data)
    })
})
}

readFile('./text1.txt').then((data)=>{
    console.log(data)
}).catch((err)=>{
    console.log(err)
})

```

Output:-

The screenshot displays the Visual Studio Code interface. On the left, the Explorer pane shows a project structure for 'PRACTICAL ASSIGNMENT-1' with folders 'node_modules', 'public', and 'Que1' through 'Que8'. Files include 'jenish.rar', 'txet.txt', 'unzip.js', 'text1.txt', 'unlink.js', 'app.js', 'package-lock.json', and 'package.json'. The central editor pane shows 'text1.txt' containing the text 'Welcome to my new home..'. The bottom terminal pane shows a PowerShell session with the following commands and output:

```

Node.js v18.17.0
PS D:\ICT_3\ICT3\Practical Assignment-1\Que1>
PS D:\ICT_3\ICT3\Practical Assignment-1\Que7> cd Que7
PS D:\ICT_3\ICT3\Practical Assignment-1\Que7> node unlink.js

```

The terminal output indicates a 'MODULE_NOT_FOUND' error, suggesting that the 'unlink.js' file is not found in the current directory or its dependencies are not resolved.

```

1  var fs = require('fs/promises')
2
3  function readFile(fpath)
4  {
5      return new Promise(function(success, fail)
6      {
7          fs.unlink(fpath, (err, data) =>
8          {
9              if(err)
10                 fail(err)
11             else
12                 success(data)
13             })
14         })
15     })
16
17     readFile('./text1.txt').then((data)=>{
18         console.log(data)
19     }).catch((err)=>{
20         console.log(err)
21     })
22
23
24

```

```

at node:internal/main/run_main_module:23:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v18.17.0
PS D:\ICT_3\ICT3\Practical Assignment-1\Que1>
* History restored
PS D:\ICT_3\ICT3\Practical Assignment-1> cd Que7
PS D:\ICT_3\ICT3\Practical Assignment-1\Que7> node unlink.js
PS D:\ICT_3\ICT3\Practical Assignment-1\Que7> node unlink.js
PS D:\ICT_3\ICT3\Practical Assignment-1\Que7>

```

Que(7):- Write a program to extract a zip file.

Ans:-

Unzip.js

```

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

function decompressZlib(inputFilePath, outputFilePath) {
    const compressedData = fs.readFileSync(inputFilePath);
    zlib.unzip(compressedData, (error, decompressedData) => {
        if (error) {
            console.error('Error decompressing data:', error);
        } else {
            fs.writeFileSync(outputFilePath, decompressedData);
            console.log('Data successfully decompressed and saved to:', outputFilePath);
        }
    });
}

// Usage example:
const compressedFilePath = './f.txt.gz'; // Replace with the path of the zlib-compressed file
const decompressedFilePath = './f.txt.ungz'; // Replace with the desired output file path

decompressZlib(compressedFilePath, decompressedFilePath);

```

OutPut:-

```

1  const fs = require('fs');
2  const zlib = require('zlib');
3
4  function compressFile(sourcePath, zipPath) {
5      const readStream = fs.createReadStream(sourcePath);
6      const writeStream = fs.createWriteStream(zipPath);
7
8      readStream.pipe(zlib.createGzip()).pipe(writeStream);
9
10     writeStream.on('finish', () => {
11         console.log('File compressed successfully:', zipPath);
12     });
13
14     writeStream.on('error', (error) => {
15         console.error('Error writing the ZIP file:', error);
16     });
17 }
18
19 // Usage example:
20 const sourceFile = './jenish.txt'; // Replace with the path of the file you want to compress
21 const zipFile = './f.txt.gz'; // Replace with the desired destination for the ZIP file
22
23 compressFile(sourceFile, zipFile);
24

```

```

PS D:\ICT_3\ICT3\Practical Assignment-1> cd Que5
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5> cd..
PS D:\ICT_3\ICT3\Practical Assignment-1> cd Que5
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5> node zip.js
File compressed successfully: ./f.txt.gz
PS D:\ICT_3\ICT3\Practical Assignment-1\Que5>

```

Que(8) Fetch data of google page using node-fetch using async-await model.

Ans:-

Fetch.js

```

async function fetchDataFromGooglePage() {
  try {
    const fetch = await import('node-fetch');
    const url = 'https://www.google.com';
    const response = await fetch.default(url);

    if (!response.ok) {
      throw new Error('Network response was not ok');
    }

    const data = await response.text();
    console.log(data);
  } catch (error) {
    console.error('Error fetching data:', error);
  }
}

```

```
}

fetchDataFromGooglePage();
```

OutPut:-

```

1
2 async function fetchDataFromGooglePage() {
3   try {
4     const fetch = await import('node-fetch');
5     const url = 'https://www.google.com';
6     const response = await fetch.default(url);
7
8     if (!response.ok) {
9       throw new Error('Network response was not ok');
10    }
11
12    const data = await response.text();
13    console.log(data);
14  } catch (error) {
15    console.error('Error fetching data:', error);
16  }
17 }
18
19 fetchDataFromGooglePage();
20
21

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS D:\ICT 3\ICT3\Practical Assignment-1\Que8> node fetch.js
<!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 co
ntent="/images/branding/google/1x/google_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="ffRPxt1iits_j3ielsqRdg">(function(){var _g=
{KEI: 'Z_XFZN7RDLQkhhIPzWm4Ag', kEXPI: '0,18168,775176,566065,6058,207,4804,2316,383,246,5,1129120,1748,13,3,1196029,601,380097,16114,28684,22431,1361,12312,17587,4998
,17075,35733,2711,887,1985,2891,4139,8221,60690,2614,3783,9708,230,1014,1,16916,2652,4,1528,2304,29062,13064,11443,2216,4437,22556,6681,7596,1,42154,2,16737,23024,56
79,1021,31122,4568,6255,23421,1252,5835,19300,8,7476,445,2,2,1,26632,8155,7381,2,15967,873,19634,7,1922,9779,5864,14776,2259,19560,20199,20136,14,82,7651,5682,6873,8
377,3787,4264,10937,550,3307,1518,3030,6111,9705,1212,592,7734,2738,2276,608,450,554,2,18735,11001,4616,7951,892,4653,347,603,543,3148,2755,5810,7914,4951,818,3626,7
048,440,1271,2041,372,1179,774,5209601,2,42,87,8797814,3311,141,795,19736,1,346,6639,502,131,204,30,3,3,14,12,4,47,85,4,12,105,33,10,7,47,45,16,23942531,579,2861448,
1182880,16672,32950,4626,520,570,724,1400550,342431,23416839,3601,255,2962,1345,3614,6722,148,1679,881,1083,836,265,696,334,963,99,198,589,500,1015,1450,1708,77,1,13
99,960,992,3,1275,256,1492,476,1605,6,452,410,32,354,70,136,585,620,60,118,1734,21,888,4,900,12,348,2333,856,1203,5,880,100,401,11,1668,71,29,766,779,441,109,132,446
,45,53,262,2568,11,67,733,670,259,98,2,361,620,285,785,782,48,3,793,36,2,722,181,2,240,2866,6,53,14,510,611,135,116,8,104,233,4,4,69,154,234,247,539,10,5,16,3,146,30
8,676,404,213,487,28,93,386,141,227,1475,431,3,5,271,142,547,786,305,171,123,81,3291', kbl: '1cU9', kOP1: 89978449});(function(){var a;(null==(a=window.google)?0:a.stvsc
?google.kei=g.kei:window.google=g;)).call(this);})();(function(){google.sn='webhp';google.khl='en-IN';})();(function(){

```

Que(9) Write a program that connect Mysql database, Insert a record in employee table and display all records in employee table using promise based approach.

Ans:-

Database.js

```

const mysql = require('nodejs-mysql').default;

const config = {
  host : "localhost",
  user : "root",
  password : "root",
  database : "employee_db"
}

```

```

const db = mysql.getInstance(config);

db.connect()
  .then(function(){
    console.log("Connected!!");

    var sql = "INSERT INTO employee (username, password, firstname, lastname, email) VALUES ('abc', 'xx', 'abc1', 'a', 'a46884@gmail.com')";

    return db.exec(sql);
  }).then(function(res){
    console.log(res);
    return db.exec("SELECT * FROM employee");
  }).then(function(result){
    for( var i in result){
      console.log("Username: ", result[i].username + " " + "Password: " + result[i].password);
      process.exit(0);
    }
  }).catch(function(err){
    console.log("ERROR: ", err);
    process.exit(0);
  });

```

Que(10) Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs application.

Ans:-

Server.js

```

const http = require('http');

const PORT = 3000;

const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, world!');
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
  }
});

```



```

    }
  });

server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});

```

Script1.js

```

// User-defined script 1
console.log('Running user-defined script 1');

```

Script2.js

```

// User-defined script 2
console.log('Running user-defined script 2');

```

Script3.js

```

// User-defined script 3
console.log('Running user-defined script 3');

```

Output:-

The screenshot shows a Visual Studio Code editor with a project named 'Practical Assignment-1'. The Explorer sidebar on the left shows a file structure with folders 'node_modules' and 'public', and files 'script1.js', 'script2.js', 'script3.js', and 'server.js'. The main editor displays the content of 'server.js', which is a Node.js server script. It imports the 'http' module, sets a port to 3000, and creates a server that listens on that port. The server has a route for '/' that returns 'Hello, world!' with a 200 status code, and a default route that returns 'Not Found' with a 404 status code. The server logs the message 'Server is running on port 3000' when it starts. The Output window at the bottom shows the terminal output of running the server with 'node server.js'. The output shows the server starting and logging the message 'Server is running on port 3000'. The terminal also shows the command 'npm run script1' being executed, which runs 'script1.js' and outputs 'Running user-defined script 1'. The terminal also shows the command 'npm run script2' being executed, which runs 'script2.js' and outputs 'Running user-defined script 2'. The terminal also shows the command 'npm run script3' being executed, which runs 'script3.js' and outputs 'Running user-defined script 3'.

```

1  const http = require('http');
2
3  const PORT = 3000;
4
5  const server = http.createServer((req, res) => {
6    if (req.url === '/') {
7      res.writeHead(200, { 'Content-Type': 'text/plain' });
8      res.end('Hello, world!');
9    } else {
10     res.writeHead(404, { 'Content-Type': 'text/plain' });
11     res.end('Not Found');
12   }
13 });
14
15 server.listen(PORT, () => {
16   console.log(`Server is running on port ${PORT}`);
17 });
18

```

```

Did you mean this?
  npm run script1 # run the "script1" package script

To see a list of supported npm commands, run:
  npm help

PS D:\ICT_3\ICT3\Practical Assignment-1\Que10> node script1.js
Running user-defined script 1
PS D:\ICT_3\ICT3\Practical Assignment-1\Que10> node script2.js
Running user-defined script 2
PS D:\ICT_3\ICT3\Practical Assignment-1\Que10> node script3.js
Running user-defined script 3
PS D:\ICT_3\ICT3\Practical Assignment-1\Que10>

```

Que(11) Develop an application to show live cricket score.

Ans:-

Live.js

```
const request = require('request-promise');

async function getLiveCricketScores() {
  try {
    const apiKey = '13cf787f-72cd-41ca-9e2f-3d711cb26c6e'; // Replace this with your actual
    API key
    const apiUrl = `https://cricapi.com/api/matches?apikey=${apiKey}`;

    const response = await request(apiUrl, { json: true });

    if (response.error) {
      throw new Error(response.error);
    }

    const matches = response.matches;
    if (!matches || matches.length === 0) {
      console.log('No live matches found.');
      return;
    }

    console.log('Live Cricket Scores:');
    console.log('-----');
    matches.forEach((match) => {
      const { team1, team2, score } = match;
      console.log(`${team1} vs ${team2}: ${score}`);
    });
  } catch (error) {
    console.error('Error:', error.message);
  }
}

getLiveCricketScores();
```

output:-

The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'PRACTICAL ASSIGNMENT-1' with a file tree including 'node_modules', 'public', and several 'Que' files. The main editor area shows the 'live.js' file with the following code:

```

1  const request = require('request-promise');
2
3  async function getLiveCricketScores() {
4      try {
5          const apiKey = '13cf787f-72cd-41ca-9e2f-3d711cb26c6e'; // Replace this with your actual API key
6          const apiUrl = `https://cricapi.com/api/matches?apikey=${apiKey}`;
7
8          const response = await request(apiUrl, { json: true });
9
10         if (response.error) {
11             throw new Error(response.error);
12         }
13
14         const matches = response.matches;
15         if (!matches || matches.length === 0) {
16             console.log('No live matches found.');

The terminal window at the bottom shows the following output:



```

Error: Cannot find module 'D:\ICT_3\ICT3\Practical Assignment-1\Que11\Que11'
at Module._resolveFilename (node:internal/modules/cjs/loader:1077:15)
at Module._load (node:internal/modules/cjs/loader:922:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
at node:internal/main/run_main_module:23:47 {
 code: 'MODULE_NOT_FOUND',
 requireStack: []
}

Node.js v18.17.0
PS D:\ICT_3\ICT3\Practical Assignment-1\Que11> node live.js
No live matches found.
PS D:\ICT_3\ICT3\Practical Assignment-1\Que11> .

```



The status bar at the bottom indicates the file is 'live.js' at line 1, column 26, with 15 characters selected. The system tray shows the date as 30-07-2023 and time as 11:55.


```