Name :- Jariwala Dhruvin Sanjaybhai

Roll No :- 27

Sem :- Ict 3

Subject :- Open Source Web Development

1. Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.
- Handle POST request.

Code :-

server.js

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

const port = 5000;

const server = http.createServer((req, res) => {
   const parsedUrl = url.parse(req.url);
   const pathname = parsedUrl.pathname;


   if (pathname === '/' || pathname.endsWith('.html') || pathname.endsWith('.css') ||
pathname.endsWith('.js')) {
      let filepath = pathname === '/' ? '/index.html' : pathname;
      fs.readFile(__dirname + filepath, (err, data) => {
         if (err) {
            res.statusCode = 404;
            res.end('Resource not found');
         } else {
            res.statusCode = 200;
            res.end(data);
```

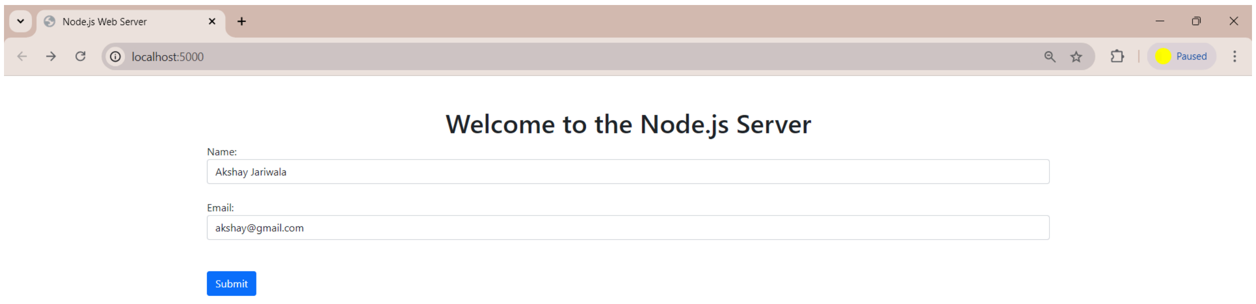
```
        }
      });
    } else if (pathname === '/get') {
      const queryParams = querystring.parse(parsedUrl.query);
      res.statusCode = 200;
      res.setHeader('Content-Type', 'application/json');
      res.end(JSON.stringify(queryParams));
    } else if (pathname === '/post' && req.method === 'POST') {
      let body = '';
      req.on('data', chunk => {
        body += chunk.toString();
      });
      req.on('end', () => {
        const parsedBody = querystring.parse(body);
        res.statusCode = 200;
        res.setHeader('Content-Type', 'application/json');
        res.end(JSON.stringify(parsedBody));
      });
    } else {
      res.statusCode = 404;
      res.end('Route not found');
    }
});

server.listen(port, () => {
  console.log(`Server running at http://localhost:${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>Node.js Web Server</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpu
uCOmLASjC" crossorigin="anonymous">
</head>

<body><br><br>
```

```
    <h1 class="text-center">Welcome to the Node.js Server</h1>
    <div class="container">
        <form action="/post" method="post">
            <label for="name">Name:</label>
            <input type="text" class="form-control" placeholder="Enter Your Name"
id="name" name="name"><br>
            <label for="name">Email:</label>
            <input type="email" class="form-control" placeholder="Enter Your Email"
id="name" name="email"><br><br>
            <button class="btn btn-primary" type="submit">Submit</button>
        </form>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/t
WtIaxVXM" crossorigin="anonymous"></script>
</body>

</html>
```

Output :-

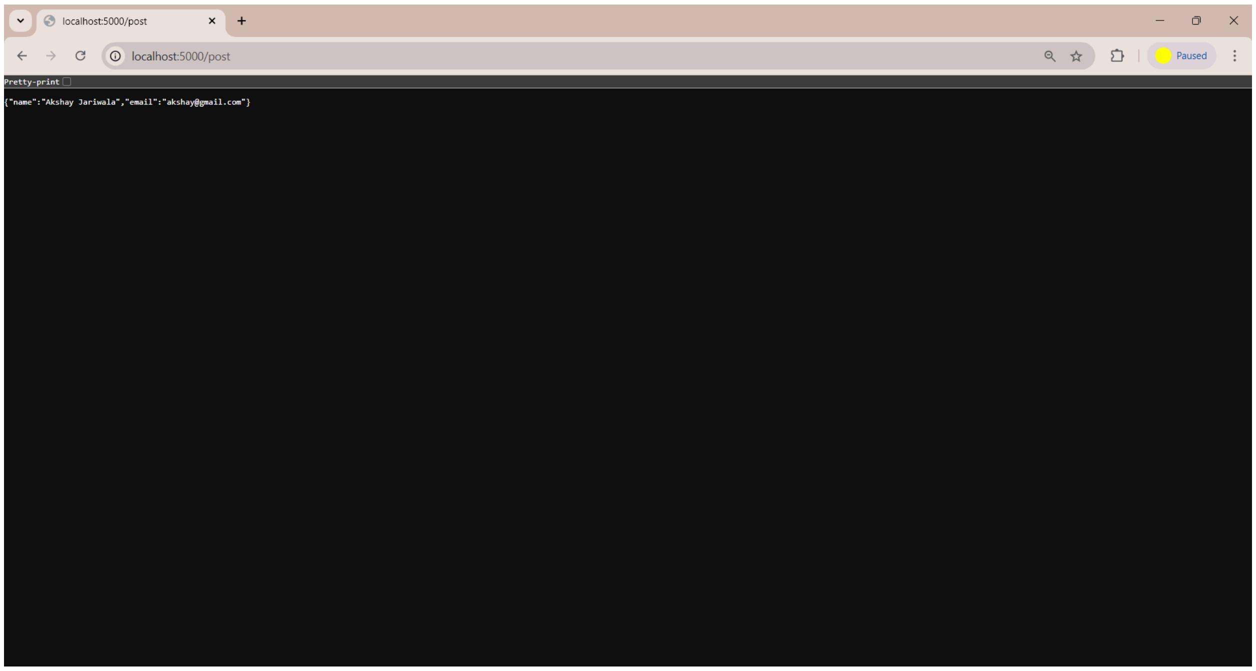


Welcome to the Node.js Server

Name:
Akshay Jariwala

Email:
akshay@gmail.com

Submit



```
{"name":"Akshay Jariwala","email":"akshay@gmail.com"}
```

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.)

Code :-

app.js

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

const app = express();

const path = require('path');

app.get('/', (req, res) => {

   res.sendFile(path.join(__dirname, 'p1.html'));
});


app.get('/gethello', (req, res) => {
   res.send('Hello Node');
});
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
   console.log(`Server is running on http://localhost:${PORT}`);
});
```

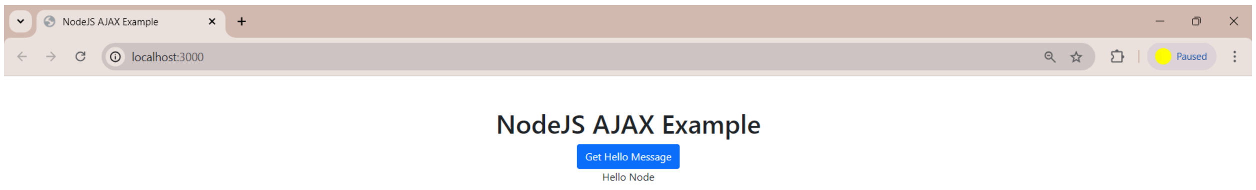
p1.html

```
<html lang="en">


<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>NodeJS AJAX Example</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohh
puuCOmLASjC" crossorigin="anonymous">
</head>
```

```
<body>

  <h1>NodeJS AJAX Example</h1>

  <button id="getHelloButton">Get Hello Message</button>

  <p id="helloMessage"></p>


  <script>

    document.getElementById('getHelloButton').addEventListener('click', function() {

      const xhr = new XMLHttpRequest();

      xhr.open('GET', '/gethello', true);

      xhr.send();

      xhr.onload = function() {

        if (xhr.status != 200) {

          alert(`Error ${xhr.status}: ${xhr.statusText}`);

        } else {

          document.getElementById('helloMessage').innerText = xhr.responseText;

        }

      };


      xhr.onerror = function() {

        alert("Request failed");
```

```
            };

        });

    </script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcX
n/tWtlaxVXM" crossorigin="anonymous"></script>

</body>



</html>
```

Output :-



NodeJS AJAX Example

Hello Node

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

Code :-

chat.js

```
class chat {
    constructor() {
        this.responses = {
            greeting: "Hello! How can I assist you today?",
            help: "I'm here to help with chatbot. You can ask me about software
installation, troubleshooting, and more.",
            software_installation: "To install software, please download the installer
from the official website and follow the on-screen instructions.",
            troubleshooting: "Can you describe the issue you're facing? I'll do my
best to assist you.",
            goodbye: "Thank you for using our chatbot. Have a great day!",
        };
    }


    getResponse(message) {
        const lowerCaseMessage = message.toLowerCase();
        if (lowerCaseMessage.includes("hello") || lowerCaseMessage.includes("hi"))
{
```

```
            return this.responses.greeting;

        } else if (lowerCaseMessage.includes("help")) {

            return this.responses.help;

        } else if (lowerCaseMessage.includes("install software")) {

            return this.responses.software_installation;

        } else if (lowerCaseMessage.includes("troubleshoot")) {

            return this.responses.troubleshooting;

        } else if (lowerCaseMessage.includes("bye")) {

            return this.responses.goodbye;

        } else {

            return "I'm sorry, I didn't understand that. Can you please rephrase?";

        }

    }

}



module.exports = chat;
```

index.js

```
const readline = require('readline');
const TechSupportChatbot = require('./chat');

const rl = readline.createInterface({
   input: process.stdin,
   output: process.stdout,
});

const chatbot = new TechSupportChatbot();
```

```
console.log("Welcome to Chatbot!");
console.log("Type your message and press Enter. Type 'exit' to quit.");

rl.on('line', (input) => {
    if (input.toLowerCase() === 'exit') {
        console.log("Goodbye!");
        rl.close();
    } else {
        const response = chatbot.getResponse(input);
        console.log(response);
    }
});
```

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

Code :-

server.js

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

const http = require('http');

const WebSocket = require('ws');



const app = express();

const server = http.createServer(app);

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



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



wss.on('connection', (ws) => {
    console.log('Client connected');


    ws.on('message', (message) => {
        console.log(`Received message: ${message}`);
        ws.send(`You said: ${message}`);
    });


    ws.on('close', () => {
        console.log('Client disconnected');
    });
});


server.listen(3000, () => {
    console.log('Server is listening on port 3000');
});
```

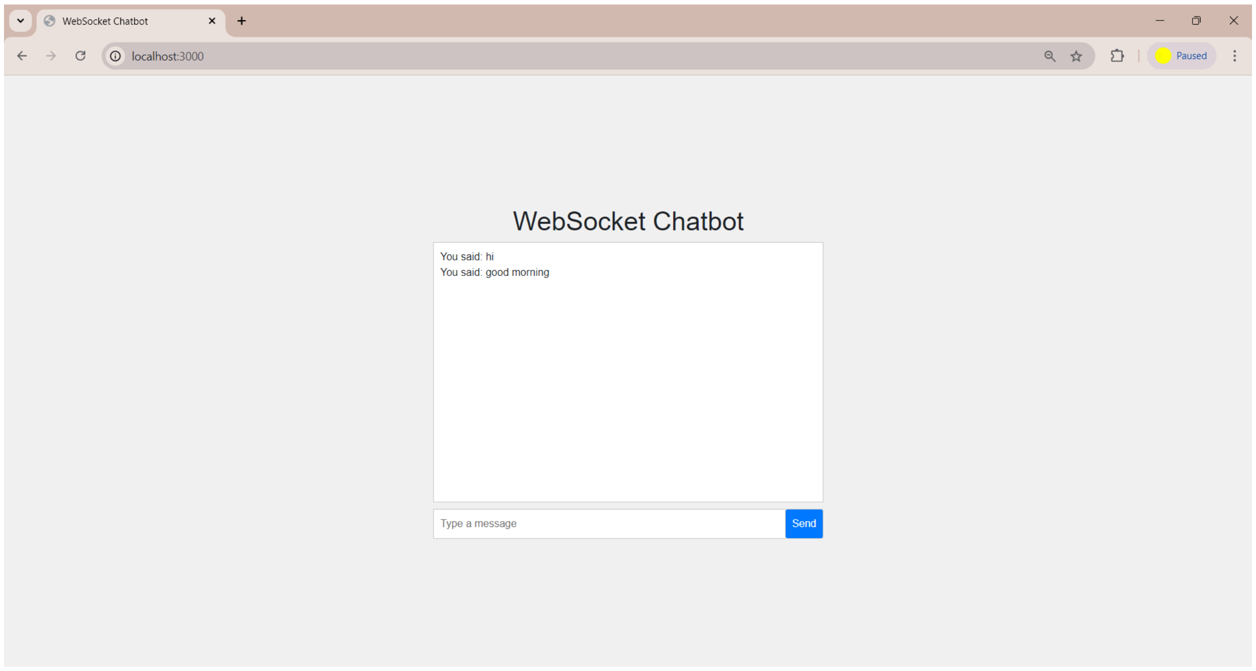
index.html

```
<!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>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65
VohhpuuCOmLASjC" crossorigin="anonymous">

    <h1>WebSocket Chatbot</h1>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            display: flex;

            flex-direction: column;

            align-items: center;

            justify-content: center;

            height: 100vh;

            background-color: #f0f0f0;

        }


        #chat {

            width: 80%;
            max-width: 600px;

            height: 400px;

            border: 1px solid #ccc;

            overflow-y: auto;

            padding: 10px;

            background-color: #fff;

            margin-bottom: 10px;

        }


        #input {

            width: 80%;

            max-width: 600px;

            display: flex;

        }


        #message {

            flex: 1;

            padding: 10px;

            font-size: 16px;

            border: 1px solid #ccc;

            border-right: none;

        }
```

```
        #send {
            padding: 10px;
            font-size: 16px;
            border: 1px solid #ccc;
            cursor: pointer;
            background-color: #007BFF;
            color: #fff;
        }
    </style>
</head>

<body>
    <div id="chat"></div>
    <div id="input">
        <input type="text" id="message" placeholder="Type a message">
        <button class="btn btn-primary" id="send">Send</button>
    </div>

    <script>
        const chat = document.getElementById('chat');
        const messageInput = document.getElementById('message');
const sendButton = document.getElementById('send');

const ws = new WebSocket('ws://localhost:3000');

ws.onmessage = (event) => {
  const message = document.createElement('div');
  message.textContent = event.data;
  chat.appendChild(message);
  chat.scrollTop = chat.scrollHeight;
};

sendButton.addEventListener('click', () => {
  const message = messageInput.value;
  ws.send(message);
  messageInput.value = '';
});

messageInput.addEventListener('keypress', (event) => {
  if (event.key === 'Enter') {
    sendButton.click();
  }
});
```

```
    </script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP
+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>


</body>


</html>
```

Output :-



WebSocket Chatbot

You said: hi
You said: good morning

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

Code : -

zip.js

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

function createZip(sourceDir, outPath) {
  if (!fs.existsSync(sourceDir)) {
    console.error(`Source directory ${sourceDir} does not exist`);
    return;
  }

  const outDir = path.dirname(outPath);
  if (!fs.existsSync(outDir)) {
    fs.mkdirSync(outDir, { recursive: true });
  }

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


  output.on('close', function() {
    console.log(`${archive.pointer()} total bytes`);
    console.log('Zip file has been created successfully');
  });


  archive.on('warning', function(err) {
    if (err.code === 'ENOENT') {
      console.warn('Warning:', err);
    } else {
      throw err;
    }
  });


  archive.on('error', function(err) {
    throw err;
  });


  archive.pipe(output);
```

```
    archive.directory(sourceDir, false);


    archive.finalize();
}


const sourceDir = 'path/to/source/folder';
const outPath = 'path/to/output/filename.zip';

createZip(sourceDir, outPath);
```

6. Write a program to extract a zip file.

Code :-

zipfile.js

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

const zipFilePath = './example.zip';

const extractDir = './extracted';

if (!fs.existsSync(extractDir)) {
    fs.mkdirSync(extractDir);
}

fs.createReadStream(zipFilePath)
    .pipe(unzipper.Extract({ path: extractDir }))
    .on('finish', () => {
        console.log('Extraction complete!');
    })
    .on('error', (err) => {
        console.error('Error extracting zip file:', err);
    });
```

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

Code : -

fs_unlink.js

```
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 ${filePath} has been deleted`);
   } catch (error) {
      console.error(`Error deleting file ${filePath}:`, error);
   }
}


const filePath = 'path/to/your/file.txt';
deleteFile(filePath);
```

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

Code :-

note-fetch.js

```
const fetch = require('node-fetch');

const fetchGooglePage = async() => {
   try {
        const response = await fetch('https://www.google.com');
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }
        const text = await response.text();
        console.log(text);
    } catch (error) {
        console.error('Error fetching Google page:', error);
    }
};

fetchGooglePage();
```

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.

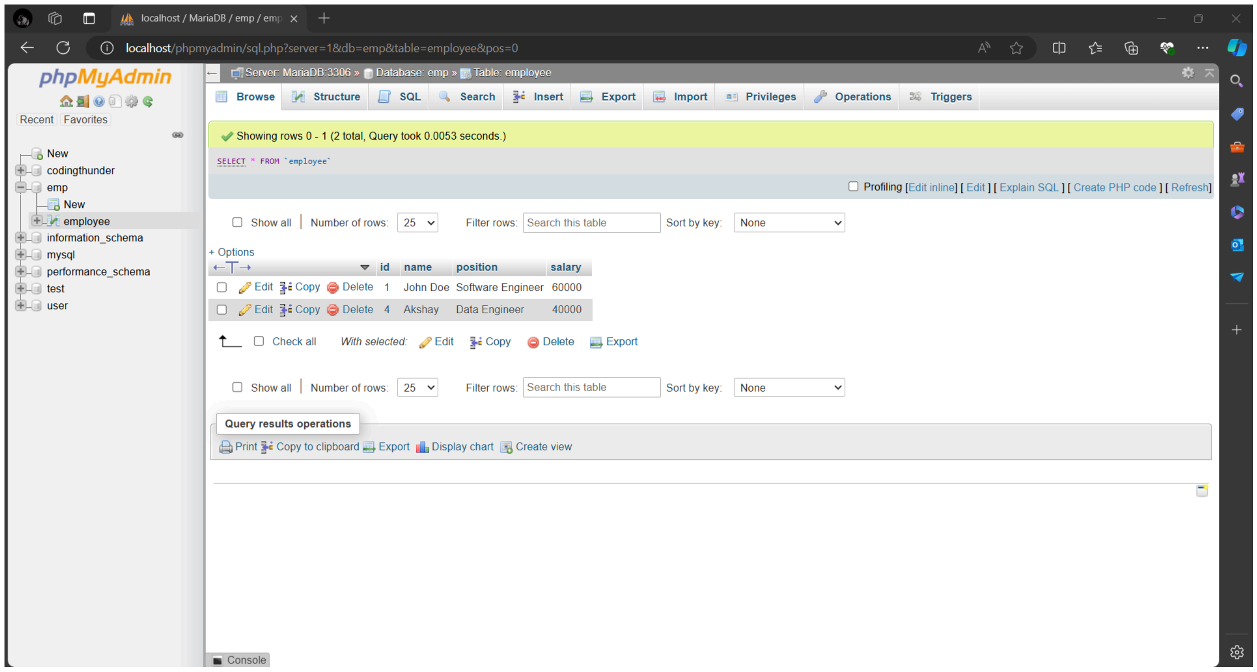
Code :-

index.js

```
const mysql = require('mysql2/promise');



const config = {

    host: 'localhost',

    user: 'root',

    password: '',

    database: 'emp'

};
```

```
async function main() {

  const connection = await mysql.createConnection(config);

  try {

    const insertQuery = 'INSERT INTO employee (name, position, salary)
VALUES (?, ?, ?)';
    const [insertResult] = await connection.execute(insertQuery, ['Akshay',
'Data Engineer', 40000]);
    console.log('Inserted record ID:', insertResult.insertId);



    const selectQuery = 'SELECT * FROM employee';
    const [rows] = await connection.execute(selectQuery);

    console.log('All employees:');
    rows.forEach(row => {
      console.log(`ID: ${row.id}, Name: ${row.name}, Position:
${row.position}, Salary: ${row.salary}`);
    });
  } catch (error) {
        console.error('Error:', error);

    } finally {



        await connection.end();

    }

}



main();
```

Output :-



✓ Showing rows 0 - 1 (2 total, Query took 0.0053 seconds.)

```
SELECT * FROM `employee`
```

	id	name	position	salary
Edit Copy Delete	1	John Doe	Software Engineer	60000
Edit Copy Delete	4	Akshay	Data Engineer	40000

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

Code :-

package.json

```
{
  "name": "my-node-app",
  "version": "1.0.0",
  "description": "A brief description of your application",
  "main": "index.js",
  "scripts": {
    "start": "node server.js",
    "test": "jest",
    "build": "webpack --config webpack.config.js",
    "lint": "eslint .",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {
    "jest": "^27.0.6",
    "webpack": "^5.38.1",
    "eslint": "^7.32.0",
    "nodemon": "^2.0.12"
  },
  "author": "Your Name",
  "license": "ISC"
}
```