# **FROM 4th September TO 10th September**

# **Project ID:**

# **2021J\_BV01\_BCI Browser**

# **Project Title:**

# **Design and development of Brain Computer Interface Browser on Web and Mobile**

# **Summary:**

* NodeJS+MySQL database connection
* Styling in reactjs

# **Detail:**

**NodeJS+MySQL Connection**

**Create a database named "mydb":**

**var mysql = require('mysql');**

**var con = mysql.createConnection({**

**host: "localhost",**

**user: "*yourusername*",**

**password: "*yourpassword*"**

**});**

**con.connect(function(err) {**

**if (err) throw err;**

**console.log("Connected!");**

**con.query("CREATE DATABASE mydb", function (err, result) {**

**if (err) throw err;**

**console.log("Database created");**

**});**

**});**

Prerequisites:

• Basic knowledge of React and Node.

• Node.js installed (version 12+).

• npm installed (version 6+).

Project Structure: This is the structure when all the modules and required files are ready.

Backend setup: Firstly we will work on our backend(NodeJS) portion. In your working folder make a file named app.js for NodeJS and package.json file to run all the modules we required.

Installing Module:

• Installing ExpressJS is a nodeJS framework:

npm install express

• Installing nodemon:

npm install nodemon

Configuration of package .json file: Add the start and dev script, which are important for running and dynamically running the code after changes made in your Node.js app respectively in package.json file

Filename- app.js: Here is the simple JavaScript code that should be written in app.js which is for NodeJS.

const express = require("express");

const app = express();

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

res.send("Hello World!");

});

const PORT = process.env.PORT || 8080;

app.listen(PORT, console.log(`Server started on port ${PORT}`));

Frontend setup: First, we have to create React app and run your app by writing the below command.

npx create-react-app demo

cd demo

npm start

Connecting: We have completed both the frontend and backend parts, now we have to connect both. Now for connecting Reactjs with Nodejs we have added this line in package.json of react app folder:

"proxy": "http://localhost:8080

Filename- package.json: The package.json file is in your React app folder. This tells React to proxy API requests to the Node.js server built with Express in our project.

{

"name": "demo",

"version": "0.1.0",

"private": true,

"proxy": "http://localhost:8080",

"dependencies": {

"@testing-library/jest-dom": "^5.11.4",

"@testing-library/react": "^11.1.0",

"@testing-library/user-event": "^12.1.10",

"react": "^17.0.1",

"react-dom": "^17.0.1",

"react-scripts": "4.0.2",

"web-vitals": "^1.0.1"

},

**STYLING IN REACTJS**

To style an element with the inline style attribute, the value must be a JavaScript object:

class MyHeader extends React.Component {

render() {

return (

<div>

<h1 style={{color: "red"}}>Hello Style!</h1>

<p>Add a little style!</p>

</div>

);

}

}

### **camelCased Property Names**

Since the inline CSS is written in a JavaScript object, properties with two names, like background-color, must be written with camel case syntax:

class MyHeader extends React.Component {

render() {

return (

<div>

<h1 style={{backgroundColor: "lightblue"}}>Hello Style!</h1>

<p>Add a little style!</p>

</div>

);

}

}

### **JavaScript Object**

You can also create an object with styling information, and refer to it in the style attribute:

class MyHeader extends React.Component {

render() {

const mystyle = {

color: "white",

backgroundColor: "DodgerBlue",

padding: "10px",

fontFamily: "Arial"

};

return (

<div>

<h1 style={mystyle}>Hello Style!</h1>

<p>Add a little style!</p>

</div>

);

}

}

## **CSS Stylesheet**

You can write your CSS styling in a separate file, just save the file with the .css file extension, and import it in your application.

**CONCLUSION**

We learnt how to style css components in ReactJS, and Nodejs and MySQL connection.