# **FROM 11th September TO 17th September**

# **Project ID:**

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

# **Project Title:**

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

# **Summary:**

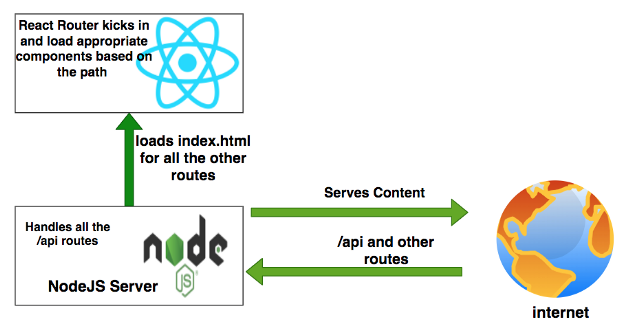
* NodeJS+MySQL database connection
* File upload in Node.js
* Forgot Password Button
* Stop button in P300 speller

# **Detail:**

**ReactJS+NodeJS+MySQL Connection**

In the development phase, we run the nodejs server and the React app on completely different ports. It’s easier and faster to develop that way. If you look at the following diagram the React app is running on port 3000 with the help of a webpack dev server and the nodejs server is running on port 3080.

React is a javascript library for building web apps and it doesn’t load itself in the browser. We need some kind of mechanism that loads the **index.html** (single page) of React application with all the dependencies(CSS and js files) in the browser. In this case, we are using node as the webserver which loads React assets and accepts any API calls from the React UI app.



all the web requests without the /api will go to React routing and the React Router kicks in and loads components based on the path. All the paths that contain /api will be handled by the Node server itself.

We need to import express and define two routes: **/api/users** and **/api/user** and the server listening on the port **3080.** Here is the server.js file. We use body-parser to handle data in the http request object.

const express = require('express');

const app = express(),

bodyParser = require("body-parser");

port = 3080;

// place holder for the data

const users = [];

app.use(bodyParser.json());

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

console.log('api/users called!!!!')

res.json(users);

});

app.post('/api/user', (req, res) => {

const user = req.body.user;

console.log('Adding user::::::::', user);

users.push(user);

res.json("user added");

});

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

res.send('App Works !!!!');

});

app.listen(port, () => {

console.log(`Server listening on the port::${port}`);

});

export async function getAllUsers() {

const response = await fetch('/api/users');

return await response.json();

}

export async function createUser(data) {

const response = await fetch(`/api/user`, {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({user: data})

})

return await response.json();

}

## File Uploads in node.js

Uploading files in HTML forms is a special case that requires an encoding type of "multipart/form-data". See [MDN’s guide to sending form data](https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data#A_special_case_sending_files) for more details about what happens with multipart form submissions.

You’ll need additional middleware to handle multipart uploads. There’s an Express package named [multer](https://www.npmjs.com/package/multer) that we’ll use here:

// routes.js

const multer = require('multer');

const upload = multer({ storage: multer.memoryStorage() });

router.post('/contact', upload.single('photo'), csrfProtection, [

// validation ...

], (req, res) => {

// error handling ...

if (req.file) {

console.log('Uploaded: ', req.file);

// Homework: Upload file to S3

}

req.flash('success', 'Thanks for the message! I’ll be in touch :)');

res.redirect('/');

});

This code instructs multer to upload the file in the “photo” field into memory and exposes the File object in req.file, which we can inspect or process further.

The last thing we need is to add the enctype attribute and our file input:

<form method="post" action="/contact?\_csrf=<%= csrfToken %>" novalidate enctype="multipart/form-data">

<input type="hidden" name="\_csrf" value="<%= csrfToken %>">

<div class="form-field <%= errors.message ? 'form-field-invalid' : '' %>">

<label for="message">Message</label>

<textarea class="input" id="message" name="message" rows="4" autofocus><%= data.message %></textarea>

<% if (errors.message) { %>

<div class="error"><%= errors.message.msg %></div>

<% } %>

</div>

<div class="form-field <%= errors.email ? 'form-field-invalid' : '' %>">

<label for="email">Email</label>

<input class="input" id="email" name="email" type="email" value="<%= data.email %>" />

<% if (errors.email) { %>

<div class="error"><%= errors.email.msg %></div>

<% } %>

</div>

<div class="form-field">

<label for="photo">Photo</label>

<input class="input" id="photo" name="photo" type="file" />

</div>

<div class="form-actions">

<button class="btn" type="submit">Send</button>

</div>

</form>

Try uploading a file. You should see the File objects logged in the console.

**Forgot Password Button**

import React from "react";

const Reset = React.createClass({

render() {

return (

<body>

<table>

<form action="/reset" method="post">

<tbody>

<tr>

<td>Enter Username</td>

<td><input type="text" name="username"/></td>

</tr>

</tbody>

<tbody>

<tr>

<td>Enter Security Answer</td>

<td><input type = "text" name = "answer"/></td>

</tr>

</tbody>

<tbody>

<tr>

<td>Enter New Password</td>

<td><input type = "text" name = "pwd"/></td>

</tr>

</tbody>

<tbody>

<tr>

<td>Confirm password</td>

<td><input type = "text" name = "cpwd"/></td>

</tr>

</tbody>

<tr>

<td colspan="2" align="center"><input type="submit" class="class1"/></td>

</tr>

</form>

</table>

</body>

)

},

});

export default Reset;

**Stop Button in P300 Speller**

**<button style="margin-top: 10px;"**

**onclick="**

**sessionStorage.setItem('stop', true)**

**"**

**>**

**Stop**

**</button>**