**WEEK 02:**

**//UPDATED MY CODE BY INSTALLING VALIDATOR:**

const validator = require('validator');

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

const { name, contact, password } = req.body;

// Validate contact as an email and password length

if (!validator.isEmail(contact)) {

return res.status(400).send('Invalid email'); // THE PART WHERE I HAD TO UPDATE

}

if (!validator.isLength(password, { min: 8 })) {

return res.status(400).send('Password too short');

}

if (name && contact && password) {

const existingUser = users.find(

(u) => u.username === name || u.contact === contact

);

if (existingUser) {

return res.json({ success: false, message: "User already exists!" });

}

const newUser = {

id: users.length + 1,

username: name,

contact,

password,

role: "member",

};

users.push(newUser);

res.json({ success: true, redirectTo: "/members", userId: newUser.id });

} else {

res.json({ success: false, message: "All fields are required" });

}

});

**HASHed PASSWORDS WITH BCRYPT:**

const bcrypt = require('bcrypt');

const validator = require('validator'); //ADDED THESE MODULES AT THE TOP OF MY FILE

//SIGNUP ROUTE

app.post("/signup", async (req, res) => {

const { name, contact, password } = req.body;

// Validate contact as an email and password length

if (!validator.isEmail(contact)) {

return res.status(400).send('Invalid email');

}

if (!validator.isLength(password, { min: 8 })) {

return res.status(400).send('Password too short');

}

if (name && contact && password) {

const existingUser = users.find(

(u) => u.username === name || u.contact === contact

);

if (existingUser) {

return res.json({ success: false, message: "User already exists!" });

}

// Hash the password before storing

const hashedPassword = await bcrypt.hash(password, 10); // ADDED LINES HERE

const newUser = {

id: users.length + 1,

username: name, // CHANGED LINES HERE

contact,

password: hashedPassword,

role: "member",

};

users.push(newUser);

res.json({ success: true, redirectTo: "/members", userId: newUser.id });

} else {

res.json({ success: false, message: "All fields are required" });

}

});

**//MODIFIED LOGIN PAGE**

app.post("/login", async (req, res) => {

const { username, password } = req.body;

// Find user by username only

const user = users.find((u) => u.username === username); // HAVE TO FIND USER BY USERNAME SINCE WE ARE USING BCRYPT

if (!user) {

return res.status(401).send('Invalid credentials');

}

// Compare plain password with hashed password

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) {

return res.status(401).send('Invalid credentials');

}

// Authentication successful

if (user.role === "admin") {

res.json({ success: true, redirectTo: "/admin" });

} else {

res.json({ success: true, redirectTo: "/members", userId: user.id });

}

});

**IMPLEMENT JWT AUTHENTICATION:**

* I also added the “**const jwt = require('jsonwebtoken')**;” at the top of the file to ensure that authentication is taking place with JWT.

//LOGIN

app.post("/login", async (req, res) => {

    const { username, password } = req.body;

    // Find user by username only

    const user = users.find((u) => u.username === username);

    if (!user) {

        return res.status(401).send('Invalid credentials');

    }

    // Compare plain password with hashed password

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {

        return res.status(401).send('Invalid credentials');

    }

    // Authentication successful

 const token = jwt.sign({ id: user.\_id }, 'HI@1M’ , { expiresIn: '1h' });

    if (user.role === "admin") {

    res.json({ success: true, redirectTo: "/admin", token });

   } else {

    res.json({ success: true, redirectTo: "/members", userId: user.id, token });

    }

});

**INSTALL HELMET :**

* Added ‘const helmet = require('helmet');’ at the top of the file.
* // Adding security 'helmet'
* app.use(helmet());