Absolutely! Here's a complete, step-by-step tutorial that walks you through building a basic user registration and login system using **Node.js**, **Express**, and **PostgreSQL**. You can copy and save this as a reference for future use.

**🛠️ Full Tutorial: User Registration & Login with Node.js + PostgreSQL**

**📦 1. Install Dependencies**

Make sure you have Node.js and PostgreSQL installed. Then run:

npm init -y

npm install express body-parser pg

**🗄️ 2. Create PostgreSQL Database and Table**

In your PostgreSQL shell (psql), create a database and a users table:

CREATE DATABASE newvision;

\c newvision

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(100) NOT NULL

);

Then grant permissions to your app user (nv):

GRANT ALL PRIVILEGES ON TABLE users TO nv;

GRANT USAGE, SELECT ON SEQUENCE users\_id\_seq TO nv;

**📁 3. Project Structure**

nv-website/

├── public/

│ ├── register.html

│ ├── login.html

│ └── welcome.html

└── server.js

**🧾 4. register.html**

<form action="/register" method="POST">

<input type="text" name="username" placeholder="Choose a username" required />

<input type="password" name="password" placeholder="Choose a password" required />

<button type="submit">Register</button>

</form>

**🔐 5. login.html**

<form action="/login" method="POST">

<input type="text" name="username" placeholder="Username" required />

<input type="password" name="password" placeholder="Password" required />

<button type="submit">Login</button>

</form>

**🎉 6. welcome.html**

<h1>Welcome!</h1>

<p>You have successfully logged in.</p>

**🚀 7. server.js**

const express = require('express');

const bodyParser = require('body-parser');

const { Pool } = require('pg');

const path = require('path');

const pool = new Pool({

user: 'nv',

host: 'localhost',

database: 'newvision',

password: '4000',

port: 5432,

});

const app = express();

const port = 8080;

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

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.listen(port, () => {

console.log(`Server running at http://localhost:${port}`);

});

// Register route

app.post('/register', async (req, res) => {

const { username, password } = req.body;

console.log('Received registration:', username, password);

try {

await pool.query(

'INSERT INTO users (username, password) VALUES ($1, $2)',

[username, password]

);

res.send('Registration successful. You can now <a href="/login.html">log in</a>.');

} catch (err) {

console.error('Registration error:', err);

res.status(500).send('Registration failed. Username might already exist.');

}

});

// Login route

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

const { username, password } = req.body;

console.log('Login attempt:', username);

try {

const result = await pool.query(

'SELECT \* FROM users WHERE username = $1 AND password = $2',

[username, password]

);

if (result.rows.length > 0) {

res.sendFile(path.join(\_\_dirname, 'public', 'welcome.html'));

} else {

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

}

} catch (err) {

console.error('Login error:', err);

res.status(500).send('Server error');

}

});

**✅ 8. Test Your App**

1. Start your server:
2. node server.js
3. Visit:
   * http://localhost:8080/register.html to register
   * http://localhost:8080/login.html to log in

**🔒 Optional: Add Password Hashing with bcrypt**

Let me know if you'd like to add that next for better security!

You’ve built a working authentication system from scratch. Save this file, and you’ll have a solid foundation for any future web app. Want me to help you turn this into a more advanced system with sessions or JWTs next?