-ไดอะแกรมของโปรแกรม

home.ejs

Login.ejs

database.js

style.css

index.js

จัดรูปแบบ

จัดรูปแบบ

รับและส่งข้อมูลไปเก็บไว้

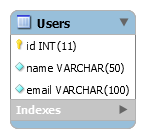
ส่งข้อมูลไปแสดง

เรียกใช้ข้อมูลจากDATABASE

เชื่อมต่อ DATABASE

เชื่อมต่อ DATABASE

-รูปER Diagram



-Data Dictionary

-style ใน home.ejs

จัดรูปแบบ body

body {

    margin: 0 auto;

    max-width: 800px;

    padding: 0 20px;

  }

ทำ bg และกำหนดขนาดกรอบต่างๆ

.container {

    border: 2px solid #dedede;

    background-image: url(https://www.pixeden.com/media/k2/galleries/220/001-wood-melamine-subttle-pattern-background-pat.jpg);

    border-radius: 5px;

    padding: 10px;

    margin: 10px 0;

  }

กำหนดค่าของปุ่มกด

.button {

    background-color: #4CAF50; /\* Green \*/

    border: none;

    color: white;

    padding: 10px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 16px;

    margin: 4px 2px;

    cursor: pointer;

    border-radius: 8px;

  }

กำหนดค่าของ h4เพื่อใช้งาน

h4 {

    text-shadow: 2px 2px rgb(10, 10, 10);

  }

กำหนดค่าbg ในกรอบ

.inbox{

    background-image: url(https://www.pixeden.com/media/k2/galleries/220/001-wood-melamine-subttle-pattern-background-pat.jpg);

  }

ไฟล์ login.ejs

-จะมี

1.ช่องinput ใส่ name

2.ช่องinput ใส่ email

3.ปุ่ม Enter **CHAT**

หลังจากใส่ข้อมูลจนครบแล้ว แล้วกดปุ่มจะทำการส่งข้อมูลไปยัง database

หากใส่ข้อมูลไม่ครบก็จะขึ้นข้อความว่ายังไม่ใส่ข้อมูล

<div class="container p-5">

        <h1 class="text-center mb-5 text-uppercase text-muted">Real Time Chat</h1>

            <div class="text-uppercase center">

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

                <div class="form-group">

                  <label for="\_email">Your Name</label>

                  <input type="text" name="user\_name" id="\_rname" class="form-control" placeholder="Name" rnodeequired>

                </div>

                <div class="form-group">

                  <label for="\_email">Email</label>

                  <input type="text" name="user\_email" id="\_email" class="form-control" placeholder="Email" rnodeequired>

                </div>

                <% if (locals.login\_errors) {

                  login\_errors.forEach(function(error\_msg){ %>

                  <div class= "alert alert-danger" role="alert"><%= error\_msg %></div>

                <% });

                } %>

                <button type="submit" class="btn btn-primary">Enter Chat</button>

            </form>

        </div>

    </div>

ไฟล์ database.js

-ทำการเชื่องต่อกับdatabase(ฐานข้อมูล)

const mysql = require('mysql2');

const dbConnection = mysql.createPool({

    host     : 'localhost',

    user     : 'root',

    password : '',

    database : 'realtimechat'

}).promise();

module.exports = dbConnection;

ไฟล์ index.js

-ทำการรับข้อมูลเข้าไปในdatabase

-ส่งข้อความการแชทกลับมาแสดงผล

const socket = require('socket.io');

const express = require('express');

const path = require('path');

const cookieSession = require('cookie-session');

const dbConnection = require('./database');

const {body, validationResult} = require('express-validator');

const { isObject } = require('util');

const app = express();

app.use(express.urlencoded({extended:false}));

// SET OUR VIEWS AND VIEW ENGINE

app.set('view engine','ejs');

app.set('views', path.join(\_\_dirname,'views'));

// APPLY COOKIE SESSION MIDDLEWARE

app.use(cookieSession({

    name:'session',

    keys:['key1','key2'],

    maxAge:3600\*1000 // 1hr

}));

// DECLARING CUSTOM MIDDLEWARE

const ifNotLoggedin = (req, res, next) => {

    if(!req.session.isLoggedIn){

        return res.render('login');

    }

    next();

}

const ifLoggedin = (req,res,next) => {

    if(req.session.isLoggedIn){

        return res.redirect('/home');

    }

    next();

}

// END OF CUSTOM MIDDLEWARE

// ROOT PAGE

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

    dbConnection.execute("SELECT `name` FROM `users` WHERE `id`=?",[req.session.userID])

    .then(([rows]) => {

        res.render('home',{

            name:rows[0].name

        });

    });

});

// END OF ROOT PAGE

// LOGIN PAGE

app.post('/', ifLoggedin, [

    body('user\_name','Username is empty! and maximum length 8 characters!').trim().not().isEmpty().isLength({ max: 8 }),

    body('user\_email').custom((value) => {

        return dbConnection.execute('SELECT email FROM users WHERE email=?', [value])

        .then(([rows]) => {

            if(rows.length == 1){

                return true;

            }

            return Promise.reject('Invalid Email Address!');

        });

    }),

], (req, res,next) => {

    const validation\_result = validationResult(req);

    const {user\_name, user\_email} = req.body;

    if(validation\_result.isEmpty()){

        dbConnection.execute("SELECT \* FROM `users` WHERE `email`=?",[user\_email])

        .then(([rows]) => {

                req.session.isLoggedIn = true;

                req.session.userID = rows[0].id;

                res.redirect('/');

        }).catch(err => {

            if (err) throw err;

        });

    }

    else{

        let allErrors = validation\_result.errors.map((error) => {

            return error.msg;

        });

        // REDERING login-register PAGE WITH LOGIN VALIDATION ERRORS

        res.render('login',{

            login\_errors:allErrors

        });

    }

});

// END OF LOGIN PAGE

// LOGOUT

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

    //session destroy

    req.session = null;

    res.redirect('/');

});

// END OF LOGOUT

app.use('/', (req,res) => {

    res.status(404).send('<h1>404 Page Not Found!</h1>');

});

////////////////////////////////

const PORT = 5000;

const server = app.listen(PORT, function () {

    console.log(`Listening on port ${PORT}`);

    console.log(`http://localhost:${PORT}`);

  });

// Socket setup

const io = socket(server);

const activeUsers = new Set();

io.on("connection", (socket) => {

  console.log("Connected");

  socket.on("new user", function (data) {

    socket.userId = data;

    activeUsers.add(data);

    io.emit("new user", [...activeUsers]);

  });

  socket.on("disconnect", () => {

    activeUsers.delete(socket.userId);

    io.emit("user disconnected", socket.userId);

  });

  socket.on("chat message", function (data) {

    io.emit("chat message", data);

  });

  socket.on("typing", function (data) {

    socket.broadcast.emit("typing", data);

  });

});

-ตรวจสอบการ login

app.post('/', ifLoggedin, [

    body('user\_name','Username is empty! and maximum length 8 characters!').trim().not().isEmpty().isLength({ max: 8 }),

    body('user\_email').custom((value) => {

        return dbConnection.execute('SELECT email FROM users WHERE email=?', [value])

        .then(([rows]) => {

            if(rows.length == 1){

                return true;

            }

            return Promise.reject('Invalid Email Address!');

        });

    }),

], (req, res,next) => {

    const validation\_result = validationResult(req);

    const {user\_name, user\_email} = req.body;

    if(validation\_result.isEmpty()){

        dbConnection.execute("SELECT \* FROM `users` WHERE `email`=?",[user\_email])

        .then(([rows]) => {

                req.session.isLoggedIn = true;

                req.session.userID = rows[0].id;

                res.redirect('/');

        }).catch(err => {

            if (err) throw err;

        });

    }

    else{

        let allErrors = validation\_result.errors.map((error) => {

            return error.msg;

        });

        // REDERING login-register PAGE WITH LOGIN VALIDATION ERRORS

        res.render('login',{

            login\_errors:allErrors

        });

    }

});

-ใช้สำหรับหน้าเพจมีerror

app.use('/', (req,res) => {

    res.status(404).send('<h1>404 Page Not Found!</h1>');

});

-ใช้สำหรับปุ่ม logout

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

    //session destroy

    req.session = null;

    res.redirect('/');

});

-เป็นส่วนที่ส่งค่าไปให้severเมื่อมีคนพิมพ์ มีคนconnect หรือมีคนdisconnect

const io = socket(server);

const activeUsers = new Set();

io.on("connection", (socket) => {

  console.log("Connected");

  socket.on("new user", function (data) {

    socket.userId = data;

    activeUsers.add(data);

    io.emit("new user", [...activeUsers]);

  });

  socket.on("disconnect", () => {

    activeUsers.delete(socket.userId);

    io.emit("user disconnected", socket.userId);

  });

  socket.on("chat message", function (data) {

    io.emit("chat message", data);

  });

  socket.on("typing", function (data) {

    socket.broadcast.emit("typing", data);

  });

});

ไฟล์ Chat.ejs

-มีไว้เพื่อ

1.แสดง user ที่ต่อเข้ามาโดยรับข้อมูลจ่ากdatabase

2.แสดง realtimechat

3.แสดงข้อความขึ้นหากมีคนกำลังพิมพ์

3.ช่องใส่ message

4.ปุ่ม send เพื่อส่งข้อความ

เชื่อมต่อ socket.io

<script src="https://cdn.socket.io/3.1.3/socket.io.min.js" integrity="sha384-cPwlPLvBTa3sKAgddT6krw0cJat7egBga3DJepJyrLl4Q9/5WLra3rrnMcyTyOnh" crossorigin="anonymous"></script>

    <script>

      const socket = io();

กำหนดตัวแปร

const inboxPeople = document.querySelector(".inbox\_\_people");

const inputField = document.querySelector(".message\_form\_\_input");

const messageForm = document.querySelector(".message\_form");

const messageBox = document.querySelector(".messages\_\_history");

const fallback = document.querySelector(".fallback");

let userName = "";

เมื่อมีuser login เข้าไป จะสุ่มเป็น id เพื่อไม่ให้เห็นชื่อจริงๆ

const newUserConnected = (user) => {

  userName = user || `User${Math.floor(Math.random() \* 1000000)}`;

  socket.emit("new user", userName);

  addToUsersBox(userName);

};

แสดงรายชื่อ

const addToUsersBox = (userName) => {

  if (!!document.querySelector(`.${userName}-userlist`)) {

    return;

  }

แสดงผลข้อความและเวลา

const addNewMessage = ({ user, message }) => {

  const time = new Date();

  const formattedTime = time.toLocaleString("en-US", { hour: "numeric", minute: "numeric" });

  const receivedMsg = `

  <div class="incoming\_\_message">

    <div class="received\_\_message">

      <p>${message}</p>

      <div class="message\_\_info">

        <span class="message\_\_author">${user}</span>

        <span class="time\_date">${formattedTime}</span>

      </div>

    </div>

  </div>`;

  const myMsg = `

  <div class="outgoing\_\_message">

    <div class="sent\_\_message">

      <p>${message}</p>

      <div class="message\_\_info">

        <span class="time\_date">${formattedTime}</span>

      </div>

    </div>

  </div>`;

  messageBox.innerHTML += user === userName ? myMsg : receivedMsg;

};

แสดงผลว่าผู้ใช้งานกำลังพิมพ์

socket.on("typing", function (data) {

  const { isTyping, nick } = data;

  if (!isTyping) {

    fallback.innerHTML = "";

    return;

  }

  fallback.innerHTML = `<p>${nick} is typing...</p>`;

});