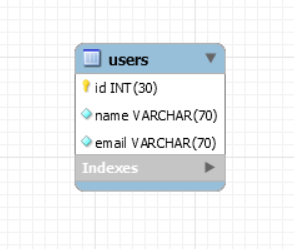


ER Diagram



style.css ใช้ตกแต่งหน้าเว็บ

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

text-shadow: 0 0 3px #ffff;

color : orange

font-family : Times, serif

}

html, body {

background-color: rgb(178, 25, 53);

overflow: hidden;

}

.container{

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

height: 100vh;

width: 100vw;

}

.inbox {

width: 600px;

height: 75%;

border-radius: 0.2em;

position: relative;

box-shadow: 1px 1px 12px rgba(255, 253, 253, 0.877);

}

form {

background: linear-gradient(120deg, rgb(60, 241, 238), rgb(20, 16, 230));

}

.inbox\_input {

width: 600px;

border-radius: 0.1em;

position: relative;

}

.inbox\_logout {

margin: 40px;

bottom: -20px;

position: absolute;

right: 0;

}

.message\_form\_\_input{

background: yellow;

position: absolute;

bottom: 0;

left: 0;

border: none;

width: 75%;

padding: 1.2em;

outline: none;

color: rgba(3, 1, 1, 0.9);

font-weight: 300;

}

.chat\_ib{

float: left;

text-align: left;

margin: -0.25em 2em;

font-size: 0.7em;

font-weight: 300;

color: rgb(255, 255, 255);

width: 200px;

}

.inbox\_\_people-list-bot{

border-radius: 0.2em;

position: relative;

}

.inbox\_\_people-list {

background: rgba(36, 40, 252, 0.404);

width: 25%;

height: 100%;

float: right;

border-top-right-radius: 0.2em;

border-bottom-right-radius: 0.2em;

}

.inbox\_\_people-list h4 {

background: rgba(18, 39, 231, 0.863);

color: rgba(255, 255, 255, 0.9);

font-size: 0.9em;

padding: 1em;

margin: 0;

font-weight: 300;

text-align: center;

}

.inbox\_\_messages::-webkit-scrollbar-track

{

-webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);

background-color: #F5F5F5;

}

.inbox\_\_messages::-webkit-scrollbar

{

width: 6px;

background-color: #F5F5F5;

}

.inbox\_\_messages::-webkit-scrollbar-thumb

{

background-color: #000000;

}

.inbox\_\_messages{

overflow-y: scroll;

overflow-x: hidden;

width: 450px;

background: rgb(185, 208, 139)

height : 91%;

border-top-right-radius : 0.2em;

border-bottom-right-radius: 0.2em;

}

.outgoing\_\_message , .incoming\_\_message {

background-color: rgb(185, 208, 139);

padding: 1em 0;

height: auto;

width: 60%;

border-radius: 5px;

margin: 2em 1em;

}

.outgoing\_\_message {

float: right;

}

.incoming\_\_message {

float: left;

}

.sent\_\_message {

color: rgb(255, 255, 255);

font-size: 1em;

margin: 0 1em;

}

.received\_\_message {

color: rgb(255, 255, 255);

font-size: 1em;

margin: 0 1em;

}

.message\_\_info{

margin: 0 2em;

}

.message\_\_author,.time\_date{

color: rgb(255, 255, 255);

font-size: 0.5em;

margin: 1 0em;

}

login.ejs จะใช้ css และ html ในการทำหน้าของ login

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

<style>

@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {

width: 360px;

padding: 8% 0 0;

margin: auto;

}

.form {

position: relative;

z-index: 1;

background: #FFFFFF;

max-width: 360px;

margin: 0 auto 100px;

padding: 45px;

text-align: center;

box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form input {

font-family: "Roboto", sans-serif;

outline: 0;

background: #f2f2f2;

width: 100%;

border: 0;

margin: 0 0 15px;

padding: 15px;

box-sizing: border-box;

font-size: 14px;

}

.form button {

font-family: "Roboto", sans-serif;

text-transform: uppercase;

outline: 0;

background: #4CAF50;

width: 100%;

border: 0;

padding: 15px;

color: #FFFFFF;

font-size: 14px;

-webkit-transition: all 0.3 ease;

transition: all 0.3 ease;

cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

background: #43A047;

}

.form .message {

margin: 15px 0 0;

color: #b3b3b3;

font-size: 12px;

}

.form .message a {

color: #4CAF50;

text-decoration: none;

}

.form .register-form {

display: none;

}

.container {

position: relative;

z-index: 1;

max-width: 300px;

margin: 0 auto;

}

.container:before, .container:after {

content: "";

display: block;

clear: both;

}

.container .info {

margin: 50px auto;

text-align: center;

}

.container .info h1 {

margin: 0 0 15px;

padding: 0;

font-size: 36px;

font-weight: 300;

color: #1a1a1a;

}

.container .info span {

color: #4d4d4d;

font-size: 12px;

}

.container .info span a {

color: #000000;

text-decoration: none;

}

.container .info span .fa {

color: #EF3B3A;

}

body {

background: #76b852;

background: -webkit-linear-gradient(right, #76b852, #8DC26F);

background: -moz-linear-gradient(right, #76b852, #8DC26F);

background: -o-linear-gradient(right, #76b852, #8DC26F);

background: linear-gradient(to left, #76b852, #8DC26F);

font-family: "Roboto", sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

</style>

</head>

<body>

<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>

</body>

</html>

home.ejs ตกแต่งหน้าหลัก โดยกำหนด div มาจาก style.css

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<meta charset="UTF-8" />

<link rel="stylesheet" href="/css/style.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Real-time Chat</title>

</head>

<body>

<h1><center>Real-Time Chat</center></h1>

<div class="container">

<div class="inbox">

<div class="inbox\_\_people-list">

<h4>User</h4>

<div class="inbox\_\_people"></div>

</div>

<div class= "inbox\_\_messages ">

<div class="messages\_\_history"></div>

<div class="fallback"></div>

</div>

<a href="/logout" class="btn btn-danger inbox\_logout">Logout</a>

</div>

<form class="message\_form inbox\_input" >

<input type="text" class="message\_form\_\_input" placeholder="Type a message" />

</form>

</div>

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

<script>

const socket = io();

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 = "";

const newUserConnected = (user) => {

userName = user || "<%=name%>";

socket.emit("new user", userName);

addToUsersBox(userName);

};

const addToUsersBox = (userName) => {

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

return;

}

const userBox = `

<div class="chat\_ib ${userName}-userlist">

<h5>${userName}</h5>

</div>

`;

inboxPeople.innerHTML += userBox;

};

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

const time = new Date();

const formattedTime = time.toLocaleString("en-US", { hour: "numeric", minute: "numeric", day:"numeric", year:"2-digit",month:"2-digit" });

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;

};

// new user is created so we generate nickname and emit event

newUserConnected();

messageForm.addEventListener("submit", (e) => {

e.preventDefault();

if (!inputField.value) {

return;

}

socket.emit("chat message", {

message: inputField.value,

nick: userName,

});

inputField.value = "";

});

inputField.addEventListener("keyup", () => {

socket.emit("typing", {

isTyping: inputField.value.length > 0,

nick: userName,

});

});

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

data.map((user) => addToUsersBox(user));

});

socket.on("user disconnected", function (userName) {

document.querySelector(`.${userName}-userlist`).remove();

});

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

addNewMessage({ user: data.nick, message: data.message });

});

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

const { isTyping, nick } = data;

if (!isTyping) {

fallback.innerHTML = "";

return;

}

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

});

</script>

</body>

</html>

database.js เป็น javascript ที่เอาไว้กำหนดให้หน้าเว็บเข้าฐานข้อมูลที่สร้างจาก mysql

const mysql = require('mysql2');

const dbConnection = mysql.createPool({

host : 'localhost', // MYSQL HOST NAME

user : 'root', // MYSQL USERNAME

password : '', // MYSQL PASSWORD

database : 'realtimechat' // MYSQL DB NAME

}).promise();

module.exports = dbConnection;

index.js

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'));

app.use(express.static(path.join(\_\_dirname, 'public')));

// 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);

});

});