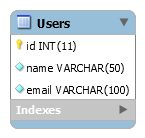
**-ไดอะแกรมของโปรแกรม**



**-รูปER Diagram**



**-Data Dictionary**

ไฟล์ style.css ใช้ทำการจัดรูปแบบ ui

-เปลี่ยนฟ้อนต์

@font-face {

font-family: 'Font';

src:url('Mali-Light.ttf') format('truetype');

}

-จัดรูปแบบbody กับ h4 และเปลี่ยน background

body{

font-family: "Font", monospace;

margin: 0 auto;

max-width: 800px;

padding: 0 20px;

background-color: #a4c8d5;

}

h4{

background-color: #5583a6;

color: rgb(255, 255, 255);

text-align: center;

border-radius: 20px;

}

-เปลี่ยนรูปแบบ button(ปุ่ม)ทั้ง 3 ปุ่ม

.button {

border-radius: 6px;

background-color: inherit;

padding: 7.5px 16px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.button1 {

color:#17395c;

}

.button1:hover {

background-color:#efa885;

color: white;

}

.button2 {

color:#17395c;

}

.button2:hover {

background-color:#efb758;

color: white;

}

.button3 {

color:#17395c;;

}

.button3:hover {

background-color:#efb758;

color: white;

}

-จัดช่องinputข้อความ

form input{

padding: 7px;

width: 400px;

border-radius: 5px;

border: none;

}

-จัดform ที่แสดงข้อความแชท

.inbox{

background-color: #ffffff63;

border-radius: 20px;

border: none;

margin: auto;

}

-ทำช่องmessage ของผู้รับกับผู้ส่ง

.outgoing\_\_message{

text-align: right;

border: 3px ;

border-radius: 6px;

background-color: lemonchiffon;

}

.incoming\_\_message{

border: 3px;

border-radius: 6px;

background-color:#cbaacb;

}

-จัดกึ่งกลาง

.center {

margin: auto;

width: 60%;

border: 3px solid #666665;

padding: 20px;

text-align: center;

border-radius: 10px;

}

ไฟล์ login.ejs

-link style.cssมาใช้

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

-จะมีข้อมูลที่ต้องกรอก

1.ช่องinput ใส่ Name

2.ช่องinput ใส่ email

3.ปุ่ม Login

หลังจากใส่ข้อมูลจนครบแล้ว แล้วกดปุ่มจะทำการส่งข้อมูลไปยัง database

หากใส่ข้อมูลไม่ครบ

* ไม่ได้กรอก Name จะขึ้นว่า USERNAME IS EMPTY! AND MAXIMUM LENGTH 8 CHARACTERS!
* ไม่ได้กรอก Email จะขึ้นว่า INVALID EMAIL ADDRESS!
* หากกรอกข้อมูลไม่ตรงตามที่ Insert ไว้ใน database จะขึ้นหน้า 404 Page Not Found!

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="icon" href="https://sv1.picz.in.th/images/2021/03/26/DrNyJS.png" sizes="16x16" type="image/png">

<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">

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

<title>Login</title>

</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">Name</label>

<input type="text" name="user\_name" id="\_rname" class="form-control" autofocus="on" 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="button button3">Login</button>

</form>

</div>

</div>

</body>

</html>

ไฟล์ database.js

-ทำการเชื่องต่อกับdatabase(ฐานข้อมูล)

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;

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'));

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('/chat');

}

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('Chat',{

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);

});

});

ไฟล์ Chat.ejs

-link ไปยัง style.css

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

-มีไว้เพื่อ

1.แสดง user ที่ต่อเข้ามาโดยรับข้อมูลจ่ากdatabase

2.แสดง realtimechat

3.แสดงข้อความขึ้นหากมีคนกำลังพิมพ์

3.ช่องใส่ message

4.ปุ่ม send เพื่อส่งข้อความ

5.แสดงจำนวนของข้อความที่สามารถพิมพ์ได้

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="icon" href="https://sv1.picz.in.th/images/2021/03/26/DrB1M0.png" sizes="16x16" type="image/png">

<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>

<div class="container">

<div class="inbox">

<h4>Real Time Chat</h4>

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

<h3><b>## Active Now ##</b></h3>

</div>

<h4>Chat Room</h4>

<div class="example1 inbox\_\_messages">

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

</div>

</div>

<form class="message\_form">

<p class="fallback"></p>

<input type="text" class="message\_form\_\_input" autofocus="on" maxlength="100" placeholder="Type a message..." onkeyup="count(this);" />

<button class="button button1 message\_form\_\_button" type="submit">

Send

</button>

<a href="/logout" class="button button2">Logout</a><br>

<div class="CoutChar">

<p id="CoutChar">100 characters remaining</p>

</div>

</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" });

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>`;

});

function count(obj){

var max = 100;

var str = obj.value.length;

var charRemain = (max - str);

document.getElementById('CoutChar').innerHTML = charRemain + 'CHARACTERS REMAINING';

}

</script>

</body>

</html>