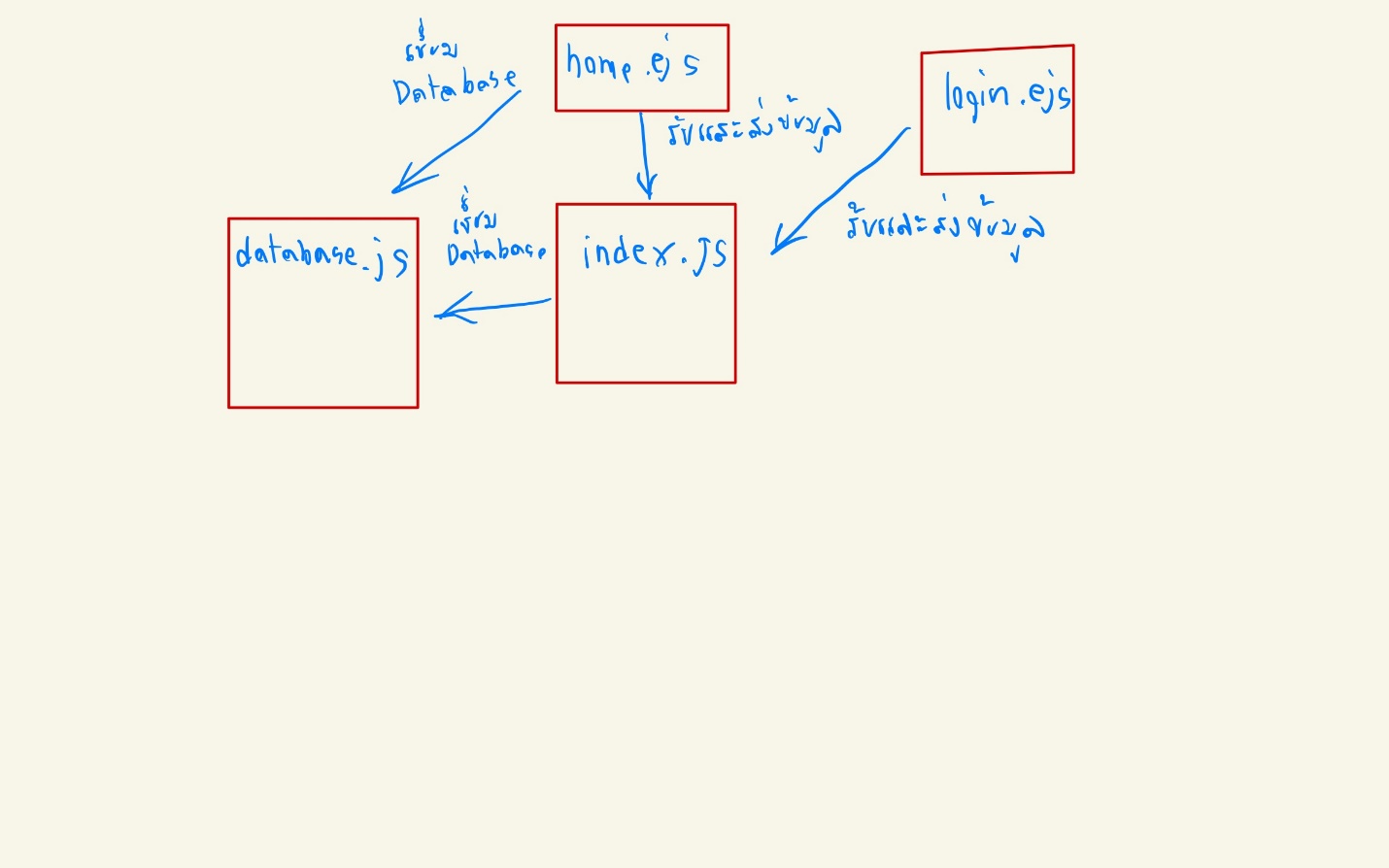
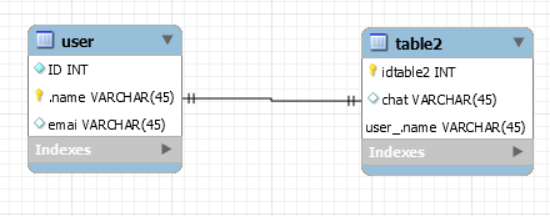
**ไดอะแกรมของโปรแกรม**



**รูป er diagram**



**Data Dictionary**

**Index.js**

1.รับข้อมูลจาก home.ejs(รับค่าเกี่ยวกับแชท) และ login.ejs(หน้าเข้าใช้)

2.นำค่าใน index.js เข้าไปใน database เพื่อเชื่อม database

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 = 3000;

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);

  });

});

**Home.ejs**

**1**.รับค่าเกี่ยวกับช่องแชท

2.ส่งค่าเกี่ยวกับ login ไป index.js

3.ส่งค่าเกี่ยวกับ chat ไป database

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

    <title>Real time Chat App</title>

</head>

<body>

    <div class="container">

      <div class="inbox">

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

          <h4>UserOnlines</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" });

  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>

**Login.ejs**

1.รับค่าlogin

2.ส่งค่าเกี่ยวกับ chat ไป index.js

<!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>

    </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>

Database.js

1.เชื่อม database ใน 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;