

# 嵌入式實驗一報告

Chat room

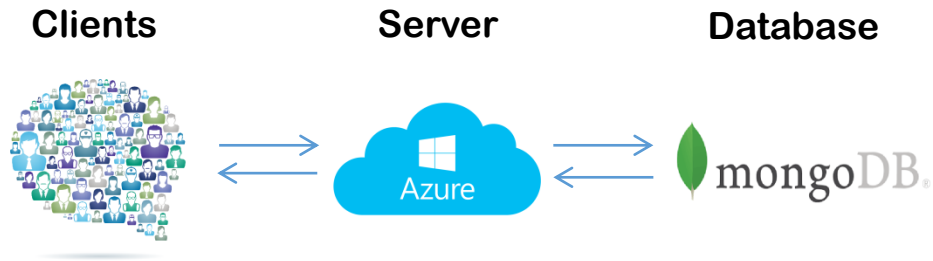
組員：電機三 林尚謙

電機三 楊耀程

指導教授：鄭振牟

繳交日期：2017/03/31

## 聊天室架構介紹



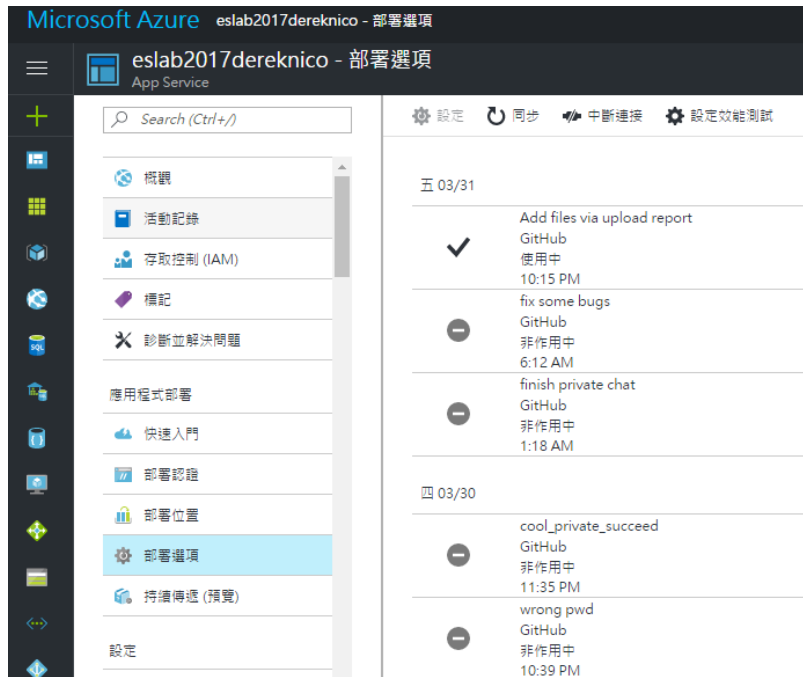
我們聊天室的架構可分為 client 端、server 端與 database，client 端即為瀏覽器上的網頁，server 端部屬在微軟 Microsoft Azure 的平台上，database 使用 MongoDB 作為雲端資料庫。Client 會先與 server 建立 socket 連線，在互動發生後(送出訊息、點擊按鈕等)，會由 client 端向 server 發出請求，再由 server 判斷該對誰(全部或特定的 socket)做出對應的回應，而 database 則負責提供 server 存取資料。我們的 client 端與 server 端分別由一個檔案(index.html 與 server.js)構成。

使用到的輔助工具：

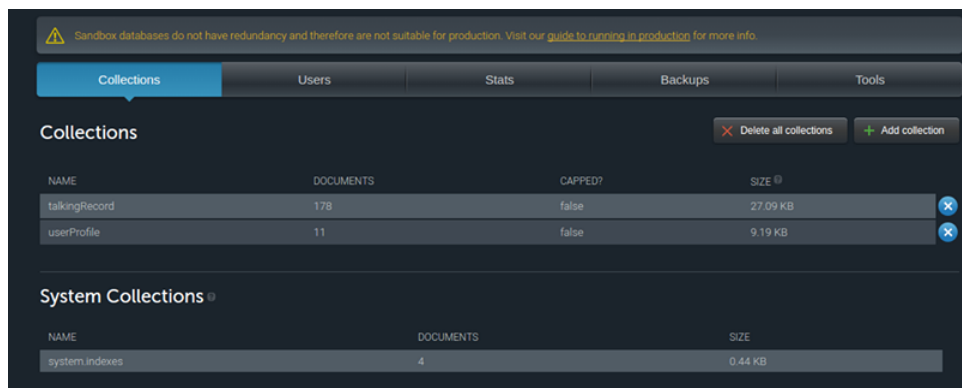
1. socket io
2. mongodb
3. jquery
4. tooltipster
5. microsoft azure

```
var express = require('express');
var app = express();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
```

```
<script src="/socket.io/socket.io.js"></script>
<script src="https://code.jquery.com/jquery-1.11.1.js"></script>
<script type="text/javascript" src="tooltipster.bundle.min.js"></script>
```



Microsoft Azure



mlab

(mongodb 雲端上的網站)

index.html(client) and server.js 由 socket io 互動 示意圖

```
socket.on('chat message', function(data) {
  AppendMessage(data);
});
```

client

```
socket.on('chat message', function(msg) { //re
  console.log(socket.username + ":" + msg);
  var mytime = mygetTime();
  io.emit('chat message', {
    u_name: socket.username,
```

server

## 特色與操作介面



### 1. 雲端的資料庫與 server

點開網址 <https://eslab2017dereknico.azurewebsites.net/>，或是在本機執行 `node server.js` 並輸入 `localhost:3000`，即可進入本聊天室。

我們的聊天室有雲端上的資料庫，server 也在雲端上執行，所以聊天並不限於本機，使用者可以用不同台電腦連上網路後，互相聊天。

```
var db;
dbClient.connect(url, function(err, tempdb) {
  db = tempdb;
});

io.on('connection', function(socket) {

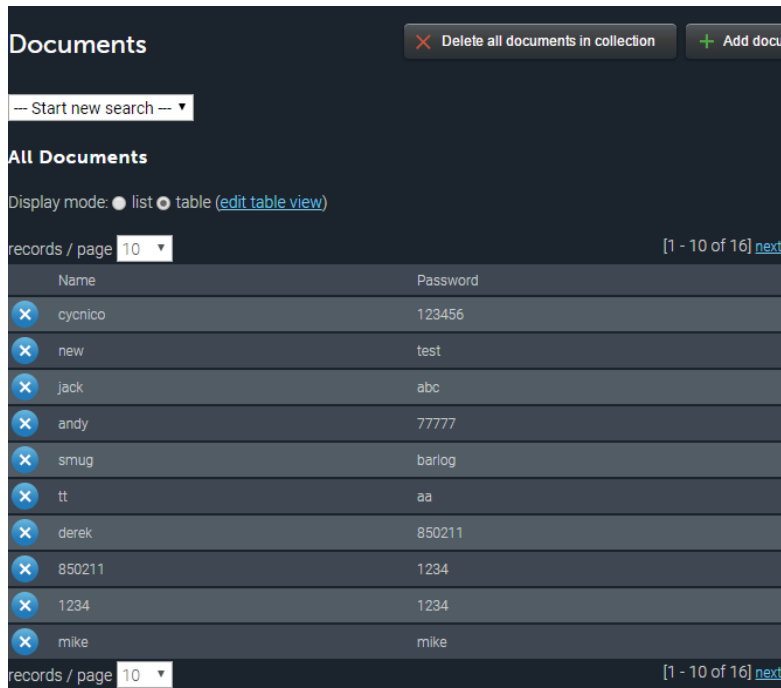
  var profiles = db.collection('userProfile'); //user profiles
  var records = db.collection('talkingRecord'); //talking records
```

在建立連線後可由 server 端存取 database，其中 `userProfile` 記錄使用者的帳號資料，`talkingRecord` 則是之前的歷史聊天紀錄。

## 2. 帳號密碼系統:

進入聊天室後，登入畫面為表單的形式，先輸入使用者名稱，在輸入使用者密碼。如果第一次登入，就會自動註冊，輸入的姓名與密碼會被記錄在資料庫中，供日後使用。

帳號密碼的資料庫由雲端的 **mongoDB** 管理，以 **json** 格式存放，只有資料庫的管理者能夠 **access**。

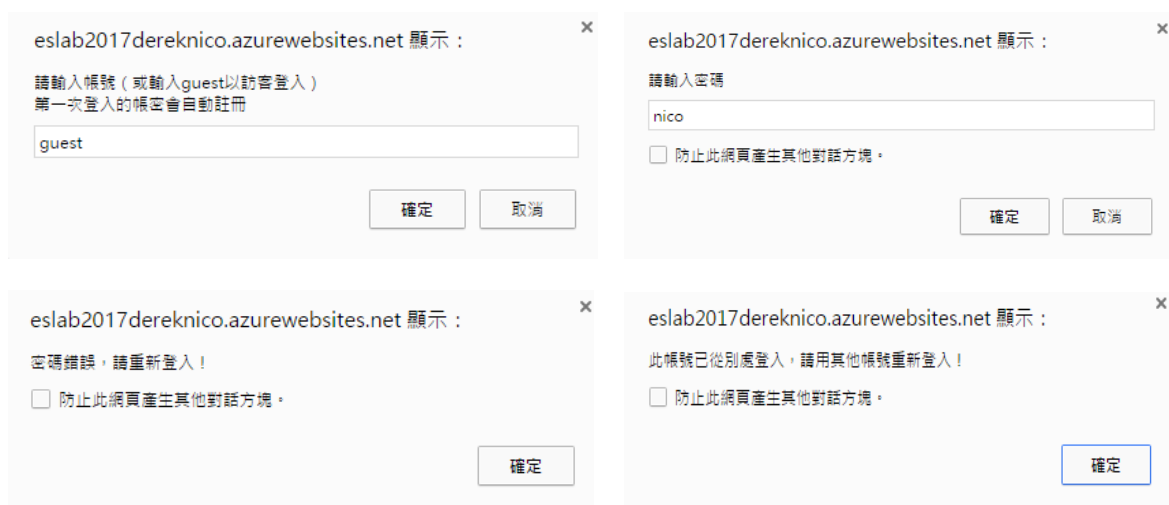


The screenshot shows the MongoDB Documents interface. At the top, there are buttons for 'Delete all documents in collection' and 'Add document'. Below that is a search bar with the text 'Start new search'. The main section is titled 'All Documents' and shows a table of documents. The table has two columns: 'Name' and 'Password'. There are 16 documents in total, with the first 10 displayed. Each document has a blue 'x' icon in the first column. The documents are as follows:

Name	Password
cynico	123456
new	test
jack	abc
andy	77777
smug	barlog
tt	aa
derek	850211
850211	1234
1234	1234
mike	mike

可由 mlab 的網站  
介面直接管理  
MongoDB 的資料

## 登入介面：



The four screenshots show the login interface for 'eslab2017dereknico.azurewebsites.net'. Each window has a title bar with a close button (X).

- Top Left:** The initial login screen. It says '請輸入帳號 (或輸入guest以訪客登入)' and '第一次登入的帳號會自動註冊'. There is a text input field containing 'guest' and two buttons: '確定' and '取消'.
- Top Right:** The screen after entering a password. It says '請輸入密碼' and has a text input field containing 'nico'. Below the field is a checkbox labeled '防止此網頁產生其他對話方塊'. There are '確定' and '取消' buttons.
- Bottom Left:** The screen after an incorrect password. It says '密碼錯誤，請重新登入！' and has a checkbox labeled '防止此網頁產生其他對話方塊'. There is a '確定' button.
- Bottom Right:** The screen after a successful login. It says '此帳號已從別處登入，請用其他帳號重新登入！' and has a checkbox labeled '防止此網頁產生其他對話方塊'. There is a '確定' button.

### 3. 即時聊天系統

即時聊天系統是本聊天室最主要的功能，能夠多人聊天並即時反應，不用重新整理就能看到新的訊息跳出來，自己的訊息會在畫面右側，其他人的訊息在左側。訊息的發送者，訊息內容，發送時間，存在雲端上的 **mongodb**，以 **json** 格式存放。

client	server
<pre>\$('#textbox').submit(function() {   if (\$('#m').val() != '') {     socket.emit('chat message', \$('#m').val());     \$('#m').val(''); // clean input   }   return false; });</pre>	<pre>socket.on('chat message', function(msg) { //receive msg from client   console.log(socket.username + ":" + msg);   var mytime = mygetTime();   io.emit('chat message', {     u_name: socket.username,     u_word: msg,     u_time: mytime   }); //when 'chat message'   records.insert({ u_name: socket.username, u_word: msg, u_time: mytime }); });</pre>

### 4. 對話記錄保存

每當 **guest** 之外的使用者登入時，此聊天室會連結 **mongodb** 中的聊天記錄，把先前的聊天訊息載入，代表即使沒在線上，也能看到其他人的對話。因此此聊天室支援離線留言的功能，即使對方不在線上，在此多人對話框發出訊息，對方上線後依然能夠看到，而且可以看到發送者，訊息內容，訊息發送時間。

### 5. 在線名單

在對話框的右側有欄在線名單，此在線名單也有即時性，不必重新整理就會更新，在線名單會顯示在線上的人，同時在線名單也會標明使用者本人，會在使用者後面加上(You)。



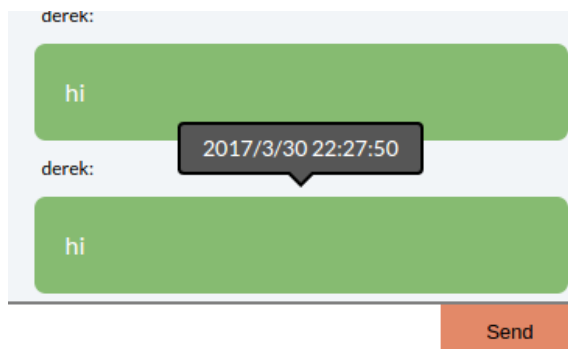
```
var getuserlist = function() {  
  const userList = [];  
  for (let i = 0; i < clientlist.length; i += 1)  
    userList[i] = clientlist[i].username;  
  return userList;  
};
```

在有人登入/登出時，server 重新更新 **userlist** 後傳給所有 **clients**

### 6. 查看訊息時間

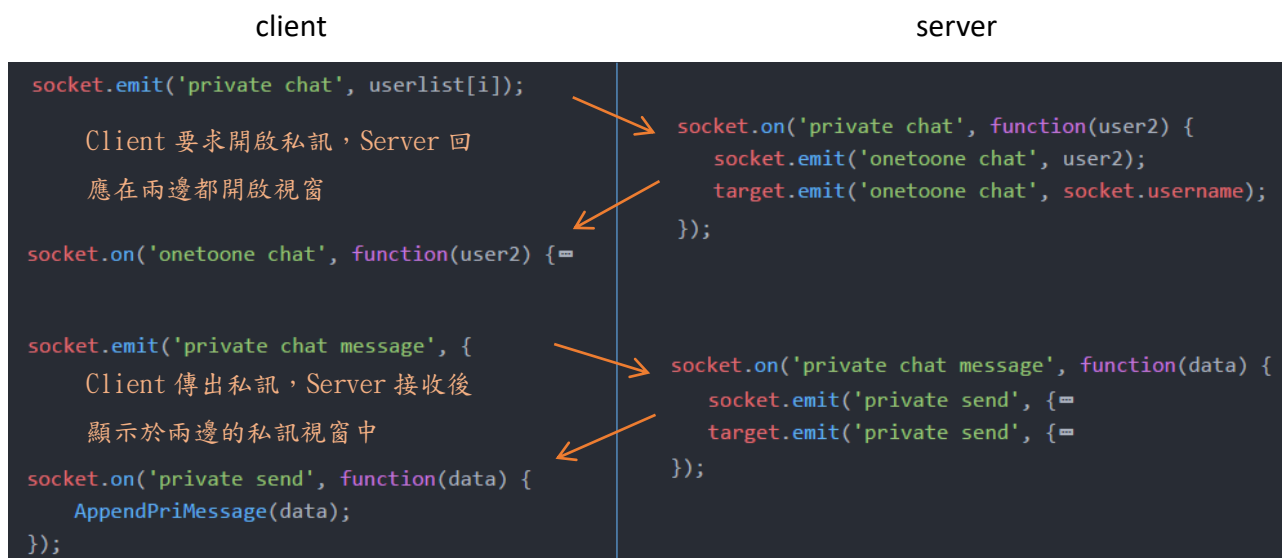
同 3 所敘述，聊天的記錄會被存放在雲端的 **mongodb**，因此聊天時間是有被保存的，即使是歷史記錄的訊息，也都有辦法顯示時間。顯示時間的方式，是利用 **tooltipster**，採用滑鼠互動模式。每當滑鼠移到訊息上方，就會有個小視窗顯示訊息的時間，當滑鼠移開，小視窗又消失。

Tooltipster 效果：



## 7. 單人即時聊天系統

單人私訊聊天的部分，我們在每次更新在線名單時，同時附加一個 **click** 的監聽事件在每個 **username** 上，當 **client** 點擊時就會對 **server** 發出一個 **private chat** 的請求，**server** 會同時回應給要求者與被要求的目標，在兩個 **client** 都開啟一個私訊視窗，視窗上會顯示你目前私訊對象的 **username**，使用者便可透過這個視窗與其他使用者進行單人私訊聊天。



## 未來可擴充之功能

1. 個人私訊時跳出來的對話框，並沒有在頁面上關閉對話框的功能，所以跟很多人聊天時，會越開越多。只要加上個 button，按下 button 就清理掉個人私訊時新增的 innerHTML，就能解決此問題，這部分再多點時間即可完成。
2. 登出系統尚未完成，現在的情況是關閉頁面即自動登出，如果增加一個登出的按鈕，然後用 cookie 紀錄登入的狀況，除非按了登出，否則不關閉瀏覽器即不自動登出，這樣與現行流行的聊天程式較為相近。我們有嘗試用 cookie 記錄一些登入相關的資料，不過後來時間趕，就沒做下去完整登出系統，cookie 相關的 code 就被先拿掉。
3. 這次聊天室的實作，大部分我們都是自己手刻，沒有用到 React，第一次接觸 javascript，花很多時間學習，發現有 React 時距離死線已經不遠，有點可惜，如果善加利用，應該可以讓聊天室的介面更加漂亮。

## 心得

我們以前都幾乎沒接觸過 javascript 與 web programming 相關技術，第一次接觸就要實作聊天室，覺得很新奇，有很多時間都在研究一些基本的概念，也花很多時間去手刻 HTML 等等。從一開始幾乎不知道要做甚麼，到有多人聊天功能、雲端的資料庫、私訊功能，最後甚至把 server 送上雲端，離開了 localhost，不同台電腦也能聊天，更像實用的聊天室了，覺得相當有成就感。總之，這次在實作中學習，覺得收穫豐富。

值得注意的是：mongoDB 免費版只能存 500MB，中途使用 heroku 免費版也受到各種限制，最終放棄使用 heroku。Microsoft Azure 則是因為參加活動獲得試用帳戶，內有 2000 多的餘額，真的多虧有 Azure 我們才成功把 server 放到網路，還能利用 Azure 做良好的更新與維護，自動與 github 同步。我們發現，這些雲端相關運用的資源，大多不是免費，如果真的有心想做規模大一點的東西放到網路上，又希望有方便的使用介面與完整的功能，大概得花錢在這些雲端資源上。

Thanks for your reading!!!