嵌入式系統實驗

實驗一報告

聊天室 app

組別：16

組員：b03505006 蔡宇傑

b03901164 鄧惇益

教授：鄭振牟

壹、 功能

一、登入頁面

在命令提示字元使用 node 開啟 index. js 檔，再開啟瀏覽器輸入正確位址，可以看到登入頁面，若為初次登入使用者，自行輸入帳號及密碼即可自動註冊，如輸入之帳號已被別人使用過，則會跳出警告視窗要求重新輸入。

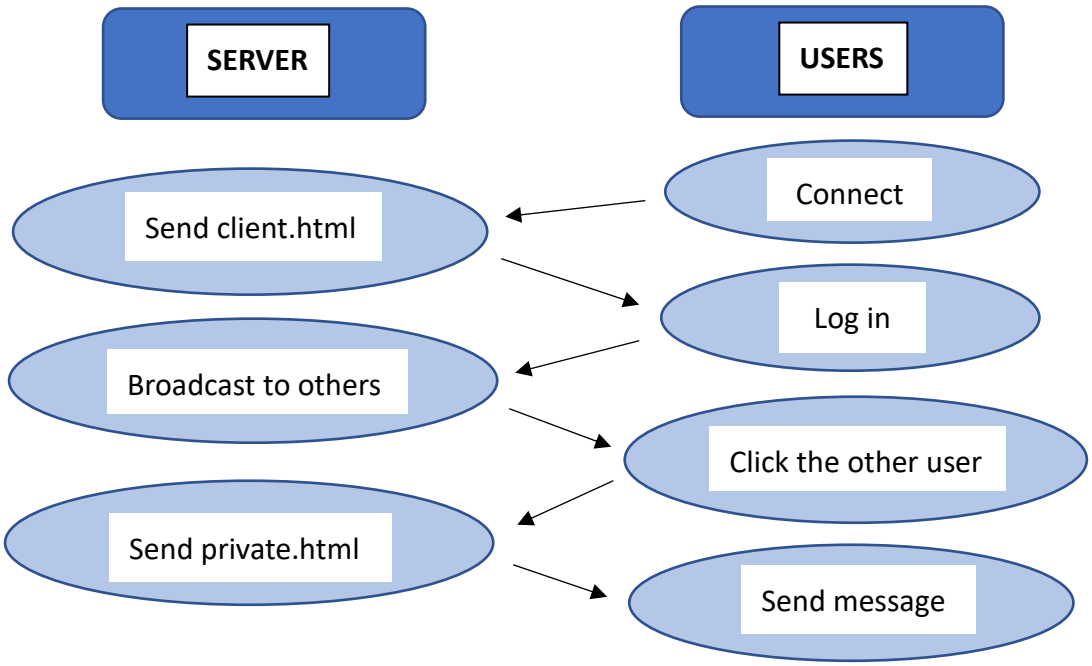
二、群體聊天介面

成功登入後，會切換到群體聊天室，主要分為三大區塊，最下方可輸入訊息，左方顯示聊天紀錄，右方則顯示當前線上的使用者名單。在聊天框中，自己打的訊息會顯示靠右，別人的訊息則會連同 ID 一起顯示靠左；線上名單則是依照登入的時間先後順序顯示，若有人加入或離開聊天室，聊天框中會顯示靠左的系統訊息告知剩餘使用者，並即時更新線上名單。

三、私訊功能

在群聊介面中，點選線上名單中的使用者 ID 將視為對該使用者啟用私訊功能，兩邊都自動跳出一個新的聊天視窗，在這些新聊天視窗中所輸入的聊天訊息，其顯示的格式同群體聊天，但不會顯示在群體聊天介面中。另外，同一位使用者可同時被不同使用者私訊，每有一位使用者便會新增一個視窗。當有一方將私訊視窗關閉後，另一邊的視窗將會顯示系統訊息通知對象離開了私聊，並於五秒鐘後自動關閉視窗。

四、流程示意圖



貳、 主要架構

一、主要程式之檔案的彼此關係

(a) Index.js：此為主程式，同時也是 Server。
(b) Client.html：負責群聊頁面的 style 等設定，並和 Server 溝通。
(c) Private.html：私聊頁面，由 Client.html 開啟並連上 Server。
(d) Account.csv：儲存帳號和密碼，並以 SHA256 的方式加密，

二、帳號密碼之儲存

由於我們是將密碼存在本機，為了避免被他人輕易得知密碼而盜取帳密資料，我們為密碼進行了加密的動作。當使用者輸入帳號密碼後，Server 會去查找並判斷使用者為第一次加入註冊或再次登入，再將結果傳給前端，使得聊天窗口顯示出不同的系統訊息。

三、登入帳密以及切換至群聊

登入的介面及群體聊天的介面都是存在 client.html，放入不同的 div 區塊中，在 style 中設定外觀與顯示位置的初始值，其中，預設讓群體聊天區塊隱藏，使得使用者一打開時會先看到登入畫面，再利用「addme」標籤與後端溝通，將登入參數傳給伺服器，判斷帳號是否存在以及密碼的正確性，再將結果回傳給前端觸發相對應的函數印出不同的系統訊息或警告視窗，若是登入或註冊成功，則改變登入介面與群體聊天區塊的顯示順序以達到切換的效果，切換部分的程式碼如下：

```
92          socket.on('check id', function(username, data){
93              loginmain.style.display='none';
94              chatroom.style.display='inline';
95              my_id=username;
96              $('#messages').append($('<p align="left">').text(data));
97          });
```

四、訊息傳遞

輸出部分，我們利用 html 內建的 form 來獲得資料，並配合 button 來觸發函數以傳送訊息，這些 form 及 button 被包入 div 並放入對應介面的母 div 中，總共有分 login 跟 message，前者負責傳遞登入資訊，後者負責傳送聊天訊息。當伺服器收到聊天訊息時，便向所有視窗丟出該訊息以及傳出此訊息的使用者 ID，不同的群聊窗口再根據此 ID 判斷該訊息要印在右側還是左側。程式碼見下頁：

```
61      $(function () {
62          var socket = io.connect('/');
63          socket.on('connect', function(){
64              $('#loginform').submit(function(){
65                  socket.emit('addme', $('#usernameinput').val(), $('#passwordinput').val());
66                  //$('#usernameinput').val('');
67                  $('#passwordinput').val('');
68                  return false;
69              });
70          });
```

五、更新線上名單

線上名單被包入一個 div 中置於右邊，每當後端偵測到有人登入成功或斷線，便呼叫名單更新函數，由於我們希望做到私訊功能，因此我們將每個 ID 都用 button 來顯示負責事件的觸發。在下頁的程式碼中，第 100 行負責將整個 div 刷新，用空字元來覆蓋舊有的線上名單。之後根據名單總人數來跑迴圈，迴圈內新增 button 屬性的 element，並且將其 value 設置為帳號名稱，方便後續私訊事件的觸發所用(於下一分項詳述)，第 107 行便是設定該按鍵被按下後須觸發的函數。最後，由於自己不需私訊自己，因此我們不用按鍵來表示，用 else 將其分開，全程式碼如下：

```
99          socket.on('updatenames', function(data){
100             document.getElementById('users').innerHTML="";
101             for(var d = 0; d < data.length; d++ ) {
102                 if(data[d]!==my_id)
103                   { var dc_but=document.createElement("button");
104                     var t = document.createTextNode(data[d]);       // Create a text node
105                     dc_but.appendChild(t);
106                     dc_but.value=data[d];
107                     dc_but.onclick = Test;
108                     $('div#users').append(dc_but);
109                     $('div#users').append("<br />");
110                     }
111                 else
112                   { var sb = data[d] +"(You)" + "<br />";
113                      $('div#users').append(sb);
114                   }
```

六、私訊功能

當私訊按鈕被按下時會觸發函數，開啟新的聊天視窗並向伺服器發送私訊要求，同時傳遞帳號名稱，如第五項所述，按鈕內的值為使用者名稱，前端可將此值傳給伺服器來找尋對應的使用者，使其也開啟新的私訊視窗。所有前端若要開啟私訊視窗，都使用 private.html 檔，在訊息傳送的部分與 client.html 雷同，唯有部分標籤為了與群聊做出區分而相異。因為所有私訊窗口所使用的標籤是相同的，因此需記住呼叫與被呼叫的使用者 ID，使得各私訊間不會互相干擾，此呼應前述傳遞帳號名稱的部分。

ID 之設定程式碼如下：

```
124         socket.on('private_message start', function(trans, receive){
125             if(my_id===receive){
126                 chat_partner=trans;
127                 window.open('private');
128             }
129             if(my_id===trans){
130                 chat_partner=receive;
131                 window.open('private');
132             }
133         });
```

七、離線

當伺服器偵測到有使用者關閉視窗(disconnect)後，便會做出相對應的動作：

1. 使用者將群聊關閉：在伺服器中有一個陣列儲存了在線者的名單，當有人斷線時，便將該名單中的對應使用者移除，再向其他使用者的群聊介面顯示離線系統訊息。
2. 使用者將私聊關閉：私聊的 html 內存了聊天對象的 ID，將此 ID 傳給伺服器令其搜尋對應的私聊視窗，使其顯示離線，並呼叫函數印出頁面即將關閉的提示訊息，時間到後自動關閉。

參、 實作時最初遇到之問題與解決

一、使字體自由靠右或靠左顯示

因為最初版本是使用 list，所以不容易設定顯示位置，最後我們發現使用<p>元件，其位置設定較為簡便，且背景樣貌也可以藉由設定其 style 來進行美化。

二、各區塊之顯示位置

原先我們並沒有用到許多的 div 區塊分類，而是一起寫在 body 之中，後來發現無法將格式設定分開，且不管怎麼調整參數，線上名單的顯示窗格都無法處於正確的位置。在上網搜尋後，才發現 div 區塊的強大之處，最後得以整合出聊天介面，並把一些區塊間明顯的縫隙消掉。

三、登入頁面與群聊之切換

我們想要在進入聊天室之前，設計一個登入頁面窗口，起初我們打算用不同的 html 檔來解決，但後來發現不好實行導致進度停滯。後來在調整區塊顯示發現了 div 的方便之處，因此決定同樣利用 div 的顯示隱藏來達到切換效果，此法的優點是較為簡易，但缺點是在登入界面時聊天室其實就已經在顯示，安全性較為不足。

四、加入私訊功能

完成群聊功能後，client.html 檔中的 div 結構已經開始有點複雜，因此我們決定再次嘗試使用新的 html 檔，並另外配置一個網址為私訊功能所用。首先遇到的問題便是如何觸發新的 html 檔，我們的構想是點擊線上名單上的名字來叫出私訊視窗，一開始我們使用超連結，但後來緊接著發現新的私訊視窗與呼叫者間的連線增為多對後便難以維繫。後來我們決定將原本純文字顯示的線上名單改為按鈕顯示，使其同時有觸發以及傳遞訊息的功能，讓新開的私聊視窗方便記錄是那些使用者呼叫的，缺點就是在剛跳出私聊視窗時，Server 需要花費資源來處理兩邊私聊的聯繫。

肆、 參考資料

(a)http://www.blueshop.com.tw/board/FUM20041006161839LRJ/BRD20111004234353G96.html

(b)http://iosdevelopersnote.blogspot.tw/2012/09/socketio.html

(c)https://socket.io/get-started/chat/