一、檔案架構

這次的作業分為 6 個主要檔案，分別是: app.js , httpserver.js , main.js ,main.css , data.db , index.html

app.js
負責處理 client 端(main.js)的 request 以及 server 端(httpserver.js)的 response，藉由 set_handler 這個 funtion 的兩個 input，來作為 client 與 server 之間的橋樑

httpserver.js
作為 server 端，當中最重要的 function 就是 do_respond_to_an_HTTP_request，他負責針對來自 client 端的 request 做出相應的 respond，其中，responder 負責回應 respond，而 reqhandler 則負責解析 request

main.js
作為客戶端，裡面使用 DOM 的 API 來幫助我們將定義 events, buttons 等等的 elements，而每一個 button (event)都有一個相對應的 listener，藉由呼叫 http_post 來送出 request，同時操作一些客戶端的 element

data.db
裡面儲存了 JSON 格式的資料，分別是 nickname, emoji, text, time。這些資料是藉由 app.js 當中的 do_submit 接收到 request_body 之後，加以解析之後寫進檔案當中的；而這些資料也會藉由 app.js 的 do_readall 和 main.js 的 data_from_server_callback_readall 來顯示在 html 的對話框當中

index.html
這個檔案是構成網頁外表的重要檔案，同時，他定義了元素 id、性質、大小等特性，如果這部分做得好，main.js 就能夠非常容易地透過 DOM 來操縱 html 上面的元素

main.css
CSS 與<script>標籤有些類似，他能夠統一管理 html 當中的元素，給予更多細節上的性質設定

二、功能 function 架構

```
var configs = function (set_port, set_hostname, set_handler) {
  set_port(2015);
  set_hostname('127.0.0.1');
  set_handler('GET /', do_output_html);
  set_handler('GET /index.html', do_output_html);
  set_handler('GET /signup.html', do_output_signup);
  set_handler('GET /login.html', do_output_login);
  set_handler('GET /main.css', do_output_css);
  set_handler('GET /main.js', do_output_js);
  set_handler('GET /jquery.js', do_output_jquery);
  set_handler('GET /favicon.ico', do_output_favicon);
  set_handler('POST /echo', do_echo);
  set_handler('POST /submit', do_submit);
  set_handler('POST /read_all', do_readall);
  set_handler('POST /cleanup', do_cleanup);
  set_handler('POST /filter',do_filter);
  set_handler('POST /nickname',do_nickname);
  set_handler('POST /signup',do_signup);
  set_handler('POST /login',do_login);
  set_handler('POST /userdata',do_user);
};
```

以上為所有 http request，分別對應了各個按鈕的功能

POST /echo：

原有功能，為傳回一個與 request body 相同的 response，現在已沒有再使用

POST /submit：

將 request body(包含了 nickname, text, emoji)解析過後加上 time 的資訊，包裝成 JSON 作為 response，並把資料寫入 data.db

POST /read_all：

把 data.db 讀近來，並整個作為 respond 回傳

POST /cleanup：

清空 data.db 檔案

POST /filter：

把 data.db 讀近來，經過篩選過後(與 request body 比對)，將相同 nickname 的 JSON 資料回傳

POST /nickname：

Default 的 request，回傳現有的所有 nickname，讓 client 端能顯示有哪些 nickname 來篩選

POST /signup：

判斷用戶名稱是否註冊過，若否，就將 request body 寫入 user.db

POST /login：

將 request 與 user.db 中的資料比對，並回傳比對結果

POST /userdata：

用來 login 用的，作為判斷現在的使用者是誰

三、 加分題

除了作業規定的功能之外我們還增加了 bonus1, bonus3 以及圖片顯示功能:

Bonus1

Bonus1 的 nickname 篩選功能，在 main.js 的 client 端我們透過 listener 來發送出”nickname_selector”的 request，同時呼叫 data_from_server_callback_readall 來刷新留言板，以顯示特定的 nickname 和他的留言

```
<form action="" method=POST id="form1" name="form1">
<select id="nickname_selector" >
```

```
selector_elm.addEventListener('change',function (){
 console.log(selector_elm.value);
 http_post('/filter',selector_elm.value,data_from_server_callback_readall);

});
```

而篩選的功能則是放在 app.js 的 do_filter fuunction 當中，他會從 data.db 檔案裏面讀取 JSON 資料，經過與 request 內容比對後，把篩選後的 JSON 資料 respond 回 client 端

```
var do_filter = function (send_response,request_body,request_headers){
 var username = request_body.toString();
 var content_type_default = 'application/octet-stream';
 var content_type = request_headers['content-type'] || content_type_default;
 console.log("filter");

 require('fs').readFile('data.db',function (err, data) {
  if (err) throw err;
  else{
   var jsonobj = JSON.parse(data.toString());
   var send_obj = new Array;
   for (var i =0; i<jsonobj.length;i=i+1){
    if(jsonobj[i].nickname === username)
     send_obj.push(jsonobj[i]);
   }
   var json_str = JSON.stringify(send_obj);
   console.log(json_str);
   send_response(new Buffer(json_str), {'Content-Type': content_type});

  }
 });
};
```

Bonus2

Bonus2 的註冊登入功能，我們新增了兩個檔案來進行實現，分別是 signup.html 以及 login.html。

1. Signup

signup 時，client 端 main.js 的 signup listener 會開啟新的網頁視窗

當你輸入完符合規定的帳號密碼後，按下”submit”，signup.html 的 submit listener 會將帳號密碼等資訊打包成 JSON 並將 request 傳給 server

```
var do_signup = function (send_response, request_body, request_headers){
var jsonobj = JSON.parse(request_body.toString());
var content_type_default = 'application/octet-stream';
var content_type = request_headers['content-type'] || content_type_default;

var nick_arr = new Array;
require('fs').readFile('user.db',function (err, data){
 if(err) throw err;
 else{
  if(data.toString().length === 0 ){
  }
  else{
   console.log(data);
   nick_arr = Array.prototype.slice.call(JSON.parse(data.toString()));
  }
  //console.log(nick_arr);
  var check = true;
  for(var i = 0;i<nick_arr.length;i++){
   if(nick_arr[i].username === jsonobj.username){
    console.log(nick_arr[i].username);
    send_response(new Buffer("username is used!!!"),{'Content-Type': content_type});
    check = false;
   }
  }
  if(check){
  nick_arr.push(jsonobj);
  send_response(new Buffer("OK"),{'Content-Type': content_type});
  require('fs').writeFile('user.db',JSON.stringify(nick_arr),function (err, data){
    if(err) throw err;
    else
     console.log("success write");
  });
  }
 }
)
```

另一方面，app.js 的_do_signup 會根據 request 將 username(也就是 nickname)還有相應的 password 以 JSON 的格式存入 user.db 當中。

```
submit.addEventListener('click',function(){
 var reg_nickname = /^[a-z0-9]{3,10}$/;
 var reg_password = /^[a-z0-9]{5,12}$/;
 if(!reg_nickname.test(username.value)){alert("username fomat error!!!");return;}
 if(!reg_password.test(password.value)){alert("password fomat error!!!");return;}
 var jsonobj = {"username":username.value,"password":password.value};
 var jsonstr = JSON.stringify(jsonobj);
 console.log(jsonstr);

 http_post('/signup',jsonstr,function (result){
  if(result === "OK") {console.log("it'sOK");close()}
  else
   alert(result);

 //close();
 });
});
```

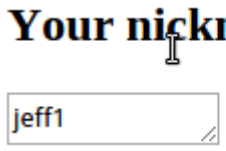
2. Login

而 login 時，當你按下 login button 後，client 端 main.js 的 login listener 會開啟新的網頁視窗，當你輸入正確的帳密之後，會將 request 交由 app.js 的 do_login 來進行分析比對


```
var do_login = function (send_response, request_body, request_headers){
 var jsonobj = JSON.parse(request_body.toString());
 var content_type_default = 'application/octet-stream';
 var content_type = request_headers['content-type'] || content_type_default;

 var nick_arr = new Array;
 require('fs').readFile('user.db',function (err, data){
  if(err) throw err;
  else {
   if(data.toString().length === 0 ){
   }
   else{
    console.log(data);
    nick_arr = Array.prototype.slice.call(JSON.parse(data.toString()));
   }
   //console.log(nick_arr);
   var check = true;
   for(var i = 0;i < nick_arr.length;i++){
    if(nick_arr[i].username === jsonobj.username){
     if(nick_arr[i].password === jsonobj.password){
      user = nick_arr[i].username;
      send_response(new Buffer("OK"),{'Content-Type': content_type});
      return;
     }
     else{
      console.log(nick_arr[i].username);
      send_response(new Buffer("Incorrect password!!!"),{'Content-Type': content_type});
      check = false;
      return;
     }
    }
    else if(i === nick_arr.length-1)
     send_response(new Buffer("No username!!!"),{'Content-Type': content_type});
```

如果正確無誤，login listener 當中的這行`nickname_textarea_elm.value = result;`就會將 nickname 寫到 nickname 的欄位(這裡的 textarea 設定為 readonly 因此必須要登入才能使用)，如下圖



四、 功能介紹

1. 登入系統：

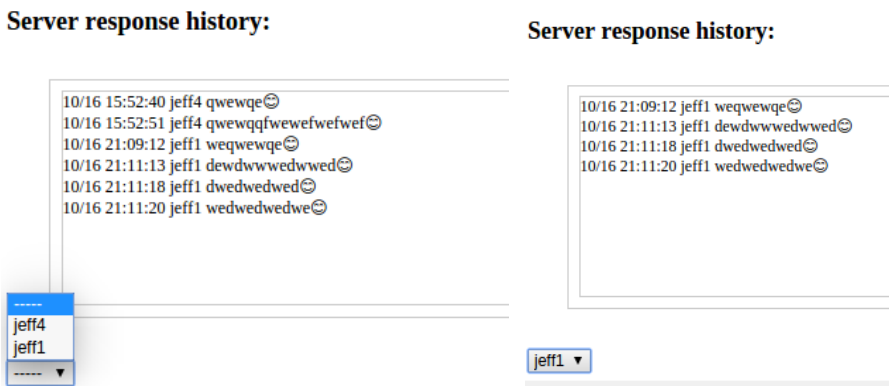
目前僅支援飛成簡單的註冊系統與登入系統，都是利用 json 包裝一個 username&password，並沒有經過任何加密，也是很簡單的確認是否有相同的 username 在 server 中，還有確認密碼是否完全正確，格式的確認則是在 html 端就篩選完畢，主要利用了/POST signup/POST login 兩個 http request 並針對與資料庫比對後的結果回傳 response。

第一次使用時，需要先註冊後登入，分別有 signup button 和 login button，按下按鈕後都會跳出一個視窗，輸入帳號(nickname)3~10 個英文或數字、密碼 5~12 個英文或數字，就能開始使用留言版

2. 留言板部分：

submit 會傳回由 nickname message emogi 組成的 json 並且在 server 端加上 timestamp 儲存在 data.db，readall 則會將所有 data.db 的資料傳回瀏覽器顯示在 response-area 中，clean 功能單純是為了實作上的方便，增加一個能夠把資料庫全部刪除的功能，所以只要 clean 之後再按 readall 留言版 client 端的留言就都會消失。當然，如果你已經以一個帳號登入了，你所有的留言都會是同樣的 nickname，無法改變。

而最底部有篩選用戶的功能，能夠從所有留言當中，找出特定 nnickname 的留言



3. 上傳檔案功能:

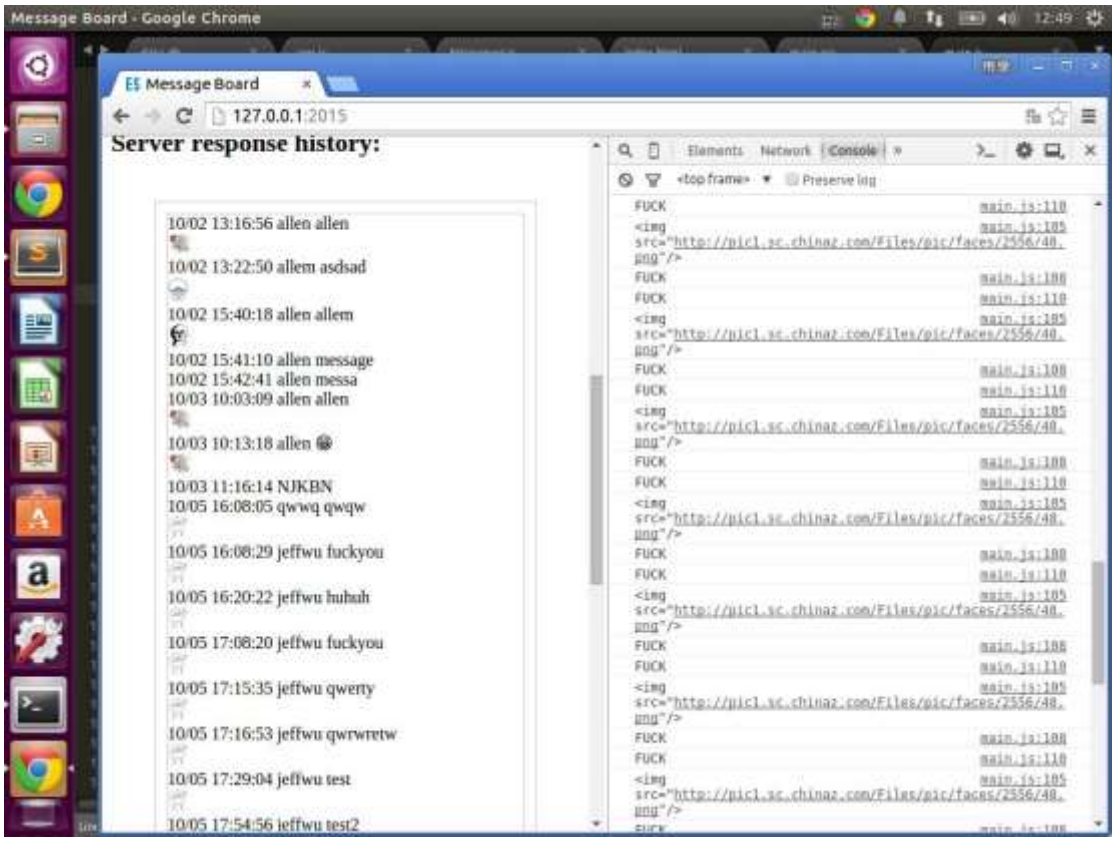
這部分由於時間和技術因素，我們尚未完全做完，目前所做到的有:

a. 可插入貼圖的 div，我們原本留言版所使用的元素是 textarea，但由於 textarea 有著諸多限制，而且無法加入圖片，因此利用 index.html 和 main.css 建立了 log-area 的區塊，可以利用 main.js 裡的 function，將圖檔 append 給留言版

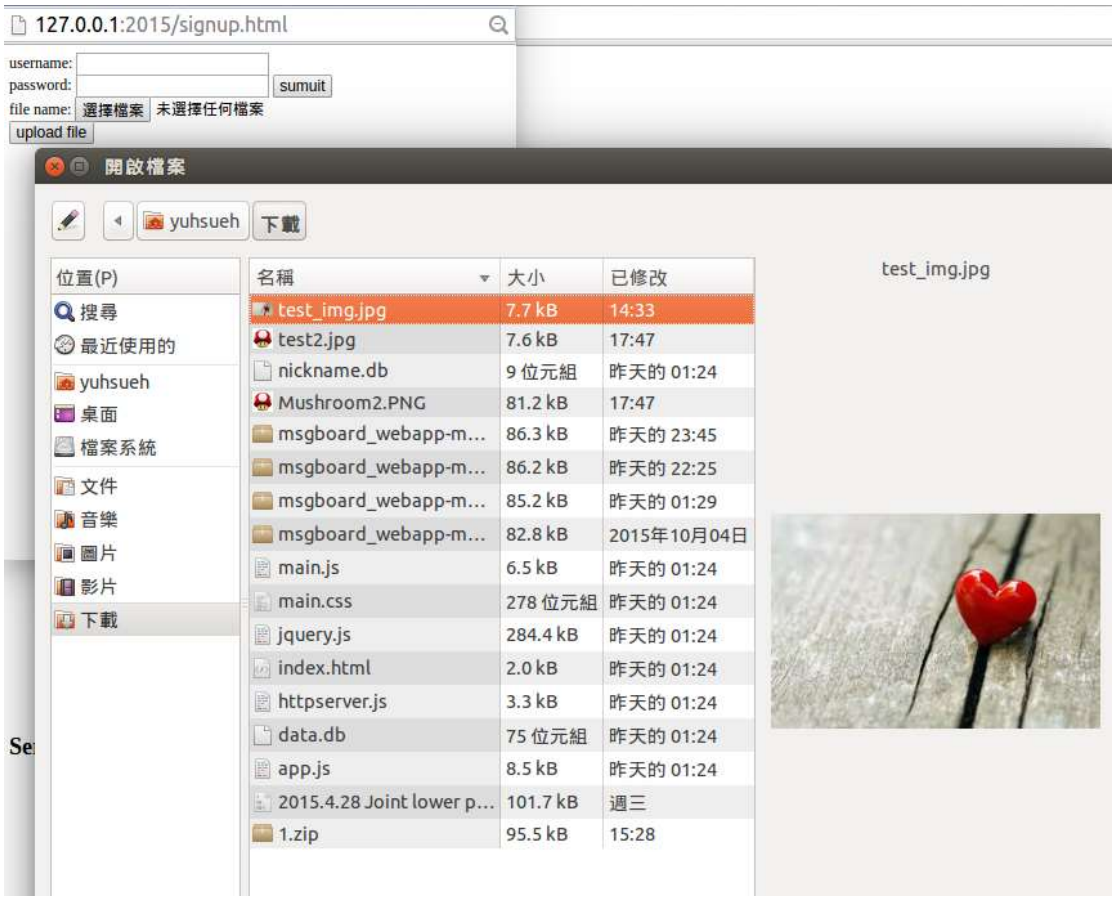
```
<div >
  <h2>Server response history:</h2>
<div id="log-area_outer">
<div id="log-area" class="response-zone" contenteditable="true"  hidefocus="true"></div>
</div>
 <form action="" method=POST id="form1" name="form1">
 <select id="nickname_selector" >
 </select>
 </form>
```

```
var x = function (pic_src){
//src = '<img src="http://cfile26.uf.tistory.com/image/2522F535538E804C29A143"/>';
 $("#log-area").append(pic_src);
 };
```

成果如下



b. 讀檔系統，目前已經可以從電腦目錄當中將想要的圖檔讀取進來了，然而卻仍然無法順利將圖檔 request 給 server 讓 server 將圖檔儲存起來，若能克服這點，就可以對應 nickname 與圖檔，做到幫用戶設定大頭貼的功能



五、心得

這次的實驗讓我們先自己從完全陌生到熟悉一個語言，發現熟悉了語言的邏輯以後，其實後面開始學習其他不同語言要比第一次快速許多，而且 JavaScript 真的是一個比 C++還要簡單的一個語言，不過在很多寫網頁的用法還是有很多不瞭解的地方，也是靠了 google 學習了很多 JS 和 html 的語法，在一開始對 http 和 JS 的架構還很不了解的時候也利用了 code cademy 學習了一些基礎，其實在現在網路這麼發達的世代，學習已經不再僅止於非得要上課了，許多網路資源只要我們懂得利用，也能達到很好的學習效果，這次實驗雖然只是往後實驗二實驗三的基礎，但很重要的是讓我們對於一個新的語言產生興趣，與對於 JS 所能做到的事情感到好奇，我想這也會是我們這學期學習新知與不斷探索下去的動力。

謝謝助教!