雲端機房管理系統－CRMS

專題動機：

為了使各位辛苦的網管同仁能夠不再為自家的硬體設備管理感到困擾，我們決定要將進銷存管理的概念引入，並結合現代的智慧型手機讓網管們可以做到「移動式管理」的層次。就算人不在自己的工作場所，也能夠及時了解機房的狀態，更可以利用建立起來的系統在外快速查詢硬體的詳細資訊，讓網管們進行以往繁複的硬體排錯能夠更加順利。

專題目的：

此專題將分成用戶端及伺服器兩部分來進行，用戶端的目標是開發一套能夠在Android系統上運行的查詢介面，利用Android的圖像辨識便能夠及時讀取事先做好QR code的bar扣，如果事先知道編碼，也能夠以輸入的方式作查詢，此用戶端介面也會有web版本，方便沒有智慧型手機的同仁能夠直接使用電腦進行操作。

伺服器端則需要資料庫的建立，以及伺服器的管理。

當這套系統建立起來後，能夠使網管們更有效率的工作。

專題範圍：

資料庫：

負責伺服端操作資料庫用API。

管理後台：

後端包括使用者帳號權限以及前端資料庫控制。

PM：

專案管理，整合統整兩組專案進度以及進度追蹤,訂定開會時間以及版本控制。

條碼辨識：

QR Code與機器序號轉換。

介面：

Android平台介面進行必要之資訊查詢。

DATA：

客戶端底層負責和伺服端溝通,以及JSON解析。

WEB：

跨平台的網頁客戶端介面,提供無法執行ANDROID應用程式之網管之替代方案。

系統需求分析

需求分析方法與工具：

於機房進行實際訪察，了解機房內部的器材及工具。

問題陳述：

如何系統性的管理機房內的設備。

功能性需求：

有登記的帳戶才能夠進行管理，必須輸入帳號密碼才能夠存取資料庫。

利用智慧型手機(Android)對QR Code進行掃描，反分析，並取得物品ID。

智慧型手機透過無線網路的連線讀取資料庫，並且能夠對設備進行「新增」、「查詢」、「移除」、「編輯」等動作。

智慧型手機能夠顯示出各個機房的設備，以便瀏覽。

WEB介面能夠利用設備ID自動生成QR Code。

WEB介面也能夠對設備進行「新增」、「查詢」、「移除」、「編輯」等動作。

WEB介面能夠對帳戶進行管理，手機端則無此功能。

設備ID要能夠自行生成，故需要特定編碼方式生成ID。

能夠對設備編輯的資訊：「品名」、「價格」。

非功能性需求：

介面操作簡易。

系統穩定。

務必確保伺服器端的運作正常。

系統操作及需求：

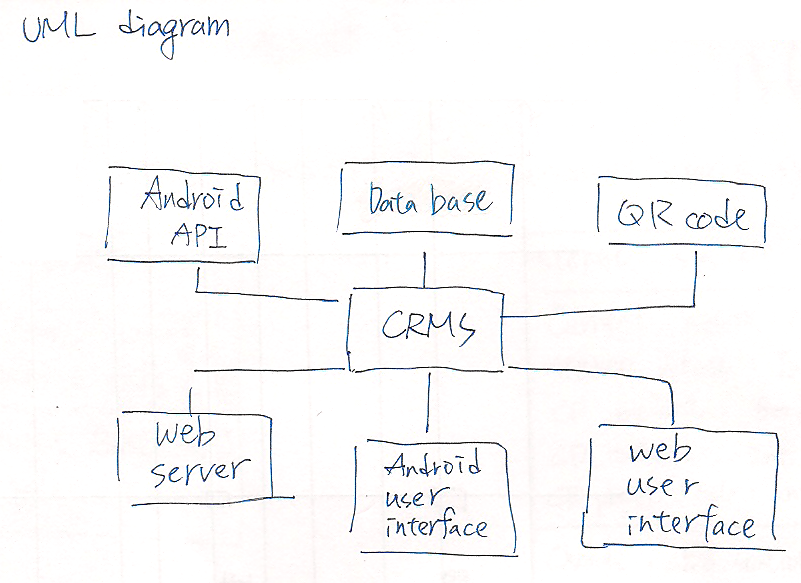
Android phones

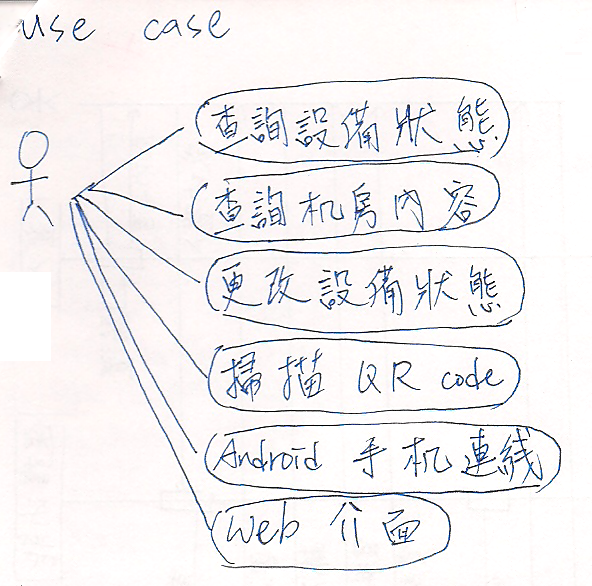
QR Code標籤。

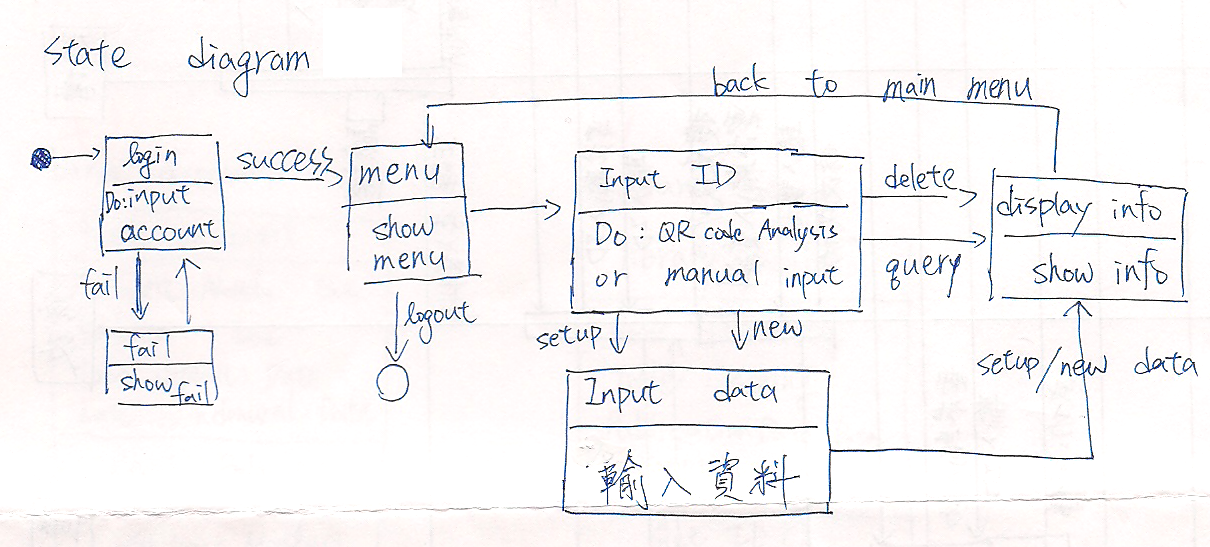
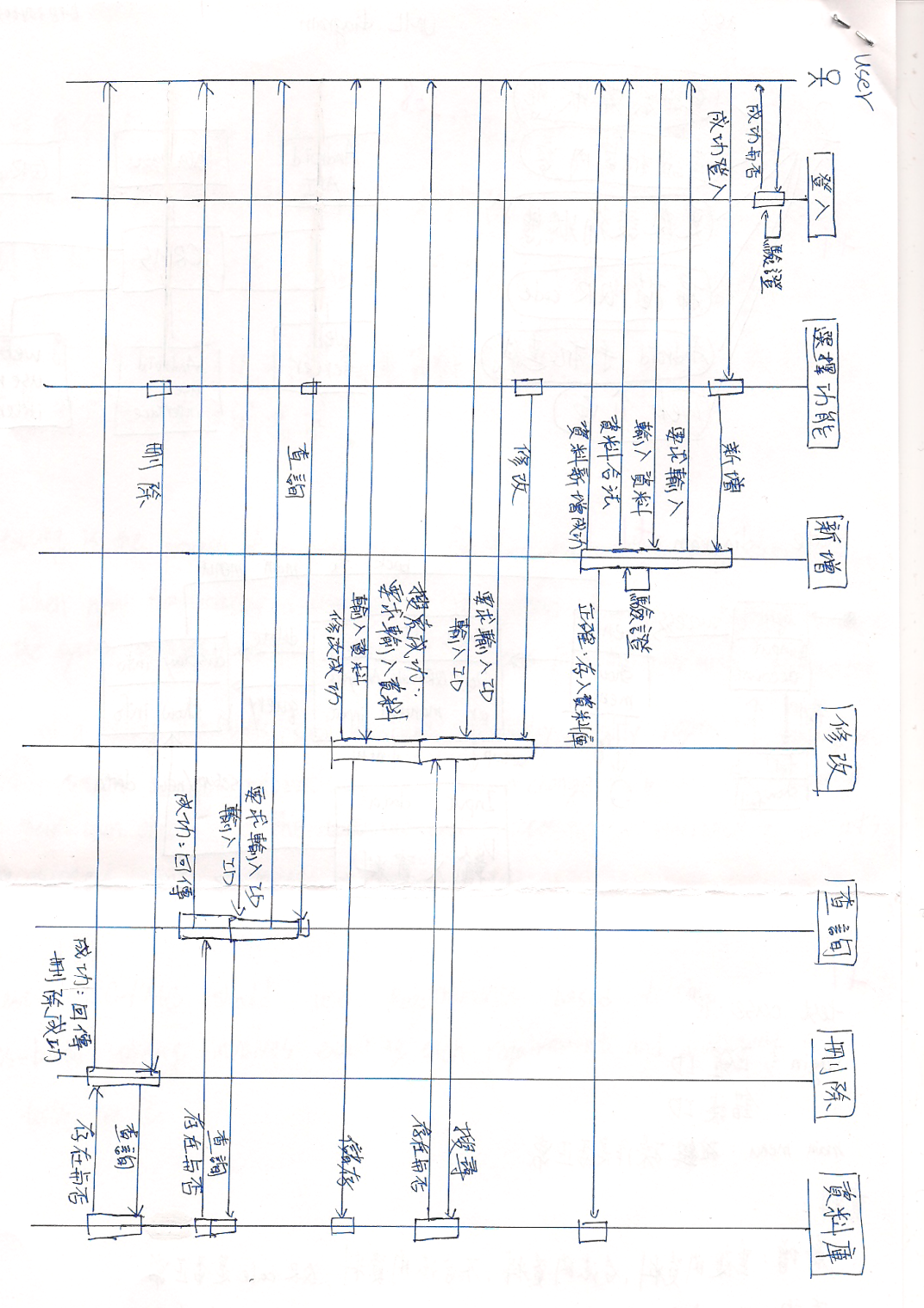
Apache網頁伺服器，及MySQL資料庫。

網頁瀏覽器。

UML diagrams







使用工具

作業系統：

Windows

Linux-Ubuntu

編輯器：

VIM

NotePad++

Eclipse

版本控制：

Git

GitHub

安卓：

Android

ZXing

adt

網頁技術：

jQuery

CSS

JavaScript

PHP

Google API

Ajax

REST

資料庫與伺服器：

MySQL

Json

Apache

測試工具：

Wireshark

Firebug

瀏覽器:

Safari, Firefox, Opera, Chrome

測試資料與方法

登入：

正確/錯誤之使用者名稱

正確/錯誤之密碼

安卓程式主畫面：

按鈕是否接成正常使用。

正確/錯誤資料傳輸

新增：

重複的資料

合法/不合法的資料

QR Code是否正確讀取

查詢：

於資料庫增加幾筆測資

輸入存在/不存在的查詢資料，並檢查能否正確回傳

刪除：

存在/不存在的資料

不合法的資料

修改：

存在/不存在的資料

修改的資料的合法性

資料庫：

是否正常連線

不同方式存取

多重存取

版本控制心得

我們這組測次選擇利用Git進行版本控制，和以往作業不一樣的地方是：這是一份團體的作業，我們很有可能會遇上一個或多個人同時修改一份或多個檔案。

所以用MSN或者是USB快閃傳輸的話，將會使原始碼交換產生嚴重的問題，假如是使用傳統的FTP進行原始碼管理，那有可能造成多人同時修改同個錯誤，或者是剛修改的檔案被覆蓋的慘劇，所以我們選擇了有版本控制功能的Git。

一開始的時候，由於沒有相關訓練，使用上有一些障礙，但是漸漸熟悉之後，對於commit、push以及pull的功能變得比較不那麼陌生，甚至可以快速地將伺服器部屬至多部電腦，而且只要該電腦有基本的Apache、PHP、MySQL 就可以進行測試，也就是就算電腦脫離網路，也可以進行專題。

Git除了版本控制外，還有強制要求上傳的人註解的功能，這個動作方便我們瞭解其他組員的進度，例如某某功能實作完成，或者是修正某某錯誤，甚至可直接看出他改了那些行，此時我們只需要對他修改的功能進行除錯即可，此時我們只需要對他修改的功能進行除錯，大大降低了除錯的複雜度。

網頁技術使用心得

在以往，我們必須利用JavaScript慢慢做才能夠做到華麗的特效，甚至需要使用Flash，但是目前較新的平台，已經開始不支援Flash，其占用的系統資源較多，在行動平台上的表現不如預期。

所以我們採用JavaScript作為前端用的語言，而jQuery正是加速JavaScript開發的工具。JavaScript與Java其實是根本上不同的東西，就如同印度和印地安人一樣，他們只是名字相近罷了。JavaScript可以直接控制瀏覽器行為，使得部分負載可以移至客戶端，像是基本的資料型別檢查　特效　按鈕顯示等等，這樣可以增加CS架構中C的量。

我們這次使用jQuery包含了特效以及Ajax的部分，特效包括首頁的圖片可以根據瀏覽器大小而自行縮減、使用Google API使QR code自動產生，並使圖片自行放大。Ajax的部份讓我們可以用比較便捷的方式進行登入，也可以在送出前檢查是否有欄位重複，我們也利用Ajax達到不換頁的載入資訊，也就是我們「只傳送必要的資訊」，如此可以大大節省伺服器端所需負荷的流量成本，甚至可以配合DNS的負載平衡達到雲端的效果。

另外jQuery本身算是十分輕便的語言，所以除了初期的開發成本較低，對於後面的維護也可以較其他解決方案有更低的成本。

QR Code

我們這次專題用上了QR Code，這對於台灣而言算是近兩年來才開始應用的技術，但在日本卻已流行多年，主要是日本人的高度手機依賴性，因為QR Code雖然可以儲存的東西很多，但是他通常放的地方是在海報、貼紙或平面廣告等「唯讀的資訊接觸點」，不像我們平常的廣告可以藉由使用者互動來更改其外觀，所以QR Code要放什麼資料就變得十分重要。

假如我們在QR Code裡面放入經常更改的資料，那麼則意味著我們必須經常重新製作這張QR Code，而製作一張QR Code需要一定的成本，為避免產生不必要的浪費，我們決定只放入機器的統一編號。

主要原因有其二：

1.就像身份證字號，我只要修改資料庫中此編號的相關屬性即可，貼紙放的東西越少，錯誤校正的成功率就越高，所以我們QR Code就只儲存機器的新增日期以及流水號共13碼。

2.當使用者看到一個QR Code時，他必須要有能夠掃描的設備，掃描後必須連上網路以登入主資料庫，在這個階段也可以避免，不必要的資訊洩漏，造成安全問題。

在產生QR Code的部分，我們使用GOOGLE的API，由於QR Code的演算法已經有很多實作版本，效率也都差不多，所以我們不需要重新發明一次輪子。

關於安全性的部分，由於我們只傳送數字，外流的資訊頂多讓Google得知存在此編號，所以並不構成任何威脅。

讀取的部分基於不重新發明輪子的理由，我們選擇reuse　Android上現有的套件：Zxing。考量到可能的讀取失敗，所以我們參考了一般一維條碼的解決方案－在QR Code旁放入可人工辨識的數字，讓條碼難以辨識時可以進行人工輸入。