資管二乙第一組 訪談內容

1. **受訪者姓名：**董書妤、周佳欣

2. **從系統分析與設計課程中遇到甚麼問題?**

因組內許多成員都有拖延的習慣，導致整合時間相當匆忙，debug 時間不足，常常都是期限快到的時候才完成。學姐們也有提到，濟聰老師有建議同學們在系統發表前不應再增加新功能，但因為學長姐整合及除錯時間常不足，所以常常都在快要發表時還在新增功能。

3. **從系統開發中遇到甚麼問題? 如何解決?**

學長姐在製作系統時初次嘗試使用 Firebase 和 React 這兩種語言，因此對操作尚未完全熟悉。在開發記帳功能時遇到了一些問題，導致資料庫的讀取量超過負荷，最終使得資料庫停止運作。出現這樣的問題主要是因為 Firebase 的免費版本有讀取量的限制，同時在學長姐撰寫的程式碼中存在錯誤，導致 React 的‘useEffect‘機制在每次讀取後都不斷觸發資料的讀寫操作，進而造成資料庫超載。

學長姐認為，在開始操作前應更深入了解系統架構，並在發現錯誤時逐一排除以進行除錯。同時，建議使用 GitHub 建立分支，並在本地端進行測試後再進行整合，以減少錯誤發生的可能性。

(1.) **使用 git 嗎? 遇到甚麼問題?**

有使用，但因擔心負擔過重而未建立分支，導致組員間在使用時出現程式碼上的衝突，例如其中一位組員直接上傳本地端資料而沒有更新雲端的資料，導致主要幹道上原先撰寫好的功能被覆蓋掉，得額外花時間進行處理。

(2.) **使用哪個語言、資料庫開發? 為何? 優缺點是?**

只有使用到 Firebase 和 React，且沒有再加其他的框架。使用的語言不管前後端都是以前沒有接觸過的。

	React	Firebase
優點	與 JS 和 HTML 相比，React 更簡單且模組化，能夠一次性在原始頁面進行重複元素的修改，維護相當方便。目前在業界也被廣泛應用。	屬於 Nosql，且 CRUD 等等大功能都已經存在，功能相當齊全，使用者可直接進行使用。
缺點	在未曾接觸過的情況下，需要花時間自行摸索和熟悉，並且	Firebase 使用上很便利，但有成本、限制和依賴問題。

	得完整了解架構，以減少出錯機會。(邊做邊學)	對大型應用的擴展性有挑戰。
建議	要了解一個新的語言前可以先了解它是如何運作的，了解一些概念後參考他人的文件並進行實作。	

(3.) 如何定期追蹤組員進度? 遇到甚麼問題?

整體運作屬於扁平化組織，組員遇到任何問題都會馬上提出並討論，各自會處理好自己的部分，且有餘力會互相幫助。但因為時常拖到期限前才完成，彼此間無法完整的進行進度追蹤，且在 git 上未建立分支，因此整合會花上很多時間。

4. (到目前為止，)專題遇到的最大問題是甚麼? 如何解決?

學長姐正式在製作專題時，前端和後端都換成了新的語言，因此前期花費了些許時間熟悉新語言。然而，整個製作過程遇到最大的問題是：為了實現專題預期的效果，學長姐決定加入時下新穎的 AI 工具 Microsoft azure 做使用。因為採用的是最新的架構，很難上網找到更好的參考對象以及資源來更好的運用這些服務。

5. (到目前為止，)系統開發中遇到甚麼問題? 如何解決?

學姐們指出，系統運作的流程需要經過討論，但有時討論不夠明確或者意見不一致時，可能需要重新設計。由於學長姐們的專題需要使用到 AI 技術，各人對於設計的預設想法可能存在差異，因此達成共識十分重要。透過詳細的說明可以讓自己的想法更清晰地被理解。同時，若有問題可立即提出，以便全組同學共同討論和解決。

(1.) 使用 git 嗎? 遇到甚麼問題?

有，且同時使用 Devops 來記錄以及分配代辦事項給其他組員，並以其為命名依據。此舉改善了沒有建立分支的問題。

(2.) 使用哪個語言、資料庫開發?跟 SA 使用的一樣嗎?為何?

因為專題製作上有產學合作，需要配合對方使用一樣的語言，因此前端、後端以及資料庫都是使用和先前不同的語言，前端使用的是 Vue、後端使用.net、資料庫則是使用 sqlserver。

(3.) 如何定期追蹤組員進度? 跟 SA 一樣嗎? 遇到甚麼問題?

和 SA 一樣，學長姐們沒有定期進行進度追蹤，直到開會時才能了解所有人的進展。然而，這樣的做法會增加不確定性，並導致遇到的問題被延遲解決。

6. 對各位的建議是?

-要在 Git 上建立分支，以便於後續合併。此外，分工應根據自己的系統來劃分，並且分工必須明確且公平。

-應確保每個人都清楚自己的任務，而不是假設大家的認知都是一致的。學姐們建議選出一位組長，以確保每個人知曉當前應該執行的任務。

-老師注重功能的合理性以及使用者體驗，並且很在意同學們對該作業的態度。

Deeplink 心得

深度學習-居家智慧照顧系統 心得

這場深度學習專題分享會分成三個部分，分別為簡介及技術分享、在過程中遇到的問題以及如何解決、如何將所學知識用在專題上。

目前深度學習在機器領域上是相當熱門的產業之一，但可惜的是，現今深度學習多被使用於人臉辨識，而高齡化的社會下，深度學習未被廣泛用於居家照顧上，所以學長想往未發展成熟的居家智能領域發展，並以此為專題研究，以建立居家照顧系統，來增加社會福祉。

在分享會中，學長提到機器學習與深度學習最大的不同之處是，機器學習是人們給照片和特徵讓機器去學習，而深度學習則是，除了人們給的特徵外，它還會自己去找人類沒發現的特徵。深度學習仰賴背後的資料集，只要資料集的數量越多，那深度學習能判斷出的東西也越準確。此次專題運用了三個技術，包含 LebelImg、Python、Line notify，其中 Line notify 是我們最有興趣的部份，它是一個免費且普及的軟體，在說明會中，學長利用 Line notiify 結合人臉辨識，使得當偵測到人臉關節點時，Line notify 會跳出偵測訊息，對普遍有 Line 的台灣人來說相當方便。

在遇到的問題上，學長覺得專題有三個點是最重要的，第一，**主題必須是自己有興趣的領域**，而不是隨便教授丟給你一個主題就做。第二，分配工作上不該分太明確，這樣大家都只做完自己的工作，卻不知道對方的工作而導致**成品無法組合**起來。第三，在遇到問題時，我們必須自己學會**在網路上查找解決方法**，自己解決問題，而不是等待其他人給你解答，只有不斷的自我學習，從實作中練習，才能做出一個好的專題。

訪談結果和 deeplink 分享比較後，我們可能遇到的問題&預防方法

在學長姐訪談和 deeplink 中我們得知，**組員定期進度追蹤**以及**分配工作**是最為重要的，不管是在影片亦或是現場訪問學姊，這兩件都是組中基本會的問題。

我們認為拖延的部分在我們組內不會發生，因為在之前的課上合作時，組員們都是盡快完成作業交給下一位組員去進行，甚至提早完成或當天分配完工作當天就完成，我們都很信任組員，此外我們若有時程上的問題都會第一時間提出，並由其他組員從旁協助。第二，在分配工作的問題上，我們認為在開始分配工作前，大家都必須去熟悉所有東西後，再進行分配的工作，這樣可以避免互相都不熟悉彼此的業務，此外，我們中午吃飯時也會一起吃飯並確認對方進度，以及有無可以改善的地方。

第 28 屆資服競賽得獎經驗分享-捐捐不息 心得

透過這次的分享讓我受益最多的部分是上台發表自己的作品和面對評審們的提問該如何準備和面對；首先讓我印象最深的一點是柏儒學長說**選擇上台報告的人不是用抽籤和爬梯，而是要有禮貌、能言善道且要強大的舞台魅力**，但一直以來誰上台報告的問題，我們通常是用爬梯子來選擇的，只能說學長真的是一針見血直接戳到我們的痛處了；要有一場精彩的報告除了報告者要有上述三點特質，學長還提到了很多我之前都沒想到的小撇步，像是模擬情境不斷演練，之前我在練習報告時都是自己一個人面對著 ppt 練習，但是就像學長說的，**面對著組員報告**會更貼合真正報告時的情境和情緒，讓自己更習慣真正上場報告的感受；第二個讓我印象深刻的是學姐們分享面對評審 QA 的問題，這也是我報告中最怕但也最需要克服的環節，我害怕如果評審問出了一個我不知道該如何回答的問題時要怎麼辦，但透過這次學姊的分享給了我一個典範，讓我可以更好的去解決這個問題；那學姊主要分享了三個 QA 小撇步，首先就是**把 QA 問題分類**，讓多個人分擔，這樣可以讓彼此各自更專注在一大類的問題也可以分散壓力；第二個是自己組在討論或是**老師、朋友等對專題提出疑問時，就把問題記下來並加上小組討論出的解決方案**，但如果真的遇到無法回答的問題，就要展現自己積極的態度像是回答：「謝謝評審的建議，為了我們系統的長期發展，我們會回去討論並解決這個問題。」而不是：「沒有討論到這個問題」，同個意思給評審的感受卻是天差地遠，所以好的表達是很重要的；最後是與其他組交流，感覺有些部分可能是我們太順著自己的邏輯走而疏忽掉一些細節，所以跟其他組互相分享並聽取他們的問題和建議，可以減少燈下黑的情況發生。透過上述的經驗分享，讓我們未來可以為上台報告有更出色的表現和準備的方向。

訪談結果和 deeplink 分享比較後，我們可能遇到的問題&預防方法：

從訪談結果中，我覺得我們最有可能跟學姐一樣面臨到**資料覆蓋的問題**，之前都有跟現在的組員彼此合作過很多次，我們大多都是分配好工作後就個自把自己部分完成，然後最後在整合上傳，但在整合的過程中就可能遇到彼此的資料庫的資料不同，命名不同等一些細節上的問題，但就是因為這些小細節可能造成系統出錯、畫面跑不出來等令人頭痛的問題，所以之後我們會吸取學姊給我們的建議就是**開分枝，先各自做好且確認都沒問題後才可以整合進主幹道**，確保主幹道的功能都是可以正常運作沒有問題的；那透過 deeplink 我覺得我們最有可能面臨的問題就是誰報告的問題，因為我們通常不會主動去負責報告，所以都用被動也就是抽籤的方式決定，但這樣就可能會錯過最合適的人選，所以**接下來可以讓大家試著去報告，為將來報告的人選做準備。**

訪談結果和 deeplink 分享比較後，我們可能遇到的問題&預防方法：

工作分配前

問題：
工作分配的目標不明確
預防方法：
組員都必須清楚了解所有工作內容後再進行工作分配，建議可以使用 DevOps 分配各個代辦事項，可以更好的了解系統目前的進度，除了可以避免組員不熟悉彼此的業務，還可以避免每個人做出來的產出和預想的架構有落差，還需另外花時間修改的狀況。

問題：
資料覆蓋、無法相容
預防方法：
工作分配可以使用 Git 進行內容的分支，除了清楚的討論，每個人的程式要經確認後再上傳至主幹道，以確保主幹道的內容可以完整呈現，減少程式整合時產生的非預期問題。

成果發表

問題：
大家都不願意接受口頭報告的工作
預防方法：
每位組員需要嘗試口頭報告的經驗，在未來正式專題發表時，應該由最適合、最能夠掌握舞台的人進行報告，讓我們的成果能夠完整的呈現給評審。