2018

智慧化企業整合 Project2—APP 設計

學生餐廳服務 APP

106034803 趙　敏樺

107034536 張簡宇傑

107034541 施　伯穎

第 7 組

2018/11/29

內容

1 主題描述.....1
2 現況分析和目標模型(As-Is vs To-Be).....2
2.1 現況分析(As-Is).....2
2.2 目標模型(To-Be).....3
3 Flexsim 模擬分析.....5
3.1 基本假設.....5
3.2 現況模型(As-Is).....6
3.3 改善模型(To-Be).....9
4 基本功能—APP 頁面功能.....12
4.1 會員註冊和登入[C、R].....12
4.2 訂餐[C、R、D].....13
4.2.1 瀏覽店家並點餐.....13
4.2.2 查看訂單.....13
4.3 商店相關功能[C、R、U、D].....14
4.3.1 店家管理和權限.....14
4.3.2 維護產品列表.....15
4.3.3 訂單管理.....16
5 額外功能—APP 完整架構.....17
5.1 Fragment.....17
5.2 JAVA 物件導向(Object-Oriented, OO)商業架構設計.....19
5.2.1 類別圖.....19
5.2.2 訂單狀態圖.....20
5.3 資料庫連線.....20
5.3.1 透過 PHP 連接 MySQL.....20
5.3.2 透過 JDBC 函式庫(library)連接 MySQL.....21
5.3.3 資料取用物件(Data Access Object, DAO).....22
5.4 單元測試(Unit Test).....24
5.4.1 目的.....24
5.4.2 在 Android Studio 實作單元測試.....24
5.5 小結.....27
6 聊天機器人(Chatbot).....28
7 結論和討論.....29

圖目錄

圖 1. 1 5W1H 分析 1

圖 2. 1 As-Is 服務藍圖 2
圖 2. 2 As-Is VSM 3
圖 2. 3 To-Be 服務藍圖 3
圖 2. 4 To-Be VSM 4

圖 3. 1 改善前 Flexsim 模型 6
圖 3. 2 改善前各店家平均等候時間 7
圖 3. 3 現況流程時間 8
圖 3. 4 改善後 Flexsim 模型 9
圖 3. 5 改善後各店家平均等候時間 10
圖 3. 6 改善後流程時間 11

圖 4. 1 登入和註冊頁面 12
圖 4. 2 點餐頁面 13
圖 4. 3 買家管理頁面 13
圖 4. 4 無權瀏覽頁面 14
圖 4. 5 店家管理頁面 15
圖 4. 6 編輯商品頁面 15
圖 4. 7 刪除商品頁面 16
圖 4. 8 訂單管理頁面 16

圖 5. 1 底部導航欄 17
圖 5. 2 頂部導航欄 18
圖 5. 3 商業架構類別圖 19
圖 5. 4 訂單狀態圖 20
圖 5. 5 PHP 連線 MySQL 架構 21
圖 5. 6 JDBC 架構 21
圖 5. 7 manifest 設定取用網路 22
圖 5. 8 JDBC 函式庫在 android 專案的位置 22
圖 5. 9 資料層類別圖 22
圖 5. 10 資料連線設定 23
圖 5. 11 DAO 抽象類別 23
圖 5. 12 會員 DAO(MemberDao)實作 24

圖 5. 13 單元測試路徑.....25
圖 5. 14 CRUD 測試程序.....26
圖 5. 15 CRUD 測試結果看板.....27
圖 5. 16 Android APP + JAVA 整體架構.....27

圖 6. 1 Chatbot 英文版介面.....28
圖 6. 2 Chatbot 建議餐點對話示意.....28

表目錄

表 2. 1 As-Is 花費時間表.....2

表 3. 1 各時段模擬人數.....5

1 主題描述

學生餐廳幾乎是所有清大學生每天都會造訪的地方，對學生的重要性不言可喻。然而由於清大學生人數眾多，只要一下課，尤其是中午，要在午休短短的一小時二十分鐘內點完餐、拿到餐點、找到位置、吃完餐點，時間實在是不太夠；有些特別好吃的店家人潮特別多，或是想買飲料喝又要再排一次隊，無形之中就浪費了許多時間。希望能夠透過學生餐廳 APP，提供學生一個學餐服務平台，能夠省下更多時間。透過此 APP，學生在下課先時查看店家，找尋自己鐘意的餐廳進行點餐，從教室走到店家時，學生可以直接領取餐點以節省時間。

本專題透過 5W1H 分析，釐清主題並分析現況。如圖 1. 1 所示。

- ✓ Who：經常在學校吃飯的教職員生，以及為他們提供餐點的店家。
- ✓ What：顧客透過手機 APP 進行點餐，點餐完成後的資料立即回傳至店家的手機 APP 中，使店家可以預先為顧客製作餐點。
- ✓ Why：因傳統現場點餐極其耗時，顧客時常在點餐的那一刻才再猶豫要吃什麼，且廚房在櫃台完成結帳後才開始製作餐點。
- ✓ When：餐廳開放的用餐時間。
- ✓ Where：校內的學生餐廳。
- ✓ How：使用者預先下載學生餐廳服務 APP，並註冊成為會員後，即可在用餐時間時預先至 APP 上進行點餐，而店家在收到訂單後即可製作餐點。

圖 1. 1 5W1H 分析

2　現況分析和目標模型(As-Is vs To-Be)

本章透過服務藍圖和價值溪流圖(Value Stream Map, VSM)比較 As-Is 和 To-Be 的模型。

2.1 現況分析(As-Is)

根據自身經驗和統計，計算午休時間用餐過程的花費時間表如表 2. 1。

表 2. 1 As-Is 花費時間表

項目	執行時間	等待時間	總和
下課至餐廳	10.0	0.0	10.0
點餐	0.8	7.0	7.8
取餐	0.8	10.0	10.8
找座位	3.0	0.0	3.0
用餐	30.0	0.0	30.0
總和	24.6	17.0	61.6

在花費約一小時吃午餐的過程之中，等待時間就佔了 17 分鐘，這 17 分鐘就是瓶頸(Bottleneck)。如能夠簡短等待時間，中午休息時間可以增添 17 分鐘，可以小睡片刻，下午上課會更加有精神及效率。圖 2. 1 為 As-Is 服務藍圖，圖 2. 2 為現況的 VSM。從圖(二)中，可以看到一般從下課到吃完午餐所花費的總時間為 61.6 分鐘，其中加值時間只佔了 31.6 分鐘。

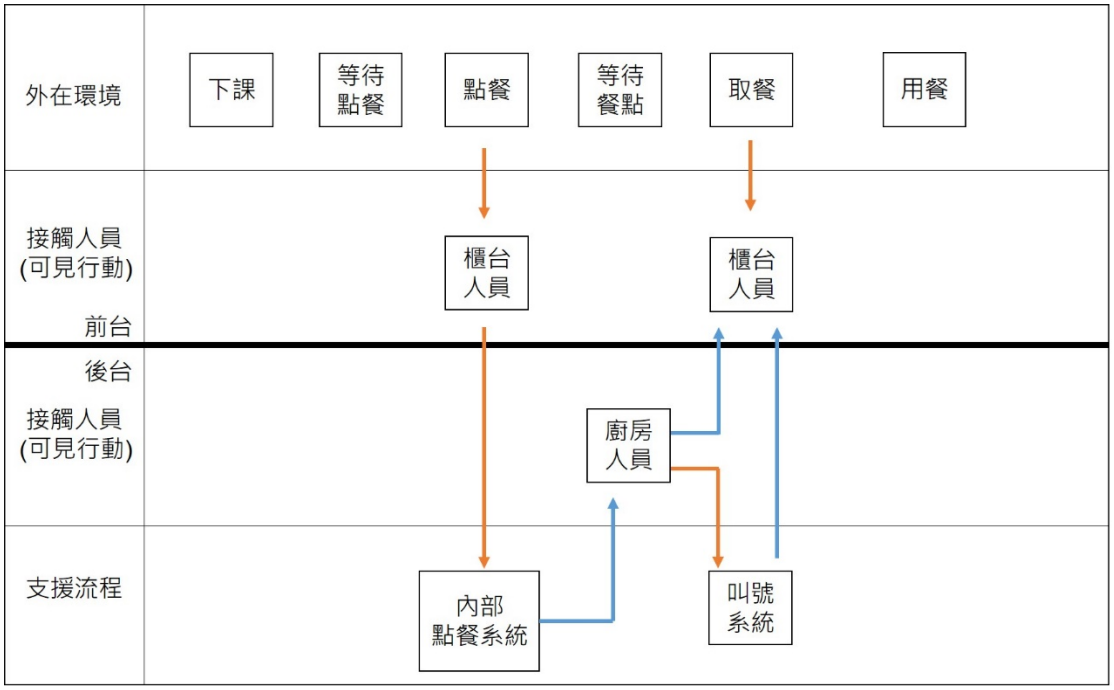


圖 2. 1 As-Is 服務藍圖

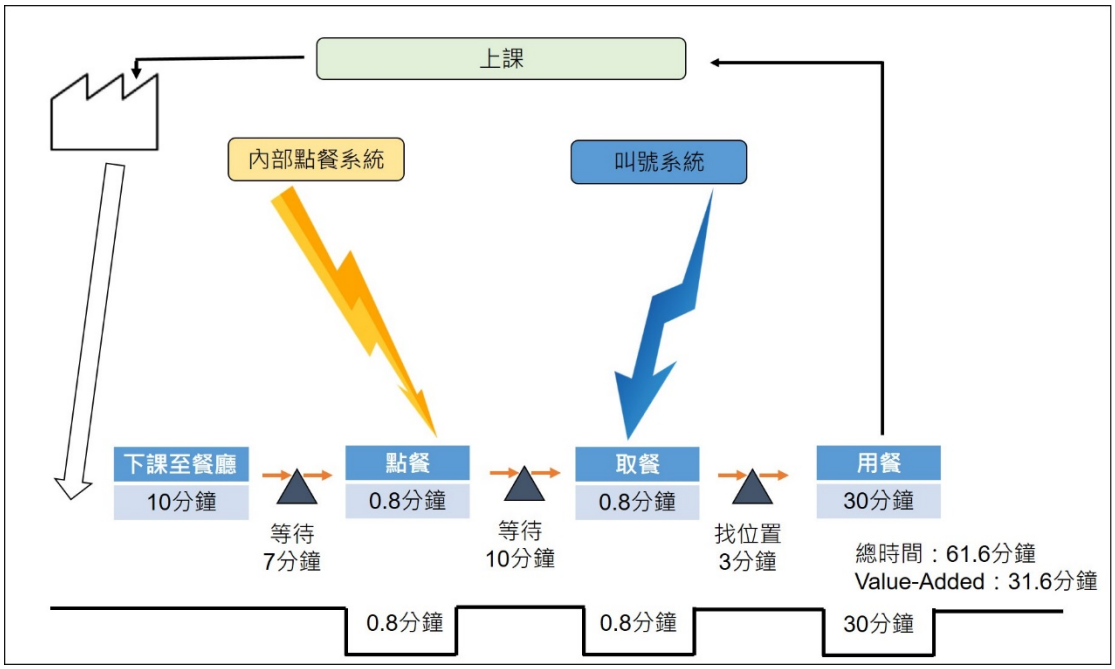


圖 2. 2 As-Is VSM

2.2 目標模型(To-Be)

採用線上訂餐 APP 後花費時間如表 2 所示。總花費時間從原本的 61.6 降低至 45.8，其中的瓶頸從 17 分鐘將到 1 分鐘，這 1 分鐘為到達店家後領取餐點的時間，不必和其他人一起排在點餐隊伍，可以直接領餐櫃台出示 App 進行取餐，並直接享用餐點。圖 2. 3 為 To-Be 服務藍圖、圖 2. 4 為 VSM。

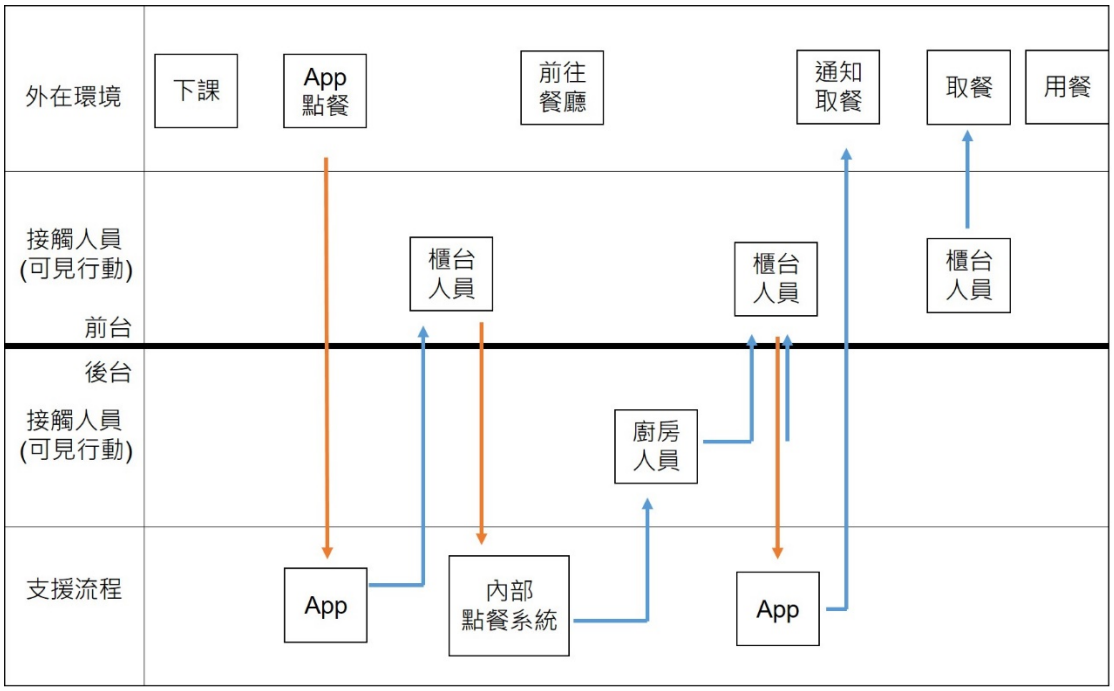


圖 2. 3 To-Be 服務藍圖

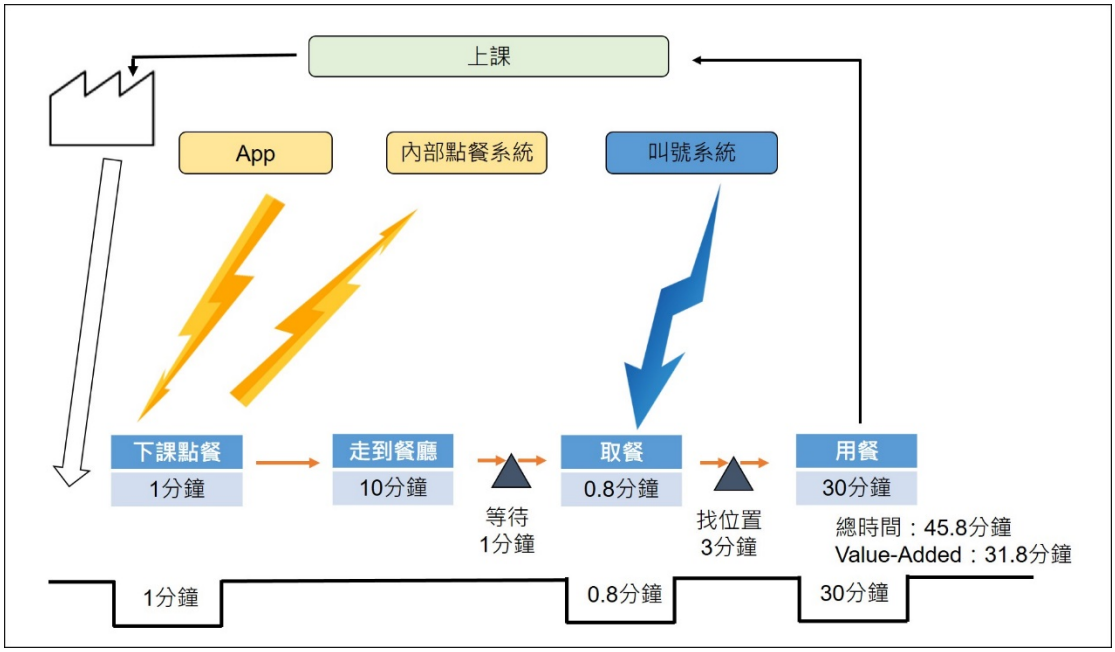


圖 2. 4 To-Be VSM

從圖 2. 4 中可以得知，總時間花費從大幅減少了 15.8 分鐘，其中的加值時間為 31.8 分鐘，約 70%，相對於 As-Is 加值時間所佔的比例大幅提高

3 Flexsim 模擬分析

本章說明採用 Flexsim 模擬並分析學生餐廳運作現況。首先在 3.1 說明基本假設、並分別在 1.1 和 1.1 展示現況和改善後的 Flexsim 模擬分析結果。

3.1 基本假設

以下條列基本假設。

- ✓ 時間以「分鐘」為單位。
- ✓ 顧客點餐皆為內用，限一人點一份餐點，且選定店家的方式為隨機選擇。
- ✓ 餐廳內有 3 家店家，每一家店有一位櫃台人員負責點餐及叫號領餐、一個廚房負責製作餐點、一個可容納 60 個座位的用餐區。
- ✓ 用餐區沒有佔位情形發生，且當有等待情形發生時，一有空位就會被補滿。
- ✓ 櫃台人員點餐時間為平均 0.8 分鐘的常態分配。
- ✓ 廚房製作餐點時間分別為 Shop A 4 分鐘、Shop B 5 分鐘、Shop C 6 分鐘的常態分配。
- ✓ 櫃台人員點餐時間為平均 0.8 分鐘的常態分配。
- ✓ 顧客用餐時間為平均 30 分鐘的常態分配。
- ✓ 模擬時間為餐廳白天中午用餐時段，每小時顧客人數如表 3. 1 所示。

表 3. 1 各時段模擬人數

時段	人數
09:00~10:00	20
10:00~11:00	50
11:00~12:00	120
12:00~13:00	180
13:00~14:00	60

3.2 現況模型(As-Is)

模擬顧客進入餐廳，以隨機方式選擇店家並向櫃檯人員點餐，廚房收到訂單後開始製作餐點，完成後交給櫃檯人員，顧客再經由櫃台人員叫號排隊領餐，並移至用餐區享用餐點。如圖 3.1 所示。

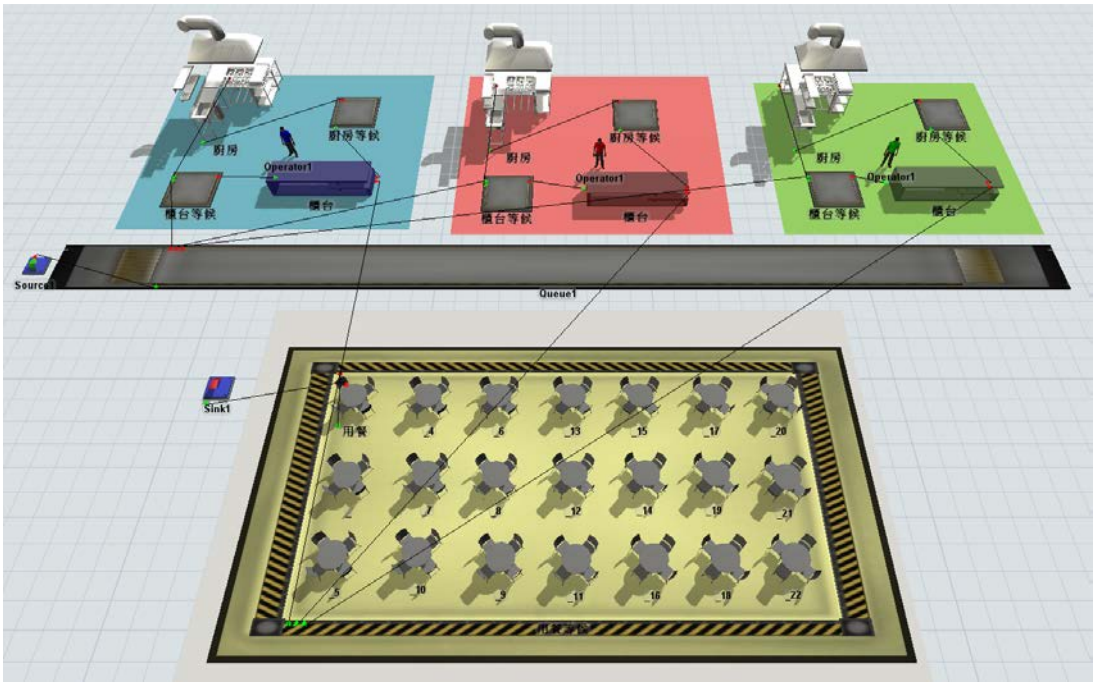


圖 3.1 改善前 Flexsim 模型

3 家店家等候時間各模擬 20 次，平均等候時間大致落在 3 至 10 分鐘。如圖 3. 2 所示。

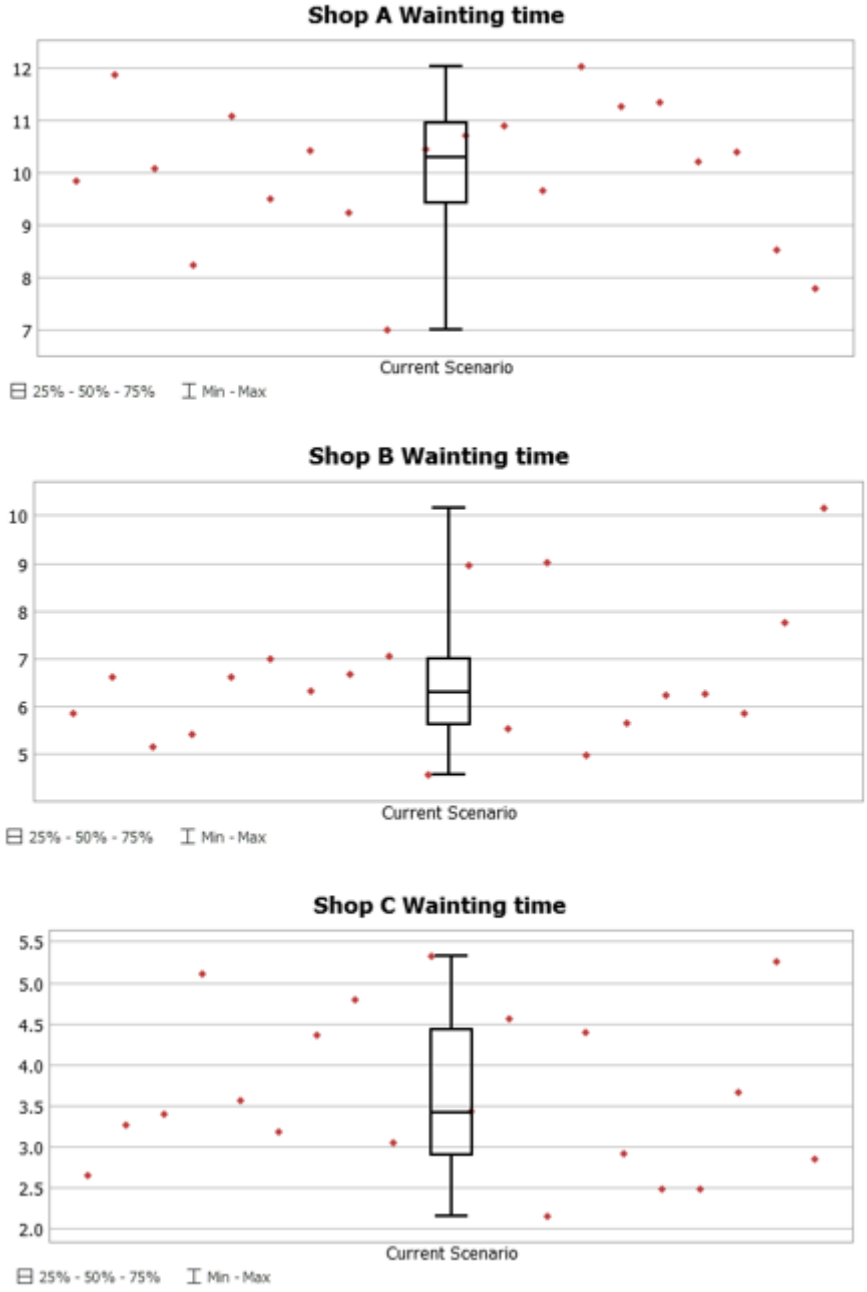
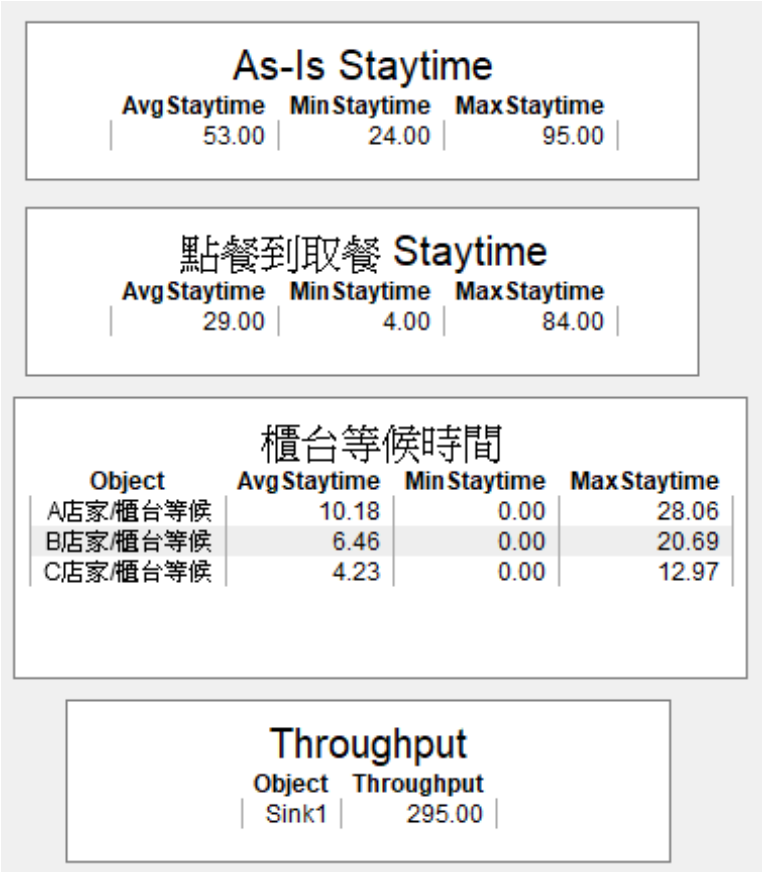


圖 3. 2 改善前各店家平均等候時間

顧客從點餐到享用完餐點的平均時間為 53 分鐘，而其中點餐到取餐的時間則占了 29 分鐘，為整體時間的 55%，餐廳店家服務的總顧客數為 295 人。如圖 3. 3 所示。



As-Is Staytime

Avg Staytime	Min Staytime	Max Staytime
53.00	24.00	95.00

點餐到取餐 Staytime

Avg Staytime	Min Staytime	Max Staytime
29.00	4.00	84.00

櫃台等候時間

Object	Avg Staytime	Min Staytime	Max Staytime
A店家/櫃台等候	10.18	0.00	28.06
B店家/櫃台等候	6.46	0.00	20.69
C店家/櫃台等候	4.23	0.00	12.97

Throughput

Object	Throughput
Sink1	295.00

圖 3. 3 現況流程時間

3.3 改善模型(To-Be)

模擬顧客進入餐廳後直接至櫃台等候取餐，接著直接到用餐區享用餐點。如圖 3. 4 所示。因顧客在前往餐廳前已在學生餐廳服務 APP 上進行點餐，店家收到資訊後則立即製作餐點，故這段時間並未算入模擬流程中。

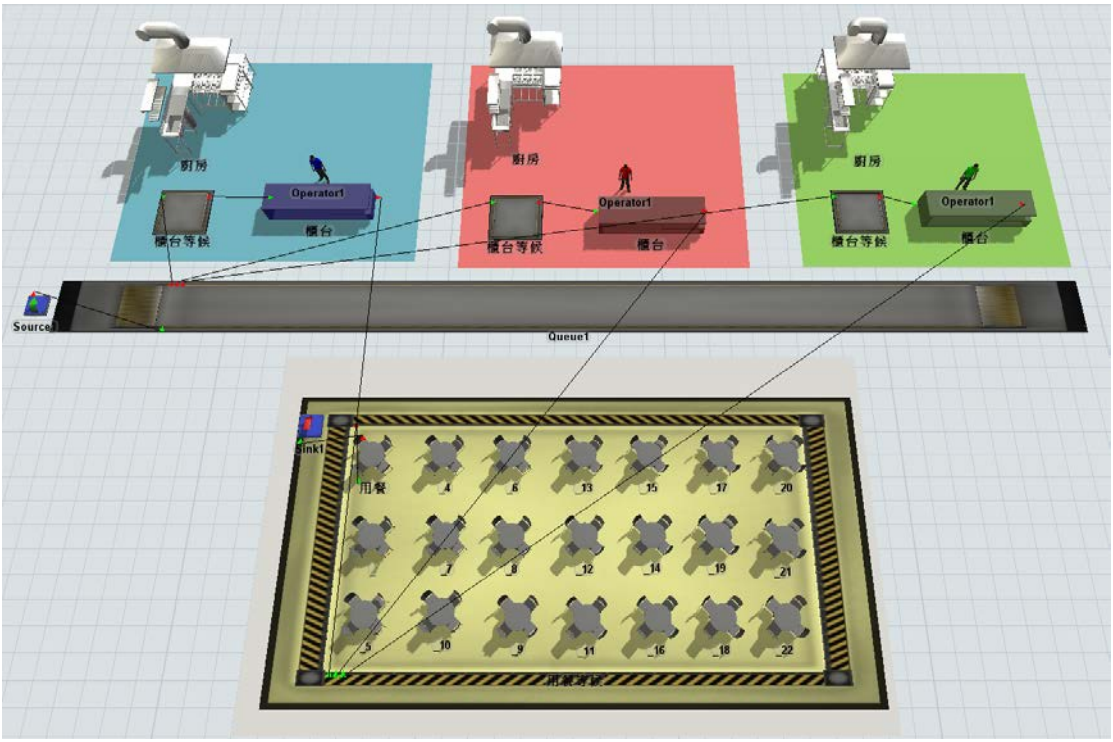


圖 3. 4 改善後 Flexsim 模型

學生餐廳服務 APP 讓顧客在進入餐廳前即可預先訂餐，省去現場排隊點餐以及等待廚房製作餐點的時間。且顧客多為學生，在中午用餐時間有限的情形下，擔心趕不上下一節課而選擇不吃午餐，或因等待的時間過長導致購買意願降低所造成時間的浪費，同時也間接影響店家的銷售量。3 家店家等候時間各模擬 20 次，平均等候時間大致落在 1 分鐘內。如圖 3.5 所示。

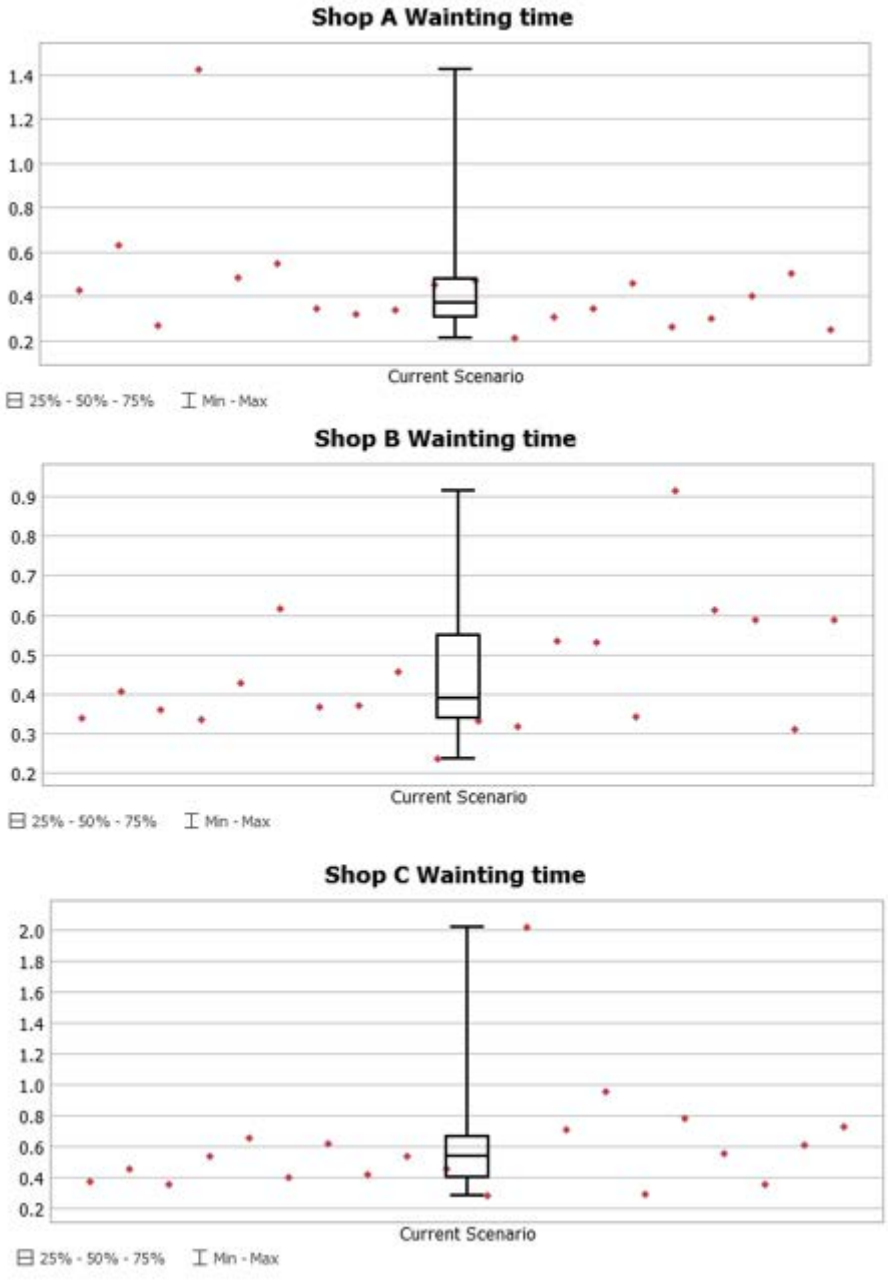


圖 3.5 改善後各店家平均等候時間

顧客從點餐到享用完餐點的平均時間為 37 分鐘，而其中點餐到取餐的時間則只占了 8 分鐘，為整體時間的 21%，餐廳店家服務的總顧客數為 366 人。如圖 3.6 所示，改善後的流程明顯減少顧客點餐及取餐所等待的時間，並且增加店家銷售餐點的數量，使得雙方皆因學生餐廳服務 APP 而受惠。

圖 3.6 改善後流程時間

4　基本功能—APP 頁面功能

本章說明此專案開發的學生餐廳服務 APP 提供的功能。由於專案期程較短，未能完整提供所有功能。以下僅說明主要的商業流程相關的功能頁面。

課堂上所教授的基本功能，聚焦在和資料庫連接並實踐新增(Create)、讀取(Read)、更新(Update)、和刪除(Delete)等 CRUD 功能。本章所提功能都會和 CRUD 對應。

4.1 會員註冊和登入[C、R]

任何在此 APP 上操作的使用者都必須申請帳號。APP 一啟動會先進到登入頁面，下方提供「登入」和「註冊」按鈕。

在登入時，系統會檢查帳號和密碼是否存在並吻合[R]，用到「新增資料」功能。註冊時需要輸入帳號、密碼、姓名、和 E-mail 等，並選擇身份，會員共有店家、學生、教職員、和管理員等身份。按下註冊按鈕後，APP 會先判斷該帳戶是否已經被申請[R]，若可申請，將會建立一筆會員帳號資料[C]。APP 畫面如圖 4. 1 所示。

圖 4. 1　登入和註冊頁面

4.2 訂餐[C、R、D]

此小節說明訂餐相關 APP 功能頁面。

4.2.1 瀏覽店家並點餐

一進到 APP 頁面，使用者可以看到視覺化的區塊呈現，列出清大主要的三個餐廳，風雲樓、小吃部、和水木餐廳。選定其中一家餐廳後，APP 會列出該餐廳內的所有店家[R]。接著進到店家頁面，從菜單中挑選想點的餐點[R]，輸入數量後，便可加入購物車[C]。在購物車內也可以點擊項目進行刪除[D]。相關 APP 功能頁面如圖 4.2 所示。

圖 4.2 點餐頁面

4.2.2 查看訂單

當使用者下訂後，可以從「買家管理」頁面選擇「我的訂單」查看所有訂單的資料和狀態[R]。如圖 4.3 所示。

圖 4.3 買家管理頁面

4.3 商店相關功能[C、R、U、D]

此小節說明商店相關功能。4.3.1 說明該功能對會員身份的權限判別和店家管理頁面，4.3.2 和 4.3.3 分別說明維護產品列表和訂單管理。

4.3.1 店家管理和權限

店家管理功能只有「店家」和「管理員」這兩種身份的會員[R]可以使用。若身份是「學生」或「教職員」，APP 將會顯示「無權瀏覽」的訊息，如圖 4. 4 所示。

圖 4. 4 無權瀏覽頁面

店家身份的會員可以管理自己擁有的店家，而管理員則可以管理所有店家。使用者選取單一店家後，便可以查看「訂單狀態」、「產品管理」、「發布公告」、「店家資料」、和「幫助中心」等頁面，如圖 4. 5 所示。

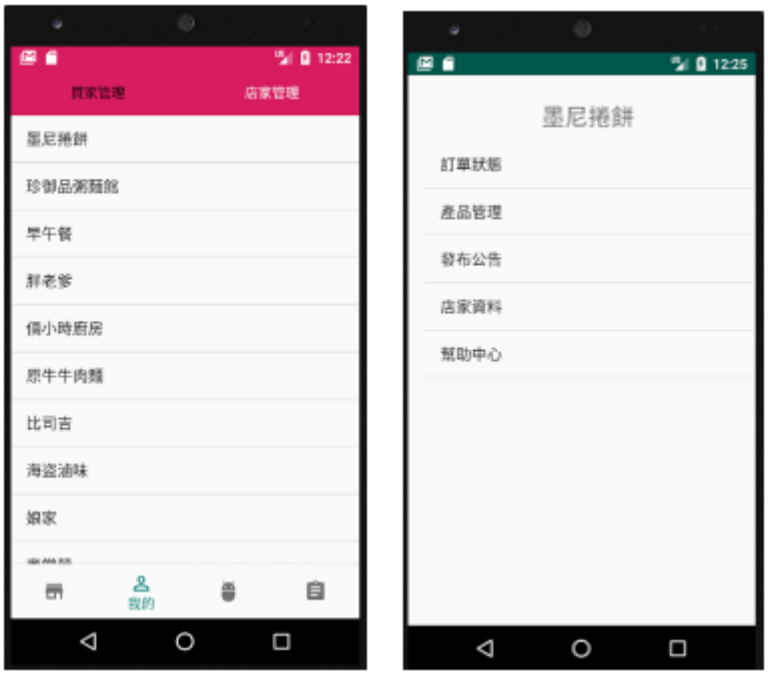


圖 4.5 店家管理頁面

4.3.2 維護產品列表

在產品管理頁面內可以查看產品列表[R]並編輯或刪除。點擊「筆」的圖示可以編輯，APP 會導向編輯產品的頁面，使用者重新確認資料後，按下右下角的「勾勾」，便能完成編輯[U]。如圖 4.6 所示。

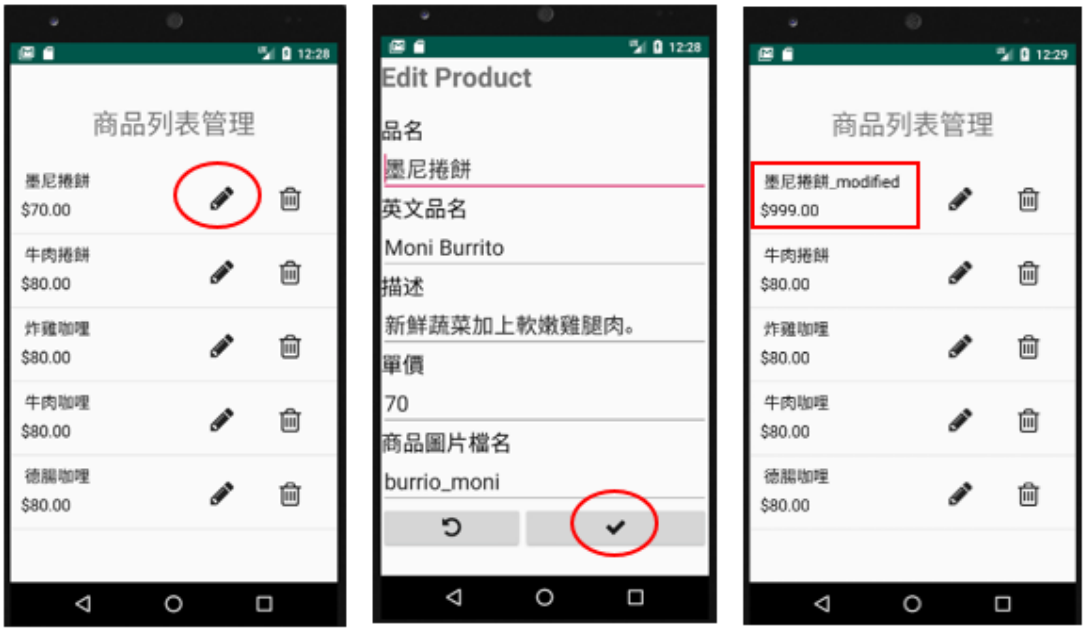


圖 4.6 編輯商品頁面

若是點擊「垃圾桶」的圖示，APP 將會跳出對話窗，請使用者確認是否要刪除，按下「確認」後，APP 將會連接資料庫刪除該筆商品[D]，並刷新商品列表顯示。如圖 4.7 所示。

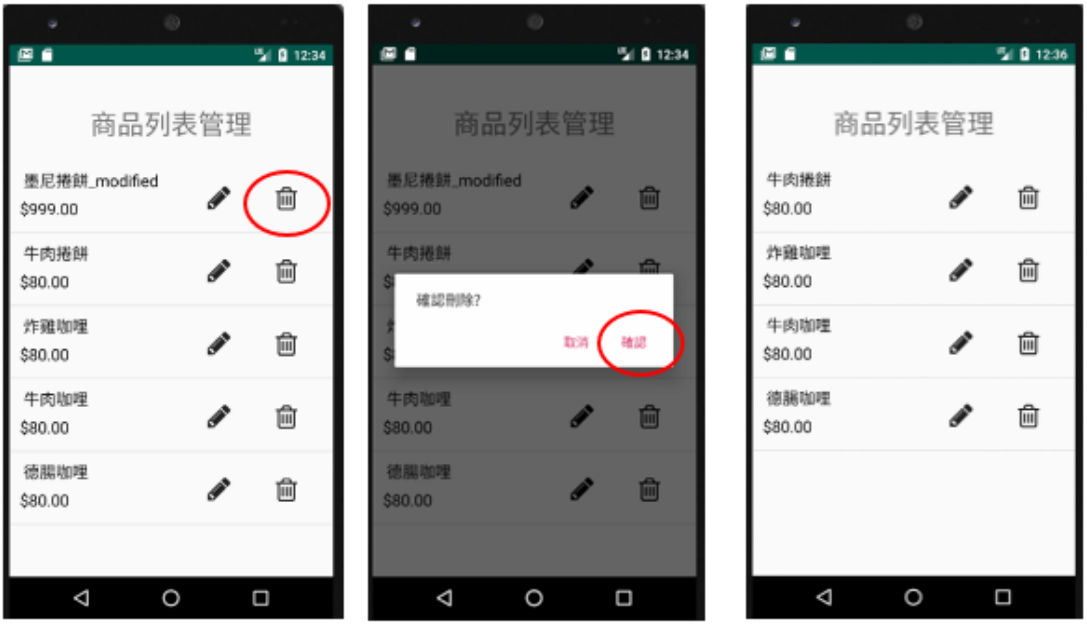


圖 4.7 刪除商品頁面

4.3.3 訂單管理

在訂單管理頁面，商家可以查看所有訂單[R]，並更新各訂單的狀態。頁面把每一筆訂單逐列呈現，並顯示當前的狀態。使用者可以點擊狀態文字，讓讓訂再前往下一個狀態。舉例來說，當狀態顯示為「待製餐」時，點擊後會讓訂單的狀態改為「製作中」[U]，如果不斷前進，直到最後「已送餐」。另外，頁面上也提供「刪除」按鈕可以刪除訂單[D]。如圖 4.8 所示。

圖 4.8 訂單管理頁面

5 額外功能—APP 完整架構

本章說明此專案用到或提供的額外功能，分為 Fragment、JAVA 物件導向商業架構設計、資料庫連線、和單元測試等部分說明，最後作一小結。為打造一健全且容易維護擴充的 APP 架構，

5.1 Fragment

Fragment 字面上的意義為「分段」，在 App 中，Fragment 的作用為將一 Activity 分段，最直接的作用是讓使用者不用一直在 Activity 間轉換，使得獲取資訊更加容易。

Fragment 在此 APP 運用之處有兩處，一為登入後的頁面，一為點餐頁面。登入後的首頁用 HomeActivity 控制，此為一 Activity，此 Activity 分割成四個 Fragment，分別為「點餐」、「我的」、「Chatbot」、和「公告」。在程式中可設定當一進入 HomeActivity 時自動進入特定的 Fragment。若沒有經過此設定，頁面將會顯示空白，需點擊下方的導航欄按鈕才會導向任一 Fragment。首頁用 Frament 實作的導航欄如圖 5. 1 所示。此 APP 在登入之後，預設進入點餐的 Fragment。可以按下下方導航欄跳至各 Fragment，不須再使用 Button 在各 Activity 跳動。若按下下方導航欄「公告」圖示即可跳至公告頁面。

圖 5. 1 底部導航欄

另一使用 Fragment 的地方在於點餐頁面，與登入後頁面不同的是，點餐頁面使用的是頂部導航欄。而在左右滑動時會更換顏色，可以讓使用者更清楚知道自己的位置，如圖 5.2 所示。相對於底部導航欄，頂部導航欄適用於 Fragment 少的時候，底部導航欄適用於 Fragment 多的時候，視覺上較於美觀。

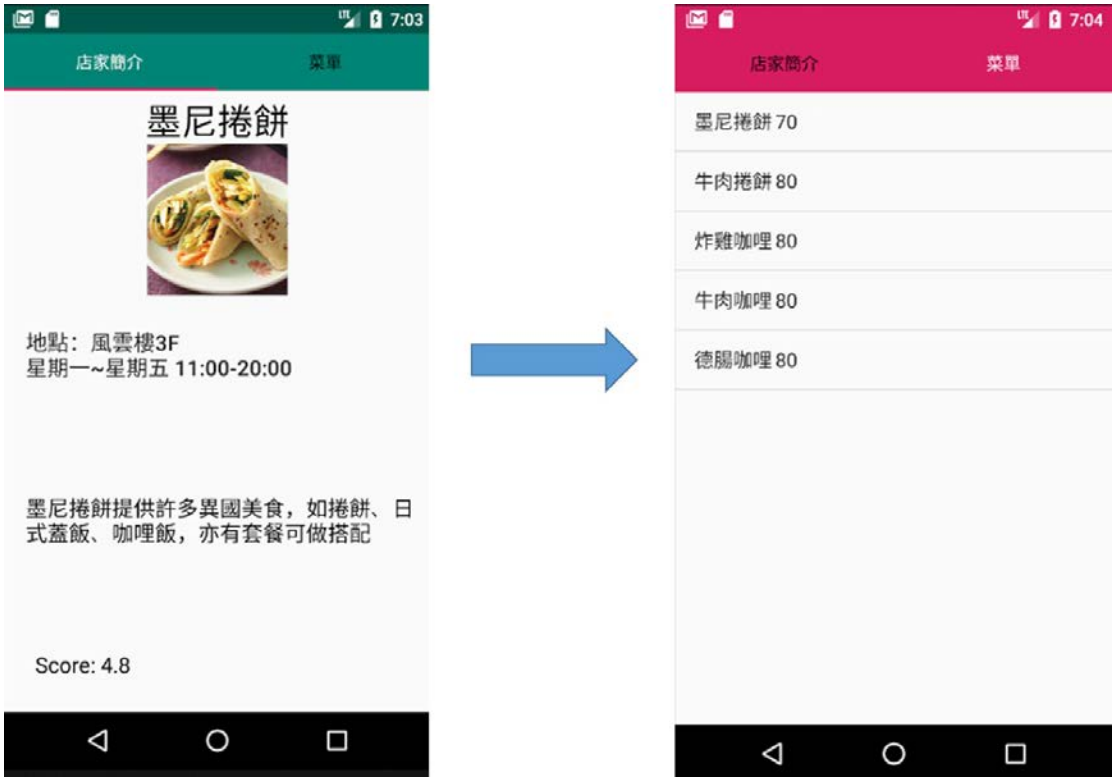


圖 5.2 頂部導航欄

5.2 JAVA 物件導向(Object-Oriented, OO)商業架構設計

此節說明本專案以物件導向概念設計的商業架構。

5.2.1 類別圖

本專案的物件導向商業模型如圖 5.3 所示。

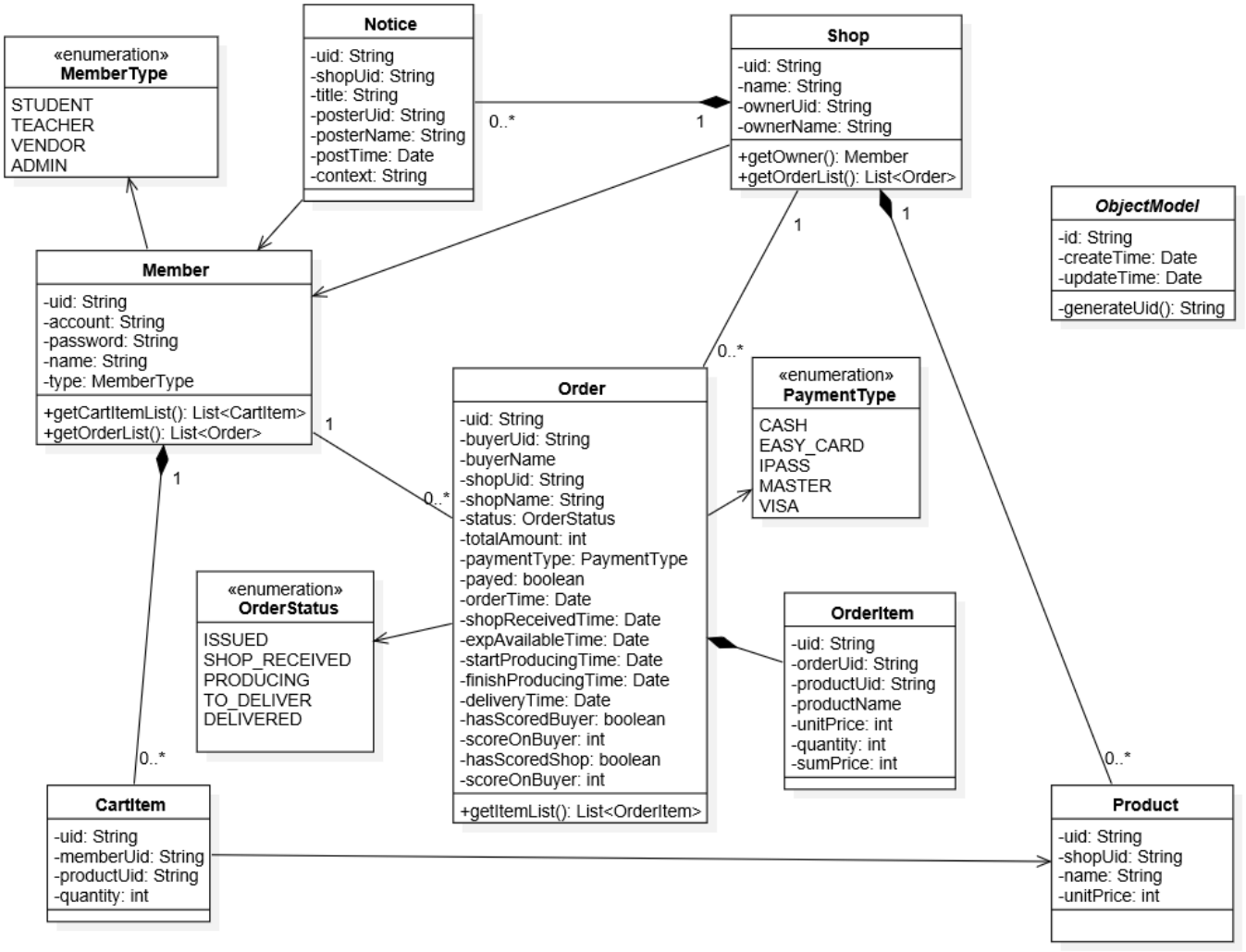


圖 5.3 商業架構類別圖

ObjectModel 是本專案最核心的抽象類別，所有其他類別都繼承它。但為了圖面呈現的簡潔，圖上並沒有畫出這些繼承的一般化(Generalization)關連。ObjectModel 定義物件統一編號(uid)、建立時間(createTime)、和更新時間(updateTime)屬性。本專案在產生物件的唯一識別時，是透過 ObjectModel 提供的 generateUid 方法由程式控制，而不依賴 MySQL 資料庫的 AutoIncrement。這樣的好處是在每個物件產生時就能知道自己的 uid，若依賴資料庫的 AutoIncrement，在從資料庫取出物件時必須依賴商業定義的唯一值(key)，但當某些類別的商業 key 值太過複雜或是不明確時，將不容易明確找到指定的物件。

其他類別屬性太過細節，將不一一介紹。要特別注意的是，為了維護或查詢的便利性，很多類別的設計並不會完全做到第三正規化(3NF)。例如店家(Shop)類別的擁有人屬性，同時記錄了擁有人 uid 和擁有人姓名，若以 3NF 的規範，不應該在此記錄姓名，但為了查詢和顯示的方便，我們選擇保留姓名欄位。

5.2.2 訂單狀態圖

訂單是這個商業模型中最核心的類別，其中涉及多個狀態之間的轉換。當買方下訂後，訂單物件產生並跳轉到「已開立(ISSUED)」狀態；當店家確定收單後，狀態轉換到「店家已收到訂單(SHOP_RECEIVED)」；接著歷經「製作中(PRODUCING)」、「待交付(TO_DELIVER)」、和「已送達(DELIVERED)」等狀態。狀態圖如圖 5. 4 所示。

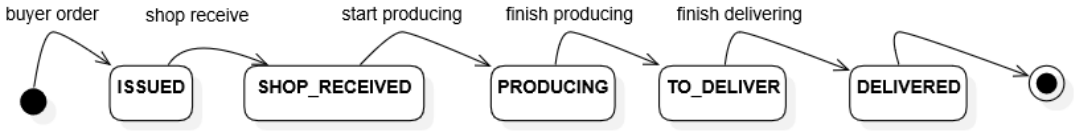


圖 5. 4 訂單狀態圖

5.3 資料庫連線

本專案評估後採用 JDBC 連接 MySQL 資料庫，而非課堂範例採用的 PHP。當然兩者的優劣和適用性在網路上都可以查到廣大的討論，並沒有絕對的好壞。本節詳細說明考量的原因和 JDBC 資料庫連線方式。

5.3.1 透過 PHP 連接 MySQL

在之前的 Project1 中是開發網頁的應用程式，PHP 和 HTML 有非常強力的結合，搭配 MySQLi 模組，便能有效地和 MySQL 資料庫溝通。考量教學的方便和一致性，繼續沿用此架構是相當合理的。

但本次的 Project2 是開發 Android 應用程式，而 Android 主要採用 JAVA 程式語言撰寫，若要沿用此架構，必須額外架設 PHP 的站台，先把指令由 JAVA 呼叫外部的 PHP，透過 PHP 網頁來連接 MySQL 資料庫，把回傳的結果包成 JSON 格式再回到 JAVA 程式中解析。如圖 5. 5 所示。

圖 5. 5 PHP 連線 MySQL 架構

這樣的作法在安全性和開發效率上都有疑慮。安全性的考量而言，Android 程式碼無法掌控 PHP 的內容， PHP 的內容暴露在外部站台，若有心人士想要竊取或竄改檔案是更加容易的。以開發效率而言，開發者必須要額外建立一個 PHP 站台來連接 MySQL 資料庫，且還要透過 JSON 包裝再解析，當程式出現錯誤時不易追蹤且不好維護。

5.3.2 透過 JDBC 函式庫(library)連接 MySQL

為解決 5.3.1 提到的問題，本專題採用 JAVA 提供的 JDBC 函式庫連接 MySQL 資料庫，示意如圖 5. 6 所示。JAVA 應用程式，例如 Android，透過 JDBC API 和對應的驅動程式(driver)便能直接和資料庫溝通。甚至如果哪天想要更換使用的資料庫，只要抽換相對應的 driver 即可，程式需要改動的幅度通常不大。

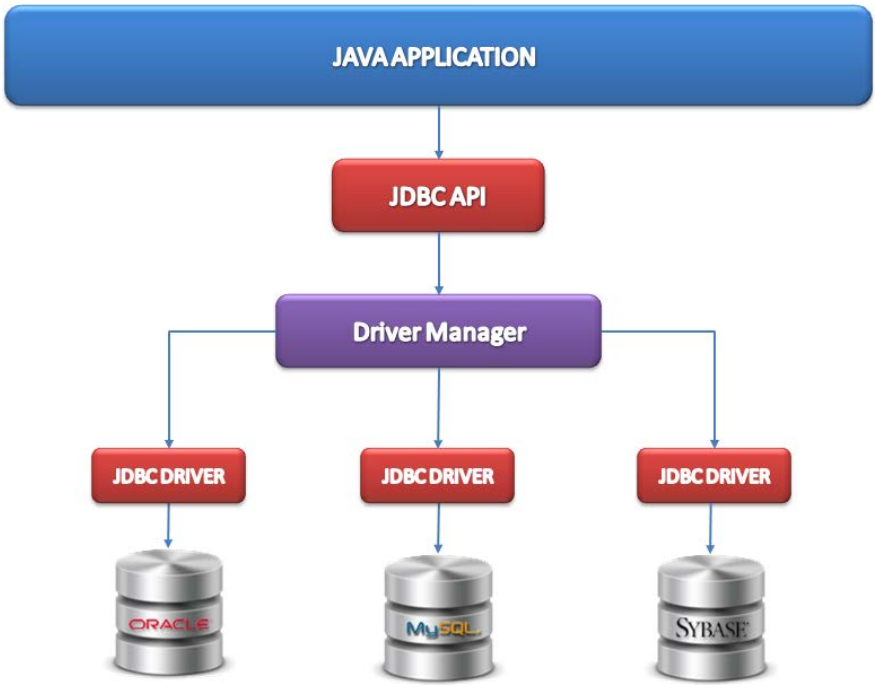


圖 5. 6 JDBC 架構

接著說明在 Android Studio 專案中的設定步驟。首先，Android Studio 默認是不允許訪問 Internet 的，因此需要在/app/src/main/AndroidManifest.xml 中的<manifest>...</manifest>區域內添加允許網路連線的宣告，如圖 5.7 所示。而這部分和透過 PHP 連線時相同。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.elebc_gpu_server.group7_project2">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <application
```

圖 5. 7 manifest 設定取用網路

接著，只要下載 mysql-connector-java 的 jar 檔後，放置到/app/libs 路徑下就設定完成，如圖 5. 8，不需要另外架設 PHP 的站台。(這部分其實我們沒有自己實作，因為助教已經先幫我們架好了。)

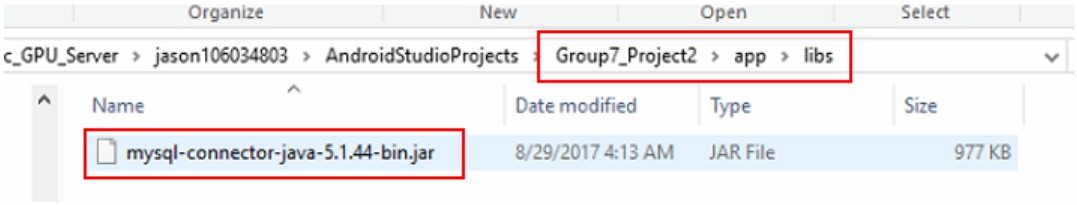


圖 5. 8 JDBC 函式庫在 android 專案的位置

最後就是撰寫程式來存取 MySQL 資料庫。

5.3.3 資料取用物件(Data Access Object, DAO)

DAO 的職責是和資料庫溝通並下達 SQL 指令。本專案的資料庫架構類別圖如圖 5. 9 所示。

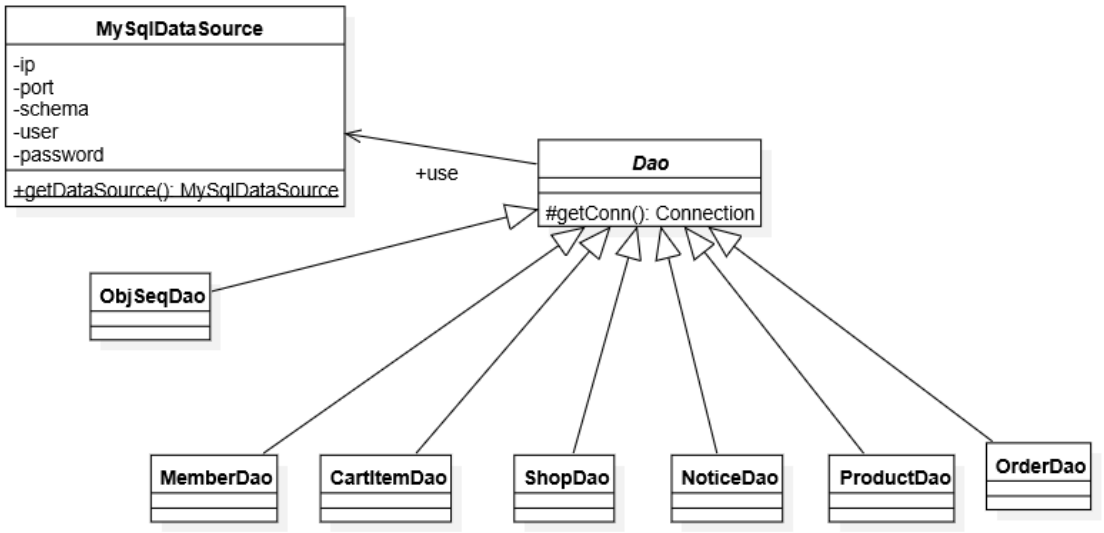


圖 5. 9 資料層類別圖

資料庫連線設定

在/app/src/main/{project_package}/data/MySqlDataSource.java 類別統一設定資料庫連線參數，包含 ip、port、schema(db)、user、和 password 等參數，並提供一靜態的 getDataSource 方法回傳相關設定。全專案不會需要在其他地方重覆輸入這些設定參數。程式碼示意如圖 5. 10 所示。

```
public class MySqlDataSource {
    //----------------------------------------------------------------
    private final static MySqlDataSource DS_178 = new MySqlDataSource( ip: "140.114.54.178", port: "3306", schema: "group7", user: "group7",
            password: "07836");

    public final static MySqlDataSource getDataSource(){return DS_178;}
```

圖 5. 10 資料連線設定

Dao 抽象類別和實作

在/app/src/main/{project_package}/data/Dao.java 抽象類別提供全專案統一的資料庫連線介面，提供測試連線(testConn)、取得連線(getConn)、和關閉連線(close)等方法供子類別實作。另外，本專案所有的商業物件都會繼承 ObjectModel 抽象類別，該類別通用的欄位也會在 Dao.java 中宣告，集中管理。如圖 5. 11 所示。本專案繼承 Dao.java 的子類別目前有 ObjSeqDao、MemberDao、CartItemDao、ShopDao、NoticeDao、ProductDao、和 OrderDao 等，每一個 DAO 分別宣告資料表(table)和各欄位(column)名稱統一管理，並負責存取各類別的物件。以 MemberDao 為例，首先用靜態變數宣告資料表名稱和各欄位名稱，並提供公開(public)的方法包括儲存會員(saveMember)、刪除會員(deleteMember)、和取得會員(loadMember)等。如圖 5. 12 所示。

```
package com.example.elebc_gpu_server.group7_project2.data;

import ...

public abstract class Dao {
    public void testConn() {...}

    protected Connection getConn() {...}

    protected final void close(Connection conn, PreparedStatement pstmt, ResultSet rs) {...}


    //----------------------------------------------------------------
    protected final static String COL_UID = "uid";
    protected final static String COL_CREATE_TIME = "create_time";
    protected final static String COL_UPDATE_TIME = "update_time";

    protected final boolean deleteObject(String _table, String _uid) {...}

}
```

測試連線

取得連線

關閉連線

宣告ObjectModel
通用的欄位名稱

圖 5. 11 DAO 抽象類別

圖 5. 12 會員 DAO(MemberDao)實作

5.4 單元測試(Unit Test)

本專案有採用單元測試，5.4.1 說明單元測試的目的，5.4.2 說明在 Android Studio 專案中的實作。

5.4.1 目的

維基百科對單元測試的說明如下：在電腦編程中，單元測試是針對程式模組（軟體設計的最小單位）來進行正確性檢驗的測試工作。程式單元是應用的最小可測試部件。在程序化編程中，一個單元就是單個程式、函式、過程等；對於物件導向程式設計，最小單元就是方法，包括基礎類別（超類）、抽象類、或者衍生類別（子類別）中的方法。通常來說，程式設計師每修改一次程式就會進行最少一次單元測試，在編寫程式的過程中前後很可能要進行多次單元測試，以證實程式達到軟體規格書要求的工作目標，沒有程式錯誤；雖然單元測試不是什麼必須的，但也不壞，這牽涉到專案管理的政策決定。

5.4.2 在 Android Studio 實作單元測試

Android Studio 的專案建立之後，預設會產生單元測試的模組在/app/src/test/{project_package}的路徑下。本專案針對每一個商業類別的新增(Create)、讀取(Read)、更新(Update)、刪除(Delete)，簡稱 CRUD，進行單元測試。確保 JAVA 的商業物件類別和 MySQL 資料庫之間的溝通是正常的。

透過完整的單元測試區分範疇，當編程過程中出現 bug 時，能更容易識別問題發生的位置，增進開發效率。如圖 5. 13 所示。

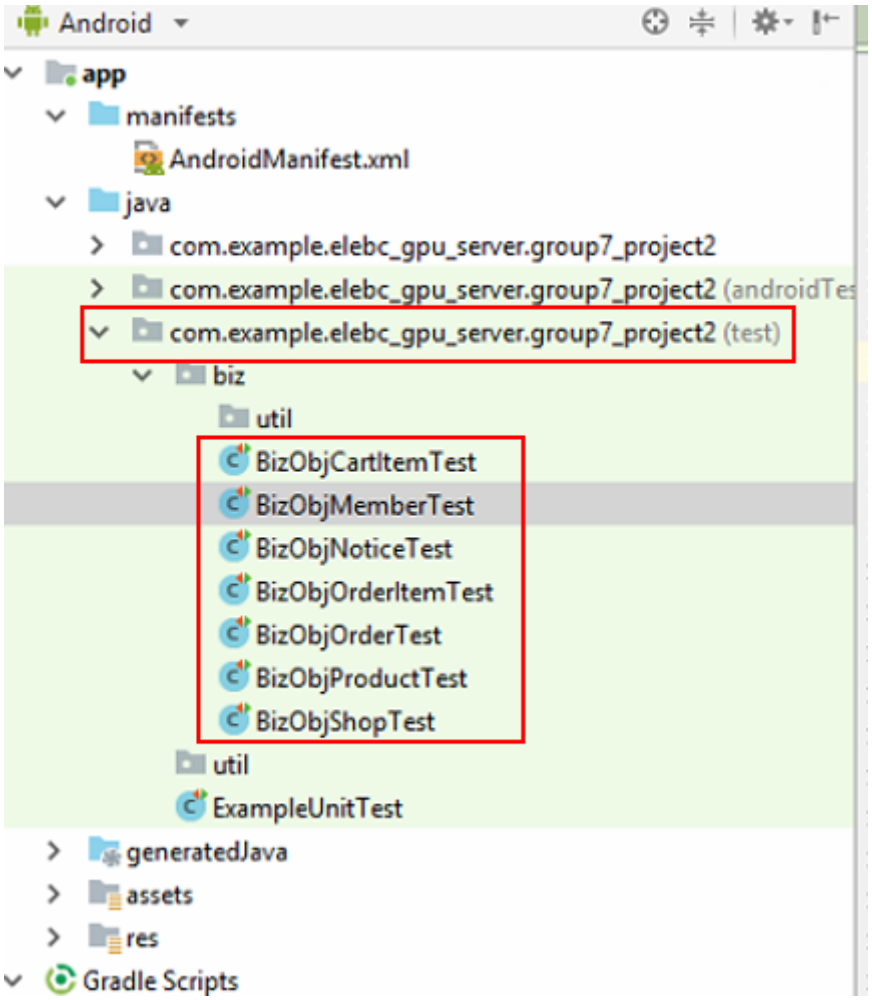


圖 5. 13 單元測試路徑

以 BizObjMemberTest.java 為例，可以看到內含的主測試程序包含新增、更新、和刪除，這樣在執行完一輪測試後，資料庫不會殘留多餘的資料。而讀取的程序則在新增和更新後執行比對時會呼叫到。如圖 5. 14 所示。

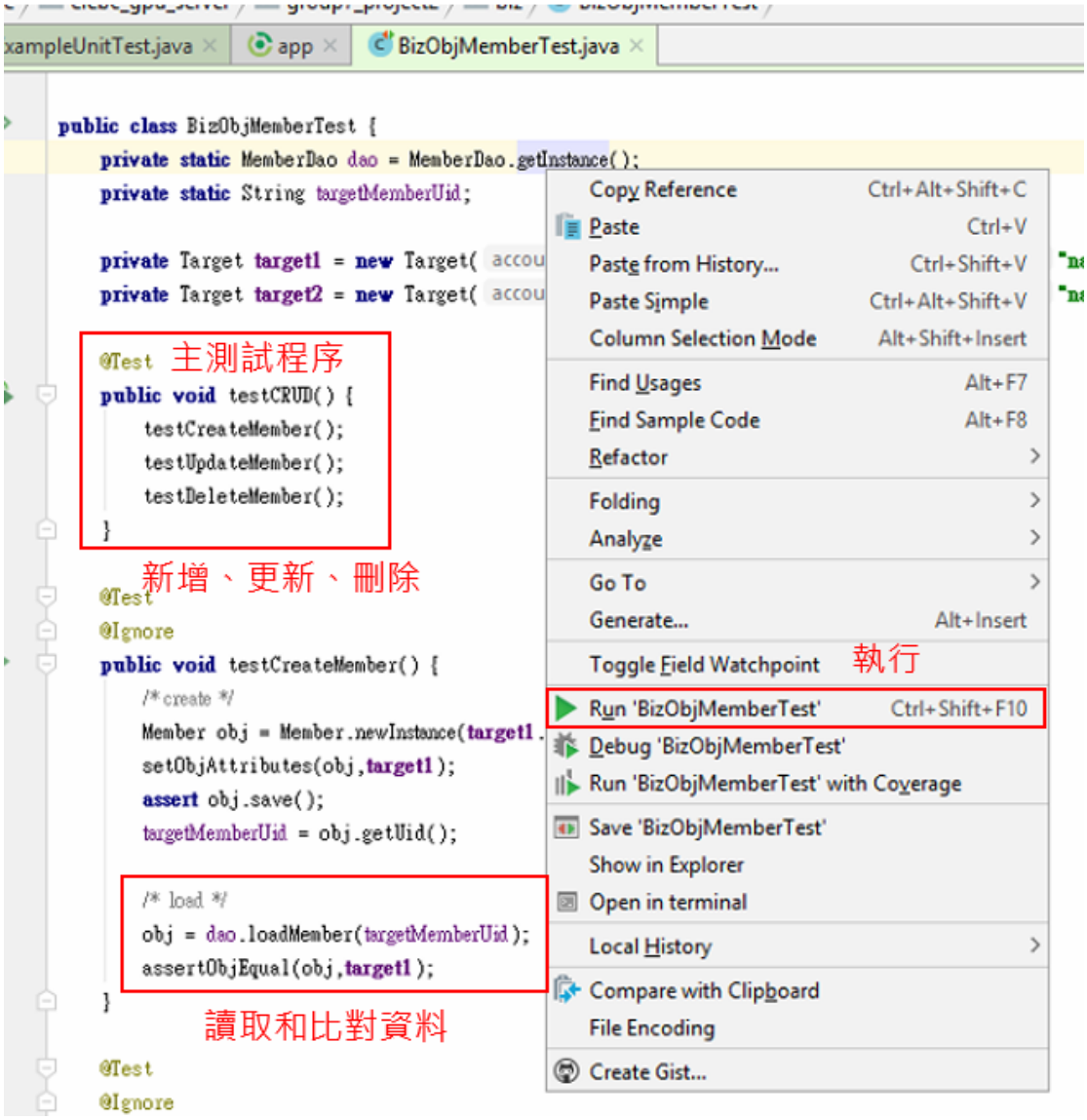


圖 5. 14 CRUD 測試程序

單元測試的執行結果也可以輕易地在 Android Studio 的介面上看到。如圖 5. 15 所示。

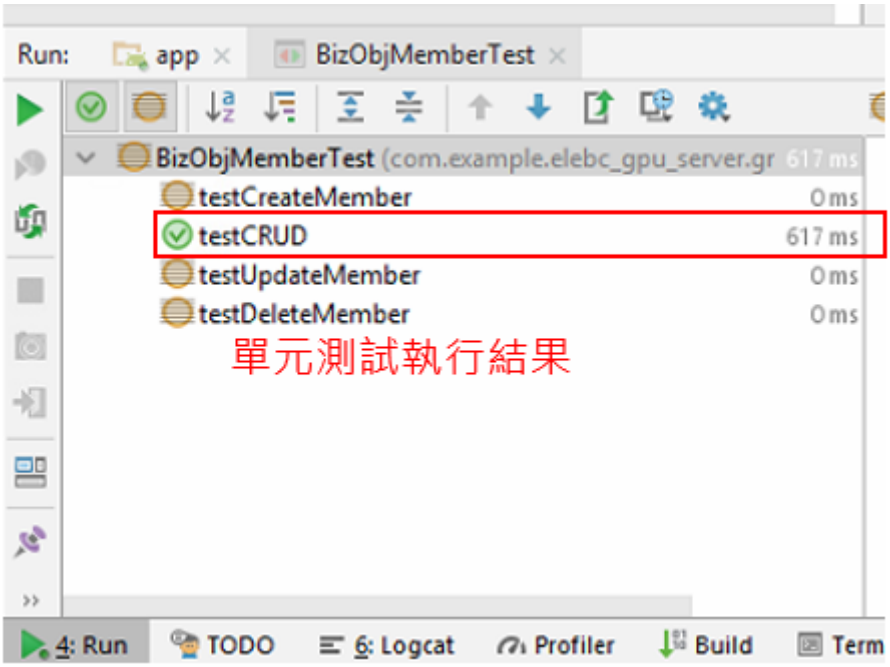


圖 5. 15 CRUD 測試結果看板

5.5 小結

除了助教課堂上的基本示範之外，本章詳細說明完整商用 Android APP 的架構。從前端的頁面設計、商業層的 JAVA 物件導向程式設計、到後端的資料庫連線模組，如圖 5. 16 所示。明確區分各層的範疇，再搭配單元測試的應用，創造一個完整且容易維護的專案架構。

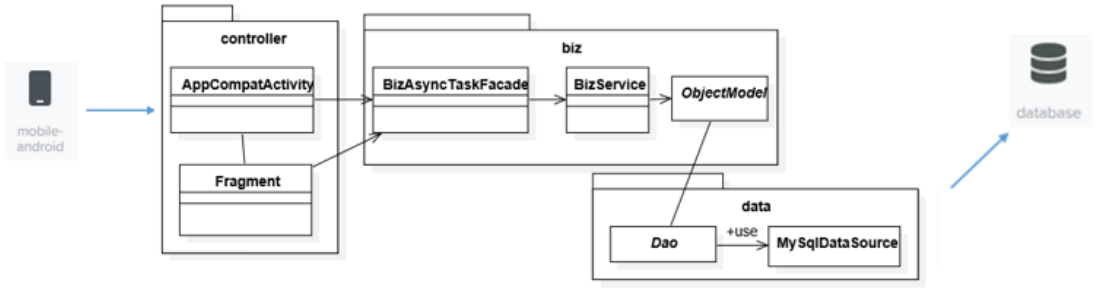


圖 5. 16 Android APP + JAVA 整體架構

6 聊天機器人(Chatbot)

本次的聊天機器人主旨在於解決同學們常常有的煩惱—「不知道要吃甚麼？」而聊天機器人具有雙語功能，輸入「嗨」會出現中文版服務，而輸入「hi」則會出現英文版服務。如圖 6. 1。以下使用英文版做說明。

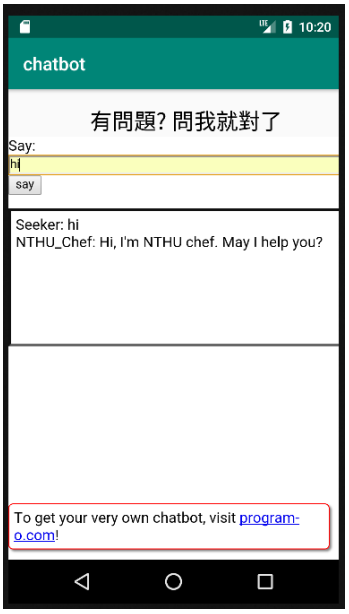


圖 6. 1 Chatbot 英文版介面

在對話框輸入「hi」時，機器人便會回應，問是否需要幫忙。使用者進而輸入不知道要吃甚麼的煩惱，而機器人便會給予建議，詢問牛肉麵如何，使用者可以再進一步詢問價錢。使用者輸入「ok thank you」時即可結束服務，使用者可以前往點餐或者詢問其他問題。對話示意如圖 6. 2 所示。

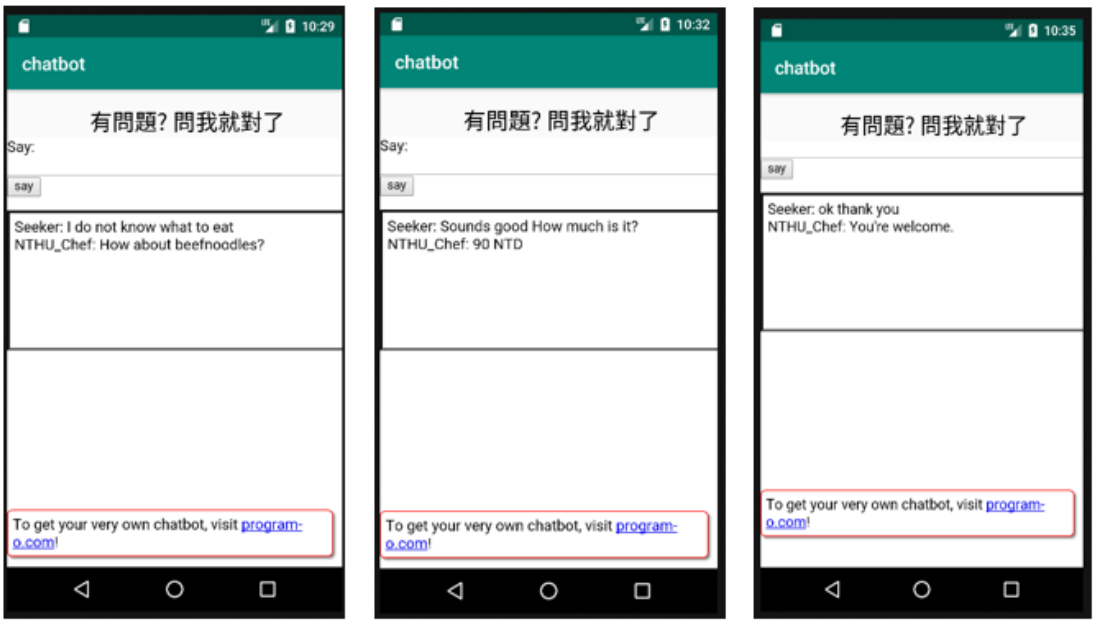


圖 6. 2 Chatbot 建議餐點對話示意

7 結論和討論

本章依成果、限制、適用性、和未來發展討論。

- 成果

學校師生可以透過此點餐平台，大幅減少中午尖峰時的等候時間，省下來的時間可以進行午休，讓下午更有精神；因為不用等待了，加強師生在學生餐廳用餐意願，同時也為商家帶來更多商機。

透過公告功能，店家可以發布各種優惠資訊，吸引顧客上門，同時管理員也可以使用公告功能，發布餐廳營業資訊，省下白跑一趟的麻煩。

- 限制

店家在中午尖峰時期人手不足，可能會無暇理會 APP 上的訂單，因為會造成此 APP 英雄無用武之地。

- 適用性

由於可以選擇餐點後再到店家付款取餐，怕會有惡作劇的發生，例如點餐後不取，或者大量訂購後不取餐，對商家造成極大損失，也對 App 會產生不信任。

- 未來發展

未來發展希望能夠發展到校外店家，提供學校師生更方便的服務。上面部分提到的信任問題，未來希望能夠加入評分及防止惡作劇機制，例如使用者會有信用分數，如有多次點餐未取紀錄就會該帳號就會遭到停權，以及如果訂單的數量或者金額到達一定數量，使用者就必須要親自聯絡店家，以防止惡作劇的產生，對雙方都有保障。