

國立成功大學
工業與資訊管理學系
專題論文

線上訂餐系統之整合開發
- 以 FOODGEN 平台為例

指導教授：劉任修 老師

組員：劉恩孟、楊甯文、王毓淳、陳逸帆

中華民國一百零七年十二月廿七日

摘要

近年來，電子商務產業蓬勃發展，新的商業模式崛起，帶動新一波經濟發展，餐飲市場的經濟效益十分可觀，根據行政院經濟部統計處的資料，2017年度的餐飲業營業額約為四千五百億元台幣左右，且近十年來餐飲業的營業額除了2009年度因受到全球性經濟衰退影響而為負成長外，其他年度的營業額皆呈現持續增加的趨勢，但從增加的幅度可以發現到有趨緩的現象，因此本專題以探討餐飲業為例，試圖找出可能增加餐飲業營業額的機會。目前已有許多消費者在選擇餐廳時，習慣先搜尋餐廳的相關評價，傳統掛布條、發傳單的行銷方式所觸及的消費者數量遠不及網路評價的影響，越來越多消費者會根據網路上的評價與部落客分享來決定是否前往該餐廳(陳威珞, 2018)，可見得在餐飲業的競爭中考量的因素並不僅止於所提供的餐點品質，更應因應消費者對於獲得資訊的來源及方式有更加全面的對策。

本專題會藉由發放問卷已獲得目前消費者對於現有的訂餐APP的看法，進而分析各類型的訂餐APP以找出最可能符合消費者使用需求的模型。根據本專題的分析結果歸納出：「良好的使用者介面」及「良好的服務流程」是模型中必須具備的兩項要素，因此在實作設計上會著重於此。透過敏捷開法的方式實作出期望的訂餐APP雛形—「FOODGEN」線上訂餐系統，並為了相容於各式中終端裝置，本專題採用Web APP的方式以達到跨平台使用之目的；在資料庫的選擇上也根據消費者、商家的資料特性，分別選用SQL及NoSQL，使得系統更有彈性；最後，搭配簡易的UI、UX設計，讓即使第一次使用FOODGEN者也可以輕易上手。

關鍵字：線上訂餐、WEB APP、電子商務

目錄

壹、緒論	1
1.1 研究背景與動機	1
1.2 研究問題定義	2
1.2.1 研究問題	2
1.2.2 研究限制	2
1.3 研究目的	2
1.4 研究架構與流程	3
貳、文獻探討	4
2.1 敏捷開發 (Agile)	4
2.1.1 敏捷宣言 (The Agile Manifesto)	4
2.1.2 敏捷原則 (The Agile Principles)	5
2.2 Angular	5
2.2.1 Angular 演進	6
2.2.2 Angular 特色	6
2.3 資料庫管理系統	7
2.3.1 資料庫的歷史	8
2.3.2 資料庫管理系統的結構及特性	8
2.3.3 SQL與NoSQL的比較	9
參、系統分析與設計	11
3.1 系統需求分析	11
3.1.1 現有訂餐系統 APP 比較	11
3.1.2 問卷分析	19
3.1.3 可行性分析	22
3.2 系統架構	23

3.3 系統設計	24
3.3.1 Usecase diagram	24
3.3.2 Activity diagram	25
肆、系統實作與展示	26
4.1 系統技術	26
4.1.1 前端架構	26
4.1.2 後端架構	26
4.1.3 資料庫實作	27
4.2 系統介面	28
4.2.1 商家端系統介面	29
4.2.2 顧客端系統介面	31
4.3 系統流程	33
4.3.1 服務流程	33
4.3.2 現金流向	33
4.4 系統特色	34
伍、研究結論與未來展望	35
5.1 研究結論	35
5.2 未來展望	36
陸、參考文獻	37
附錄	38

圖目錄

圖一 研究流程圖	3
圖二 研究架構圖	3
圖三 Google Play商店搜尋「美食APP」結果	11
圖四 MOS Order使用介面	12
圖五 Google Play上MOS Order評論	13
圖六 Foodpanda使用介面	14
圖七 Foodpanda台南地區搜尋結果	14
圖八 食在方便使用介面	15
圖九 GOMAJI使用介面	16
圖十 微碧首頁	17
圖十一 黑色香蕉商家頁	17
圖十二 問卷調查結果 - 「您有使用過訂餐APP嗎？」	19
圖十三 問卷調查結果 - 「用過哪些訂餐APP？」	20
圖十四 問卷調查結果 - 「有哪些可以改善的部分？」	20
圖十五 問卷調查結果 - 「為什麼沒有安裝使用過訂餐APP？」	21
圖十六 問卷調查結果 - 四種策略可否提升使用者意願	21
圖十七 系統架構圖 ①	23
圖十八 系統架構圖 ②	23
圖十九 Usecase diagram	24
圖二十 Activity diagram	25
圖二十一 後端架構圖	26
圖二十二 顧客資料庫	28
圖二十三 商家資料庫	28
圖二十四 首頁	29
圖二十五 登入頁	29
圖二十六 註冊頁	29

圖二十七 店家端介面	29
圖二十八 商品管理	29
圖二十九 商品上架	30
圖三十 公告與優惠管理	30
圖三十一 商家訂單管理	30
圖三十二 歷史紀錄	30
圖三十三 使用者端介面	31
圖三十四 購物車	31
圖三十五 會員專區	32
圖三十七 評論	32
圖三十六 顧客訂單管理	32
圖三十八 結帳	32
圖三十九 訂餐流程圖	33
圖四十 金流圖	33
圖四十一 login_signup activity diagram	38
圖四十二 third party activity diagram	38
圖四十三 customer edit data activity diagram	39
圖四十四 customer comment activity diagram	39
圖四十五 vendor edit data activity diagram	39
圖四十六 vendor edit announcement activity diagram	39
圖四十七 vendor edit discount activity diagram	40
圖四十八 vendor order management activity diagram	40
圖四十九 vendor update menu activity diagram	40

表目錄

表一 MySQL和MongoDB比較表	10
表二 微碧、MOS、Foodpanda比較表	18

壹、緒論

本節將敘述本專題之研究背景與動機，以及將欲研究之內容和說明研究過程中所遭受之限制，最後列出研究目的及建立研究架構及流程。

1.1 研究背景與動機

近年來,電子商務產業蓬勃發展,新的商業模式崛起,帶動新一波經濟發展,餐飲市場的經濟效益十分可觀,根據行政院經濟部統計處的資料,2017年度的餐飲業營業額約為四千五百億元台幣左右,且近十年來餐飲業的營業額除了2009年度因受到全球性經濟衰退影響而為負成長外,其他年度的營業額皆呈現持續增加的趨勢,但從增加的幅度可以發現到有趨緩的現象,因此本專題以探討餐飲業為例,試圖找出可能增加餐飲業營業額的機會。目前已有許多消費者在選擇餐廳時,習慣先搜尋餐廳的相關評價,傳統掛布條、發傳單的行銷方式所觸及的消費者數量遠不及網路評價的影響,越來越多消費者會根據網路上的評價與部落客分享來決定是否前往該餐廳(陳威珞, 2018),可見得在餐飲業的競爭中考量的因素並不僅止於所提供的餐點品質,更應因應消費者對於獲得資訊的來源及方式有更加全面的對策。

另一方面,行動訂餐APP的出現為餐飲業提供了新興的商業管道,但是根據專題組員共同的使用經驗發現,現今的餐飲相關的APP眾多,可是實際提供訂餐服務的APP往往僅提供單一店家,這意味著如果要訂購不同店家的餐點就比需再安裝另一個APP,這點對於消費者而言相當不便。針對提供外送服務的APP,因為相關企業在台灣尚在發展階段以至於並非各縣市皆能使用此服務。綜合以上,本專題將進一步探討在行動餐飲APP上的發展要如何在提供商家便利的銷售平台,同時亦能滿足消費者需求及系統可行性。

1.2 研究問題定義

在進行研究之前，本專題先將欲著手進行研究之間題定義明確，並說明因實作面之侷限所導致之研究限制。

1.2.1 研究問題

根據觀察及實際測試後本專題歸納出以下幾個問題：

- 單一店家的APP僅提供自身的訂餐服務，使用者不會想安裝每個APP，導致各個APP使用皆不廣泛。
- 使用者體驗不佳（例如：介面設計不直觀、流程繁雜等）。
- 訂單眾多且又分成內用、外帶及電話預訂時，店家容易搞混，考慮現實問題，APP使用不廣泛和店家搞混訂單，不是系統可解決之問題，因此本專題主要將針對「使用者體驗不佳」的問題進行研究

1.2.2 研究限制

本專題雖然旨在解決各家平台不整合之間題，但由於涉及商業經營，無法實際實施於各商家，此外，本專題雖然試圖解決缺失，建立整合平台，但並不會考慮現實商業利益及商家經營，一切假設立於消費者角度，可能產生的需求為發展參考構面及命題，但於經營方仍有相當程度的探討。

1.3 研究目的

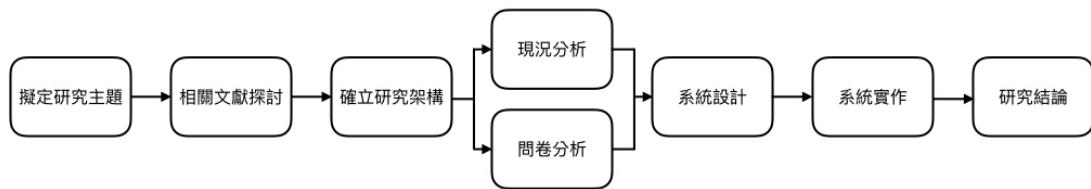
以台灣（台南為主）可能使用訂餐APP之族群為研究對象，敘述如下：

- 瞭解現有訂餐APP之間的功能差異及服務定位。
- 瞭解消費者對於現有訂餐APP之使用經驗。
- 探討消費者所期望系統具有之功能。
- 藉由本專題實際建構之系統作為未來行動訂餐APP發展方向。

1.4 研究架構與流程

本專題在第一階段先搜集既存訂餐 APP 使用狀況，並確立研究動機、目的及問題特性，再針對目前台灣訂餐電商平台優缺點及現況，分析行動訂餐市場在台灣不盛行之原因，進而研擬研究計畫書；第二階段，本專題在第一階段的基礎上進行系統設計，探討服務流程，定義服務範圍；第三階段，本專題將以前述之系統設計為基底，嘗試發展一套訂餐電商模式；第四階段，實作測試系統，優化使用者體驗。

研究架構及流程可歸納如圖一及圖二所示：



圖一 研究流程圖



圖二 研究架構圖

貳、文獻探討

本節將針對本專題所採用之系統開發方法（敏捷開發）、前端開發工具（Angular）及資料庫管理系統進行介紹。

2.1 敏捷開發 (Agile)

本專題採用敏捷開發的方式，以緊密的團隊合作、共同作業和頻繁的溝通及修改軟體版本完成開發。

敏捷開發，是一種從1990年代開始逐漸引起廣泛關注的一些新型軟體開發方法，是一種應對快速變化的需求的一種軟體開發能力。它們的具體名稱、理念、過程、術語都不盡相同，相對於「非敏捷」，更強調程式設計師團隊與業務專家之間的緊密共同作業、面對面的溝通、頻繁交付新的軟體版本、緊湊而自我組織型的團隊、能夠很好地適應需求變化的程式碼編寫和團隊組織方法，也更注重軟體開發過程中人的作用。

敏捷軟體開發描述了一套軟體開發的價值和原則，在這些開發中，需求和解決方案皆通過自組織跨功能團隊達成。敏捷軟體開發主張適度的計畫、進化開發、提前交付與持續改進，並且鼓勵快速與靈活的面對開發與變更。這些原則支援許多軟體開發方法的定義和持續進化。「敏捷軟體開發宣言」定義了相關的價值和原則。敏捷軟體開發的框架不斷的發展，兩個最廣泛被使用的是Scrum與Kanban。

2.1.1 敏捷宣言 (The Agile Manifesto)

- 獨立的工作成員與人員互動勝於流程與工具的管理。
- 工作產生的軟體勝於廣泛而全面的文件。
- 客戶的合作勝於契約的談判。
- 回應變動勝於遵循計畫。

2.1.2 敏捷原則 (The Agile Principles)

- 最為優先的事情是透過早期與持續交付有價值的軟體來以追求客戶滿意度。
- 歡迎需求的變動，即使是在開發的晚期。敏捷式流程駕馭變動來作為客戶的競爭優勢。
- 頻繁交付工作產生的軟體，自數週至數月，週期越短越好。
- 領域專家與開發成員必須一同作業，並貫穿整個專案開發時期。
- 使用積極的工作成員來建構專案，給予適當的環境以及支援所需的一切，然後信任對方能夠完成工作。
- 在開發團隊中最快也最有效的傳遞資訊方法就是面對面的溝通。
- 工作產生的軟體是衡量進度最主要的依據。
- 敏捷式流程倡導水平一致的軟體開發。
- 專案發起者，開發人員以及使用者都必須持續的維持專案進度。
- 持續重視技術的優勢以及設計品質。
- 最好的架構、需求以及設計會出現在能夠自我管理的團隊裡。
- 在規律的反覆之間，團隊會反省與思考如何更有效率，然後相對的來調整與修正團隊的開發方式。

2.2 Angular

AngularJS是一款由Google開發出來的一款開源 JavaScript 框架，用來協助單一頁面應用程式運行。它的目標是透過MVC模式（MVC）功能增強基於瀏覽器的應用，使開發和測試變得更加容易。

2.2.1 Angular 演進

Angular 共有 1、2、4、5 這麼多版本，1 正名為 AngularJS，而 2、4、5 版為 Angular。Angular 是一個前端開發平台，可以幫忙本專題更輕鬆的構建 Web 應用程式。Angular 集聲明式模板（declarative templates）、依賴注入（dependency injection）、端到端工具和一些實踐方式於一身，可以解決開發層面的各種挑戰。Angular 為開發者提升構建 Web、手機或桌面應用程式的能力。

2.2.2 Angular 特色

函式庫讀取包含附加自訂（標籤屬性）的HTML，遵從這些自訂屬性中的指令，並將頁面中的輸入或輸出與由JavaScript變量表示的模型綁定起來。這些JavaScript變量的值可以手工設定，或者從靜態或動態JSON資源中獲取，是一個“ALL-IN-ONE”的框架，這就意味著只要掌握了Angular 就可以完成大量的前端工作了。Angular 最顯著的特徵就是其整合性。一體化框架涵蓋了M、V、C/VM等各個層面，不需要組合、評估其它技術就能完成大部分前端開發任務。這首先得益於它對各種技術的封裝，讓使用者不用關心它的實現細節。Angular 隔離了瀏覽器的細節，大多數工作甚至都不知道 DOM 等前端知識就可以完成。這樣可以有效降低決策成本，提高開發速度，對需要快速起步的團隊是非常有幫助的。

Angular 選擇了 TypeScript 作為主語言。簡單來說 TypeScript 可以說是 Super 版的 JavaScript，目的是改用一些結構良好或是更輕鬆的語言來開發應用程式，開發完後再編譯回 JavaScript，更重要的是 TypeScript 完全相容 JavaScript。如果是個 C# 工程師，一定會對它的語法感覺似曾相識。TypeScript、C#、Delphi 都是傳奇人物 Anders Hejlsberg 創建，強類型、類、接口、註解等等。

Angular在前端實現了服務與依賴注入的概念。依賴注入的觀念就是將所有東西先在「外面」準備好，然後再帶入「內部」的程式中，如此一來就能夠在檢視程式碼的時候，一目了然地知道這個程式依賴著哪些類別。

Angular對團隊作戰提供了良好的支持，比如模板與程式碼的分離、樣式表的局部化、組件化的設計、服務與依賴注入體係等。這些特性讓工程師可以先專注於可以先處理模型、交互邏輯、樣式、模板等等技術中自己擅長的部份，自己不擅長的部分則交給隊友。

Angular 中將一個網站應用程式拆成一個一個組件 (Component) ，而每個組件又可以是更小的組件所構成，而這種方式在現代前端開發框架中已經非常常見，在 React 與 Vue 中也可以看到差不多的概念。好處是一個功能為一個組件，彼此獨立，在修改、抽換上都比較方便，日後維護和拓展也方便。再者，Angular 中除了 Component 還有 Service、Directive、Pipe Router 等等物件。而每個組件由三塊所構成，分別為 component.html、component.css、component.ts，由 ts 來做邏輯處理，並串連 HTML 模板和 CSS。模板藉由和 ts 數據綁定以及套上 Directive 和 Pipe 使的網頁呈現和邏輯更加強大。而所有組件整合在一起可以變成一個模組 (module)，一個 Angular 開發的應用程式可以同時使用好幾個模組。總之，整個架構就像疊積木一樣，積木又可以是更小的積木，最後蓋出成品。

2.3 資料庫管理系統

一種針對物件資料庫，為管理資料庫而設計的大型電腦軟體管理系統。具有代表性的資料管理系統有：Oracle、Microsoft SQL Server、Access、MySQL 及 PostgreSQL 等。通常資料庫管理師會使用資料庫管理系統來建立資料庫系統。

2.3.1 資料庫的歷史

早期電腦剛被發明出來時，尚未有完整的資料庫管理系統，電腦的功能充其量只能用於儲存資料、檢索這兩項功能而已，而儲存資料的方式必須透過赫爾曼·何樂禮（Herman Hollerith）所發明的打孔卡將資料轉換成電腦可以讀取的訊息進行紀錄，然而這些功能不足以構成一個資料庫管理系統；直到1960年，查爾斯·巴赫曼（Charles William Bachman）所開發出第一代網狀式資料庫管理系統是針對階層式資料庫進行改良而成的，實際應用層面追求最大化效能，雖然在處理巨量資料時可以非常有效率，然而在隨著資料量的增加會使得資料的結構愈趨複雜化，不易資料庫管理者理解。

1970年代，埃德加·弗蘭克·科德（Edgar F. Codd）提出關聯式資料庫為現代的資料庫管理系統奠定了基礎，大幅改善了網路式資料庫不易理解的問題，直到現在使用頻率最高的資料庫管理系統，諸如IBM DB2、Oracle、MySQL和Microsoft SQL Server，都是應用關聯式資料庫。到了1990年代，伴隨者物件導向的程式設計逐漸流行，並且發現到關聯式資料庫不利於處理巨量資料的索引等規模大的資料，因此提出了NoSQL的概念用以避免SQL進行JOIN所產生的問題，而現在使用NoSQL為架構的代表是MongoDB及Apache Cassandra。2011年，451 Group的分析師Matthew Aslett提出NewSQL是現今資料庫領域的最新概念，在NewSQL中與過往關聯式資料庫的差異有三種類型，分別是新架構、SQL引擎及透明分片。

2.3.2 資料庫管理系統的結構及特性

使用資料庫管理系統來儲存資料可以達到許多好處，主要的原因可以總結成五項：降低資料重複性（Redundancy）、達成資料一致性（Consistency）、資料獨立性（Data Independence）、確保資料完整性（Integrity）、資訊安全性（Security）。依據ANSI/SPARC所提出的架

構可以將資料庫分成三層，目的是為了將使用者所操作的系統與資料庫實體分開以助於管理者了解系統架構，這三層分別是外部層（external level）、概念層（conceptual level）以及內部層（internal level），而資料庫的使用者可以檢視的部分為外部層，資料庫的設計者會在概念層針對資料庫本身進行功能更新或修改，至於電腦實際是如何儲存資料是屬於內部層。

資料庫管理系統在傳遞事務（transaction）的過程中，透過資料庫的四個特性來確保該事務的完成，此四個特性為：

① 原子性（Atomicity）：

一個事務只可能有兩種狀態，一是事務完全沒被執行，二是事務執行完成，並不會有執行到中途停止的事務被記錄下來的情況，如果事務執行途中被終止時，會退回該事務尚未執行前的狀態。

② 一致性（Consistency）：

一個事務開始與結束時，資料庫的完整性沒有被破壞。

③ 隔離性（Isolation）：

又稱為獨立性，在資料庫允許多個事務同時進行時，為了避免不同事務同時修改資料時導致資料不一致，因此會將事務隔離成不同的級別以決定修改同一資料的優先順序。

④ 持久性（Durability）：

事務一旦執行完成，修改便會永久存在並不會因為系統故障而導致資料遺失。

2.3.3 SQL與NoSQL的比較

SQL是Structured Query Language的縮寫，是一種介於關聯代數與關聯演算之間的結構化語言，因為在語法上允許巢狀結構使得語言的彈性極大，現今多被用於關聯式資料庫管理系統。關聯式資料庫的代表是

MySQL是一個小型、精簡且容易使用的資料庫伺服器，多被用於中小型企業或為個人使用。

NoSQL是Not only SQL的所寫，目的是用於處理巨量級別的資料所使用，在使用上並沒有像SQL一樣有強制規定要使用模式（Schema）的方式來新增表格（table），以現在較為熱門的MongoDB為例，儲存資料是使用文本的方式儲存，強調的是關鍵值（Key）與數值（value）的組合，儲存格式與JSON格式完全相同，由於規範並沒有SQL嚴謹，也使得NoSQL的靈活性更為提升。

MySQL	MongoDB
ACID Transactions	ACID Transactions
Table	Collection
Row	Document
Column	Field
Secondary Index	Secondary Index
JOINS	Embedded documents, \$lookup & \$graphLookup
GROUP_BY	Aggregation Pipeline

表一 MySQL和MongoDB比較表

參、系統分析與設計

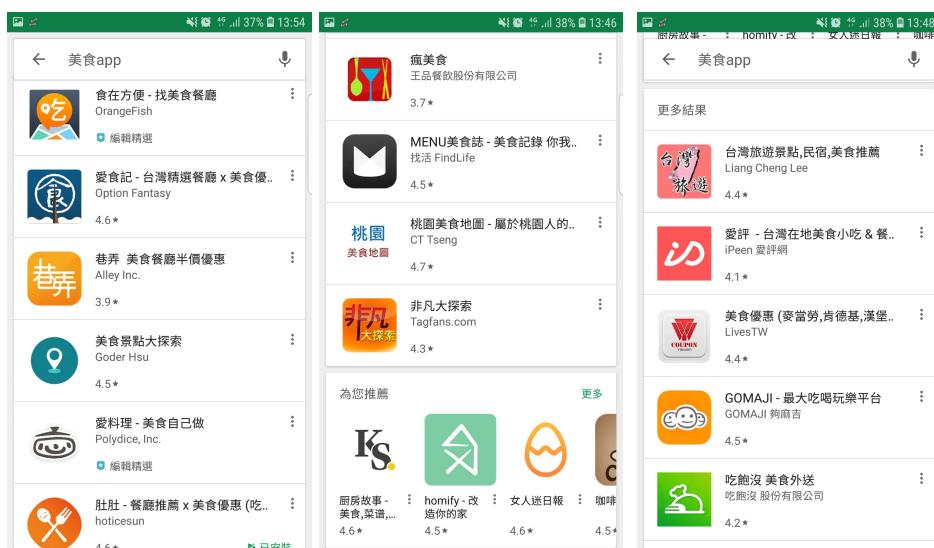
本節將進行系統開發之前置作業，包含進行需求分析、建立系統架構及規劃系統設計，並繪出所需之 Usecase diagram 和 Activity diagram。

3.1 系統需求分析

為了能實際了解使用者需求，本專題首先比較現有的訂餐相關 APP 以了解各個平台上既存的功能，針對各功能進行比較以歸納出使用者可能期望的訂餐平台架構及功能，透過問卷的調查來檢視及映證本專題所提出之對現有平台功能的質疑是否確實存在，以及可能解決方案是否為消費者期望之架構。

3.1.1 現有訂餐系統 APP 比較

根據 Google Play 商店搜尋「美食APP」的結果(如圖三)可以發現到有十幾種相關的APP，為了清楚了解各類型的APP所提供的服務，本專題成員依照下載次數前七名的APP實際下載並進行功能分析，本專題將這些APP分成四大類分別為「自有品牌」、「專職外送」、「探索型」及「折價券型」。另外，為了後續進行問卷調查，故增加一間成大學生較為熟悉的早餐店所使用的APP一同進行分析與比較



圖三 Google Play 商店搜尋「美食APP」結果

(1) 自有品牌

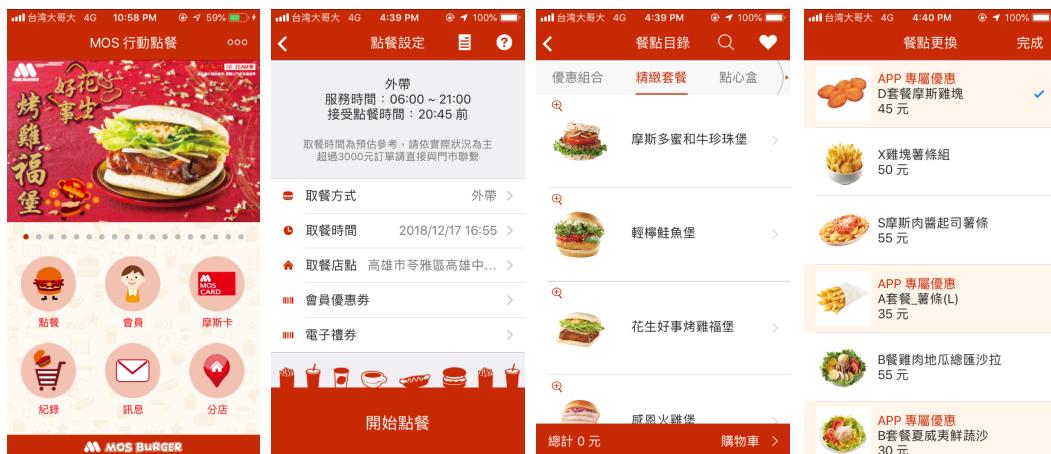
本專題定義此類別為自有餐飲企業品牌針對自身所開發且具有點餐功能的APP。由於搜尋美食APP的結果中符合的僅有「瘋美食」，因此本專題增加麥當勞的APP—「麥當勞歡樂送」及摩斯漢堡的APP—「MOS Order」一同比較。而經過使用後發現到MOS Order在使用上的流程最為人性化，且可以提供訂餐服務，故以分析此APP作為此類型的代表。

[MOS order]



① 簡介：

摩斯漢堡向來以「米漢堡」的創始企業聞名，「米漢堡」是一種結合日本傳統飯糰與西式漢堡概念而成的產品，以米壓製成的米板取代麵包來製作漢堡。與其他速食業求快速、求廉宜的訴求不同，強調「嚴選素材」與「現點現做」的原則，以較佳的品質受到許多顧客的青睞



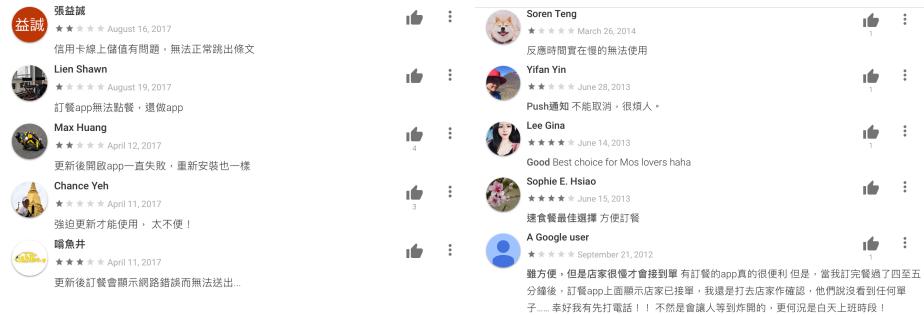
圖四 MOS Order使用介面

② 優點：

- 取餐方式多樣：外帶、內用、外送、車道。
- 可定位顧客所在地，尋找最近的 MOS 商家。

③ 缺點：

- 時常出現程式 bug。(如圖五)
- 只有單一店家,食物種類選擇侷限,導致消費者下載 APP 欲望降低



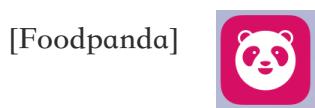
④ 點餐服務流程：

點選「點餐」按鈕 → 選擇取餐方式、時間及地點 → 選擇餐點

→ 訂單確認 → 送出訂單

(2) 專職外送

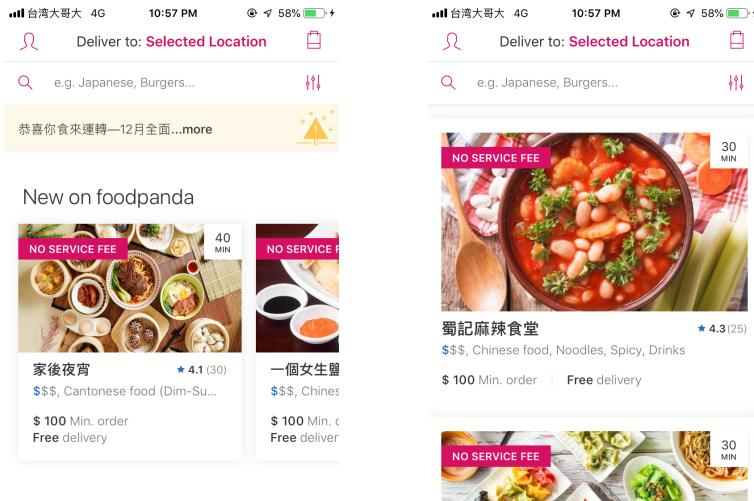
本專題定義此類別為專門設計為提供餐廳外送服務且提供消費者訂餐服務之APP。由於搜尋結果中僅有「吃飽沒美食外送」，因此本專題加入知名外商外送APP「Foodpanda」及「UberEATS」作為比較。而又以Foodpanda成立時間最早，因此推測其可能累積較多的經營經驗，因此選擇它作為此類型的代表進行深入的分析。



① 簡介：

為全球知名電子商務平台APP,其功能主要為提供第三方外送線上訂餐服務。Foodpanda Taiwan是台北最大的線上外送訂餐平台,全台有

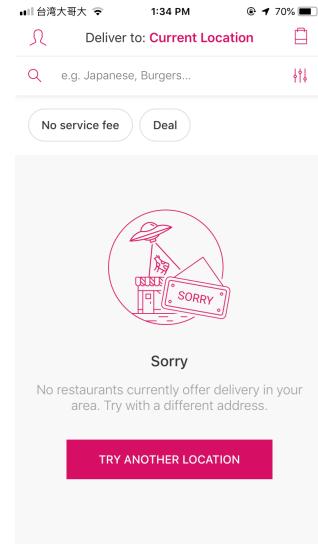
7000多間簽約店家,但服務範圍僅於台北、新北、桃園、新竹、台中及高雄,目前台南並不在服務範圍內。



圖六 Foodpanda使用介面

② 優點：

- 為第三方外送
- 食物種類多、店家選擇多
- 可直接搜尋附近店家



圖七 Foodpanda台南地區搜尋結果

④ 點餐服務流程：

選擇餐廳 → 選擇餐點 → 確認餐點及訂餐資訊 → 送出訂單

(3) 探索型

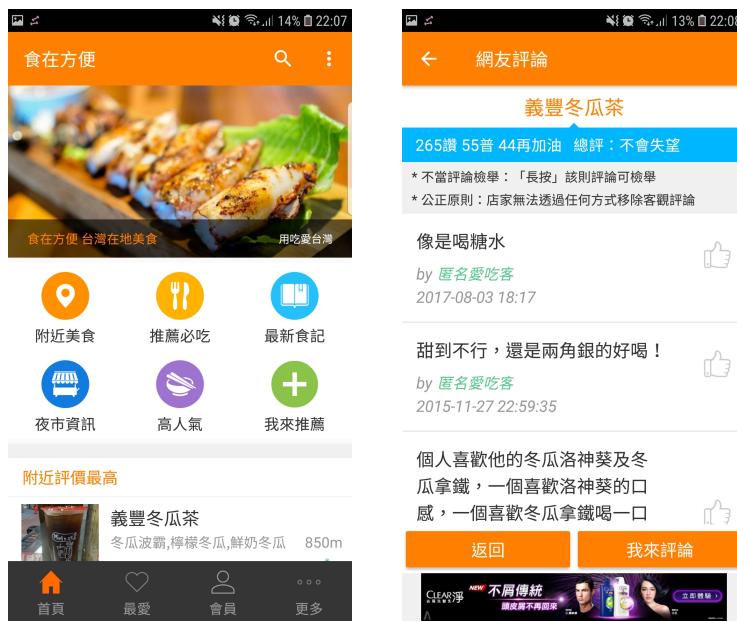
本專題定義此類別為提供消費者透過關鍵字搜尋或地圖搜尋的方式探索可能感興趣的餐廳但不提供訂位或點餐服務之APP。搜尋「美食APP」的結果中，幾乎都屬於這一類型，因此本專題挑選下載次數最多之APP—「食在方便」作為此類型的代表。

[食在方便 - 找美食餐廳]



① 簡介：

為一款由OrangeFish所開發的免費APP，此APP使命是「將台灣的吃發揚光大，對消費者與店家帶來幫助」，在Google Play的「旅遊與在地資訊」類別連續30個月第一名，且曾獲得多項APP設計相關獎項。



圖八 食在方便使用介面

② 優點：

- 使用者可以探索周邊餐廳的功能，找尋離自身最近且感興趣的店家
- 針對不清楚要吃甚麼的使用者，提供餐廳清單供其參考
- 可以依據特定地區、種類、評分等方式進行篩選目標餐廳

(3) 缺點：

- 餐廳經營資訊更新不及時，導致有餐廳已歇業但仍顯示營業中
- 有部分使用者認為使用介面不佳

(4) 折價券型

本專題定義此類別為提供顧客購買各家餐廳的折價券但不提供訂位或點餐服務之APP。根據搜尋結果符合有「巷弄」、「GOMAJI」，又以「GOMAJI」在Google Play商店中所顯示的下載次數較多及評分較高，故選擇其作為此類型的代表。

[GOMAJI - 最大吃喝玩樂平台]



(1) 簡介：

全台最大吃喝玩樂平台GOMAJI以活化閒置資源、協助店家賺錢、提供消費者省錢且美好體驗為成立初衷。面對網路產業的瞬息萬變，GOMAJI不斷面對挑戰積極發展更全面的O2O模式，將努力帶給消費者更豐富、更便利、更優惠的生活消費體驗。



圖九 GOMAJI使用介面

② 優點：

- 提供各種餐廳的優惠券，折扣後的金額可能不到原價的一半
- 時常有限時的搶購優惠活動
- 提供除了餐廳以外的優惠券

③ 缺點：APP類型與本次專題方向較不相符，因此不予以探討

(5) 早餐店—「黑色香蕉」所使用之APP

[微碧智慧店面(行動方案)]



① 簡介：

「微碧智慧店面」以iPad POS 點餐系統為主體，延伸出「行動版微碧愛點餐」，以月租的方式提供店家使用「微碧愛點餐」點餐 App，消費者可從手機直接傳送訂單到店家。支援功能包含庫存管理(月租費會增加)、電子發票(月租費會增加)、掃描結帳、信用卡、行動支付、帳務分析、雲端資料同步並備份。目前多為早餐店但總使用店家數不多。



圖十 微碧首頁



圖十一 黑色香蕉商家頁

② 優點：

- 整合店家後端庫存管理
- 行動應用程式整合POS系統，資訊可以完整被記錄於後端資料庫內

③ 缺點：

- 使用者介面不佳：無法瀏覽店家，必須知道店名或店家代碼
- 加入行動方案之店家數不多，多為早餐類
- 無法定位消費者目前位置

④ 點餐服務流程：

輸入店家代碼 → 點餐 → 送出訂單

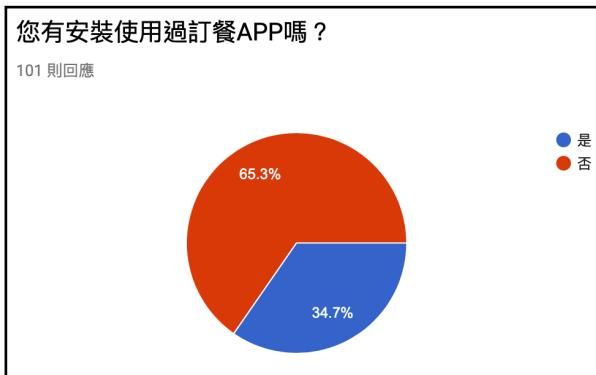
針對以上五個APP進行比較，結果整理成下表：

	微碧智慧店面 (行動方案)	MOS 行動點餐	Foodpanda 訂餐&外送	食在方便	GOMAJI
店家數	全台：900-1000 台南：10-20	全台：261 台南：8	全台：7000 台南：0	無實際資料，估計至少20000以上	5000家品牌以上，但包含餐廳之外的類別
是否外送	否	大部分有	是	否	否
是否可預約 內用/外帶	是	是	否	否	否
食物種類	早餐為主	單一	多元	多元	多元
後端管理	是	是	否	否	是，GOMAJI 店家系統
定位顧客目前位置	否	是	是	是	否

表二 微碧、MOS、Foodpanda比較表

3.1.2 問卷分析

本專題總共回收了101份有效問卷，針對消費者部分（問卷結果主要以18-30歲經常外食的學生族群為主，台南地區的使用者為多數），將其分為使用過訂餐APP及未使用過訂餐APP兩大類，分別調查使用者體驗及使用者需求，目的是為了了解消費者使用習慣及相關APP可改善方向，作為後續系統設計之依據。而根據調查結果（如圖十）可以發現到約有三分之一的消費者有使用過相關的APP，另外三分之二的消費者未曾使用過訂餐APP，後續本專題將於①及②深入分析這兩類消費者，於③分析目前消費者對於訂餐APP的需求。



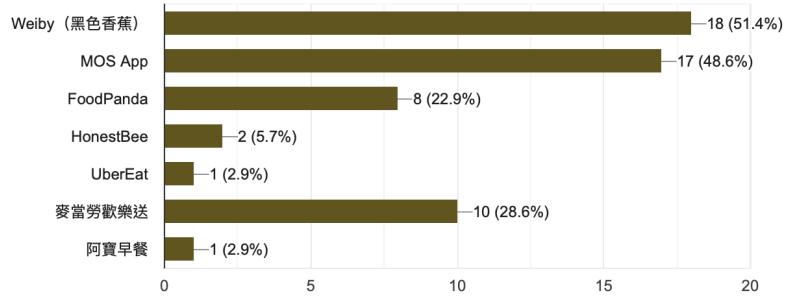
圖十二 問卷調查結果 - 「您有使用過訂餐APP嗎？」

① 使用過訂餐 APP 之消費者

在使用過訂餐APP之消費者中，本專題開放此問題為複選題以了解消費者使用各種APP的情況；根據調查結果（如圖十一），使用過Weiby之消費者居首，MOS APP次之，麥當勞歡樂送第三，FoodPanda第四，而剩餘的APP則佔少數。由於問卷填答的對象主要為成大學生，因此在臺南沒有提供服務的FoodPanda及UberEat比例上會相對較其他有提供之地區低，此外，Weiby所提供之服務主要以早餐店為主，而成功大學附近恰巧有一間使用該APP之早餐店，所以Weiby再填答的比例而言相對其他區域會相對較高。

用過哪些訂餐APP？

35 則回應

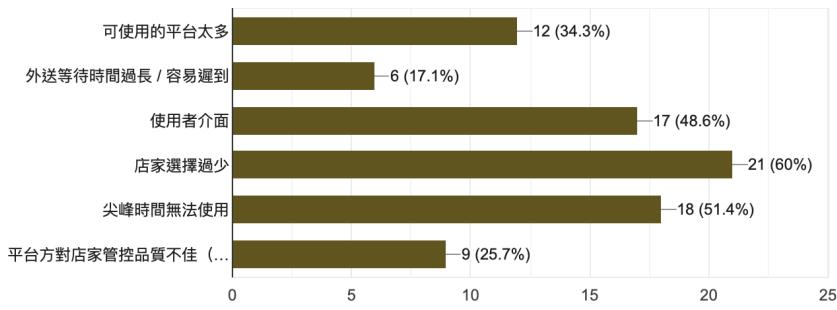


圖十三 問卷調查結果 - 「用過哪些訂餐APP？」

為了能更深入了解有相關經驗的消費者對既有APP可改善之處的想法，在問卷中提供幾個現有APP可能存在的問題，透過調查來了解是否是大多數使用者在過去使用經驗中也曾面臨到這些類型的問題。從調查的結果（如圖十二）可以明顯看出最多人認同的問題是現有APP可以選擇點餐的店家數過少，而尖峰時間無法使用及使用者介面的問題位居第二及第三，而尖峰時段當機的問題牽涉許多原因，不在本專題討論範圍。

有哪些可以改善的部分？

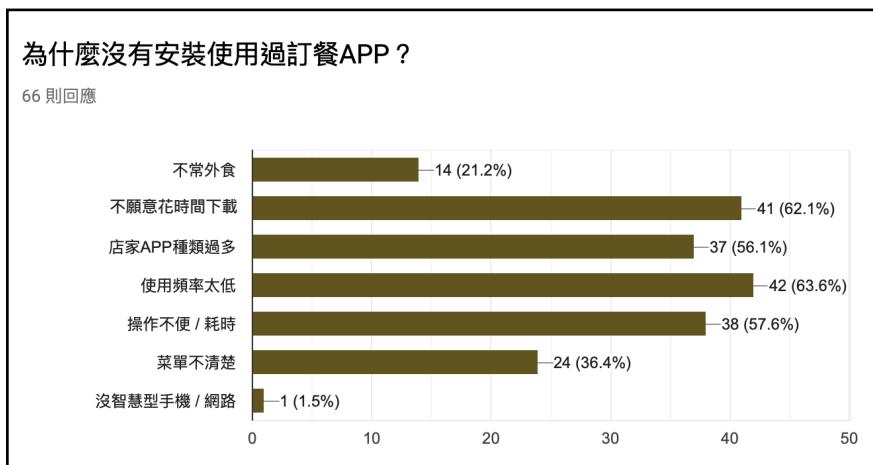
35 則回應



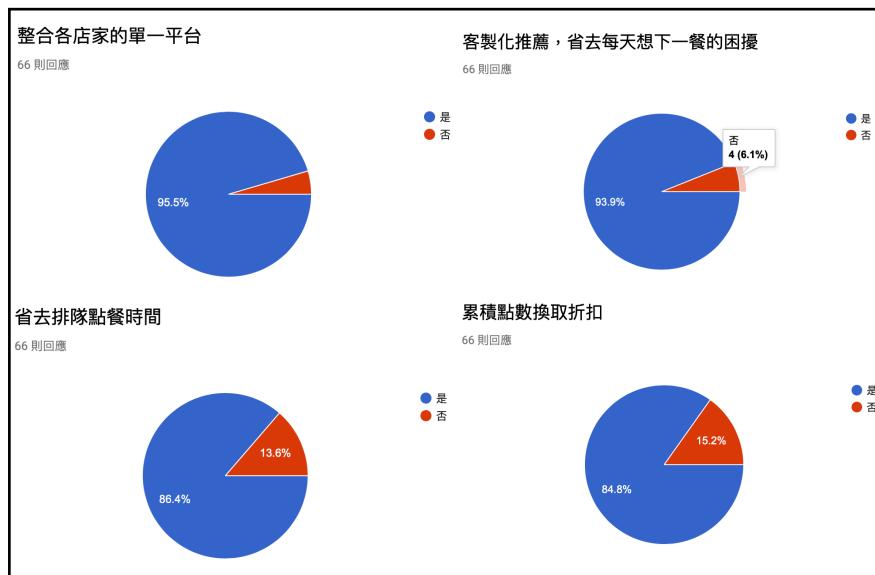
圖十四 問卷調查結果 - 「有哪些可以改善的部分？」

② 未使用過訂餐APP之消費

在未使用過訂餐APP的消費者中，不曾安裝的原因以使用頻率太低及懶得下載為主，操作不便或耗時及店家APP種類過多次之（如圖十三）。此外，本專題亦調查了四種改善方法及策略是否能提升使用者的使用意願，分別是整合各店家形成單一平台、省去排隊點餐時間、累積點數換取折扣及客製化推薦，結果顯示四種策略均能或多或少提升消費者意願（如圖十四），因此本專題將由此為基礎，作為本系統之開發方向。



圖十五 問卷調查結果 - 「為什麼沒有安裝使用過訂餐APP？」



圖十六 問卷調查結果 - 四種策略可否提升使用者意願

③ 消費者需求分析

從問卷結果分析可歸納出目前消費者需求，整理成兩大部分：

第一，目前訂餐APP待改善的主要三點：

- 店家選擇過少
- 尖峰時間無法使用
- 使用者體驗

第二，有助於鼓勵消費者使用點餐 APP 的四個誘因：

- 積累點數換折扣
- 客製化推薦
- 單一平台
- 增加效率

3.1.3 可行性分析

本專題會實際建置系統，為了能確立使用者需求是否可以在系統中實現，故進行可行性分析來篩選可以實際達成之需求，如無法實現會提出其不可行的原因，並會提供可能解決之方向：

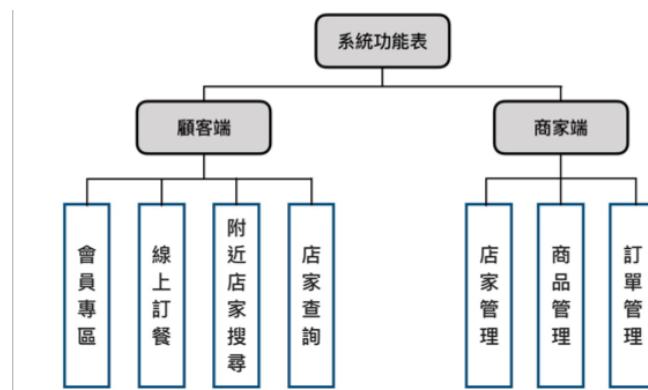
- 店家選擇過少：牽涉現實店家參與意願，本系統並無實際上線運作，無法討論這項問題。
- 尖峰時間無法使用：由各種因素導致，包括系統負載量、網路穩定性...等，問題過於複雜，目前不在本專題研究範圍。
- 使用者體驗：本專題將針對「介面」及「流程」進行改善。
- 使用設備限制：本專題將利用 Web APP，不管是Android、IOS 手機亦或是電腦，都可使用。

Wang et al. (2018) 認為在行動餐飲APP中感知價值對於網路口碑的影響相較使用者滿意度大；而使用者滿意度對於再次使用APP的影響

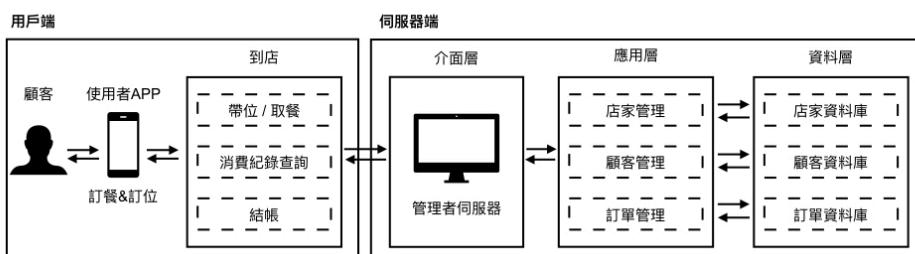
相較感知價值大。由於本專題著重於設計及開發APP，並無實際與商家合作，因此在消費者的感知價值無法在本專題中實踐，因此會將重心放在「使用者介面」及「服務流程」。

3.2 系統架構

本專題之研究架構為圖十五及圖十六，首先在用戶端顧客登入本系統後，可使用線上訂餐、訂位、外送等店家所提供的服務，接著顧客透過第三方支付確認付款後，將訂單傳送給店家，待店家確認訂單，開始準備餐點，依預約時間送餐，顧客在等待或到位用餐前，查詢目前訂單進度，用餐完畢後會有紅利點數累計，並紀錄該用戶歷史消費，日後作為推薦依據。伺服器端主在於提供用戶端訂餐系統內容相關資料查詢、顯示、管理等功能，透過介面層與應用層的連接，應用層存取的資料再回傳給介面層，最後再由介面層傳遞給系統作資料的顯示，應用層的資料來源是由資料層取得會員資料、菜單資料、訂單資訊、訂位資訊等，回傳給介面層，最後由進行新增訂單、查詢等功能。



圖十七 系統架構圖①



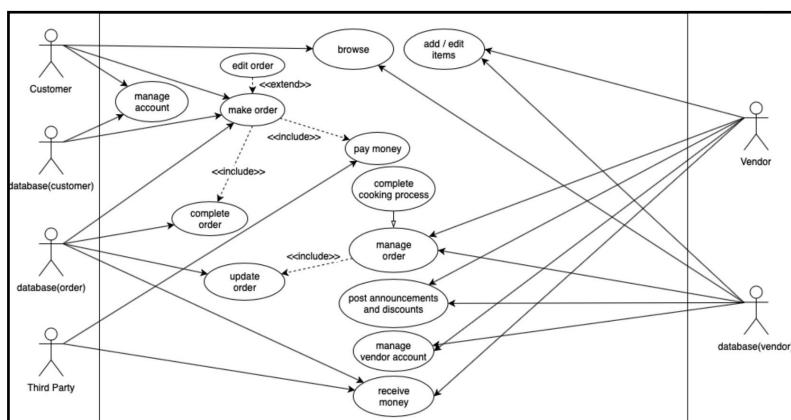
圖十八 系統架構圖②

3.3 系統設計

系統設計將繪出 Usecase diagram 及 Activity diagram，作為之藍圖，以此為基礎進行後續實際系統之開發。

3.3.1 Usecase diagram

應用敏捷開發的方法在設計系統架構中，強調快速生產出可以實際運行的軟體，因此本專題利用 Usecase diagram 呈現 FOODGEN 系統中的使用者與系統之間的互動關係。Usecase diagram 中有三個種類的使用者及三個資料庫，使用者分別是消費者、商家以及第三方支付公司，而資料庫分別是消費者資料庫、商家資料庫及訂單資料庫，在此先簡述各個使用者可以操作的功能。消費者可以使用的功能包含瀏覽商家、下餐點訂單、管理個人帳戶的功能；瀏覽店家時，會存取商家資料庫內的商品資訊；下餐點訂單時，會將訂單存入訂單資料庫，而消費者在下訂單後必須要付款才能使訂單成立；管理個人帳戶可以修改個人基本資料。商家可以使用的功能包含商品管理、訂單管理、公告與優惠管理、商家帳戶管理；商品管理中可以新增、修改及刪除商品並會對應更動商家資料庫；訂單管理可以檢視各種類型的訂單以及各訂單的狀態，這些資料會從訂單資料庫內提取；公告與優惠管理及商家帳戶管理均是對應到商家資料庫。第三方支付公司會在訂單成立時暫時保管從消費者端所支付的款項，直到消費者取完餐點並確認無誤後再將款項支付至商家的帳戶。

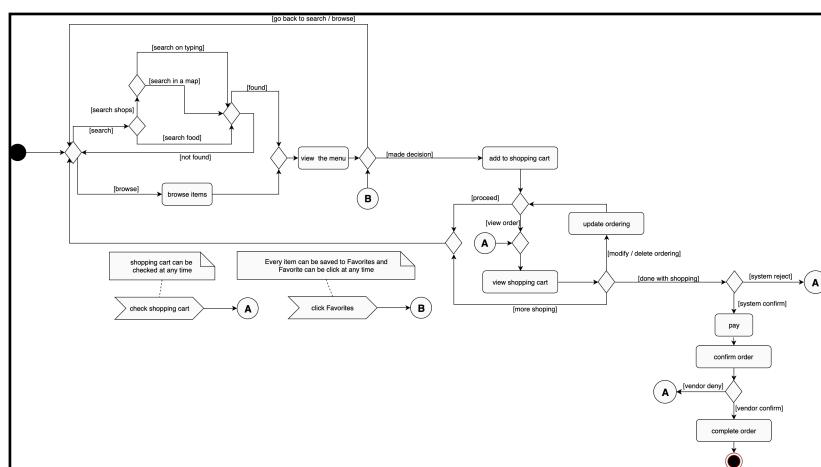


圖十九 Usecase diagram

3.3.2 Activity diagram

本專題從消費者角度出發重新設計線上訂餐系統，下圖為本系統訂餐之活動圖，消費者進入首頁後，可以選擇隨意瀏覽店家（本專題將會針對時段推薦合適選擇，例如：下午時段推薦甜點），或是選擇搜尋店家及餐點，搜尋店家除了可使用關鍵字搜尋，也可用地圖定位找出離消費者較近的店家，搜尋成功可點擊照片進入店家菜單，若無搜尋結果將返回首頁；在過程中，隨時可返回首頁，也隨時都可以點擊餐點圖片上的訂餐按鈕及收藏按鈕，加入購物車或收藏，而購物車也一直顯示在頁面上，供消費者隨時確認當前選購內容，收藏內容則可在會員頁面查看。

進入購物車後，可修改訂單內容，確認當前訂購內容無誤後，便可送出訂單，若未被系統拒絕訂單¹，消費者接著要進行信用卡付款的作業（款項會由第三方平台託管，待訂單完成才會匯入商家帳戶），確認款項後，通知商家訂單，若未被商家拒絕訂單²，則訂單成立，商家開始製作餐點，消費者可由終端頁面查看目前進度，待取餐時間，消費者拿到餐點確認無誤，便可按下完成按鈕，即完成訂單，或是在一定時間未按下完成按鈕，系統默認為訂單完成；若是未在指定時間拿到餐點，系統會對商家進行不良計點，消費者可以選擇延長取貨時間，或是取消訂單進行退款。



圖二十 Activity diagram

¹ 訂單被系統拒絕的狀況包含：店家目前接單數已超過設定的負荷量、當前系統有狀況..等等不可預期之情況。

² 商家拒絕訂單的狀況：逾時未按下確認鍵、突發狀況無法接單...等。

肆、系統實作與展示

本節將介紹 FOODGEN 實際建構所使用之技術，並展示實際系統畫面和主要系統流程說明。

4.1 系統技術

本專題所使用之技術分為前端、後端、資料庫，以下將依序進行介紹。

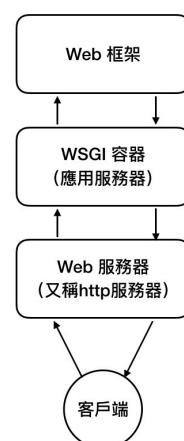
4.1.1 前端架構

前端架構的選擇上，本專題在 Angular 和 React 兩者間抉擇，AngularJS 是由 Google 所開發的前端 JavaScript 框架，可用 HTML 實現 Declarative templates 與 data-binding，控制介面的運作方式，且提供完整的專案環境與工具，可快速上手。React 是由 Facebook 所開發的 Javascript 程式庫，使用 JSX 建立 HTML 模板，強調模組化、以及將網頁劃分為獨立的區塊，並同樣能實現資料繫結。但 React 並不提供所有架構專案需要的工具，需使用第三方套件才能建立功能完整的網頁。

Angular 和 React，一個是框架，一個是程式庫。框架能夠提供比程式庫更完整的功能與環境，因此本專題最終選擇使用 Angular 為開發工具。

4.1.2 後端架構

- Web 框架：flask，方便開發Web應用程序，HTTP 請求的動態數據庫就是由Web框架來提供的。
- WSGI 容器：guincorn，為Web服務器與Web框架間的溝通協議，遵守WSGI溝通協議便可以自由選擇Web框架及web服務器，而Web框架及Web服務器在開發上也可以分開進行不需考量兩方的具體實現。



圖二十一 後端架構圖

- Web 服務器：Nginx，客戶端瀏覽網站時，Web服務器會處理請求並將文件回傳瀏覽器，並附帶訊息告知瀏覽器如何開啟該類型文件。
- 客戶端：瀏覽器或APP。

在建構後端伺服器上，本專題透過Amazon EC2所提供的IaaS來實踐，在Web伺服器的選擇上有Apache與Nginx可做選擇，而Nginx在效能上較Apache佳且具有異步驅動的功能，此外Nginx能支援靜態文件的處置並作為反向代理服務器，有助於APP功能的實行，故選用Nginx。在Nginx接收請求後會先緩衝及響應，然後再透過unix socket將請求整理好發給gunicorn，gunicorn會以多執行序的方式執行python flask，本專題的核心APP服務都會由python flask提供。

python flask的特色是輕量級的網路框架，提供給使用者自定義的服務，相較於Django很多方法都以定義完成而言，輕量級的服務框架較符合本專題的開發需求。gunicorn在設定上較uWSGI容易且gunicorn不須編寫及配置文件，開發難度較低。最後使用supervisor來管理伺服器背景程式。

4.1.3 資料庫實作

本專題使用到了三個資料庫，分別為顧客資料庫、訂單資料庫以及商家資料庫，在資料庫的選擇上會根據資料內容的特性來決定。

考量到顧客資料庫因為欄位內的資料變動不頻繁，利用SQL的格式儲存可以使資料結構化有助於日後的維護，又因為本專題建構系統的目的在於提供可行的解決方向，並沒有要進行大量的模擬測試實際應用情況，故採用專為中、小型企業或個人資料庫使用的MySQL。

而針對商家資料庫本專題原先也考慮使用MySQL作為其資料庫管理系統，然而使本專題放棄使用MySQL的原因有二，其一為考量到商家的資料在使用上經常會需要進行JOIN，但是MySQL在進行大量且頻繁的JOIN時效率非常不佳，其二是商家有建立菜單的需要，使用MySQL進行儲存會

因為要符合正規化的因素而使得單一商家需要多個 table 來儲存，對於資料庫的管理上概念較為複雜，因此選用 NoSQL 的資料庫管理系統，而 MongoDB 是目前物件導向資料庫的代表之一，故本專題採納此系統為商家端的資料庫管理系統。

Field	Type	Null	Key	Default	Extra
<code>id</code>	<code>mediumint(9)</code>	<code>NO</code>	<code>PRI</code>	<code>NULL</code>	<code>auto_increment</code>
<code>username</code>	<code>varchar(20)</code>	<code>NO</code>		<code>NULL</code>	
<code>password_hash</code>	<code>varchar(128)</code>	<code>NO</code>		<code>NULL</code>	
<code>email</code>	<code>varchar(120)</code>	<code>YES</code>		<code>NULL</code>	
<code>mobile</code>	<code>varchar(20)</code>	<code>YES</code>		<code>NULL</code>	
<code>date_of_birth</code>	<code>date</code>	<code>YES</code>		<code>NULL</code>	
<code>gender</code>	<code>enum('m','f','a')</code>	<code>YES</code>		<code>NULL</code>	
<code>first_name</code>	<code>varchar(20)</code>	<code>NO</code>		<code>NULL</code>	
<code>last_name</code>	<code>varchar(20)</code>	<code>NO</code>		<code>NULL</code>	
<code>middle_name</code>	<code>varchar(20)</code>	<code>YES</code>		<code>NULL</code>	
<code>credit_card_number_h</code>	<code>varchar(128)</code>	<code>YES</code>		<code>NULL</code>	
<code>credit_card_expiration_date</code>	<code>char(7)</code>	<code>YES</code>		<code>NULL</code>	
<code>credit_card_security_code_h</code>	<code>varchar(128)</code>	<code>YES</code>		<code>NULL</code>	
<code>join_date</code>	<code>timestamp</code>	<code>YES</code>		<code>CURRENT_TIMESTAMP</code>	

圖二十二 顧客資料庫

```
> db.vendor.find()
[{"_id": ObjectId("5c051a527671ca3d36ee46fc83"), "products": [{"name": "湯類", "key": "5c051a527671ca3d3ca3a38b31", "index": 1}, {"name": "飯類", "key": "5c051a5871ca3d3ca2c700b5", "index": 2}, {"name": "熱炒", "key": "5c051a6171ca3d3ca5444fab", "index": 3}], "info": {"name": "文章牛肉湯", "phone": "06 228 4626", "address": "台灣台南安平區安平路590號", "tags": ["中式"], "open_hours": {"others": "0000 2400", "TUE": "1000 0000", "MON": "0000 1400"}}]
> db.items.find()
[{"_id": ObjectId("5c051a527671ca3d3ca3a38b31"), "new_item_index": 2, "items": [{"pic": "src", "base_price": 100, "options": [{"op0": {"ctr": 0}}, {"description": "", "name": "招牌牛肉湯", "index": 1}, {"pic": "src", "base_price": 100, "options": [{"op0": {"ctr": 0}}]}, {"description": "", "name": "牛腩湯", "index": 2}], "vendorID": ObjectId("5c051a527671ca3d36ee46fc83"), "category_name": "湯類", "item_order": [1, 2]}], {"_id": ObjectId("5c051a5871ca3d3ca2c700b5"), "new_item_index": 2, "items": [{"pic": "src", "base_price": 120, "options": [{"op0": {"ctr": 0}}]}, {"description": "", "name": "牛肉炒飯", "index": 1}, {"pic": "src", "base_price": 120, "options": [{"op0": {"ctr": 0}}]}, {"description": "", "name": "牛肉燴飯", "index": 2}], "vendorID": ObjectId("5c051a527671ca3d36ee46fc83"), "category_name": "飯類", "item_order": [1, 2]}], {"_id": ObjectId("5c051a6171ca3d3ca5444fab"), "new_item_index": 2, "items": [{"pic": "src", "base_price": 120, "options": [{"op0": {"ctr": 0}}]}, {"description": "", "name": "芥藍牛肉", "index": 1}, {"pic": "src", "base_price": 120, "options": [{"op0": {"ctr": 0}}]}, {"description": "", "name": "蔥爆牛肉", "index": 2}], "vendorID": ObjectId("5c051a527671ca3d36ee46fc83"), "category_name": "熱炒", "item_order": [1, 2]}]
```

圖二十三 商家資料庫

4.2 系統介面

- ① 首頁：點選上方的「搜尋附近店家」可以依據使用者GPS定位顯示附近店家。「店家查詢」可以直接輸入關鍵字來搜尋符合的店家。
- ② 登入：輸入帳號及密碼即可登入。
- ③ 註冊：如果尚未有帳號，可以輸入基本資料、帳號、密碼來建立新的帳戶，再完成手機驗證後即可從「登入」頁面登入。



圖二十四 首頁



圖二十五 登入頁



圖二十六 註冊頁

4.2.1 商家端系統介面

- ① 店家端介面
- ② 商品管理：修改商品資訊、刪除商品。
- ③ 商品上架：新增商品。
- ④ 公告與優惠管理：新增修改刪除公告、設定優惠及適用日期。
- ⑤ 商家訂單管理：檢視訂單資訊。
- ⑥ 歷史紀錄：查看已完成的訂單。



圖二十七 店家端介面



圖二十八 商品管理



圖二十九 商品上架



圖三十一 商家訂單管理



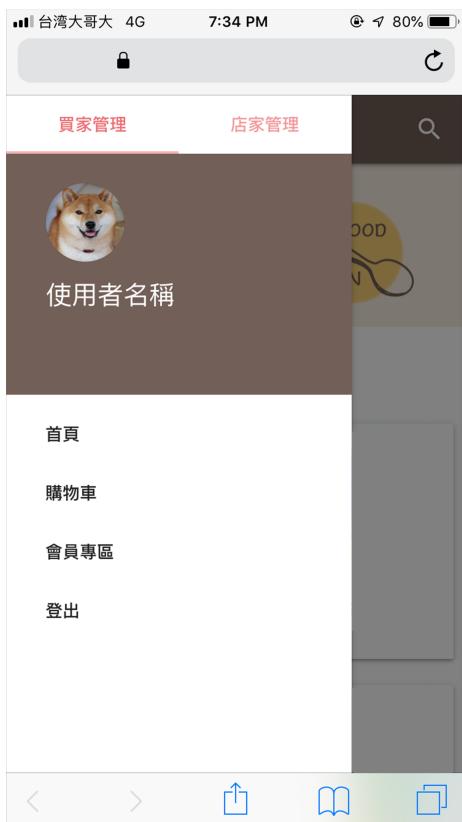
圖三十 公告與優惠管理



圖三十二 歷史紀錄

4.2.2 顧客端系統介面

- ① 使用者端介面
- ② 購物車：欲訂購的餐點可以存放在購物車內。
- ③ 會員專區：基本資料修改、紅利點數查詢。
- ④ 評論：可以評論購買過的商品或商家。
- ⑤ 顧客訂單管理：歷史紀錄查詢。
- ⑥ 結帳：付款方式及確認其他資訊。



圖三十三 使用者端介面



圖三十四 購物車



圖三十五 會員專區



圖三十六 顧客訂單管理



圖三十七 評論



圖三十八 結帳

4.3 系統流程

FOODGEN 之主要系統流程為結合消費者及商家端的中介服務流程，和結帳時所使用到之第三方支付的獻金流向兩大部分。

4.3.1 服務流程

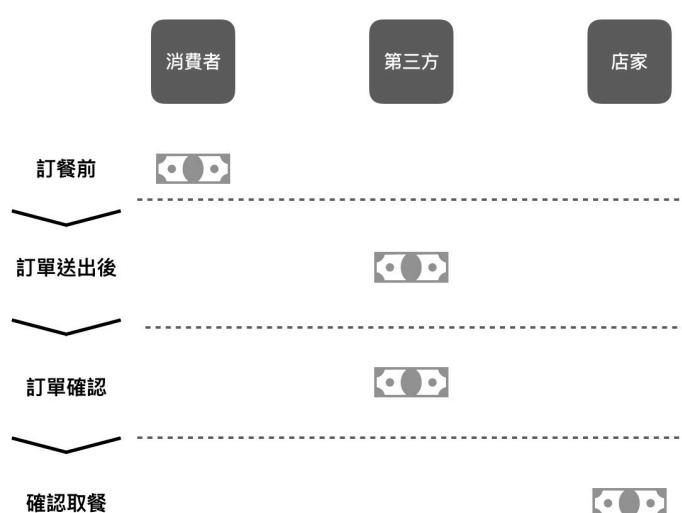
消費者在進入本系統後，先行搜尋或瀏覽，找尋目標店家，而後進行點餐及下訂單，確認付款後商家端會接收及處理訂單，開始餐點製作，最後消費者即可享用。



圖三十九 訂餐流程圖

4.3.2 現金流向

本系統採取第三方支付方式收付款，消費者在下訂單時，將錢轉入第三方，確認完成訂單後再由第三方將款項轉給店家，以此保障買賣雙方的權益，以下為此作法之現金流程圖：



圖四十 金流圖

完成訂單情況：前提為店家已按下餐點完成按鈕

- ① 顧客在取餐後按下完成訂單按鈕
- ② 顧客在取餐後仍未按下完成訂單按鈕，則採以下兩種判斷方法
 - 內用 → ($\max\{\text{預約時間}, \text{店家完成}\} + \text{兩小時}$)後，確定付款給店家
 - 外帶 → ($\max\{\text{預約時間}, \text{店家完成}\} + \text{兩小時}$)後付款給店家

4.4 系統特色

本專題開發之系統，為參考各現有的APP後加上對功能的發想和使用者體驗，經實作達成以下三項與市場相異之特色：

- 彈性的資料庫：同時運用SQL (My SQL)及 NOSQL(Mongo DB)，能夠更靈活的建構資料。
- 友善的使用者介面及服務流程：設計簡約直觀使用介面，即使初次使用也不會不知所措。
- 不受終端設備限制：使用 Web APP，不論 Android 、 IOS 或 PC ，皆可藉由瀏覽器直接使用。

伍、研究結論與未來展望

本節為本專題之整體研究結論，及簡述在未來期望可繼續發展之方向。

5.1 研究結論

伴隨系統開發工具的功能愈趨完善，要開發出可以使用的系統難度降低，然而系統開發的目的在於了解系統使用者的需求並予以滿足，如此建置出來的系統才具有競爭力，本專題透過問卷調查及現有 APP 分析，以深入了解市場現況避免開發出來的系統不符合使用者的期望。根據實作出的 FOODGEN 整合型線上訂餐系統，針對商家端而言，提供容易操作的平台管理頁面，且因單一帳號可以同時具有商家及消費者雙重身分，因此商家在管理自身店家的同時，亦可以消費者的角度瞭解其他商家的經營狀況。針對消費者而言，再瀏覽完各式店家或餐點後，可以即時透過 APP 訂位或訂餐，無須再因透過電話方式訂餐而可能遇到店家忙碌無法接聽而白白浪費時間等待的問題，且因商家資訊是由商家自身管理，因此可以大幅降低資訊更新不及時的問題。

將 FOODGEN 系統與現有訂餐APP進行比較，可以將 FOODGEN 定位於「探索型」，此外因為結合訂餐的功能，不但使消費者可以便利地搜尋後立即訂餐或訂位，對於商家而言亦可以省去額外的資金投資開發 APP，這項優勢無論是對於既有餐廳或是新開張的餐廳都提供了獲取客源的管道。最後，透過本專題為了達成所歸納出消費者所期望之系統應具備「良好的使用者介面」及「良好的服務流程」這兩項要素，在 FOODGEN 系統設計中，透過參考 MOS Order 簡潔且清晰的畫面設計及融合知名購物平台，諸如露天拍賣、蝦皮商城等方便使用者瀏覽且從選購商品至完成結帳流程快速的特點，因此即使是首次使用 FOODGEN 系統亦可以快速完成訂餐的程序。

5.2 未來展望

經由這次實作，本專題開發出一個簡單的整合性訂餐電商平台，而在此專題中主要以使用者體驗為主軸進行分析研究，且由於現實與時間的限制，仍有許多需求無法及實現，因此，本專題希望在可期的將來，針對以下幾點進行研究，致力將其完善：

- 接單效率(定義、量化)：與電話訂餐的比較(錯誤率、花費時間)、內用
用QR code點餐。
- 行動支付：讓消費者可以更便利的在線上進行支付，以減少現金流動所
造成的失誤與時間消耗。
- 客製化推薦：依照顧客訂餐紀錄，運用演算法預測顧客喜好進行推薦。
- 優惠設定更加彈性：讓系統相容各式各樣優惠形式。
- 更佳的 UX 體驗
- 建立各項機制，因應各種非常規情況，如下：
 - ① 拒絕訂單的情況
 - 顧客送出訂單後3分鐘內店家未接單則自動取消訂單
 - 店家主動拒絕訂單
 - ② 店家未能準時完成餐點之情況
 - 超過預約時間但在10分鐘內：店家可以傳訊息告知顧客預計完成時間
 - a) 顧客接受→更新預約時間
 - b) 顧客不接受→取消訂單→款項退還給顧客
 - 超過預約時間10分鐘店家未按下完成餐點鈕：顧客可無條件取消訂單
 - 超過預約時間2小時店家未按下完成餐點鈕：系統自動取消訂單

隨著人手一「機」的時代到來，企業也須以不同的商業模式應對，然而，轉變原有的消費習慣並不是一件容易的事，因此往往無法馬上被大眾接受，若本專題藉由不斷的改善線上訂餐平台、增加平台上的商家數，期望促進行動訂餐的普及。

陸、參考文獻

98年3月批發、零售及餐飲業動態調查（民國98年4月23日）。經貿透視雙周刊。民107年12月12日，取自<https://www.trademag.org.tw/page/newsid1/?id=506994&iz=6>

台灣趨勢研究(民國107年)。產業分析：餐飲業發展趨勢（2018年）。檢自http://www.twtrend.com/share_cont.php?id=63 (民國107年12月12日)

呂中力(民國104年1月8日)。你真的搞懂了什麼叫敏捷式(Agile)開發嗎。檢自<https://www.projectup.net/article/view/id/15726> (民國107年11月19日)

吳仁和(民107)。資訊管理：企業創新與價值創造。臺北市：智勝文化。

常見問題 | FOODPANDA(無日期)。常見問題 | FOODPANDA。檢自<https://www.foodpanda.com.tw/contents/contact.htm> (民國107年11月25日)

微中子(民國106年1月15日)。[Day 31] Angular 2 給初學者的學習指南。檢自<https://ithelp.ithome.com.tw/articles/10189032> (民國107年11月19日)

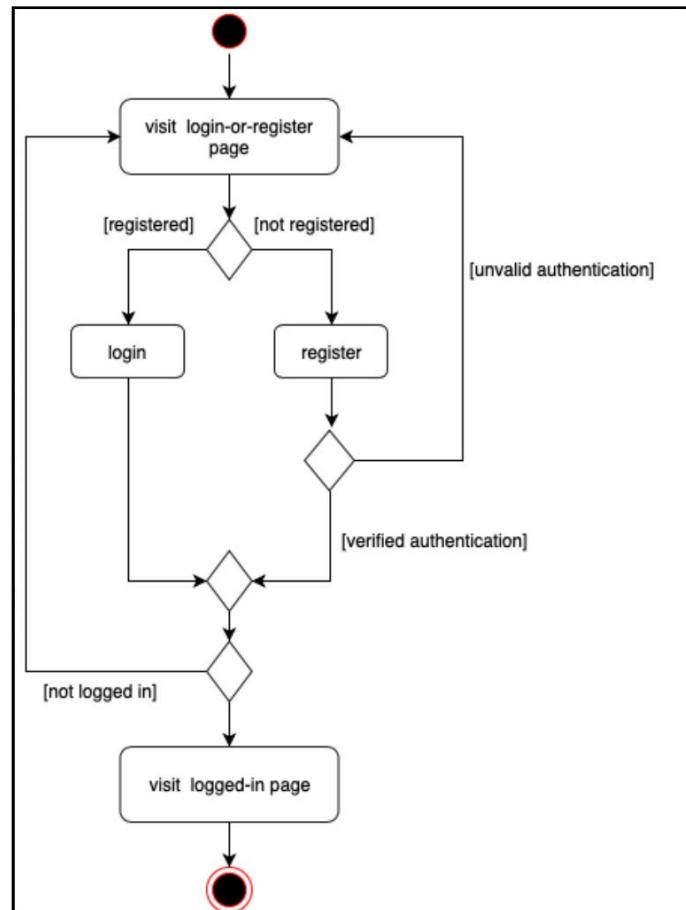
微碧智慧店面(無日期)。Features。檢自<https://weiby.tw/features>(民國107年11月25日)

摩斯漢堡(無日期)。APP 訂餐。檢<https://www.mos.com.tw/shop/mosapp.html> (民國107年11月25日)

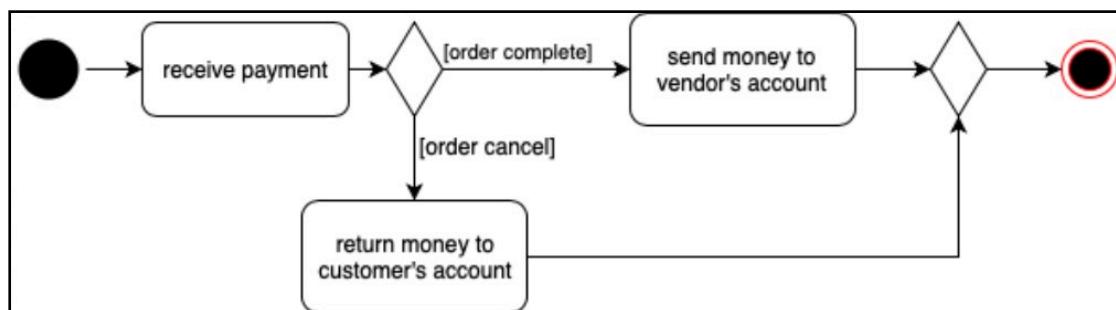
MongoDB and MySQL Compared.(n.d.).MongoDB and MySQL Compared. Retrieved from <https://www.mongodb.com/compare/mongodb-mysql>(Oct 30, 2018)

Wang, Y.-S., International Journal of Hospitality Management (2018),
<https://doi.org/10.1016/j.ijhm.2018.06.002>

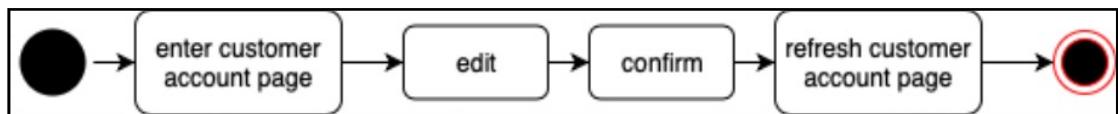
附錄



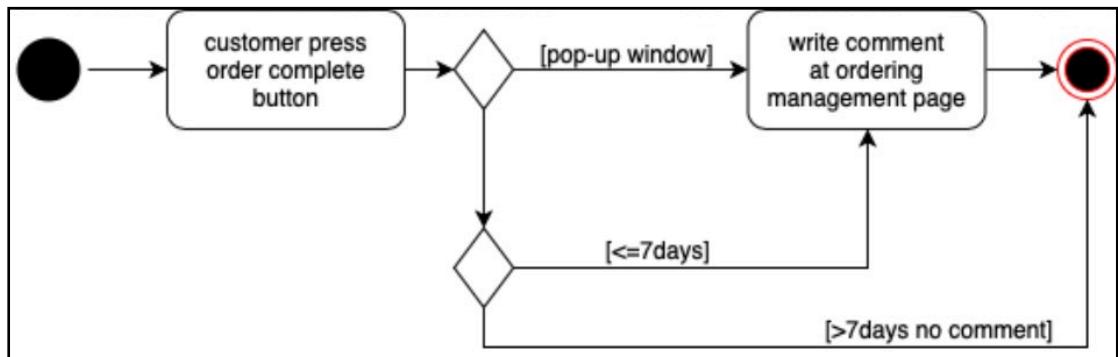
圖四十一 login_signup activity diagram



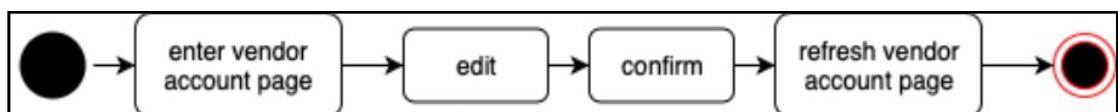
圖四十二 third party activity diagram



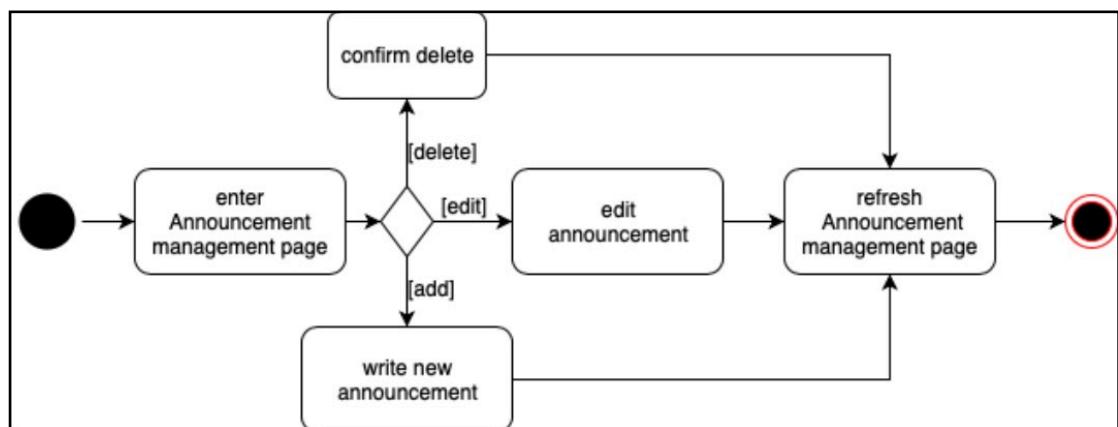
圖四十三 customer edit data activity diagram



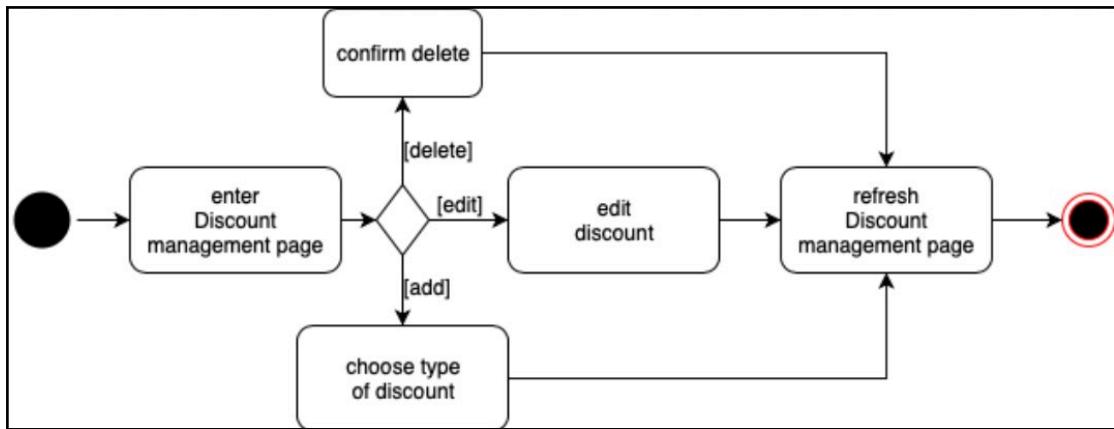
圖四十四 customer comment activity diagram



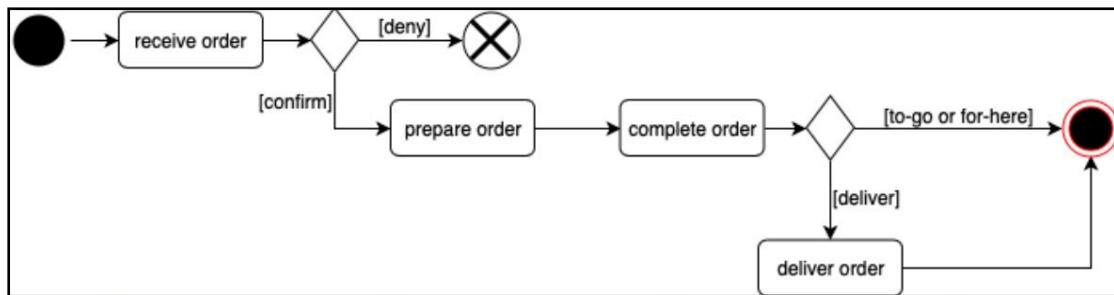
圖四十五 vendor edit data activity diagram



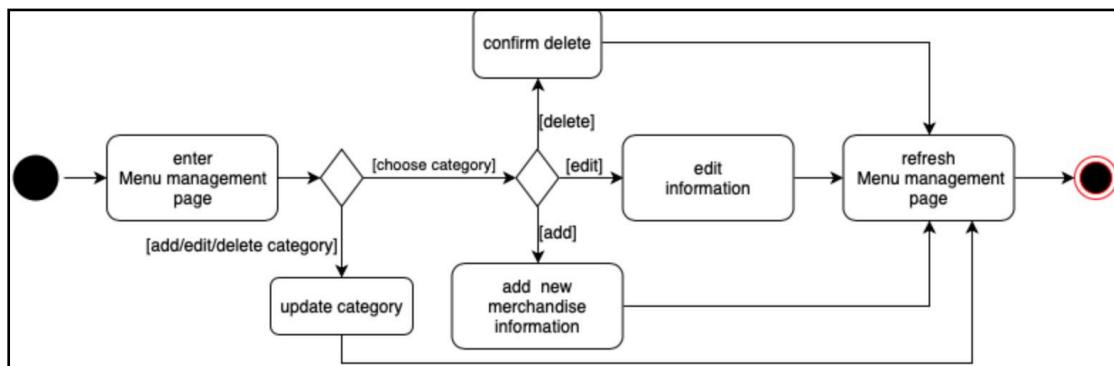
圖四十六 vendor edit announcement activity diagram



圖四十七 vendor edit discount activity diagram



圖四十八 vendor order management activity diagram



圖四十九 vendor update menu activity diagram