

Group7: Reservation and Leftover Food

# 餐餐不剩

統計二 陳柏翰

統計二 林哲宇

統計二 林承佑

地政四 胡詠琪

# 目錄

---

<b>一、背景與動機</b>	<b>1</b>
1. 背景與動機	1
2. ESG理念落實SDGs	1
• 減少食物浪費(SDGs 2.終結飢餓 & SDGs 3.健康與福祉)	1
• 提高餐廳運營效率(SDGs 8.就業與經濟成長)	1
• 有效利用資源(SDGs 12.責任消費與生產 )	2
<b>二、主題介紹與描述</b>	<b>3</b>
1. 主題介紹	3
2. 問題描述與需求分析	3
• 問題描述	3
• 需求分析	3
3. 系統分析與安排	5
<b>三、系統描述</b>	<b>6</b>
1. 架構(UML)	6
2. 功能	7
3. 概念描述(簡報三、資料表)	15
• 前端主架構	15
• 後端主架構	17
• 資料庫與資料表	17
• 各項概念描述	17
4. 特色(簡報四、簡報後面幾個特色)	21
• 結構式取餐代碼	21
• 使用者數量等於系統刷新速度	21
• 雙向奔赴的設計	23
<b>四、結論與未來展望</b>	<b>24</b>
1. 程式設計面向	24
2. 應用程式面向	24
<b>五、分工</b>	<b>26</b>
<b>六、其他相關資料</b>	<b>27</b>
1. Milestone1	27
2. Milestone2	27
3. Presentation	27
4. Demo Video	27
5. System operation manual	27

## 一、背景與動機

### 1. 背景與動機

我們發現在臉書「政大交流版」中，常會有許多演講、會議、活動等提供的餐盒或飲料有剩餘的情況，且通常僅在交流版中發文自由領取，沒有一個系統能夠直接且明確地得知剩食的狀況與種類。其次，依照過去領取或發送剩食的經驗，許多都會先私訊發文的同學先保留餐食，才有辦法取得；然而，當同時有許多活動有剩食狀況時，容易導致無法有效發完，甚至不知道要等待到何時，只能選擇丟棄食物的情形。

另外，基於剩食的預約餐時與整合的概念，我們亦想到與自身息息相關的中午買飯與上課的問題。在政大周遭的餐廳在中午，時常會發生買飯時間過久，排隊甚長的問題；同時，有許多的助教課或小考都會安排於中午時段，常導致我們無法準時參與或準時用餐。因此，我們便覺得可以將類似的功能結合於同一個系統中，將一般餐廳整合與加入訂餐系統，讓餐廳可以有效地提前準備餐食，亦增加銷售量外；學生也可以提前將餐食預約完成，靈活運用中午的時間參與助教課程，或找尋地方休息準備下午的課程。

因此，我們希望製作一個系統，整合剩食與訂餐的問題，來解決食物浪費，買飯與排隊時間過長的問題。期待將整合兩者功能，讓系統擁有更多使用者願意使用，亦期待透過整合的想法，讓使用者更直覺，且更專一地使用與接收餐食消息，不需要於交流版內翻找現在是否有剩食的狀況，以利提升與解決剩食發放的問題；同時，訂餐系統達成讓學生可以更加靈活運用中午的休息時間，加入助教課程，或休息與移動至下堂課的教室等。

### 2. ESG理念落實SDGs

ESG(環境、社會、治理)理念的落實，是實現可持續發展目標(SDGs)的重要手段。ESG理念強調企業在追求經濟效益的同時，必須考慮對環境的保護、社會的責任以及企業治理的完善。本文聚焦於如何通過減少餐廳食物浪費和提高餐廳效率來落實ESG理念，並推動可持續發展。

- 減少食物浪費(SDGs 2.終結飢餓 & SDGs 3.健康與福祉)

減少食物浪費是實現可持續發展目標中的一個重要方面，因為食物浪費不僅導致資源的浪費，還會增加環境負擔。此系統可以幫助餐廳通過加強供應鏈管理來確保食材的供應量與需求量匹配，避免過量訂購和食材過期。利用我們的數據分析工具，餐廳可以根據歷史數據和需求預測進行精準採購，從而大幅減少過剩食材，提高資源利用效率。並且剩食系統，幫助餐廳或學生團體將多餘的食物捐贈給有需要的人，這不僅能避免浪費，還能履行餐廳的社會責任。

- 提高餐廳運營效率(SDGs 8.就業與經濟成長)

提高餐廳運營效率有助於降低成本、提高顧客服務質量，並減少對環境的負面影響。本系統通過優化訂單管理、提升預訂效率和員工培訓來實現這一目標。訂單管理系統可精確記錄和追蹤訂單，避免出錯和浪費；預訂系統則根據客流量預測調整人員安排和備料量，減少等待時間和資源浪費；此外，定期對員工進行培訓，提高其工作效率和服務質量，促進整體運營效率的提升。這

些措施不僅提高了運營效率，還有助於實現可持續發展目標，體現了餐廳在環境和社會責任方面的承諾。

- **有效利用資源(SDGs 12.責任消費與生產 )**

在餐廳運營過程中，資源的高效利用不僅可以降低運營成本，還能減少對環境的影響，符合ESG理念中的環境保護要求。我們的系統通過能源管理、水資源管理和廢棄物管理來實現這一目標。採用節能設備和技術，減少能源消耗，降低碳足跡；優化用水管理，使用節水設備和技術，減少水資源浪費；實行廢棄物分類和回收，減少垃圾填埋和環境污染。這些措施不僅提升了資源利用效率，還體現了我們對環境保護和可持續發展的承諾。

通過減少食物浪費、提高運營效率和有效利用資源，餐廳可以在落實ESG理念的同時，實現可持續發展目標。這不僅有助於提升餐廳的經濟效益，還能增強其社會責任感和品牌形象，實現長遠發展。

## 二、主題介紹與描述

### 1. 主題介紹

基於系統中兩大主要概念，剩食與訂餐，我們以「餐餐不剩」作為系統與主題名稱。

關於「剩食」，我們期待透過系統的整合，甚至系統的通知發放，使用者可以即時得知目前剩食的狀況與項目，並且以簡單的操作介面，直覺方便地預約與解決剩食問題。而發放剩食的人，可以藉由系統設計與統計，立即得知目前的預約與剩餘狀況，以及即時扣除現場領取的數量，達到預約與現場領取的整合，提升剩食發放的效率，達成剩食皆發放完畢「不剩」的情況！

關於「訂餐」，我們期待整合政大周遭的餐廳加入，學生可以於開放預約的時間，對系統中的餐廳下單，讓餐廳可以依照取餐時間，有效地安排餐點製作的順序與時間，解決中午與課餘時間買飯的「盛」況空前的情況，希望可以每餐都讓學生可以節省更多的時間，有效安排時間參與更多課程外，讓時間不「剩」地同時，亦達成餐食的購買與用食完畢，

### 2. 問題描述與需求分析

- 問題描述：

- 剩食：

- 解決活動未發放玩的餐食
    - 建立剩食狀態、項目與數量(並整合於訂餐同介面)
    - 開放預約與即時取餐(直覺與有效使用)

- 訂餐：

- 結合政大周遭餐廳(提升平台實用度)
    - 預約訂餐與結構式編號

- 需求分析：

- 針對系統本身進行「SWOT」思考

S	<ul style="list-style-type: none"> <li>● 方便校內學生在課餘時間訂購午餐，解決用餐時間不足的問題。</li> <li>● 可使用學號作為識別碼，建立可靠的用戶資料庫，亦可做相關的分析。</li> <li>● 不收取額外手續費，降低使用門檻</li> </ul>
W	<ul style="list-style-type: none"> <li>● 系統同步更新的挑戰，可能造成商家和使用者資料不一致</li> <li>● 取餐以及現場購買的人可能會造成店家混亂</li> </ul>
O	<ul style="list-style-type: none"> <li>● 整合校門口附近餐廳，提高系統覆蓋率和使用人數</li> </ul>

	<ul style="list-style-type: none"> <li>與行動政大整合，簡化登入流程，提高使用率</li> </ul>
T	<ul style="list-style-type: none"> <li>現有餐廳已有一定銷售渠道，可能不願加入新系統</li> <li>餐廳可能已與其他訂餐系統(LINE結合的「你訂」)或外送平臺合作</li> </ul>

- 若系統實際公開運作，思索帶來的競爭優勢「五力分析」

新進入者的威脅	<p>進入難易程度『中』；</p> <ul style="list-style-type: none"> <li>現有店家可能對一個不成熟的系統接受度不高</li> <li>剩食本身就可以在交流版做處理，如果使用系統人數過低就沒有達到預期的正向效果</li> </ul>
替代品的威脅	<p>被其他產品替代程度『低』；</p> <ul style="list-style-type: none"> <li>雖然現今有很多相關的訂餐系統(LINE你訂)，但都是針對單一店家的設計，沒有全部的整合；或是外送平台的訂餐，但又會多收取服務費或對店家抽成。</li> </ul>
購買者(使用者) 議價能力	<p>議價能力『中』</p> <ul style="list-style-type: none"> <li>因為系統會盡可能收集與整合政大附近就近餐廳，對於學生可以有效並提早做訂餐是一個很大的吸引力；</li> <li>剩食方面，則可以有效解決負責處理的同學的等候或處理時間，也有效處理「真的」被浪費的食物</li> </ul>
供應商(店家) 議價能力	<p>議價能力『中』</p> <ul style="list-style-type: none"> <li>對店家而言或許可以提升出餐的效率，而且不需要多餘的手續費</li> <li>但沒有完善的系統資訊建立，如果有跑單的行為沒辦法有效處理，對店家而言是一種浪費與問題</li> </ul>
現存的競爭者	<p>現存競爭程度『低』</p> <ul style="list-style-type: none"> <li>現今的系統除了LINE的訂餐系統較為直覺且沒有太高的營運成本；</li> <li>外送平台會對店家與使用者都收取一定程度的費用；</li> <li>而最原本電話訂餐的模式對學生而言，在上課期間有些不便</li> </ul>

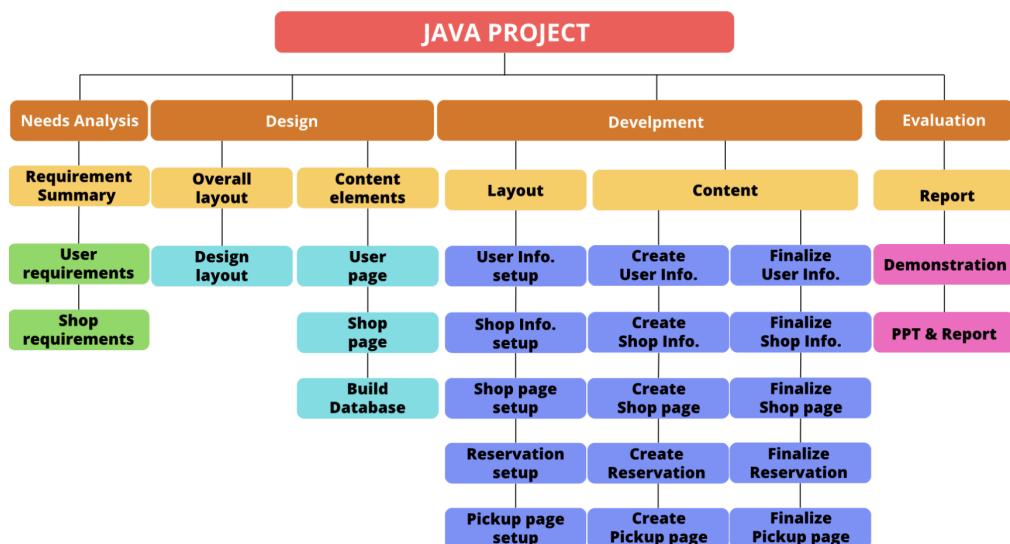
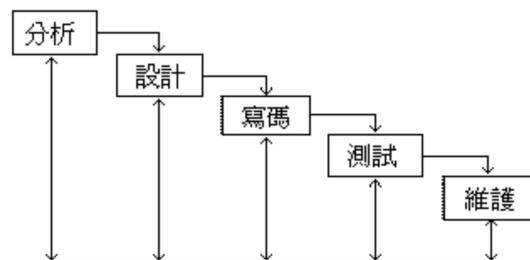
- 同學之間的調查

■ 現在交流版就可以有效解決剩食的問題，用這個系統會不會有

點多此一舉與沒有必要性？

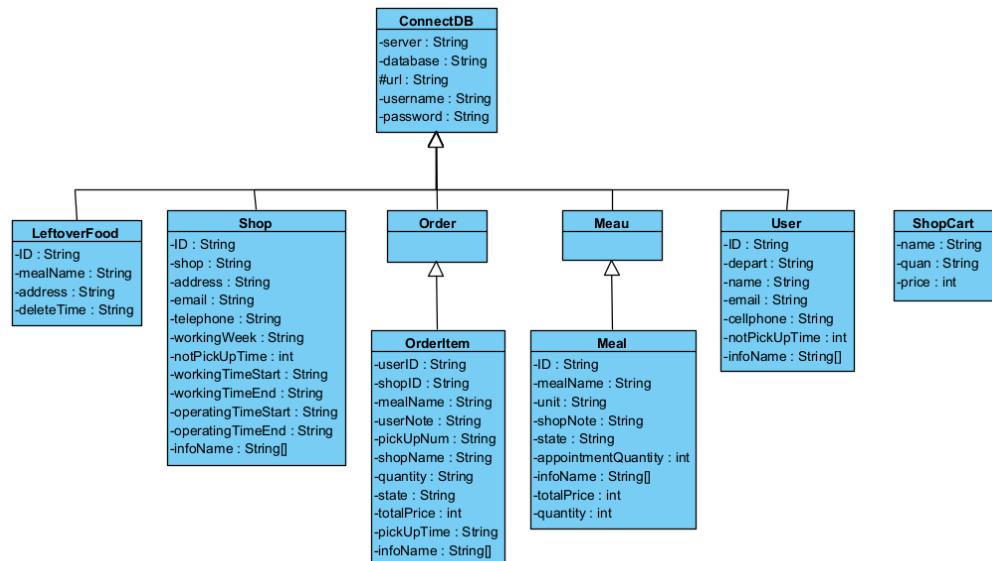
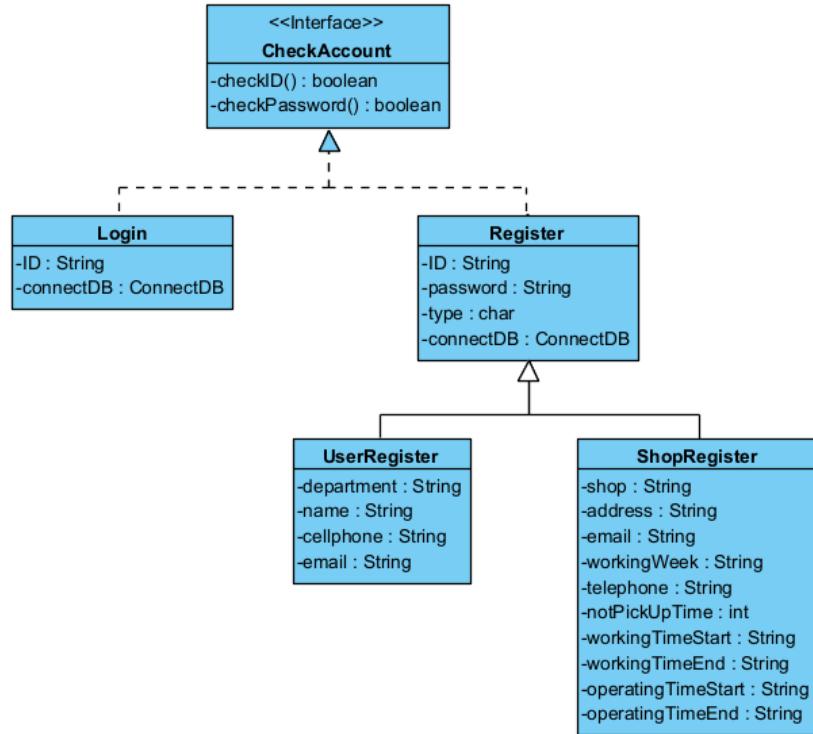
- 訂餐部分，如果店家可以有效地分配時間，且可以準時出餐，確實會是很方便的系統！
- 如果在尖峰時期（多個會議或演講）同時舉辦，會同步有多個剩食問題，這樣的設計或許可以有效解決和分配？
- 如果系統使用上能夠很直覺簡單，會更實用。

### 3. 系統分析與安排



### 三、系統描述

#### 1. 架構(UML)



## 2. 功能

CheckAccount << interface >>		
Return type	Method or Variable	description
<b>Abstract Methods</b>		
boolean	checkID	確認帳號
boolean	checkPassword	確認密碼

ConnectDB		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	server	資料庫相關資料
ConnectDB	database	
String	url	
String	username	
String	password	
<b>Methods</b>		
	ConnectDB ()	constructor
ArrayList<Object>	connectDB(String query, String condition)	取得資料表所有內容，並輸入相關條件(該資料名稱)
boolean	insertDeleteDB(String query)	執行插入或刪除操作的 SQL 查詢。
ArrayList<Object>	showResultSet(ResultSet resultSet, Object condition)	將結果集中的資料轉換為 ArrayList<Object>。
ResultSet	executeQuery(String query)	執行給定的 SQL 查詢並返回結果集。
ArrayList<ArrayList<Object>>	fetchAll(String query)	執行給定的 SQL 查詢並返回所有結果。

LeftoverFood		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID, mealName, address, deleteTime	剩食帳號, 剩食名稱, 剩食地點, 剩食刪除時間
int	appointmentQuantity	可預約數量
String	setUpTime	建立時間
String[]	infoName	所有剩食資料表的各項資料名稱
<b>Methods</b>		
	LeftoverFood()	constructer(創建新帳號時使用)
	LeftoverFood(String ID)	constructer(已經有帳號, 取得資料使用)
boolean	checkIDExist()	確認帳號是否存在於資料表中
boolean	initializeInfo(String mealName, int appointmentQuantity, String address)	初始化所有資料內容與設定
boolean	deleteID()	刪除該剩食帳號、菜單中符合剩食ID的資料、訂單中符合剩食ID的資料
boolean	setSomeInfo(String infoName, Object newInfo)	設定某項資料, 並輸入該資料於資料表的名稱
void	refreshInfo()	刷新所有資料, 並進行未取餐次數的紀錄計算。
boolean	checkSetUpTime()	確認建立的時間是否超過兩小時, 超過則呼叫 deleteID()
void	checkAllDBTime()	確認剩食資料表中所有項目的狀況, 呼叫 checkSetUpTime()作為判斷
	get...() * 5	

Login
-------

Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID	帳號
ConnectDB	connectDB	
<b>Methods</b>		
	Login()	constructer
boolean	checkID()	確認帳號是否相同
boolean	checkPassword	確認密碼是否相同
char	getType	取得種類
boolean	setPassword (String orginalPassword, String newPassword)	設定新密碼。

Meal		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID, mealName, unit, shopNote, state	商家(剩食)帳號, 餐點名稱, 單位, 商家註記, 餐點狀態
int	appointmentQuantity, unitPrice	可預約數量, 單位價格
String[]	infoName	所有菜單資料表的各項資料名稱
int	totalPrice, quantity	總金額, 總數量
<b>Methods</b>		
	Meal(String ID, String mealName, int appointmentQuantity, String unit, int unitPrice, String shopNote, String state)	Constructer(給刷新所有菜單時使用)
	Meal(String ID, String mealName, int appointmentQuantity, String unit, int unitPrice, String shopNote)	Constructer(給商家建立餐點時使用)

	Meal(String ID, String mealName, int appointmentQuantity)	Constructor (給剩食建立餐點時使用)
boolean	setSomeInfo(String infoName, Object newInfo)	設定某項資料，並輸入該項資料在資料表中的名稱
boolean	deleteMeal(String ID)	刪除商家帳號欄位中，符合輸入的帳號之所有菜單資料。(給剩食刪除時使用)
boolean	deleteMeal(String ID, String mealName)	刪除某商家帳號中的某個特定餐點(餐點名稱不會重複)。
	get...() * 9	
	set(...) * 2	

Menu		
Return type	Method or Variable	description
<b>Methods</b>		
	Menu()	constructor
ArrayList<Meal>	refreshInfo(String shopID)	刷新所有該商家的餐點資料，並且儲存為meal型態
boolean	insertMenuDB(String ID, String mealName, int appointmentQuantity, String unit, int unitPrice, String shopNote)	用於輸入菜單資料表中，給商家新增菜單時使用。
boolean	insertMenuDB(String ID, String mealName, int appointmentQuantity)	用於輸入菜單資料表中，給剩食新增菜單時使用。

Order		
Return type	Method or Variable	description
<b>Methods</b>		
	Order()	constructor
ArrayList<OrderItem>	refreshInfo(String shopID)	刷新所有該“商家帳號”的訂單資料，並儲存為OrderItem的型態。

ArrayList<OrderItem>	userRefreshInfo(String user)	刷新所有該“一般使用者帳號”的訂單資料，並儲存為OrderItem的型態。
ArrayList<OrderItem>	checkState(ArrayList<OrderItem> all)	確認所有訂單的狀態，如果今日的日期與取餐代碼的前四碼的日期不同，且狀態非已完成，則增加使用者的未取餐次數。
boolean	insertOrderDB(String userID, String shopID, String pickUpNum, String mealName, String quantity, int totalPrice, String note, String pickUpTime)	用於輸入訂單資料表中，給新增商家訂單時使用。
boolean	insertOrderDB(String userID, String shopID, String pickUpNum, String mealName, String quantity, String pickUpTime)	用於輸入訂單資料表中，給新增剩食訂單時使用。
String	generatePickUpNum(String shopID)	結構式生成取餐編號

OrderItem		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	userID, shopID, mealName, userNote, pickUpNum, shopName, quantity	使用者帳號，商家帳號，餐點名稱，使用者備註，取餐編號，商店名稱，數量
String	state	餐點狀態
String[]	infoName	所有訂單資料表的各項資料名稱
int	totalPrice	總金額
String	pickUpTime	取餐時間
<b>Methods</b>		
	OrderItem(String userID, String shopID, String pickUpNum, String mealName, String quantity,	Constructer(給刷新資料)

	int totalPrice, String userNote, String pickUpTime, String state)	
	OrderItem(String userID, String shopID, String mealName, String quantity, int totalPrice, String note, String pickUpTime)	Constructer(給商家)
	OrderItem(String userID, String shopID, String mealName, String quantity, String pickUpTime)	Constructer(給剩食)
boolean	deleteOrder()	刪除該筆訂單資料
boolean	setSomeInfo(String infoName, Object newInfo)	設定某項資料，並輸入該項資料在資料表中的名稱
String	generatePickUpNum(String shopID)	結構式生成取餐編號
void	insertOrderToDB()	輸入所有訂單資料到訂單的資料表中(一般)
void	insertOrderToDB(boolean isSurplus)	輸入所有訂單資料到訂單的資料表中(給剩食)
	get...() * 10	

Register		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID	帳號
String	password	密碼
char	type	種類
ConnectDB	connectDB	
<b>Methods</b>		
	Register(String ID, String password, char type)	constructer
boolean	checkID( String ID)	確認帳號是否重複
boolean	checkPassword (String password)	確認與儲存註冊密碼
String	getID()	取得帳號

char	getType()	取得種類(按下不同按鈕後存放不同的種類)
boolean	insertAccountDB()	將帳號、密碼與種類存入帳號的資料表中。

Shop		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID, shop, address, email, telephone, workingWeek	商家帳號、商家名稱、地址、信箱、手機或市話、營業星期
int	notPickUpTime	未取餐次數
String	workingTimeStart, workingTimeEnd	營業時間起始, 營業時間結束
String	operatingTimeStart, operatingTimeEnd	開放預約時間始, 開放預約時間末
String[]	infoName	所有商家資料表的各項資料名稱
<b>Methods</b>		
	Shop(String ID)	constructor
void	refreshInfo()	刷新與取得所有商家資料
boolean	setSomeInfo(String infoName, Object newInfo)	設定某項資料, 並輸入該項資料在資料表中的名稱
	get...() * 11	

ShopCart		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	name	所有餐點名稱(,隔開)
String	quan	各個餐點數量(,隔開)
int	price	總金額
<b>Methods</b>		
	ShopCart(String n, String q, int p)	constructor
String	getName()	取得購物車中的餐點名稱。
String	getQuan()	取得各個餐點數量

int	getPrice()	取得總金額
-----	------------	-------

ShopRegister		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	shop, address, email, telephone, workingWeek	商家名稱、地址、信箱、市話、營業星期
String	workingTimeStart, workingTimeEnd	營業時間起始, 營業時間結束
String	operatingTimeStart, operatingTimeEnd	開放預約時間始, 開放預約時間末
<b>Methods</b>		
	ShopRegister(String ID, String password)	constructor
void	setInfo(String shop, String address, String email, String workingWeek, String telephone, int notPickUpTime, String workingTimeStart, String workingTimeEnd, String operatingTimeStart, String operatingTimeEnd)	設定所有基本資料
boolean	insertShopInfoDB()	將商家各項資訊存入商家的資料表中。

User		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	ID, depart, name, email, cellphone	一般人帳號, 系級, 姓名, 信箱, 手機
int	notPickUpTime	未取餐次數
String[]	infoName	所有一般人資料表的各項資料名稱
<b>Methods</b>		
	User(String ID)	constructor

ArrayList<Shop>	getAllShop()	取得所有可顯示的商家名單，依照未取餐次數進行判斷，大於者不行。
ArrayList<LeftoverFood>	getAllLeftoverFood()	取得所有目前剩食存在的名單，並針對各項剩食進行建立時間的判斷(判斷是否要刪除)
void	refreshInfo()	刷新與取得所有一般人帳號的資訊
boolean	setSomeInfo(String infoName, Object newInfo)	設定某項資料，並輸入該項資料在資料表中的名稱
	get...() * 6	

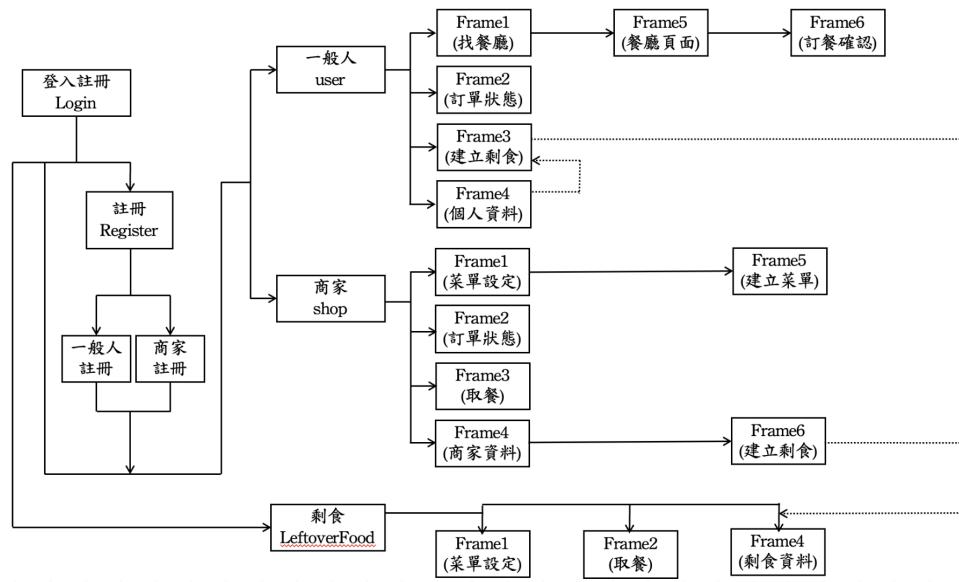
UserRegister		
Return type	Method or Variable	description
<b>Instance variable</b>		
String	department	系級
String	name	姓名
String	cellphone	手機
String	email	信箱
<b>Methods</b>		
	UserRegister(String ID, String password)	constructor
void	setInfo(String dp, String name, String phone, String mail)	設定所有基本資料
boolean	insertUserInfoDB()	將一般人各項資訊存入一般人資訊的資料表中

### 3. 概念描述(簡報三、資料表)

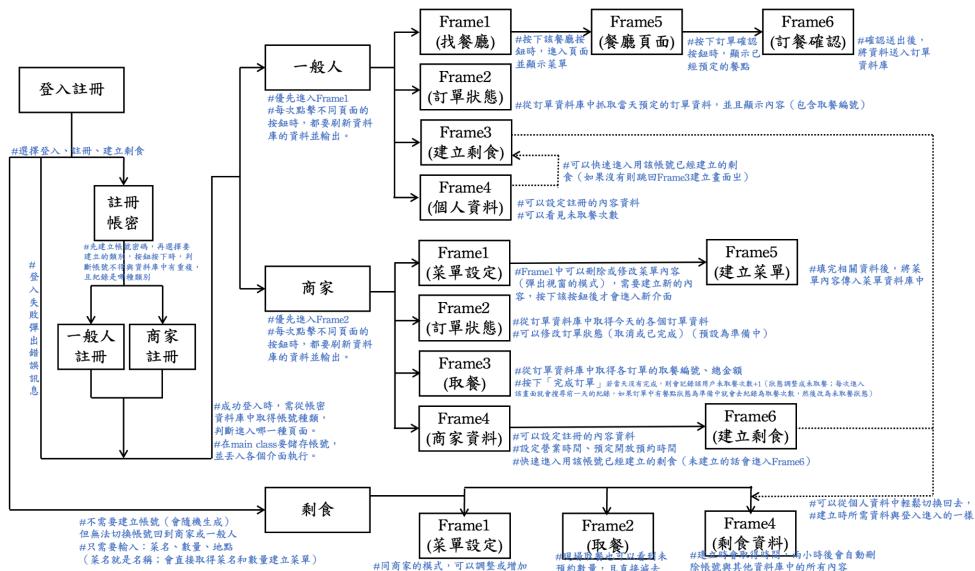
- 前端主架構：

下圖中，顯示依使用者的不同操作，將會切換至不同的頁面與顯示不同的功

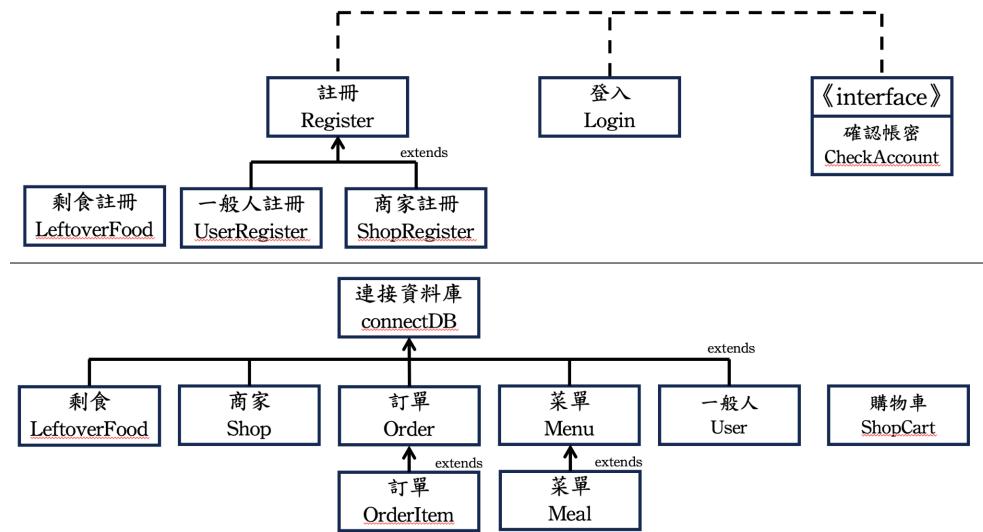
能圖(詳細操作方式與內容請見「系統使用手冊」)。



而下圖中的藍字，則是註記與顯示各個頁面的切換，主要經由什麼樣的過程與判斷式進行。



- 後端主架構:



- 資料庫與資料表:

透過帳號作為大部分資料表的primary key, 而訂單資料表忠則是以取餐編號(結構式不重複的取餐編號)作為primary key, 並於後續說明該編號的結構為何。

帳號 Account							
帳號ID ID	密碼 Password	種類 Type					
商家基本資訊 ShopInformation							
商家帳號ID ID	店名 Shop	地址 Address	電話 Telephone	信箱 Email	營業時間 WorkingTime	開放預約時間 OpeningHours	未取餐次數小於多少 NotPickUpTimes
一般人基本資訊 UserInformation							
一般人帳號ID ID	系所 Department	姓名 Name	手機 Cellphone	信箱 Email	未取餐次數 NotPickUpTime		
訂單 Order							
一般人 ID	商家. 剩食 ID	取餐編號 PickUpNumber	餐點名稱 MealName	數量 Quantity	價格 TotalPrice (該項總額)	備註 Note (一般人)	取餐時間 PickUpTime
菜單 Menu							
商家或剩食 帳號ID	菜名 MealName	開放預約數量 AppointmentNumber	量詞 Unit	每單位價格 UnitPrice	備註 (商家) ShopNote	品項狀態 State	
剩食 LeftoverFood							
剩食帳號 ID	菜名 MealName (名稱)	數量 Quantiy	地點 Address	建立時間 SetupTime			

- 各項概念描述

(僅針對較為複雜, 或難以理解的部分作解釋, 其餘基本的除錯(防呆)與基本判斷式等, 皆包含於本專案之中!)

- **inheritance & interface**
- 描述:

- 純粹註冊功能的類別為父項，商家或一般人用戶，由於需要輸入不同的內容，且都為註冊功能下的細分，所以為子項，形成第一個繼承模式
- 由於所有的資料都來自各個資料表中，每筆資料都需要透過連接資料庫，所以將所有資料表繼承於連接資料庫底下。
- 而菜單與訂單再細分為第二層，是將父項列為指定某商家或一般人的菜單與訂單內容，而子項(菜單細項或訂單細項)則是將每筆資料的內容做細分與取得，建立成一個物件，方便取用。
- interface放置於登入與註冊的兩者環節，是基於我們註冊的順序為，先註冊帳號密碼，按下不同的使用者類別按鍵，進入不同的資料註冊頁面；因此，不論是登入或註冊，皆需要做帳號密碼的確認，前者為找到相同的帳號，後者則為找到完全不相同的帳號。

```

UserRegister.java
1 package backEnd;
2
3 public class UserRegister extends Register {
4
5     private String department;
6     private String name;
7     private String cellphone;
8     private String email;
9
10    public UserRegister (String ID, String password) {
11        super(ID, password, 'U');
12    }
13
14    public void setInfo (String dp, String name, String pho
15        this.department = dp;
16        this.name = name;
17        this.cellphone = phone;
18        this.email = mail;
19    }
20
21    public boolean insertUserInfoDB() {
22
23
24
25
26
27
28
29
2
CheckAccount.java
1 package backEnd;
2
3 public interface CheckAccount {
4
5     boolean check (String ID);
6
7     boolean checkPassword (String password);
8
9 }
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
5
Register.java
1 package backEnd;
2
3 import java.util.ArrayList;
4
5 public class Register implements CheckAccount {
6
7     private String ID;
8     private String password;
9     private char type;
10
11     protected ConnectDB connectDB;
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
2

```

- **connect database**

- **描述：**

- 所有資料庫藉由串連的程式碼進行連結與取得，並在連結資料庫的類別中，建立多種適用於各個情況的方式，來供子項取得與使用。

```

import java.sql.*;
public class ConnectDB {
    private String server = "jdbc:mysql://140.119.19.73:3315/";
    private String database = "111304019"; // change to your own database
    protected String url = server + database + "?useSSL=false";
    private String username = "111304019"; // change to your own username
    private String password = "bovwk"; // change to your own password
    public ConnectDB() {}
    public ArrayList<Object> connectDB(String query, String condition) {
        ArrayList<Object> result = new ArrayList<Object>();
        try (Connection conn = DriverManager.getConnection(url, username, password)) {
            Statement stat = conn.createStatement();
            ResultSet rs = stat.executeQuery(query);
            while (rs.next()) {
                result.add(rs.getObject("Column1"));
            }
        }
    }
}
public class User extends ConnectDB {
    private String ID;
    private String name;
    private String cellphone;
    private String email;
    private int notPickUpTime;
    private String[] infoName = { "Department", "Name", "Cellphone", "Email" };
    public User(String ID) {
        this.ID = ID;
    }
    public ArrayList<Shop> getAllShop(){
        ArrayList<Shop> allShop = new ArrayList<Shop>();
        String query = "SELECT ID FROM `ShopInformation`";
        System.out.println("Executing query: " + query);
        ArrayList<Object> allID = super.connectDB(query, "ID");
        for (Object id : allID) {
            Shop shop = new Shop();
            shop.ID = id.toString();
            allShop.add(shop);
        }
    }
}

```

- **overloading (以meal為例)**

- **描述：**

- 第一個meal，是提供商家或一般人，在刷新資料表中菜單資訊時，需要取得所有的資訊並建立一個meal的物件。
- 第二個meal，是提供商家在創建菜單時使用，由於預設為販賣中，所以不需要輸入狀態欄為，因此輸入的參數數量不同。
- 第三個meal，是提供剩食商家在創建菜單(帳號)時使用，由於剩食不需要單價等詳細資訊，且在兩小時內一律為販賣中，不需要更動狀態，因此輸入的參數更少亦不同。
- 因此，由於輸入參數的數量不同，但都坐同一件事情(運用在不同情境)，所以形成overloading。

```

    // 刪新資料使用
    public Meal(String ID, String mealName, int appointmentQuantity, String state) {
        this.ID = ID;
        this.mealName = mealName;
        this.appointmentQuantity = appointmentQuantity;
        this.unit = unit;
        this.unitPrice = unitPrice;
        this.shopNote = shopNote;
        this.state = state;
    }

    // 商家新增菜單
    public Meal(String ID, String mealName, int appointmentQuantity, String state) {
        this.ID = ID;
        this.mealName = mealName;
        this.appointmentQuantity = appointmentQuantity;
        this.unit = unit;
        this.unitPrice = unitPrice;
        this.shopNote = shopNote;

        super.insertMenuDB(ID, mealName, appointmentQuantity, unit, unitPrice, shopNote, state);
    }

    // 約食新增菜單
    public Meal(String ID, String mealName, int appointmentQuantity) {
        this.ID = ID;
        this.mealName = mealName;
        this.appointmentQuantity = appointmentQuantity;
        this.unit = "份";
    }

```

- **object & down casting & ArrayList**

- **描述:**

- 由於每次要取得的資料和內容非常多，所以我們嘗試減少重複打query的內容(SQL語法)；因此以object先作為取得所有資料的型態，並先建立好array固定取得資料的順序與內容(各個資料的名稱)，再將所有取得的資料先存入arraylist，再依序做downcasting(因為已經固定好取得的順序，因此可以確定arraylist中每個位置會需要做哪種型態的casting)，讓所有資料存在它應該存入的型態之中。
- 另外，希望前端只需要呼叫各個類別中refresh系列的method一次，就可以將所有資料存到該物件當中，不需要一個一個進行呼叫與設定。

```

public ArrayList<OrderItem> refreshInfo(String shopID) {
    ArrayList<OrderItem> allOrder = new ArrayList<>();

    try {
        String query = "SELECT * FROM Orders WHERE ShopID = '" + s
        System.out.println("Executing query: " + query); // 输出查询语句
        ArrayList<ArrayList<Object>> result = super.fetchAll(query);

        for (ArrayList<Object> row : result) {
            String userID = (String) row.get(0);
            String shopID2 = (String) row.get(1);
            String pickUpNum = String.valueOf(row.get(2));
            String mealName = (String) row.get(3);
            String quantity = (String) row.get(4);
            int totalPrice = (int) row.get(5);
            String userNote = (String) row.get(6);
            String pickUpTime = (String) row.get(7);
            String state = (String) row.get(8);

            OrderItem order = new OrderItem("LF", userID, shopID2,
                pickUpTime, state);
            allOrder.add(order);
        }
    }
}

```

- **ActionListner & Event & Anonymous class**

- **描述:**

- 許多按鍵按下接收 (ActionListner) 後的Event, 具有明確目的且專一的性值, 因此建立Anonymous class, 將該事件放入裡頭, 以利程式碼的閱讀。

```

MealTable.setCellFactory(col -> new TableCell<Meal, Void>() {
    private final Label quantityLabel = new Label();
    private final Button incrementButton = new Button("+");
    private final Button decrementButton = new Button("-");
    private final HBox hbox = new HBox(5, decrementButton, quantityLabel);

    {
        incrementButton.setOnAction(event -> {
            int quantity = Integer.parseInt(quantityLabel.getText());
            quantity++;
            quantityLabel.setText(String.valueOf(quantity));
            updateMealQuantity(getIndex(), quantity);
        });
        decrementButton.setOnAction(event -> {
            int quantity = Integer.parseInt(quantityLabel.getText());
            if (quantity > 0) {
                quantity--;
                quantityLabel.setText(String.valueOf(quantity));
                updateMealQuantity(getIndex(), quantity);
            }
        });
        hbox.setAlignment(Pos.CENTER);
    }

    @Override
    protected void updateItem(Void item, boolean empty) {
        super.updateItem(item, empty);
        if (empty) {
            setGraphic(null);
        } else {
            Meal meal = getTableView().getItems().get(getIndex());
            quantityLabel.setText(String.valueOf(meal.getQuantity()));
            setGraphic(hbox);
        }
    }

    private void updateMealQuantity(int index, int quantity) {
        Meal meal = getTableView().getItems().get(index);
        meal.setQuantity(quantity);
        meal.setTotalPrice(meal.getUnitPrice() * quantity);
        tableView.refresh();
    }
});

```

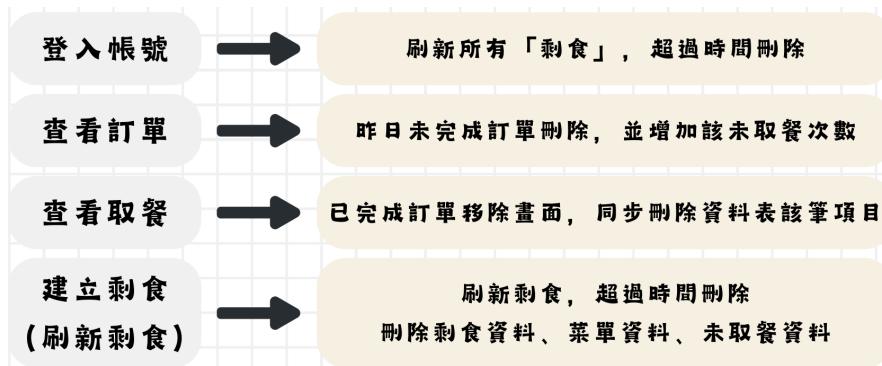
#### 4. 特色(簡報四、簡報後面幾個特色)

- 結構式取餐代碼

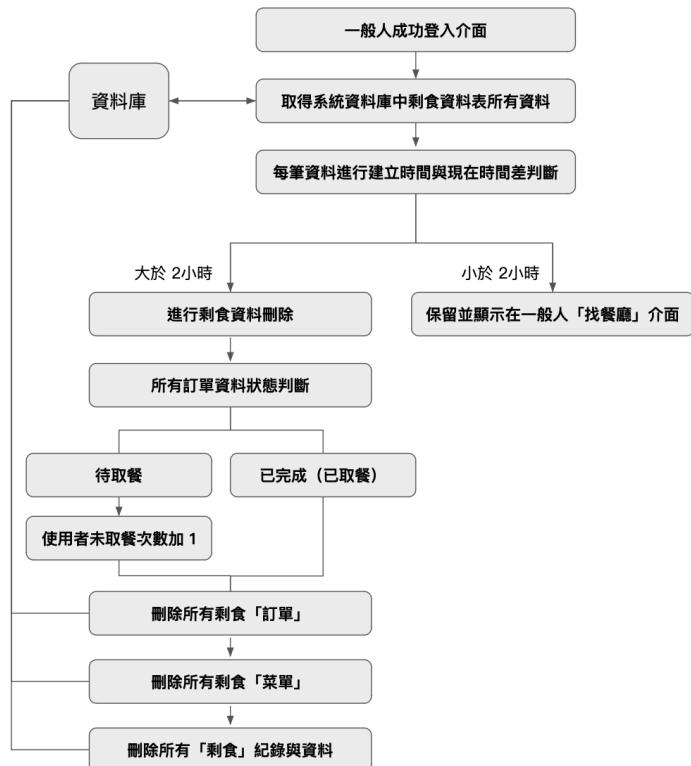


- 前四碼為當天日期，由於日期數字於當年度中，只會隨之增長，因此可以藉由介面的設計，讓訂單編號由大到小排列，更加方便查看訂單順序。
- 中間三碼屬於當天訂單數量。原訂為各個店家各自獨立計算，但為減輕系統與資料庫連線的負擔，與耗時過久問題；目前暫定為整個系統建立的訂單數量。否則將提供商家觀看當日使用本系統預約本餐廳的訂單數量。
- 最後隨機生成兩位數，避免取餐編號內容容易猜測，或使用者報錯號碼，導致餐點發送錯誤的情形。

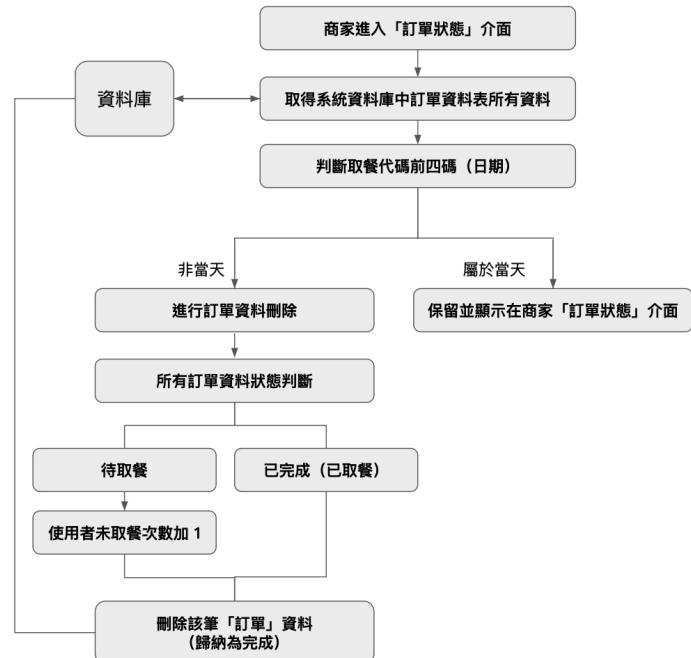
- 使用者數量等於系統刷新速度



- 剩食帳號自動刪除機制：

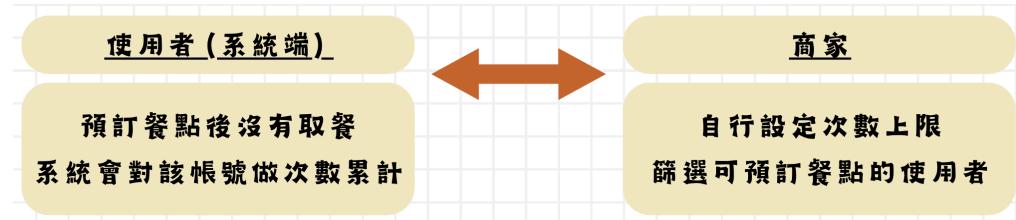


- 刷新菜單與歸納為完成：



- 通常刪除的動作會發生在，商家當天第一次進入訂單狀態的時候。
- 由於訂單做刪除的動作時，會直接從資料表中，刪除非當天的訂單記錄，因此不會有重複計算未取餐次數，或重複刪除的問題。

- 當日已完成訂單仍會顯示在訂單狀態介面，但會消失在取餐介面。
- 雙向奔赴的設計



- 目的: 篩選使用系統的用戶，避免慣性不取餐的用戶訂餐；亦降低商家接單的風險。
- 使用者未取餐次數不得自己設定增加或減少。；商家可以訂定自己的店家需要小於多少未取餐次數的紀錄，才可以取餐。
- 若商家刻意將未取餐次數降至最低，會造成較少數使用者可看見該餐廳；反之，若刻意將次數調高，則會使許多慣性不取餐的用戶下單，以致餐廳需負擔棄單成本。

## 四、結論與未來展望

### 1. 程式設計面向

- 特點：
  - 大量使用 Class 與 polymorphism 概念  
→ 提升程式碼的重複使用率
  - 使用 FXML (javaFX) 結合 controller  
→ 平台畫面整齊統一, ActionListener 清楚明白
- 改善：
  - 程式碼整體複雜過高, 不夠清晰易修正與使用  
→ 善用註解功能以及 javadoc 的格式, 讓不同的在修正或串接時更容易
  - 後端開發缺乏系統性建立, 部分 method 沒有達成 reuse 的理念  
→ 製作前專案的組員內應該先做草稿的溝通, 與假想架構的建立, 避免前後端串接, 或合作完成時有閒置或沒有意義的 method
  - 介面連結發現前後思索有斷層  
→ 前後端串接的經驗不足; 物件導向的架構與內容不夠明確清楚, 不立於合作完成。
  - 目前學到的方法有限, 學過的不夠了解使用方式  
→ 目前學到不少方法與概念, 但學過的不夠精熟, 或不知如何應用在專題之中, 導致許多方法都使用比較原始, 相對浪費效能的方式進行。
- 整題而言：
  - 從專案中可以更明確知道整學年的課程內容, 如何應用到實際當中的系統之中, 同時, 更加凸顯平常的練習和上課的思考訓練相當重要, 實際將物件導向的概念運用時, 才能夠確實地達成重複使用的目標, 亦讓程式碼更有效率發會更高的效能。

### 2. 應用程式面向

- 特點：
  - 有效整合同時大量剩食問題
  - 增加訂餐功能提升平台使用率
  - 雙向奔赴設計, 提升平台可信度
  - 免使用服務費且介面使用簡潔直覺
  - 系統更加完善後, 可直接實際運用, 帶給政大生更多便利性
- 改善：
  - 大量依靠連結資料庫, 使用速度切換較為緩慢
  - 部分使用者體驗不夠完善, 有些操作不夠直覺。
  - 平台獨立專一性質, 可能無法替代現有交流版。

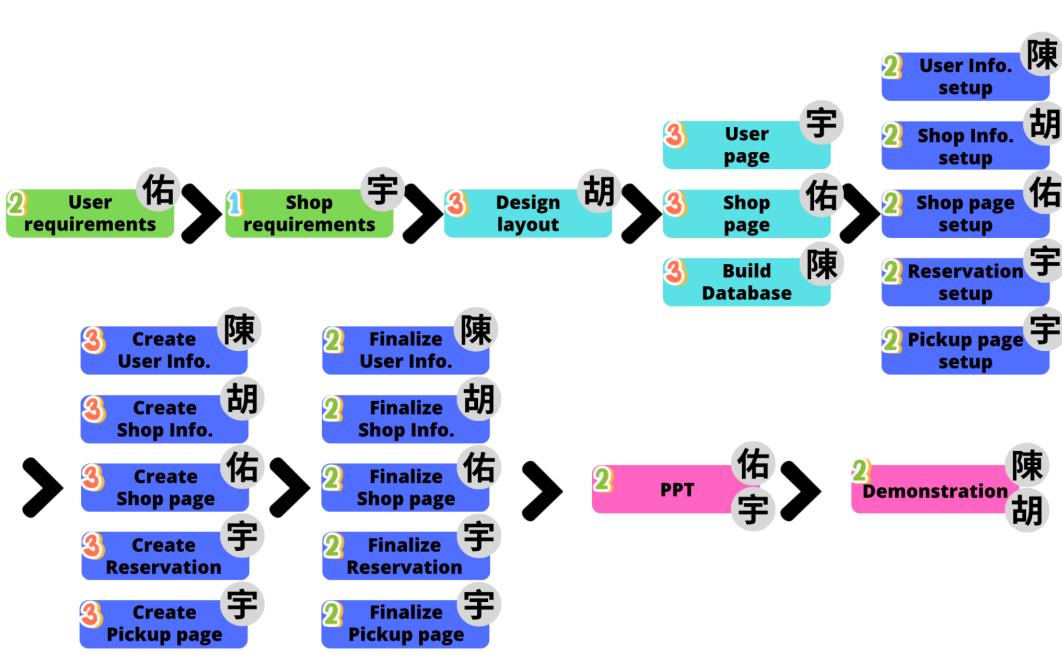
- 整體而言：

- 由於本專案只專注於訂餐和剩食兩大功能，若能結合現有其他相關的平台，如行動政大、政大通。讓使用者不需要分散或特意下載本系統時，系統的價值與實用度才會提升，以利更多使用者使用，逐漸替代與改變交流版的狀況與問題。
- 更新系統功能與修正平台架構；我們發現使用者應使用如現在所製作的手機介面的模式，而商家則可以以電腦或平板的格式進行設計，並且加入統計與圖表的分析，讓商家可以更明確知道各項餐點與使用平台的狀況。
- 調查商家意見與意願。雖然假想中的情境都相當正面，但實際運用面應主要考量的為，商家的願意的配合程度。或許商家其實會有其他考量或需求，我們應當朝此方向設計與添加，以利於提升系統與平台的吸引力，增加使用者的參與。

## 五、分工

組員		主要負責工作
111304007	陳柏翰	撰寫後端程式碼 全部程式碼檢測和除錯
111304033	林哲宇	前端畫面架構 Presentation 簡報和Demo影片製作
111304019	林承佑	前後端串接 前端畫面草圖和架構
109207412	胡詠琪	Presentation 簡報和Demo影片製作 提案思考與 Report 報告製作

註: 文字與簡報內容大多為共同製作。



## 六、其他相關資料

1. [Milestone1](#)
2. [Milestone2](#)
3. [Presentation](#)
4. [Demo Video](#)
5. [System operation manual](#)