



午餐系統及分析

壹、研究動機

之前在學校訂餐的時候，都只能用紙本點餐單來點餐，代訂常常把紙本單弄丟，而且紙本單必須手工計算數量、金額，十分不便，紙本單是不完全記名制，只能知道是哪一班點的，常常會有人忘記自己點了什麼餐，直接隨便亂拿一份餐，我們認為應該要有一個比紙本單更優秀的解決辦法，便開始著手製作午餐系統了。

午餐系統在板橋高中推行成功後，我們了解到廠商常常會備料過剩，於是我們打算建立一個模型，可以給廠商做為明天會有多少份餐點的依據。

貳、研究目的

本研究旨在於製作一套能夠取代紙本點餐單的系統，並且設計一個適當的數學模型供廠商參考明天該準備多少份餐點。

系統必須要有良好的穩定性，如果系統崩潰了，那大家今天就沒飯吃了；系統與金錢相關，所以也必須要有非常良好的安全性；可能會有上千個人在一節下課內送出點單，所以系統必須要有良好的效能；系統必須要簡單易懂，符合使用者直觀原則，讓使用者能夠輕鬆上手。

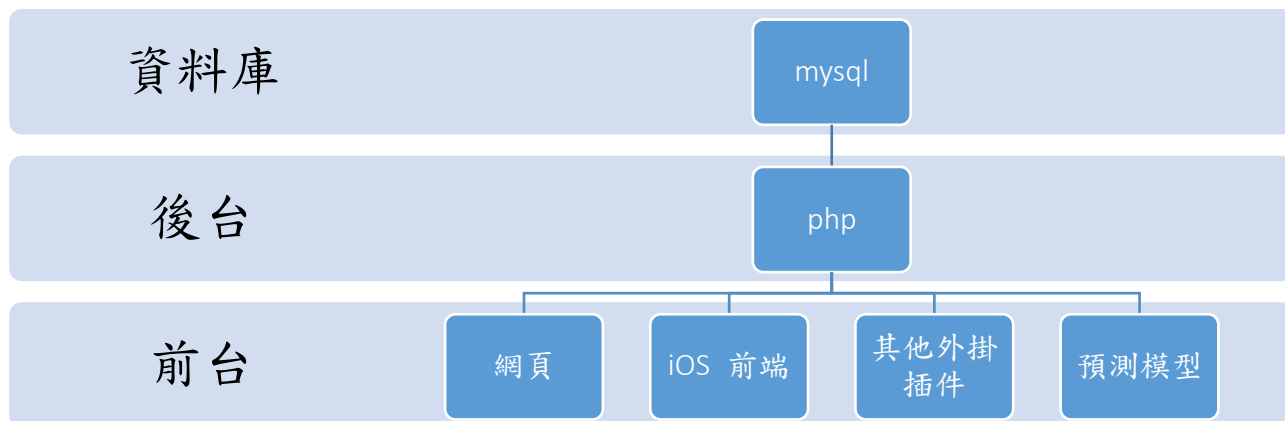
原本廠商只能估計明天要準備多少餐，不過有了數學模型就能夠更準確的預測會有多少份餐，本研究將介紹如何建立及應用數學模型。

參、實驗器材及設備

器材	用途
<i>Firebase</i>	<i>iOS App</i> 分析用的 <i>api</i>
<i>Vscode IDE</i>	製作午餐系統的 <i>IDE</i>
<i>Mysql</i>	午餐系統資料庫
<i>Php</i>	午餐系統後端
<i>Chrome</i>	網頁前端的測試環境
伺服器主機	午餐系統伺服器
<i>Noip Dynamic dns hostname</i>	午餐系統網域名稱
板橋高中	午餐系統的餐點購買對象
廠商	午餐系統的餐點提供對象
印表機	列印紙本單據
<i>Excel</i>	輸出媒體
<i>Visual studio</i>	製作外掛插件的 <i>IDE</i>
<i>Windows</i> 作業系統安裝光碟	伺服器的作業系統
<i>Mac</i>	製作 <i>iOS App</i> 的開發環境
<i>Iphone</i>	<i>iOS App</i> 的測試環境
<i>Xcode</i>	<i>iOS App</i> 的 <i>IDE</i>
<i>Android</i> 手機	網頁前端的測試環境
<i>Windows</i> 電腦	製作後端的開發環境
<i>Apple Reachability</i>	<i>iOS App</i> 的 <i>api</i>
<i>Alamofire</i>	<i>iOS App</i> 的 <i>api</i>

肆、研究過程及方式

午餐系統後端由 *Php* 作為後台，*Mysql* 作為資料庫，並且輸出結果到前後端交換介面；網頁版前台、*iOS* 前台及其他外掛插件，從前後端交換介面擷取資料，再將資料展現給使用者，下圖為午餐系統的架構圖。

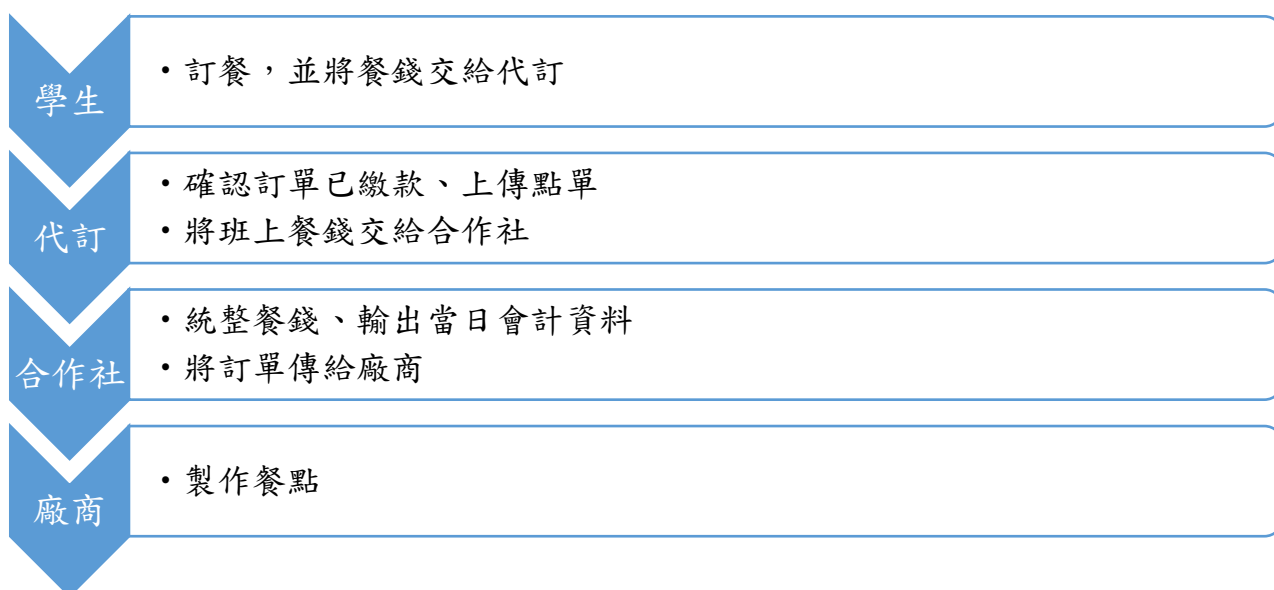


午餐系統與大多數 *POS* 系統的不同之處在於我們擁有非常良好的可擴充性，如同電子布告欄「*BBS*」，任何人都能實作自己的前端，只不過我們採用 *Json* 作為資料交換介面，而 *BBS* 採用 *Telnet* 協定。

午餐系統每天都會有大量的點餐資料，如果能對這些資料進行分析，就能夠協助廠商預測明天的餐點量，於是我們設計了一個預測模型。

一、系統使用方法

以下是使用午餐系統的使用概念，資料會跟著箭頭傳給下一個使用者。


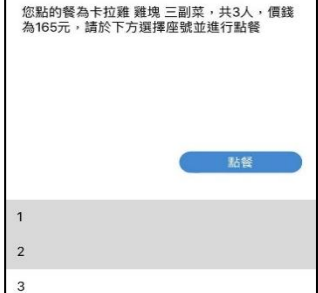
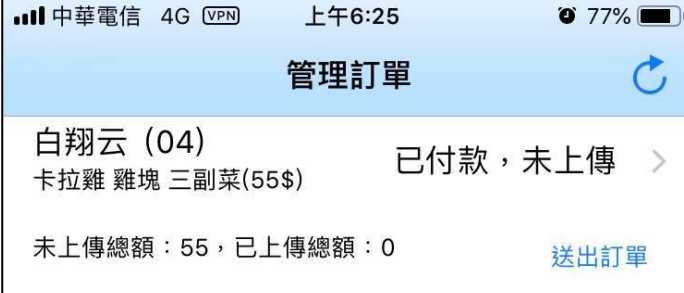


(一)、學生、代訂使用方法

學生可以點餐，代訂可以替學生點餐；學生可以在查看點單中查看自己的付款狀態，代訂在學生繳錢之後，可以將點單標記為已繳款。

行為 身分	點餐	未付款	已付款
學生			
代訂			

我們也有提供 iOS 版的前端，功能與網頁版前端類似，不過比網頁版前端簡潔，對於一些排版細節也做得比較網頁版前端好。

行為 身分	點餐	未繳款/已繳款
學生		
代訂		

代訂可以上傳資料，接下來資料就會傳送到合作社手上，上傳之後的資料沒辦法撤銷。

系統 狀態	iOS	網頁
未上傳		
已上傳		

(二)、合作社、廠商使用方法

合作社可以將上傳過來的點單標記為已繳款，確認繳款的資料會傳給廠商，廠商就可以開始製作餐點了；有些廠商因為工作環境較為油膩，不適合使用電子產品，所以我們做了額外的列印紙本功能，可以將點單資料列印成紙本。

身分 操作	合作社	廠商
顯示點單		

每間廠商都能夠列印自己的總表、班表，總表可以讓廠商知道今天總共需要生產多少份便當，藉此加快生產速度；班表可以告訴廠商今天某班需要多少份餐，將每班所需放入班級籃即可(該行為也稱為檢餐)。

身分 操作	總表	班表																																																																		
廠商	<p>總表: 飯食部</p> <table border="1"> <thead> <tr> <th>價格</th><th>名稱</th><th>數量</th></tr> </thead> <tbody> <tr><td>55</td><td>卡拉雞 雞塊 三副菜</td><td>5</td></tr> <tr><td>55</td><td>起司豬排 紅燒肉 三副菜</td><td>5</td></tr> <tr><td>55</td><td>蒜泥白肉 三副菜</td><td>3</td></tr> <tr><td>55</td><td>紅燒牛肉飯 三副菜</td><td>3</td></tr> <tr><td>55</td><td>黑胡椒烤肉飯 三副菜</td><td>11</td></tr> <tr><td>55</td><td>香煎無骨腿排 三副菜</td><td>9</td></tr> <tr><td>55</td><td>日式魚排 雞塊 三副菜</td><td>4</td></tr> <tr><td>55</td><td>蜜汁雞腿 香腸 三副菜</td><td>4</td></tr> <tr><td>55</td><td>香酥雞排 雞塊 三副菜</td><td>5</td></tr> <tr><td>55</td><td>蔥油雞飯 三副菜</td><td>5</td></tr> </tbody> </table>	價格	名稱	數量	55	卡拉雞 雞塊 三副菜	5	55	起司豬排 紅燒肉 三副菜	5	55	蒜泥白肉 三副菜	3	55	紅燒牛肉飯 三副菜	3	55	黑胡椒烤肉飯 三副菜	11	55	香煎無骨腿排 三副菜	9	55	日式魚排 雞塊 三副菜	4	55	蜜汁雞腿 香腸 三副菜	4	55	香酥雞排 雞塊 三副菜	5	55	蔥油雞飯 三副菜	5	<p>104: 飯食部</p> <table border="1"> <thead> <tr> <th>價格</th><th>名稱</th><th>數量</th></tr> </thead> <tbody> <tr><td>55</td><td>卡拉雞 雞塊 三副菜</td><td></td></tr> <tr><td>55</td><td>起司豬排 紅燒肉 三副菜</td><td></td></tr> <tr><td>55</td><td>蒜泥白肉 三副菜</td><td></td></tr> <tr><td>55</td><td>紅燒牛肉飯 三副菜</td><td></td></tr> <tr><td>55</td><td>黑胡椒烤肉飯 三副菜</td><td></td></tr> <tr><td>55</td><td>香煎無骨腿排 三副菜</td><td></td></tr> <tr><td>55</td><td>日式魚排 雞塊 三副菜</td><td></td></tr> <tr><td>55</td><td>蜜汁雞腿 香腸 三副菜</td><td>1</td></tr> <tr><td>55</td><td>香酥雞排 雞塊 三副菜</td><td>1</td></tr> <tr><td>55</td><td>蔥油雞飯 三副菜</td><td></td></tr> </tbody> </table>	價格	名稱	數量	55	卡拉雞 雞塊 三副菜		55	起司豬排 紅燒肉 三副菜		55	蒜泥白肉 三副菜		55	紅燒牛肉飯 三副菜		55	黑胡椒烤肉飯 三副菜		55	香煎無骨腿排 三副菜		55	日式魚排 雞塊 三副菜		55	蜜汁雞腿 香腸 三副菜	1	55	香酥雞排 雞塊 三副菜	1	55	蔥油雞飯 三副菜	
價格	名稱	數量																																																																		
55	卡拉雞 雞塊 三副菜	5																																																																		
55	起司豬排 紅燒肉 三副菜	5																																																																		
55	蒜泥白肉 三副菜	3																																																																		
55	紅燒牛肉飯 三副菜	3																																																																		
55	黑胡椒烤肉飯 三副菜	11																																																																		
55	香煎無骨腿排 三副菜	9																																																																		
55	日式魚排 雞塊 三副菜	4																																																																		
55	蜜汁雞腿 香腸 三副菜	4																																																																		
55	香酥雞排 雞塊 三副菜	5																																																																		
55	蔥油雞飯 三副菜	5																																																																		
價格	名稱	數量																																																																		
55	卡拉雞 雞塊 三副菜																																																																			
55	起司豬排 紅燒肉 三副菜																																																																			
55	蒜泥白肉 三副菜																																																																			
55	紅燒牛肉飯 三副菜																																																																			
55	黑胡椒烤肉飯 三副菜																																																																			
55	香煎無骨腿排 三副菜																																																																			
55	日式魚排 雞塊 三副菜																																																																			
55	蜜汁雞腿 香腸 三副菜	1																																																																		
55	香酥雞排 雞塊 三副菜	1																																																																		
55	蔥油雞飯 三副菜																																																																			

合作社能夠輸出當日會計報表，方便對帳，以下是合作社的輸出當日報表程式。

輸出程式

合作社帳號

合作社密碼

起始輸出日期

終止輸出日期

☒ 擷取實際歷史資料

輸出資料

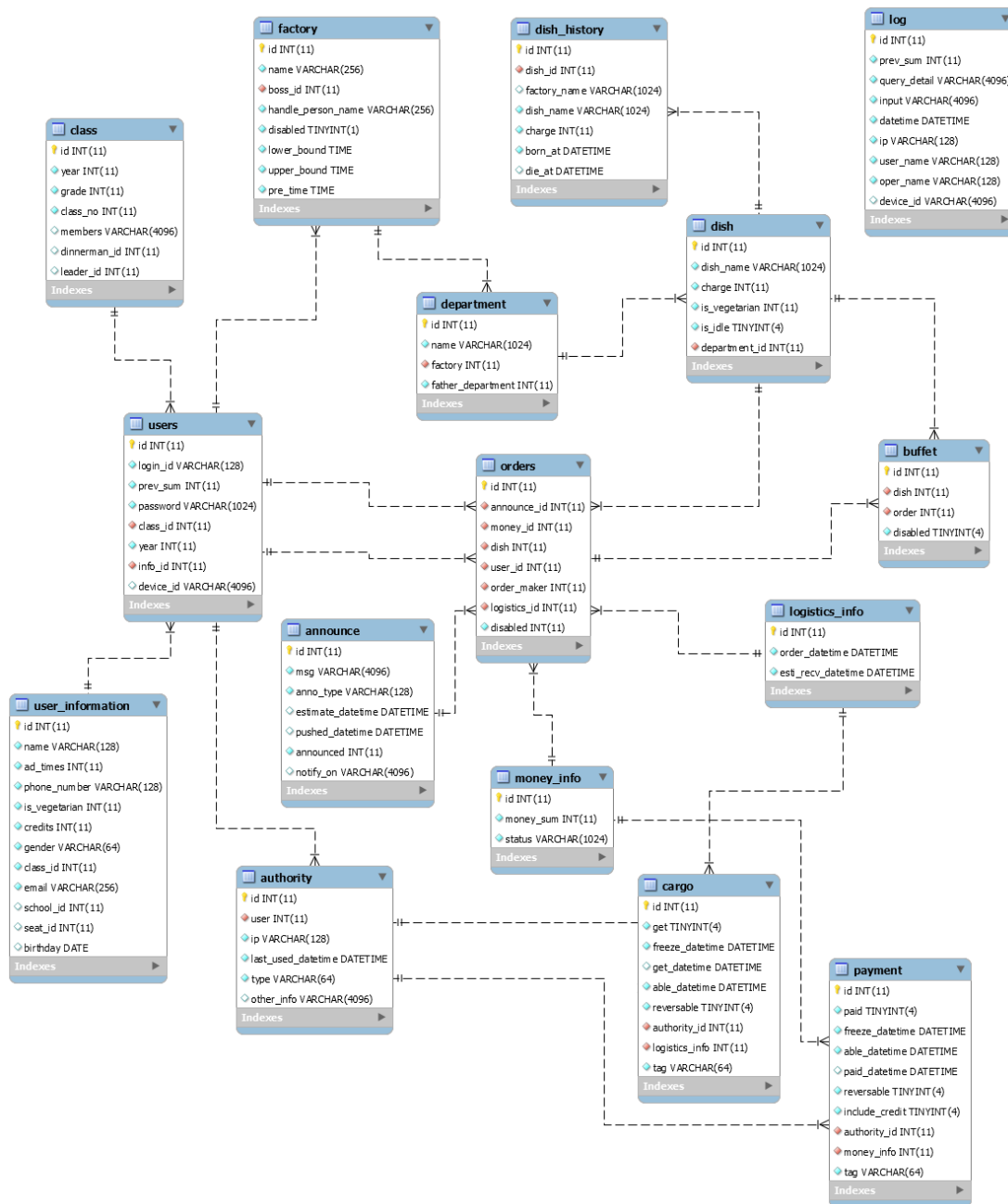
輸出結果

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA				
1		瑞益 愛佳						瑞益 愛佳						瑞益 愛佳						瑞益 愛佳						瑞益 愛佳					
2		40	55	40	55	合計			40	55	40	55	合計			40	55	40	55	合計			40	55	40	55	合計				
3	101	7	7	0	0	665	201	0	0	0	0	0	0	301	4	8	0	1	655												
4	102	4	13	1	0	915	202	0	0	0	0	0	0	302	0	0	0	0	0												
5	103	10	14	0	0	1170	203	0	0	0	0	0	0	303	0	0	0	0	0												
6	104	12	6	0	0	810	204	0	0	0	0	0	0	304	6	5	0	0	515												
7	105	16	6	0	0	970	205	0	6	0	0	330	305	0	0	0	0	0	0												
8	106	13	5	0	0	795	206	3	3	0	0	285	306	0	0	0	0	0	0												
9	107	9	8	0	1	855	207	0	0	0	0	0	307	0	0	0	0	0	0												
10	108	11	5	0	1	770	208	3	6	0	0	450	308	0	0	0	0	0	0												
11	109	4	10	0	2	820	209	0	0	0	0	0	309	0	0	0	0	0	0												
12	110	13	8	0	0	960	210	3	3	0	1	340	310	0	0	0	0	0	0												
13	111	3	8	1	3	765	211	1	0	0	0	40	311	0	0	0	0	0	0												
14	112	0	0	0	0	0	212	0	0	0	0	0	312	0	0	0	0	0	0												
15	113	14	13	1	0	1315	213	0	0	0	0	0	313	0	4	0	0	220													
16	114	4	5	1	2	585	214	0	0	0	0	0	314	0	0	0	0	0	0												
17	115	5	10	0	0	750	215	0	0	0	0	0	315	0	0	0	0	0	0												
18	116	20	8	0	0	1240	216	3	6	0	0	450	316	0	0	0	0	0	0												
19	117	16	7	0	0	1025	217	0	0	0	0	0	317	0	0	0	0	0	0												
20	118	11	9	0	1	990	218	0	0	0	0	0	318	3	8	0	0	560													
21	119	10	5	0	3	840	219	0	1	0	0	55	319	0	0	0	0	0	0												
22	120	6	5	0	0	515	220	0	0	0	0	0	320	0	0	0	0	0	0												

選定好輸出的日期之後，就能夠輸出資料到 *Excel* 上了，在 *Excel* 的右側表格可以直接輸入今天拿到多少錢，*Excel* 就會把總金額計算好。

二、資料庫

下列為午餐系統的資料結構模型，每一條線代表一個關聯性連接。



午餐系統的資料庫為關聯性資料庫，並使用 *innodb* 做為引擎，*innodb* 支援交易機制，比起 *myisam*，使用 *innodb* 更方便處理死結回溯的問題。

(一)、動態查詢

後台會根據傳入參數，使用這六種篩選語句(*syntax*)組合成所需的篩選條件，向資料庫下達 *SQL* 指令。

名稱	指令
時間下界	AND (? < LO.esti_recv_datetime)
時間上界	AND (? > LO.esti_recv_datetime)
針對特定使用者搜尋	AND (? = U.id)
針對某間廠商進行搜尋	AND (? = F.id)
針對特定列查詢	AND (? = O.id)
針對班級查詢	AND ((SELECT U.class_id FROM users AS U WHERE U.id = ?) = U.class_id)

在下達 *sql* 指令時，可以使用代稱(*alias*)方便撰寫 *sql* 語句，以下是代稱的解釋。

中文名稱	英文全名	英文縮寫
使用者	User	U
廠商	Factory	F
點單	Order	O
物流資訊	Logistic Info	LO

(二)、Procedure 優化

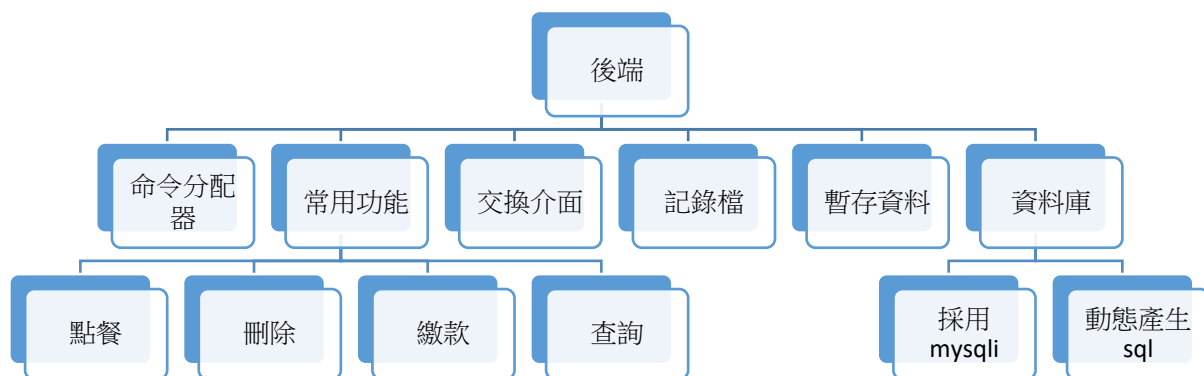
對於單語句(*syntax*)的 *sql* 操作，直接在 *php* 中呼叫資料庫即可；對於多語句的 *sql* 操作，則包裝成一個 *Procedure*，方便處理死結(*Deadlock*)回溯(*Rollback*)，也不必與資料庫伺服器多次連線。

下表為該 *Procedure* 進行四種基本操作(*insert/update/delete/select*)的數量，包裝的操作越多，越能節省與資料庫連線的時間。

	Insert	Update	Delete	Select	總計
Make order	6	0	0	9	15
Update dish	1	2	0	1	4

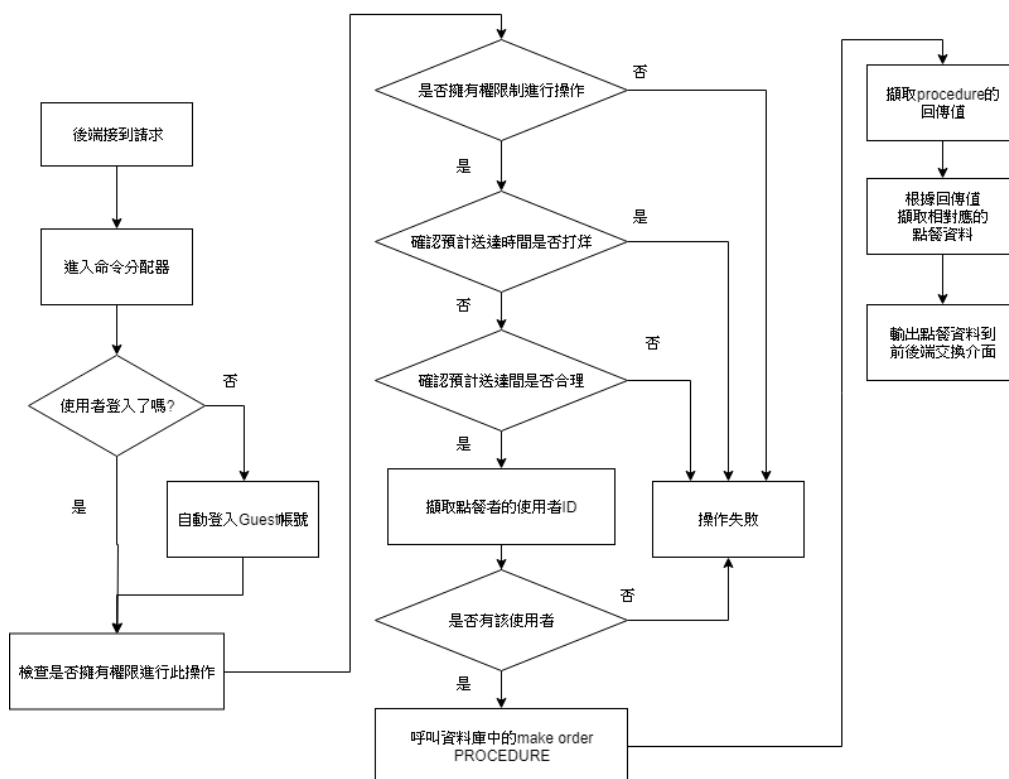
三、後端

下圖為後端的架構，我們將後端分成多個模組，方便維護，也方便開發新功能。



(一)、處理流程

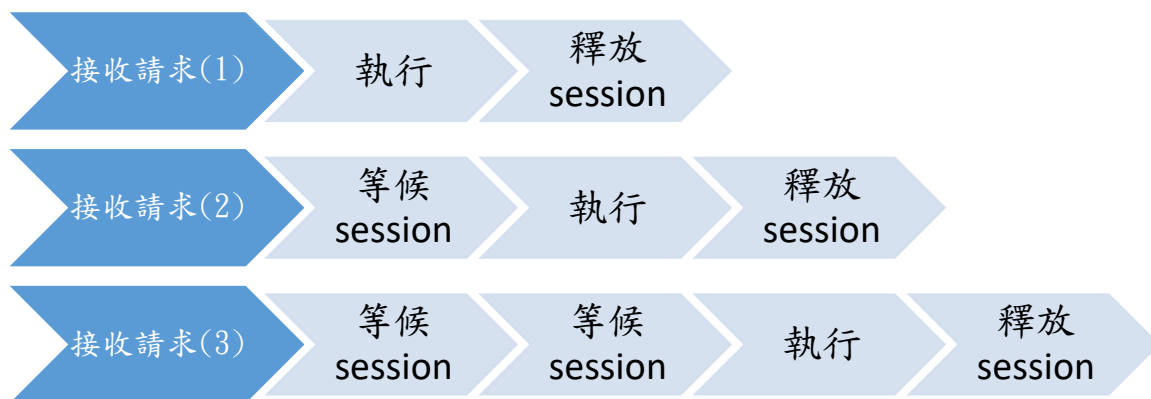
以進行點餐操作為例，系統會先進行嚴密的輸入審查，確認是否無誤後，再將資料寫入資料庫，資料庫回傳點餐 *id*，後端抓出該筆資料，經過編碼之後輸出到交換介面，以下為處理的流程圖。



(二)、效能阻塞

1. session 阻塞

php 為了保證執行緒安全，同一個 *session* 同時只能給一個請求使用。在每個請求都不會提前釋放 *session* 的狀況下，同時送出大量請求，會使得效能非常低落，如下圖所示。



上圖中，請求(3)等候前面兩個請求處理完才開始執行，成為緩慢的串行命令。如果程式會先複製好 *session* 再執行，則每個請求只需要等候其他請求複製完資料，就能夠先開始執行了，如下圖所示。



上圖中，請求(3)僅等候前面兩個請求複製資料，因為不必等候其他請求，因此能受益於 *CPU* 的平行處理，使得效能提升。

2. 資料庫存取阻塞

資料庫的存取速度遠遠低於記憶體存取速度，如果每次使用常駐資料時，都向資料庫要求一次資料，則這些常駐資料請求會拖累系統效能。後端會將常駐資料先快取於 *session*，需要使用資料時直接從 *session* 調用資料，就不必再向資料庫請求資料了。

(三)、安全性

針對後端安全，我們進行了以下幾種保護措施。

1. 密碼安全性

對於所有工作人員的密碼，皆為六個字元以上的英數混和字串，且所有的登入失敗都會寫入紀錄檔中，方便追蹤帳號是否有安全性疑慮。

2. SQL 注入

不以舊版的 *mysql* 模組操作資料庫，而以新版的 *mysqli* 操作資料庫；並將所有的 *statement* 進行 *prepare* 後 *bind_param*，再執行 *sql* 語句，不直接在 *statement* 中寫入值，如下列虛擬碼所示。

具有 <i>sql</i> 注入風險	不具有 <i>sql</i> 注入風險
<pre>\$sql="select id from orders where id = \$uid" \$database->execute(\$sql)</pre>	<pre>\$sql="select id from orders where id = ?" \$database->prepare(\$sql) \$database->bind_param("1 or True") \$database->execute()</pre>

3. XSS

在 *check_valid* 模組中，嚴格限制了每個輸入參數，凡是參數不符合規定，則立刻丟出例外狀況，並寫入記錄檔中。

4. 資料庫死結

在單線程的測試環境下，很少會遇到資料庫死結，而在系統真正運行的時候，常會遇到不可預知的死結。一個 *Procedure* 包裝了多個語句(*Syntax*)，若是在尚未執行完 *Procedure* 前，發生了死結，則可能會有不可預知的後果。

我們針對容易發生死結的 *Procedure* 加上 *start transaction*、*rollback*、*commit*，若是在 *Procedure* 尚未結束前發生死結，則回溯(*Rollback*)整個 *Procedure* 的操作。

四、前端

(一)、iOS 前端

iOS 前端為專屬開發給蘋果使用者的操作介面，由於網頁版前端較不直觀，我們開發了專屬於蘋果使用者的前端，該前端符合 iOS 的人機互動指南。

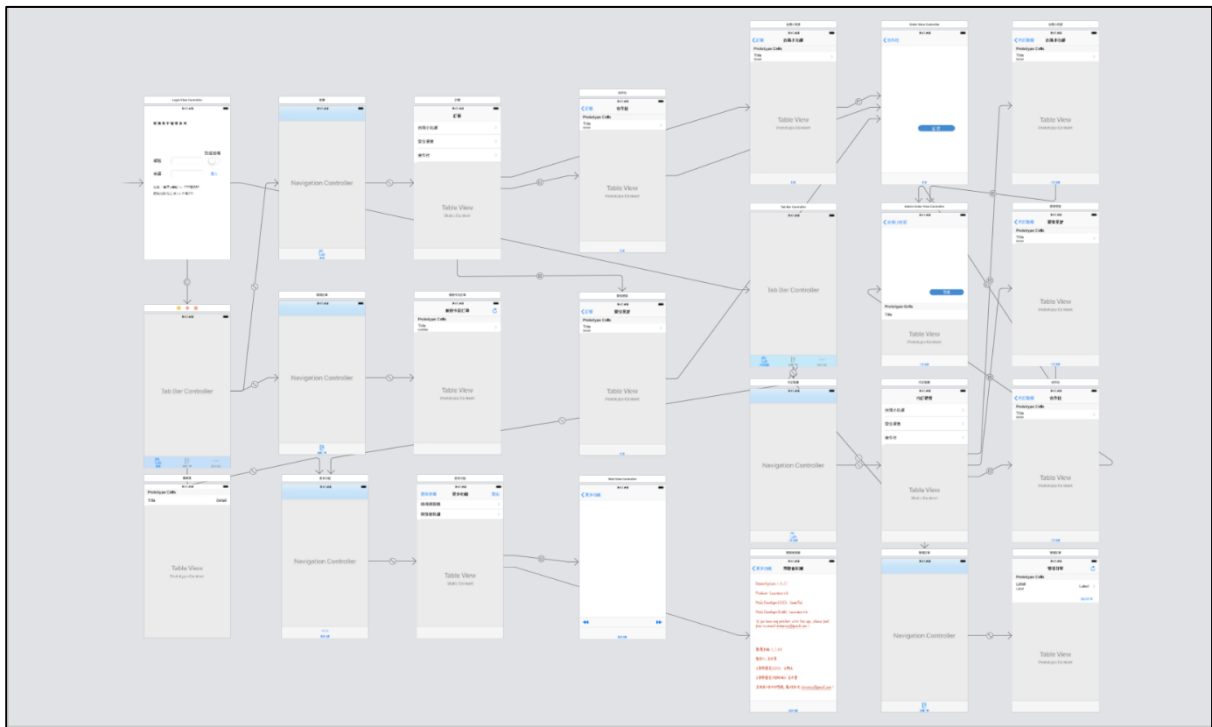
以下為 iOS 前端的登入畫面，我們可以見到前端十分簡潔，與使用者的互動簡單明瞭。



1. 呼叫後端

採用 *Alamofire* 第三方 *API(Application Interface)* 進行 *HTTP* 請求及回應，上傳時以 *get* 方法訪問(*Request*)伺服器，並使用 *Swift* 原生 *API* 中的 *JSONDecoder* 來解析伺服器回傳的 *Json* 字串。

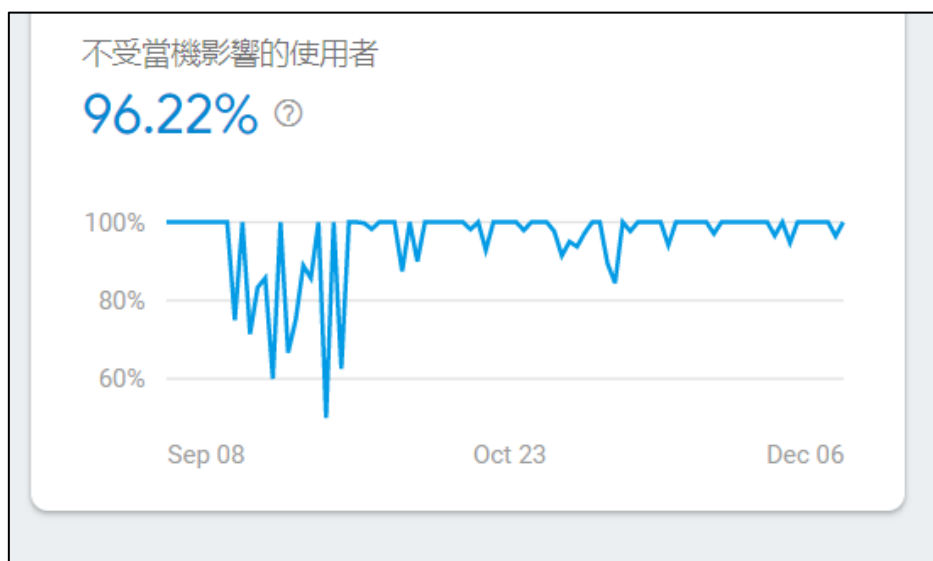
2. 頁面圖



以上為 iOS 前端的頁面(Layout)關係圖，每一條線代表觸發任何事件後可以從一個頁面(View)轉跳至下一個頁面(View)。

3. 錯誤分析

發生任何錯誤狀況時，App 會傳送資料給 *Crashlytics*，在 *Firebase* 開發者頁面中可以立即看到發生的例外狀況，*Firebase* 也會發送電子郵件給開發者，讓開發者能夠在最快的時間內修正錯誤，以下是 *Crashlytics* 的統計分析，在九月時系統剛剛上線，較不穩定，進入十月之後系統就幾乎沒有任何問題了。



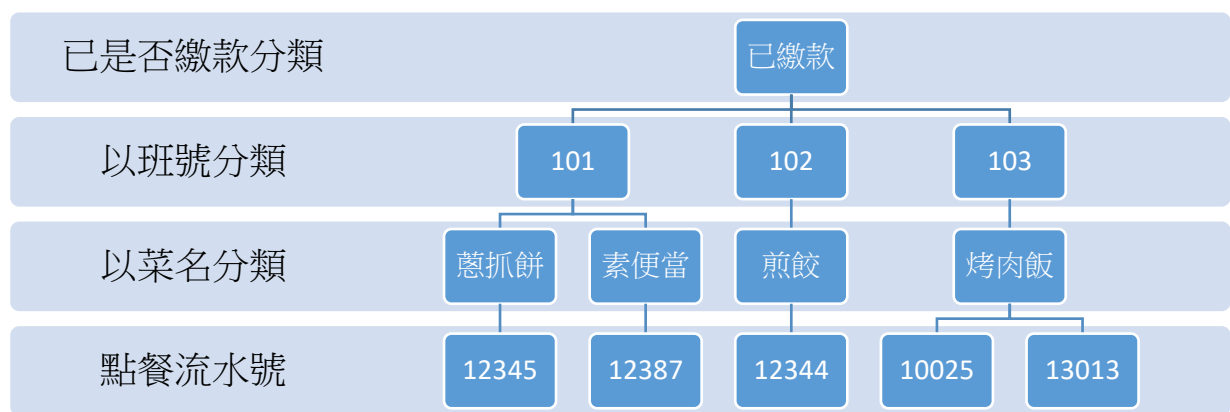
(二)、網頁前端

網頁前端擁有最完整的功能，而且當其他前端無法運作時，網頁前端會確保大多數使用者還能夠使用系統，不會使整套系統停止運作。

1. ajax

對於批次點餐、繳款、上傳，我們使用 *ajax* 技術向伺服器後端請求資料，使用 *ajax* 技術不只可以避免前後端耦合，還可以受惠於平行處理使得效能增加。

2. 分類樹



當我們想要確認繳款 101 點的所有餐時，我們不必一一查看所有點單，我們只需要查看 101 的分類就可以了，上圖為分類樹的抽象概念，下圖為實際使用分類樹。



由右圖可見，下列該分類樹的階層。

1. 班級
2. 是否付款
3. 廠商
4. 餐點
5. 點單

分類樹還會順便把金額數量加總，方便點帳。在金額無誤時，就直接按下已繳款按鈕；金額有誤時，從階層式的資料中找出是哪裡點錢點錯了即可。

3. 雲端運算

為了避免讓前端處理大量資料，我們選擇在伺服器先使用分類樹整理資料。

在前端的 *backstage.php* 中，先呼叫後端主模組 *backend_main*，抓取點餐資料，再將資料傳給 *collapsible* 進行整理並轉為 *HTML* 字串，再將已經轉為 *HTML* 碼的字串傳給前端，雖然使用了多一點的網路流量，但是能夠節省前端運算的大量時間。

(三)、輔助外掛

1. 當日會計報表

會計報表可以視為一個前端，會將點餐資料統整之後輸出給 *Excel*，以下為十月二十三號的會計報表。

我們可以看到當天的差異只有 7，而這大多是人為計算錯誤，不必太在意。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1		珊益		愛佳					珊益		愛佳					珊益		愛佳									
2		40	55	40	55	合計			40	55	40	55	合計			40	55	40	55	合計			珊益		愛佳		卡片
3	101	7	7	0	0	665	201	0	0	0	0	0	0	301	4	8	0	1	655			40	55	40	55	-55	
4	102	4	13	1	0	915	202	0	0	0	0	0	0	302	0	0	0	0	0			214	202	4	15		
5	103	10	14	0	0	1170	203	0	0	0	0	0	0	303	0	0	0	0	0			19670		985			
6	104	12	6	0	0	810	204	0	0	0	0	0	0	304	6	5	0	0	515			40	55	合計		-55	
7	105	16	6	0	0	970	205	0	6	0	0	330	305	0	0	0	0	0			小計	218	217	435		0	
8	106	13	5	0	0	795	206	3	3	0	0	285	306	0	0	0	0	0			總計	8720	11935	20655		0	
9	107	9	8	0	1	855	207	0	0	0	0	0	307	0	0	0	0	0									
10	108	11	5	0	1	770	208	3	6	0	0	450	308	0	0	0	0	0			金額	數量1	數量2	數量3	小計		
11	109	4	10	0	2	820	209	0	0	0	0	0	309	0	0	0	0	0			1000	9				9000	
12	110	13	8	0	0	960	210	3	3	0	1	340	310	0	0	0	0	0			500	3				1500	
13	111	3	8	1	3	765	211	1	0	0	0	40	311	0	0	0	0	0			100	60				6000	
14	112	0	0	0	0	0	212	0	0	0	0	0	312	0	0	0	0	0			50	59				2950	
15	113	14	13	1	0	1315	213	0	0	0	0	0	313	0	4	0	0	220			10	110				1100	
16	114	4	5	1	2	585	214	0	0	0	0	0	314	0	0	0	0	0			5	15				75	
17	115	5	10	0	0	750	215	0	0	0	0	0	315	0	0	0	0	0			1	37				37	
18	116	20	8	0	0	1240	216	3	6	0	0	450	316	0	0	0	0	0			小計	20662	0	0		20662	
19	117	16	7	0	0	1025	217	0	0	0	0	0	317	0	0	0	0	0									
20	118	11	9	0	1	990	218	0	0	0	0	0	318	3	8	0	0	560			合計	實際收	差異	正負差	結餘		
21	119	10	5	0	3	840	219	0	1	0	0	55	319	0	0	0	0	0			20655	20662	7				
22	120	6	5	0	0	515	220	0	0	0	0	0	320	0	0	0	0	0									

2. 新生資料匯入軟體

匯入軟體必須放在伺服器主機內，才能存取資料庫，新生資料匯入軟體會先從 *Excel* 資料表裡面抓取新生資料，再將資料轉成 *SQL* 語句，交給資料庫處理。

(四)、前後端交換介面

前後端交換介面確保後端與前端的獨立，任何能夠操作命令介面，與使用者互動，並且能夠擷取回傳資料，不論語言、平台撰寫成的程式，皆可被視為一個前端；任何能夠根據命令回傳出相對應的程式，不論語言、不論平台，也皆可被視為一個後端。

1. 輸出資料結構

原始輸出為 *Json* 字串，我們將 *Json* 轉換成表格，以利理解，以下表格為精簡後的點單資料結構。

Id	22410	
user	id	11184
	name	白翔云
	seat_no	20904
dish	dish_name	卡拉雞 雞塊 三副菜
	dish_id	1
	dish_cost	55

2. 命令介面

以下表格為精簡後的命令介面，「-」代表沒有參數。

操作	參數 1	參數 2
login	id=[使用者帳號]	password=[密碼]
select_self	esti_start=[時間上界]	esti_start=[時間上界]
payment_usr	oid=[點餐單號]	target=[是否繳款]
make_self_order	did=[餐點號碼]	esti_recv=[預計送達時間]

如第一列所示，其意義代表：「使用 *get/post* 方法造訪後台，以 *cmd* 為索引，傳入字串 *login*；以 *id* 為索引，傳入使用者帳號；以 *password* 為索引，傳入密碼」。

3. JSON 與編碼

採用 *UTF-8* 編碼，包含 *BOM Header*，對於 *Json* 中包含的中文字串不加以編碼，對於「*”*」字元更改為「**」，對於「**」字元更改為「**」。

五、預測模型

我們先向廠商、合作社打聽了一下平常點餐的趨勢，廠商表示訂購便當的意願隨著在學校的時間漸漸降低。我們經過初步分析，得知高一最多人點餐，高二的點餐人數較少，高三幾乎沒有人點餐，每天的點餐人數大約以 3 份餐/天緩慢下滑。

我們將預測模型拆成兩個子模型，一為比例模型，二為數量模型。顧名思義，比例模型能夠給出明天的點餐比例，數量模型能夠給出明天的點餐總數，兩個模型一起使用就能得到明天各種餐點的數量。

(一)、資料結構

模型沒有辦法直接處理前後端交換介面的資料，所以我們要對資料進行一些處理，數量模型只能接受點餐序列作為輸入，比例模型只能接受點餐圖作為輸入，本章介紹如何將前後端交換介面的資料轉為上述兩種結構。

原始的點餐資料可以視為下列表格，「-」代表沒有點餐。

	甲生	乙生	丙生	丁生	戊生
08/07	韓式拌飯	烤肉飯	-	烤肉飯	素便當
08/08	-	烤肉飯	韓式拌飯	-	素便當
08/09	素便當	韓式拌飯	素便當	烤肉飯	素便當
08/10	烤肉飯	烤肉飯	-	韓式拌飯	素便當
08/11	韓式拌飯	烤肉飯	韓式拌飯	韓式拌飯	素便當

以下根據製作該餐所需的原料進行分類，且由上往下進行篩選，例如「韓式拌飯」符合第一行的 *Regex* 篩選，故屬於調味飯類；「烤肉飯」不符合第一行的 *Regex* 篩選，而符合第三行的 *Regex* 篩選，故屬於便當類。

雜類	Regex: "(焗) (拌飯) (炒飯) (飯糰)"
麵類	Regex: "(烏龍) (麵) (湯)"
便當類	Regex: "(副菜) (飯)"
小吃類	Regex: "((餃) (蔥抓餅) (鍋貼) (板條))"
鍋類	Regex: "(鍋) (粥)"

1. 點餐序列

將甲生的點餐資料提取出來，並加以分類，再填上是否有點餐，即為點餐序列。

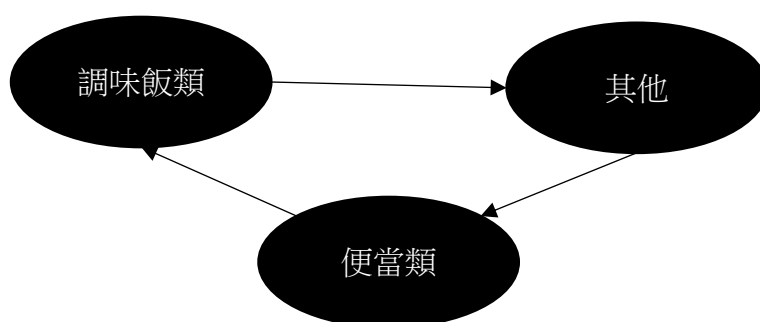
甲生	08/07	08/08	08/09	08/10	08/11
餐點類別	雜類	-	其他	便當類	雜類
是否點餐	True	False	True	True	True

2. 點餐圖

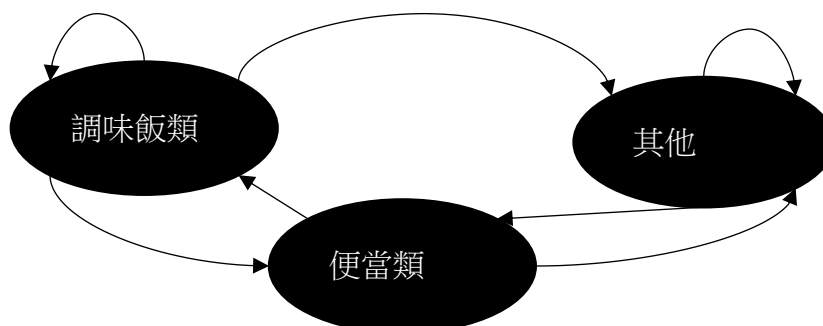
根據點餐序列，刪去沒有點餐的那一天，即如下表。

甲生	08/07	08/09	08/10	08/11
餐點類別	雜類	其他	便當類	雜類
是否點餐	True	True	True	True

將每一天的轉移視為圖上的一個邊，每一個分類視為圖上的一個節點，則可得到一個有向圖。



若將所有人的點餐序列寫入同一張圖，則可得到如同下面的有向圖，我們稱呼他為點餐圖。



(二)、比例模型

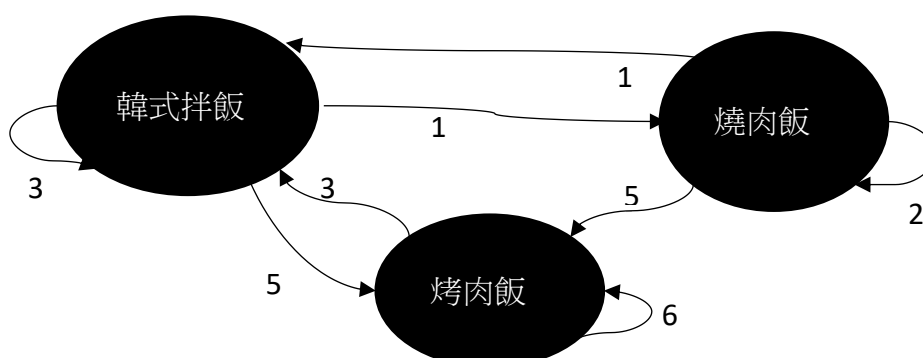
設計比例模型旨在於使用數學模型來預測各種類別的比例，我們使用機率矩陣的穩定狀態來預測比例，無法求得穩定狀態時，以矩陣快速冪代替。

我們可以將現實世界中的操作抽象化，成為圖論中的操作，方便程式預測，下表為現實世界中的操作與圖論中的操作對照表。

圖論上的操作	現實世界的意義
從圖上的任意一個點出發	第一天先隨便點一道菜
經由任意一條邊	經過了一天
到達圖上的另外一個點	第二天點了另外一道餐
經過一個自環	經過了一天，決定要吃同一道菜
到了同一個點	第三天點了同一道菜

1. 鄰接機率矩陣

對於任意有向圖，將(節點 i 到節點 j 所有邊的數量)/(節點 i 的出度)，寫入矩陣中的 M_{ij} ，則 M_{ij}^n 之值為從節點 i 移動 n 步後到達節點 j 的機率。



上圖為點餐圖，邊旁邊的數字為重邊的數量，下表為該圖的鄰接矩陣，鄰接矩陣 K_{ij} 之值為節點 i 到節點 j 所有邊的數量。

	韓式拌飯	烤肉飯	燒肉飯
韓式拌飯	3	3	1
烤肉飯	5	6	5
燒肉飯	1	0	2

上表為該圖的鄰接矩陣 K ，下表為鄰接機率矩陣，其中 $M_{ij} = K_{ij} / \sum_{k=1}^n K_{ik}$ ， M_{ij} 為從節點 i 移動到節點 j 的機率為多少。

	韓式拌飯	烤肉飯	燒肉飯
韓式拌飯	1/3	1/3	1/8
烤肉飯	5/9	2/3	5/8
燒肉飯	1/9	0	1/4

對於矩陣 M_{ij}^2 ，我們可以得知 $M_{ij}^2 = \sum_{k=1}^n M_{ik} M_{kj}$ ，即為從節點 i 經過任意節點 k 再到達節點 j 之機率和，同時也為從節點 i 到移動兩步到達節點 j 的機率。

2. 馬可夫矩陣

對於點餐向量 K ， K_i 代表原先 i 餐佔所有餐的比例，若 $KM^n = K'$ ，則 K'_i 代表經過 n 天後 i 餐的比例。我們定義 $K^{(i+n)} = K^i M^n$ ，而且經過多次轉移之後， K 會漸漸趨近於一個穩定狀態，則我們可以得到方程式 $K := KM$ 。

此時，我們稱呼 K 為矩陣 M 的穩定狀態。

3. 求穩定狀態

我們有兩種方法可以求穩定矩陣，第一種叫做「反矩陣解聯立」，第二種叫做「矩陣快速幂」，下表比較了兩種方法的優劣。

	反矩陣解聯立	矩陣快速幂
優點	能夠求出真正的穩定狀態	保證有輸出值
缺點	不一定能夠求出穩定狀態	只能求出近似穩定狀態

在反矩陣解聯立無法求出穩定狀態時，我們使用矩陣快速冪作為替代方案，由於矩陣快速冪有一定的機率會失準，我們選擇多求出幾個近似的穩定狀態，再將各個近似狀態平均，藉此取得較為精準的穩定狀態。

(1). 反矩陣解穩定狀態

根據穩定狀態的定義 $K := KM$ ，我們可以得到下列聯立方程式。

$$K_i = \sum_{j=1}^n K_j M_{ij}$$

化簡後得

$$P_{ij} = \begin{cases} M_{ij} - 1, & i = j \\ M_{ij} & i \neq j \end{cases}, 0 = \sum_{j=1}^n K_j P_{ij}$$

對於聯立方程組有以下性質， W 為給定的係數矩陣， C 為給定的值矩陣， S 為未知數矩陣。

$$C = WS, S = W^{-1}C$$

將 P^T 視為方程式中 K 的係數，且 $1 = \sum_{i=1}^n K_i$ ，我們得知有一行方程式無用，於是可以得到下式。

$$W_{ij} = \begin{cases} 1, & i = n \\ P_{ji} & i \neq n \end{cases}, C_i = \begin{cases} 1, & i = n \\ 0 & i \neq n \end{cases}$$

$$K = W^{-1}C$$

於是我們可以求得穩定狀態 K ，不過並不是每一個 W 都有反矩陣，於是我們使用矩陣快速冪作為替代方案。

(2). 矩陣快速冪

矩陣乘法具有結合律，並根據下列二式，我們可以得知只需要在 $O(\log N)$ 的時間內就能得知 M^n 的值。

$$Z_{i+1} = Z_i^2, Z_1 = M, 1 \leq i \leq \lfloor \log n \rfloor$$

$$M^n = Z_{\lfloor \log n \rfloor} M^{n-2^{\lfloor \log n \rfloor}}, n \in N^+$$

Q 為任意機率向量，給定越大的數字 n ，求出來的向量越接近穩定狀態，給定數列 N ，共有 P 個元素，其中 N_i 為一個隨機整數，且界於 N_{min} 與 N_{max} 之間。

$$N_{min} \leq N_i \leq N_{max}, 1 \leq i \leq P$$

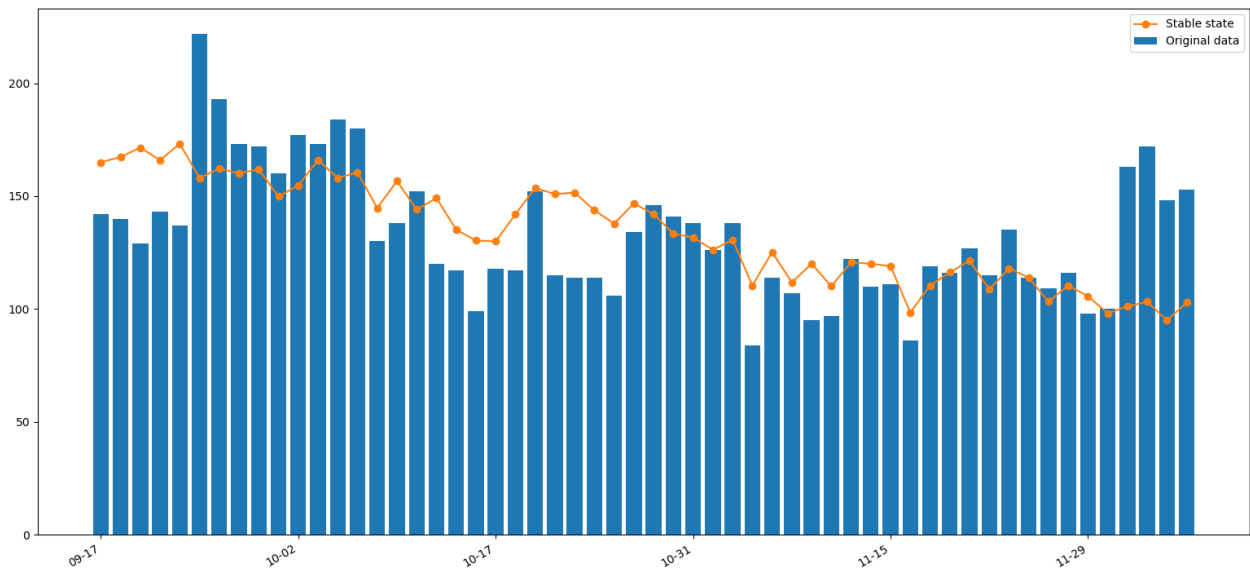
$$K = (\sum_{i=1}^P QM^{N_i})/P$$

該演算法的時間複雜度為 $O(P \log N_{max})$ ，使用越大的 P 與越大的 N_{min} 以及 N_{max} ，預測出來的 K 越接近穩定狀態，模型預設 $N_{min} = 10^7, N_{max} = 10^{10}, P = 10^3$ 。

4. 輸出圖表

下圖為比例模型對便當類的輸出圖，折線為比例模型，直方為實際資料。

折線上每一個點的值，為當天總數乘上穩定狀態的比例，模型的預測值大致與實際值相差不遠，代表本模型具有參考價值。



(三)、數量模型

設計數量模型旨在於使用數學模型來預測總餐數會有多少份，數量模型的核心為 *logistic* 模型，我們對每個人建立一個 *logistic* 模型，再使用統計演算法求出總餐數約有多少份。

1. logistic 模型

logistic 模型最主要的函數為 *sigmoid* 函數，如下。

$$F(x) = \text{Sigmoid}(x) = 1/(1 + e^{-x})$$

將輸入先經過線性變換，再交給 *sigmoid* 進行輸出，定義 M 為線性變換用的向量， X 為輸入的向量。

$$Q = \sum_{i=1}^n M_i X_i, F(M, X) = F(Q) = 1/(1 + e^{-Q})$$

一組訓練用的資料為輸入值 \hat{X}_i 以及輸出值 \hat{Y}_i ， \hat{Y}_i 為一個布林值，只會是 1 或是 0，我們可以寫出損失函數如下。

$$\text{Cost}(M, \hat{X}, \hat{Y}) = \sum_{i=1}^n \hat{Y}_i \log F(M, \hat{X}_i) + (1 - \hat{Y}_i) \log(1 - F(M, \hat{X}_i))$$

可以得知損失函數越大，該模型精確度就愈高。

我們無法找出一個 M 使得損失函數最大化，但是可以使用最大似然估計來估測 M ， k 為偏差倍率，偏差倍率越大，模型訓練越快，越容易錯過最佳解；偏差倍率越小，模型訓練越慢，越容易找到最佳解。

下式為迭代方程式， Cost' 為 Cost 對 M 的偏導函數。

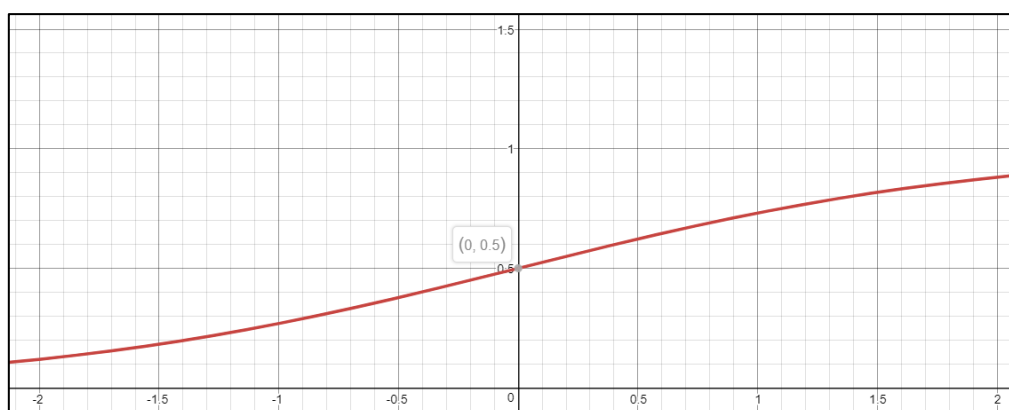
$$M^{(i+1)} = M^{(i)} + k\text{Cost}'(M^{(i)}, X, Y)$$

當 Cost' 趨近零時

$$M := M + k\text{Cost}'(M, X, Y)$$

經過大量迭代之後，我們可以得到向量 M 。

對於輸入值 $X_1 = X_2 \dots = X_n = 0$ ，線性變換後的結果必為零，sigmoid 的輸出值必為 0.5，如下圖。



這樣的模型很明顯不是我們想要的，所以我們需要一個常數來修正模型，其中 ϕ 為一個常數，將模型修正如下。

$$G(M, X) = F\left(\varphi + \sum_{i=1}^n X_i M_i\right)$$

我們使用下列方法來求出 φ ，其中 \bar{M} 為線性變換參數。

$$\bar{X}_i = \begin{cases} 1, & i = n + 1 \\ X_i & \end{cases}$$

$$F(\bar{M}, \bar{X}) = F\left(\sum_{i=1}^{n+1} \bar{X}_i \bar{M}_i\right)$$

使用最大似然估計來估測 \bar{M} ，再將 \bar{M} 還原成 M 與 φ 。

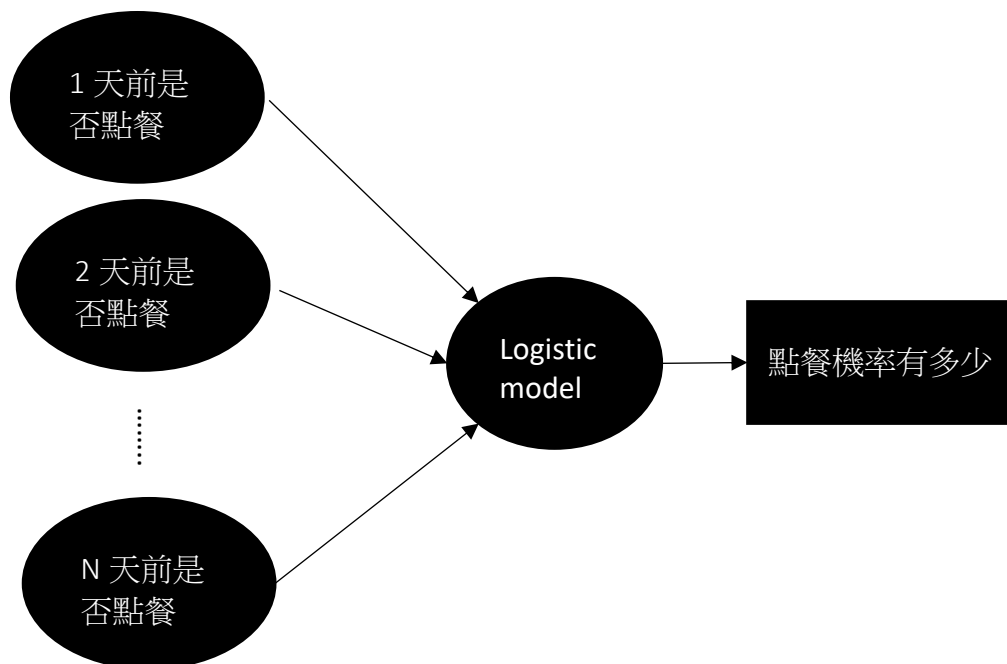
$$M_i = \bar{M}_i, 1 \leq i \leq n$$

$$\varphi = M_{n+1}$$

我們可以獲得較為精確的 **logistic** 模型，模型的訓練時間複雜度為 $O(CVN)$ ，模型預測一組資料的時間為 $O(V)$ ，其中 C 為迭代次數， V 為輸入的向量大小， N 為訓練資料數量。

2. 決策模型

決策模型只能預測特定一個人，模型會根據他之前的點餐行為，預測他今天是否會選擇點餐，以下是模型的輸入輸出圖，我們可以用點餐序列作為模型輸入。



本模型具有一定的規律鑑別能力，像是很多人是禮拜二家裡會準備便當，所以就不會訂購學校的便當，模型能夠偵測出這個人禮拜二通常都不會點餐，宏觀模型沒辦法達到這一點。

以下是一個三循環的測試資料，我們將這組資料交給決策模型進行訓練。

日期	1	2	3	4	5	6	7	8	9
是否點餐	1	0	0	1	0	0	1	0	0

以下是決策模型的訓練成果，模型最後一項的線性變換參數為 φ 。

	M_1	M_2	M_3	M_4	M_5	M_6	Loss
N=1	正相關	無相關	-	-	-	-	-3.37602795
N=2	正相關	正相關	負相關	-	-	-	-0.03053806
N=3	正相關	正相關	負相關	負相關	-	-	-0.00921238
N=4	正相關	正相關	負相關	正相關	無相關	-	-0.00816815
N=5	正相關	正相關	負相關	正相關	正相關	無相關	-0.03469504

負相關代表「該值愈大，輸出值越接近零」，正相關代表「該值愈大，輸出值越接近一」，不相關代表該輸入與輸出幾乎無關聯性，**Loss** 為損失函數值。

由上表我們可以得知，取太大或是太小的 N 容易無法判斷模式，模型預設 $N = 7$ ，因為一個禮拜有七天，大多數的規律都是七天一個循環。

4. 統計演算法

我們想要知道有幾個人點餐的機率最高，我們將甲生點餐的事件寫成 A_1 ，甲生不點餐的事件寫成 A'_1 ，總共有 i 個人點餐的事件寫成 N_i 。

$$P(N_0) = P(A'_1 \cap A'_2 \cap \dots \cap A'_n)$$

$$P(N_1) = P((A_1 \cap A'_2 \cap \dots \cap A'_n) \cup (A'_1 \cap A_2 \cap \dots \cap A'_n) \cup \dots \cup (A'_1 \cap A'_2 \cap \dots \cap A_n))$$

求出每一項 N 需要 $O(2^n)$ ，我們可以想成每個人只有點餐跟不點餐這兩個選項，枚舉每一種狀態，再將所有機率加總。演算法的時間複雜度不甚理想，我們需要對演算法優化。

我們使用 **DP** 來進行優化，優化之後只需要 $O(N^2)$ 的時間複雜度，效率大幅提升，以下是 **DP** 的遞迴關係式。

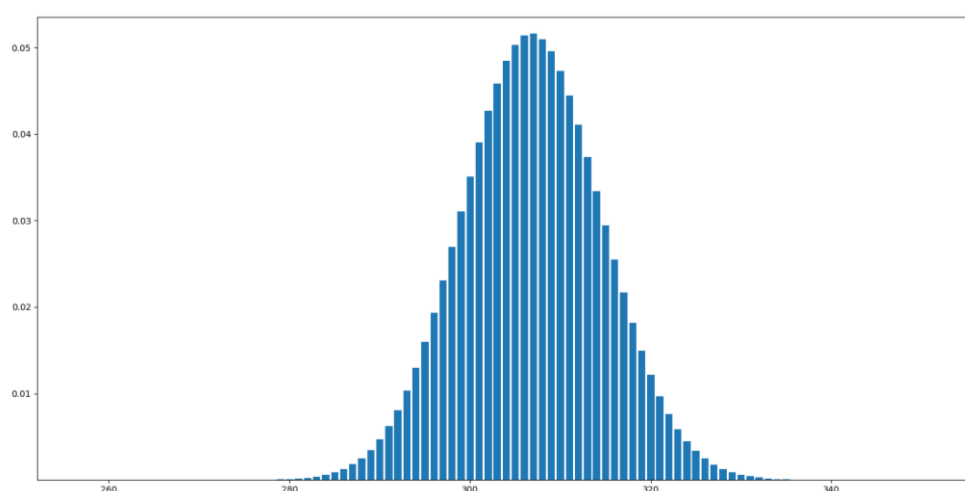
$$N_{j+1} = M_{j+1}(1 - A_i) + M_j A_i$$

$$M_0 = (1 - A_1), M_1 = A_1$$

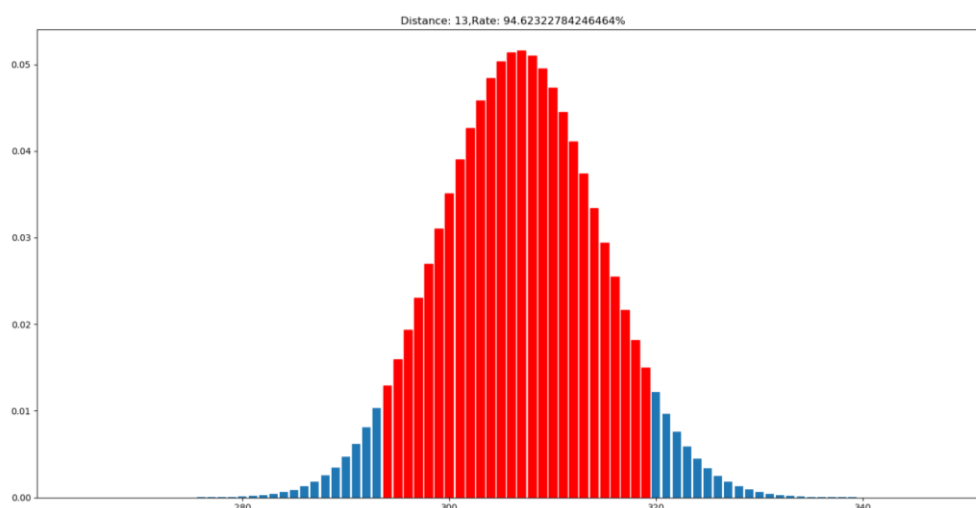
每次執行完迴圈後 N 的值會賦予 M ，最後得到的 N 就是結果。

5. 輸出圖表、信心水準

下圖為數量模型的輸出，很明顯的這是常態分佈的資料，我們將峰值 μ 視為模型的輸出值，本圖的峰值為 **308 份餐**。



下圖紅色區域為峰值上下 13 份餐，紅色區域加總約為 **94%**。



下表為模型的信心水平與信賴區間，我們保守估計數量模型在信賴區間正負十五份餐，有 **94%** 的信心水平。

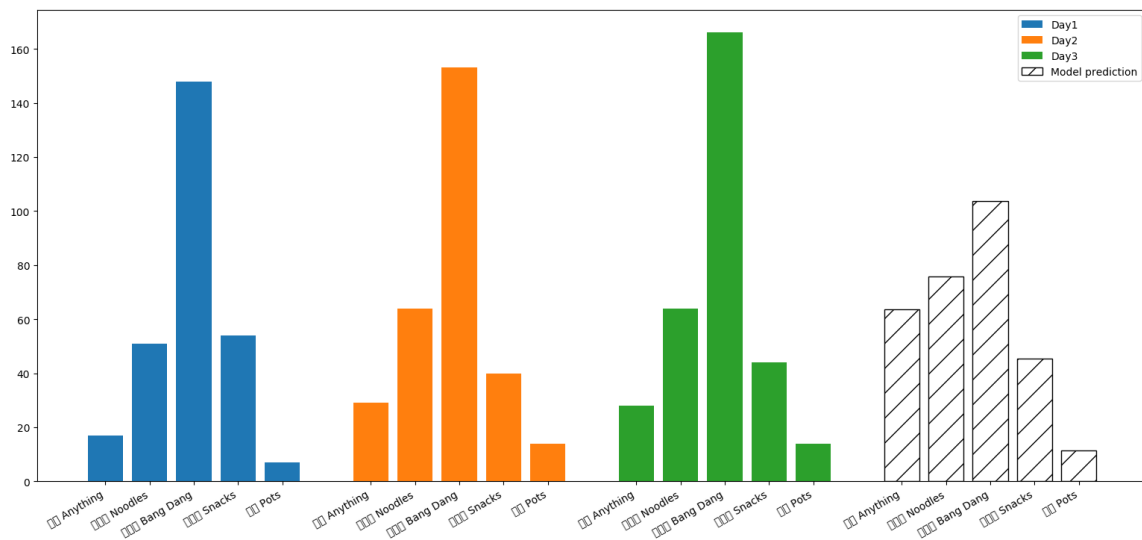
信心水平	72%	83%	87%	91%	94%
信賴區間	$[\mu-8, \mu+8]$	$[\mu-10, \mu+10]$	$[\mu-11, \mu+11]$	$[\mu-12, \mu+12]$	$[\mu-13, \mu+13]$

(四)、預測模型

比例模型可以得出比例，數量模型可以得出總數，再經由下面公式即可獲得每個分類的數量預估值。

$$\text{類別}_{\text{甲}} \text{數量} = (\text{類別}_{\text{甲}} \text{比例})(\text{總數})$$

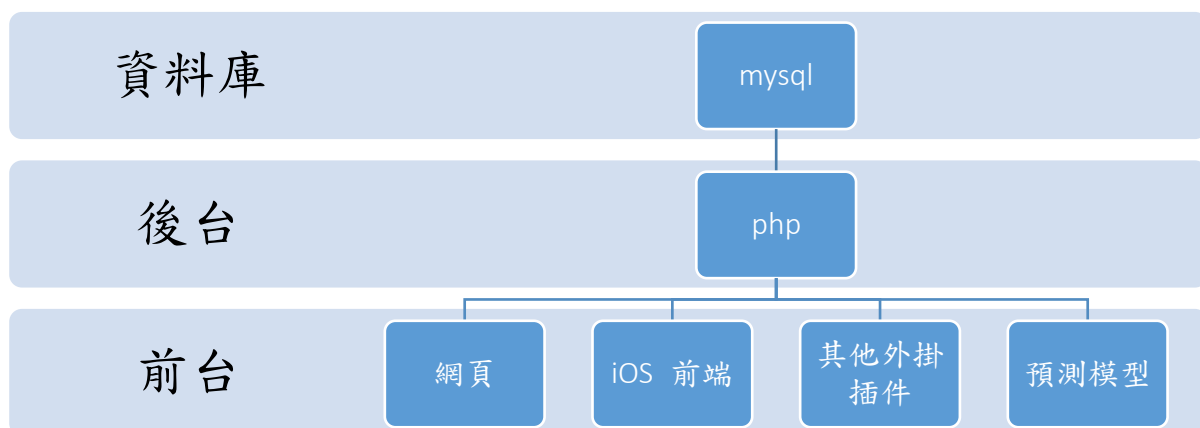
下圖為預測模型的輸出，與前三天的資料進行對比，每一種顏色代表一天的資料，每一個長條代表一種類別的數量，虛線代表預測值。



我們可以看到數量模型預測的總數與實際資料相差不遠，而比例模型預測的比例與實際資料相較平緩，因為比例模型取得鄰接機率矩陣時取平均，所以比例模型輸出值較不突出。

伍、目前研究結果

下圖為午餐系統的架構圖。



一、午餐系統

本系統在板橋高中已經成功推行，多數使用者皆有正面評價，系統總計有三萬筆訂單，約有一千個活躍使用者。

使用系統前，點餐採不記名制，少數學生會忘記自己點了什麼，於是亂拿別人的餐；使用系統後，點餐採實名制，亂拿別人的餐很容易就會被抓到，在系統上也能看到自己點了什麼餐，避免再有學生亂拿別人的餐。

總表對廠商而言是流程上的革新，總表的發明使得每個員工的平均效率大幅提升；電子化帳本省去了廠商手工對帳的麻煩，而且系統產生的電子帳本比手工對帳更為精確；原先廠商跟學生收錢需要開兩個窗口，各收半個小時，使用系統後只需要開一個窗口，收十分鐘就結束了，不只節省了學生的時間，也節省了廠商的時間。

二、預測模型

預測模型能夠給予廠商一個判斷依據，廠商的供應量一定不能少於學生的需求量，但是準備過量餐點只會造成浪費食材，如何取捨一直都是廠商的一大難題。

預測模型給給予了廠商一個量化、有根據的判斷方式，純憑第六感估計該準備多少餐不夠精確，使用數學模型能夠給予一個量化，而且有根據的預測值，方便廠商預測明天該準備多少餐，避免製作過量餐點導致廚餘浪費，也降低廠商的食材成本。

陸、參考資料及其他

板橋高中資訊培訓講義

StackOverflow

Logistic regression <https://blog.csdn.net/SzM21C11U68n04vdcLmJ/article/details/78221784>

午餐系統 <http://dinnerSystem.ddns.net>