

PUN street Universal Access (PUA)

系統需求規格書

Software Requirements Specification (SRS)

Version: 1.0

姓名	學號	E-mail
郭丞軒	110590001	t110590001@ntut.org.tw
王煥竑	110590002	t110590002@ntut.org.tw
陸芳寰	110590007	t110590007@ntut.org.tw
陳姿安	110590017	t1105900017@ntut.org.tw
劉承翰	110590018	t110590018@ntut.org.tw
歐佳昀	110590450	t110590450@ntut.org.tw

Department of Computer Science & Information Engineering
National Taipei University of Technology

12/31/2023

目錄 (Table of Contents)

1 簡介 (Introduction)	4
1.1 目的 (Purpose)	4
1.2 系統名稱 (Identification)	4
1.3 概觀 (Overview)	5
1.4 符號描述 (Notation Description)	5
2 系統 (System)	6
2.1 系統描述 (System Description)	6
2.1.1 系統架構圖 (System Architecture Diagram)	6
2.2 操作概念 (Operational Concepts or User Stories)	7
2.3 功能性需求 (Functional Requirements)	7
2.4 資料需求 (Data Requirements)	8
2.5 非功能性需求 (Non-Functional Requirements)	8
2.5.1 效能需求 (Performance Requirements)	8
2.5.2 資安需求 (Security Requirements)	8
2.6 介面需求 (Interface Requirements)	8
2.6.1 使用者介面需求 (User Interfaces Requirements)	8
2.6.2 內部介面需求 (Internal Interface Requirements)	9
2.7 其他需求 (Other Requirements)	9
2.7.1 環境需求 (Environmental Requirement)	9
2.7.2 安裝需求 (Installation Requirement)	9
2.7.3 測試需求 (Test Requirements)	10
2.8 商業規則與限制 (Business Rules and Integrity Constrains)	10
3 Conceptual Design of the Database	11
3.1 Entity Relationship ER Model	11
3.2 Data Dictionary	12
4 Logical Database Schema	17
4.1 Schema of the Database	17
4.2 Snapshot of Tables	18
4.3 SQL Statements for Database Construction	22
4.3.1 Create Table	22
4.3.2 Insert Data	24
4.4 Estimation Size of Each table	28
4.5 Expected Database Operation	29
5 Functional Dependencies and Database Normalization	30
5.1 Functional Dependencies of the Database Schema	30
6 The User Guide of the System	32
6.1 System Installation Description	32
6.1.1 Environment	32
6.1.2 Installation	32
6.2 The User Manual of the System	33
6.2.1 user	33
6.2.2 customer	36

6.2.3	seller	40
6.2.4	admin	43
7	Suggestions on Database Turning	45
7.1	View	45
7.2	Index	45
8	Additional Queries and Views	46
8.1	First Query	46
8.2	Second Query	48
8.3	Third Query	49
9	Conclusions and Future Work	51
9.1	Conclusions	51
9.2	Future Work	52
10	References	53
11	Appendix	54

1 簡介 (Introduction)

1.1 目的 (Purpose)

光華商場旁的美食街道，俗稱「潘（p'un）街」，是北科學生平日不可或缺的食物來源。每到用餐時段，街道上總是充滿著排隊人潮。為了讓學生們能夠更方便的取得潘街的食物，我們將藉由在資料庫系統課程及過往程式語言及網路通訊的基礎知識，分工合作開發一個美食訂購及外送系統—「PUN street Universal Access」，簡稱 PUA。此系統可以讓使用者在網頁上瀏覽、搜尋、並訂購餐點，讓所有人都能夠方便、迅速的享用潘街上的美食。

本系統主要目標為：

1. 顧客可以瀏覽、搜索店家並訂購餐點
2. 顧客可以查詢購物車、訂單狀況
3. 顧客可以為商店評分
4. 顧客可以註冊商店成為商店管理者
5. 商店管理者可以管理、更新餐點品項
6. 商店管理者可以計算月份總銷量、餐點銷售統計資料
7. 商店管理者可以更新訂單狀況
8. 商店管理者可以更新、查詢和刪除各種折扣政策

1.2 系統名稱 (Identification)

本專案範圍包含建置下面主系統與各項子系統，主系統為：

- 線上訂餐系統 (Web-based Ordering system, WOS)

子系統為：

- 帳號管理子系統 (Account Management Subsystem, AMS)
- 商店註冊子系統 (Shop Register Subsystem, SRS)
- 商品管理子系統 (Commodity Management Subsystem, CMS)
- 折扣管理子系統 (Discount Management Subsystem, DMS)
- 訂單配送管理子系統 (Order and Logistics Management Subsystem, OLMS)
- 銷售分析子系統 (Sales Analysis System, SAS)
- 購物車子系統 (Shopping Cart Management Subsystem, SCMS)
- 商店評價管理子系統 (Comment and Score Management Subsystem, CSMS)
- 商品搜尋推薦管理子系統 (Search and Recommend Management Subsystem, SRMS)
- 歷史訂單子系統 (Historical Order Subsystem, HOS)
- 資料庫子系統 (Database Subsystem, DBS)

1.3 概觀 (Overview)

本系統採用前後端分離架構開發，以便團隊成員分工。我們採用 svelte 作為前端框架，後端利用 golang 撰寫，資料庫管理系統則使用 postgreSQL。

1.4 符號描述 (Notation Description)

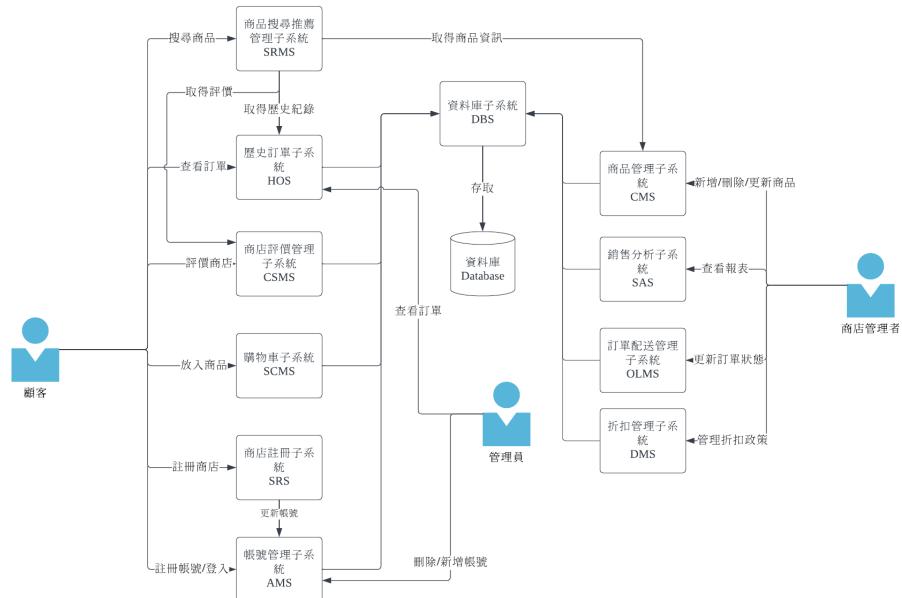
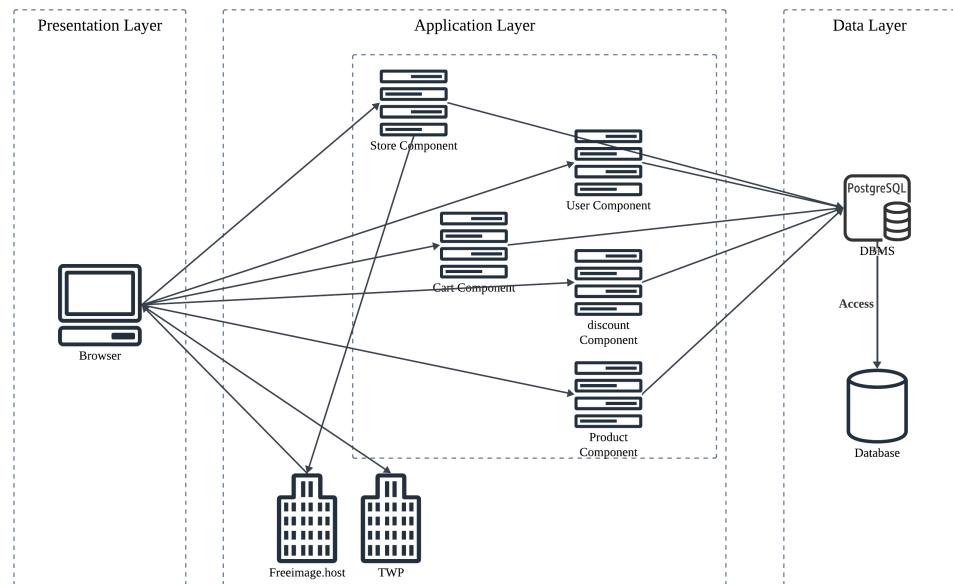
Notation	Description
WOS 1.0.0	The WOS components will be labeled with the number 1.0.0
AMS 1.1.n	The AMS components will be labeled with the number 1.1.n
SRS 1.2.1.n	The SRS components will be labeled with the number 1.2.n
CMS 1.3.n	The CMS components will be labeled with the number 1.3.n
DMS 1.4.n	The DMS components will be labeled with the number 1.4.n
OLMS 1.5.n	The OLMS components will be labeled with the number 1.5.n
SAS 1.6.n	The SAS components will be labeled with the number 1.6.n
SCMS 1.7.n	The SCMS components will be labeled with the number 1.7.n
CSMS 1.8.n	The CSMS components will be labeled with the number 1.8.n
SRMS 1.9.n	The SRMS components will be labeled with the number 1.9.n
HOS 1.10.n	The HOS components will be labeled with the number 1.10.n
DBS 1.11.n	The DBS components will be labeled with the number 1.11.n

Notation	Description
WOS-I-nnn	WOS 介面需求 (Interface Requirements)
WOS-F-nnn	WOS 功能性需求 (Functional Requirements)
WOS-D-nnn	WOS 資料需求 (Data Requirements)
WOS-N-nnn	WOS 非功能性需求 (Non-Functional Requirements)
WOS-O-nnn	WOS 其他需求 (Other Requirements)
WOS-B-nnn	WOS 商業規則或約束 (Business Rules or Constraints)

2 系統 (System)

2.1 系統描述 (System Description)

2.1.1 系統架構圖 (System Architecture Diagram)



2.2 操作概念 (Operational Concepts or User Stories)

- 顧客操作概念：

使用者可以透過帳號管理子系統 (AMS) 進行註冊，用商品搜尋推薦管理子系統 (SRMS) 尋找自己想要的商品，透過歷史訂單子系統 (HOS) 查詢自己的訂單歷史紀錄，並透過購物車子系統 (SCMS) 將商品放入購物車並購買商品。

- 商店管理者操作概念：

顧客可以透過商店註冊系統 (SRS) 進行註冊成為商店管理者，用商品管理子系統 (CMS) 管理商品，用銷售分析子系統 (SAS) 產生財務報表，並透過訂單配送管理子系統 (OLMS) 管理顧客訂單。

- 管理員操作概念：

管理員可以透過帳號管理子系統 (AMS) 管理所有帳號，透過歷史訂單子系統 (HOS) 查看所有訂單紀錄。

2.3 功能性需求 (Functional Requirements)

Notation	Description
WOS-F-001	顧客可以註冊帳號。
WOS-F-002	顧客可以搜尋商店及商品。
WOS-F-003	顧客可以將商品加入購物車。
WOS-F-004	顧客可以結帳購物車內的商品。
WOS-F-005	顧客可以查詢訂單狀態。
WOS-F-006	顧客可以在訂單完成後為商店或商品評分。
WOS-F-007	顧客可以查詢訂購歷史。
WOS-F-008	顧客可以註冊商店成為商店管理員。
WOS-F-009	商店管理者可以增加、刪減商品品項及數量。
WOS-F-010	商店管理者可以查詢日、周、月銷售量及金額。
WOS-F-011	商店管理者可以更新訂單狀況。
WOS-F-012	商店管理者可以查詢、增加及刪減各項折扣方案。
WOS-F-013	管理員可以新增、刪除、更新帳號。
WOS-F-014	管理員可以查看所有訂單。

2.4 資料需求 (Data Requirements)

需求編號	需求描述
WOS-D-001	系統需儲存使用者的相關資訊：登入 ID、姓名、密碼、電子郵件、地址
WOS-D-002	系統需儲存產品的相關資訊：產品 ID、名稱描述、其他屬性、產品圖片、價格、庫存產品數量、標籤
WOS-D-003	系統需儲存商店的相關資訊：商店 ID、名稱、地址、聯絡電子郵件、電話
WOS-D-004	系統需儲存訂單的相關資訊：訂單 ID、時間、客戶、每個採購項目的名稱和價格、每個採購項目的數量、採購總額、訂單的當前狀態（已接收、處理、運輸、關閉）
WOS-D-005	系統需儲存銷售報告的相關資訊：日期、商店名稱、訂單數量、銷售總額

2.5 非功能性需求 (Non-Functional Requirements)

2.5.1 效能需求 (Performance Requirements)

需求編號	需求描述
WOS-N-001	網頁載入時間需不超過五秒
WOS-N-002	用戶搜尋時間需不超過五秒
WOS-N-003	商店管理員更新訂單及商店狀況至用戶端反映需不超過三十秒

2.5.2 資安需求 (Security Requirements)

需求編號	需求描述
WOS-N-004	註冊密碼必須符合複雜度需求
WOS-N-005	管理員應遵守隱私權保護政策，不可隨意檢視個人資料

2.6 介面需求 (Interface Requirements)

2.6.1 使用者介面需求 (User Interfaces Requirements)

買家介面

需求編號	需求描述
WOS-I-001	搜尋食物 or 店家種類介面。
WOS-I-002	購物清單介面
WOS-I-003	點餐介面。
WOS-I-004	訂單配送管理界面。

商家介面

需求編號	需求描述
WOS-I-005	刊登商品資訊。
WOS-I-006	檢視買家訂單
WOS-I-007	管理刊登商品介面。
WOS-I-008	折扣管理介面。
WOS-I-009	檢視銷售報表介面。

2.6.2 內部介面需求 (Internal Interface Requirements)

需求編號	需求描述
WOS-I-010	SRMS 能向 CMS 接收商品資訊需求。
WOS-I-011	SRMS 能向 HOS 接收歷史訂單需求。
WOS-I-012	SRMS 能向 CSMS 接收評價需求。
WOS-I-013	HOS 與 DBS 需有傳送與接收歷史訂單需求。
WOS-I-014	CSMS 與 DBS 需有傳送與接收商店評價需求。
WOS-I-015	SCMS 與 DBS 需有傳送與接收購物車相關資訊需求。
WOS-I-016	SRS 能向 AMS 傳送更新帳號的需求。
WOS-I-017	AMS 與 DBS 需有傳送與接收帳號資訊需求。
WOS-I-018	CMS 與 DBS 需有傳送與接收商品資訊的需求。
WOS-I-019	SAS 能向 DBS 傳送取得銷售分析需求。
WOS-I-020	OLMS 能向 DBS 傳送更新訂單需求。
WOS-I-021	DMS 與 DBS 需有傳送與接收優惠資訊需求。

2.7 其他需求 (Other Requirements)

2.7.1 環境需求 (Environmental Requirement)

需求編號	需求說明
WOS-O-001	需要網路
WOS-O-002	docker
WOS-O-003	中子
WOS-O-004	電子
WOS-O-005	質子 (不會離開原子)

2.7.2 安裝需求 (Installation Requirement)

需求編號	需求說明
WOS-O-006	使用 docker compose 自動建置

2.7.3 測試需求 (Test Requirements)

需求編號	需求說明
WOS-O-007	所有子系統都應經過測試且無問題

2.8 商業規則與限制 (Business Rules and Integrity Constraints)

需求編號	需求說明
WOS-B-001	同一次購買可以應用三種類型的折扣，包括運費季節和特別活動折扣。
WOS-B-002	一種產品不能與多種特殊活動類型的折扣相關聯
WOS-B-003	產品的同一特殊活動折扣期間不能重疊。
WOS-B-004	購買的產品的數量必須是大於零的整數。
WOS-B-005	採購數量必須小於或等於該產品的當前庫存數量
WOS-B-006	使用者可以有多個身份
WOS-B-007	資料庫初始狀態要所有員工和系統管理員的資訊

3 Conceptual Design of the Database

3.1 Entity Relationship ER Model

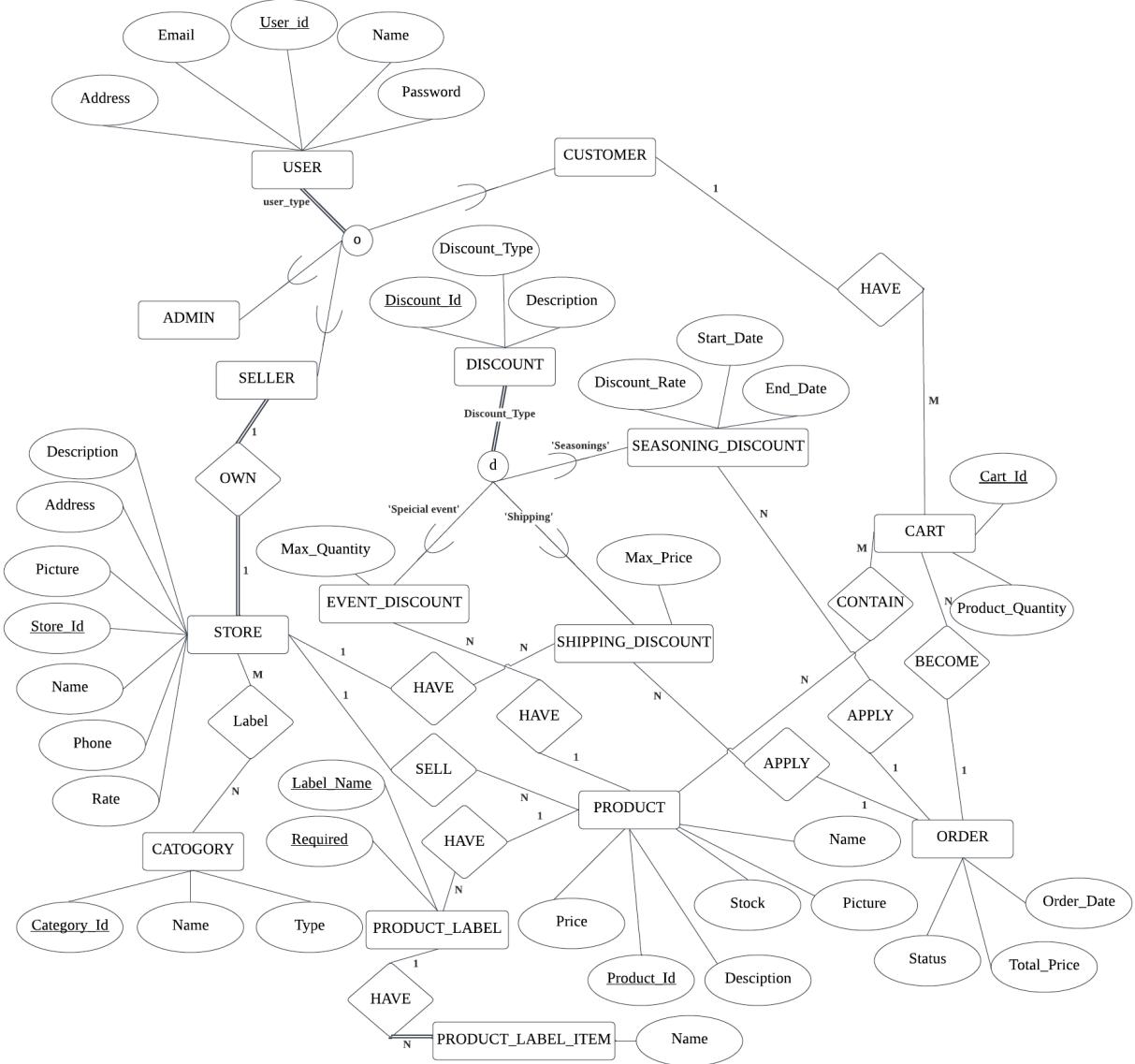


Figure 1: ER diagram

3.2 Data Dictionary

USER_DATA				
Description: 使用者相關資料				
Attribute	Type	Key	Nullable	Description
user_id	SERIAL	Primary	Not Null	使用者 ID
name	VARCHAR		Not Null	使用者名稱
email	VARCHAR		Not Null	使用者信箱
address	VARCHAR		Not Null	使用者地址
password	VARCHAR		Not Null	使用者密碼
authority	INTEGER		Not Null	使用者權限 (admin, seller, user)
phone_number	VARCHAR		Not Null	使用者電話號碼
current_cart_id	INTEGER		Not Null	使用者目前購物車
status	INTEGER		Not Null	使用者狀態 (0= 停用, 1= 啟用)
birthday	DATE		Not Null	使用者生日

CARTS				
Description: 購物車相關資料				
Attribute	Type	Key	Nullable	Description
customer_id	SERIAL	Primary Foreign	Not Null	使用者 ID
product_id	SERIAL	Primary Foreign	Not Null	商品 ID
cart_id	INTEGER	Primary	Not Null	購物車 ID
product_quantity	INTEGER		Not Null	商品數量
event_discount_id	SERIAL	Foreign	Not Null	折價券 ID
store_id	SERIAL	Primary Foreign	Not Null	購買商店 ID

PRODUCTS				
Description: 商品相關資料				
Attribute	Type	Key	Nullable	Description
product_id	SERIAL	Primary Foreign	Not Null	商品 ID
name	VARCHAR		Not Null	商品名字
description	TEXT		Not Null	商品資訊
store_id	SERIAL	Foreign	Not Null	商店 ID
picture	STRING		Not Null	商品照片
price	INTEGER		Not Null	商品價錢
stock	INTEGER		Not Null	商品庫存
status	INTEGER		Not Null	商品狀態 (0= 停用, 1= 啟用, 2= 已售完)

STORES				
Description: 商店相關資料				
Attribute	Type	Key	Nullable	Description
store_id	SERIAL	Primary	Not Null	商品 ID
name	VARCHAR		Not Null	商店名字
rate	FLOAT		Not Null	商店評價
picture	STRING		Not Null	商店照片
address	VARCHAR		Not Null	商店地址
description	TEXT		Not Null	商店描述
shipping_fee	INTEGER		Not Null	運費
status	INTEGER		Not Null	商店狀態 (0= 停用, 1= 啟用)
rate_count	INTEGER		Not Null	評價數量

LABEL				
Description: 商店標籤資料				
Attribute	Type	Key	Nullable	Description
store_id	SERIAL	Primary Foreign	Not Null	商店 ID
category_id	SERIAL	Primary Foreign	Not Null	目錄 ID

CATEGORIES				
Description: 類別資料				
Attribute	Type	Key	Nullable	Description
category_id	SERIAL	Primary	Not Null	類別 ID
name	VARCHAR		Not Null	類別名稱

EVENT_DISCOUNT				
Description: 事件折扣資料				
Attribute	Type	Key	Nullable	Description
discount_id	SERIAL	Primary	Not Null	折扣 ID
max_quantity	INTEGER		Not Null	買 x 送一
product_id	SERIAL	Foreign	Not Null	商品 ID

SHIPPING_DISCOUNT				
Description: 運費折扣資料				
Attribute	Type	Key	Nullable	Description
discount_id	SERIAL	Primary	Not Null	折扣 ID
max_price	INTEGER		Not Null	免運金額
store_id	SERIAL	Foreign	Not Null	商店 ID

DISCOUNTS				
Description: 折扣資料				
Attribute	Type	Key	Nullable	Description
discount_id	SERIAL	Primary	Not Null	折扣 ID 1= 無折扣
discount_type	INTEGER		Not Null	折扣種類 0= 無折扣 1= 運費折扣 2= 季節性折扣 3= 事件折扣
status	INTEGER		Not Null	狀態
description	VARCHAR		Not Null	折價資訊
name	VARCHAR		Not Null	折扣名稱

ORDER				
Description: 訂單				
Attribute	Type	Key	Nullable	Description
order_date	DATE		Not Null	訂單日期
total_price	INTEGER		Not Null	總金額
seasoning_discount_id	SERIAL	Foreign	Not Null	季節折扣 ID
shipping_discount_id	SERIAL	Foreign	Not Null	運費折扣 ID
cart_id	SERIAL	Primary Foreign	Not Null	購物車 ID
customer_id	SERIAL	Primary Foreign	Not Null	顧客 ID
taking_method	INTEGER		Not Null	0= 自取 1= 外送
store_id	SERIAL	Primary Foreign	Not Null	商店 ID
taking_address	VARCHAR		Not Null	運送地址
status	INTEGER		Not Null	訂單狀態 0= 未結帳 1= 已結帳 2= 接受訂單 3= 製作訂單 4= 運送中 5= 已送達 6 = 確認訂單 7= 已評分

SEASONING_DISCOUNT				
Description: 季節折扣資料				
Attribute	Type	Key	Nullable	Description
discount_id	SERIAL	Primary	Not Null	折扣 ID
start_date	DATE		Not Null	開始日期
end_date	DATE		Not Null	結束日期
store_id	SERIAL	Foreign	Not Null	商店 ID
discount_percentage	INTEGER		Not Null	折扣比例 (70=30%off)

PRODUCT_LABEL				
Description: 商品標籤				
Attribute	Type	Key	Nullable	Description
product_id	SERIAL	Primary Foreign	Not Null	商品 ID
label_name	VARCHAR	Primary Foreign	Not Null	類別名稱
required	BOOLEAN		Not Null	是否為必填

PRODUCT_LABEL_ITEM				
Description: 商品標籤選項				
Attribute	Type	Key	Nullable	Description
product_id	SERIAL	Primary Foreign	Not Null	商品 ID
label_name	VARCHAR	Primary Foreign	Not Null	類別名稱
name	VARCHAR	Primary	Not Null	選項名稱

4 Logical Database Schema

4.1 Schema of the Database

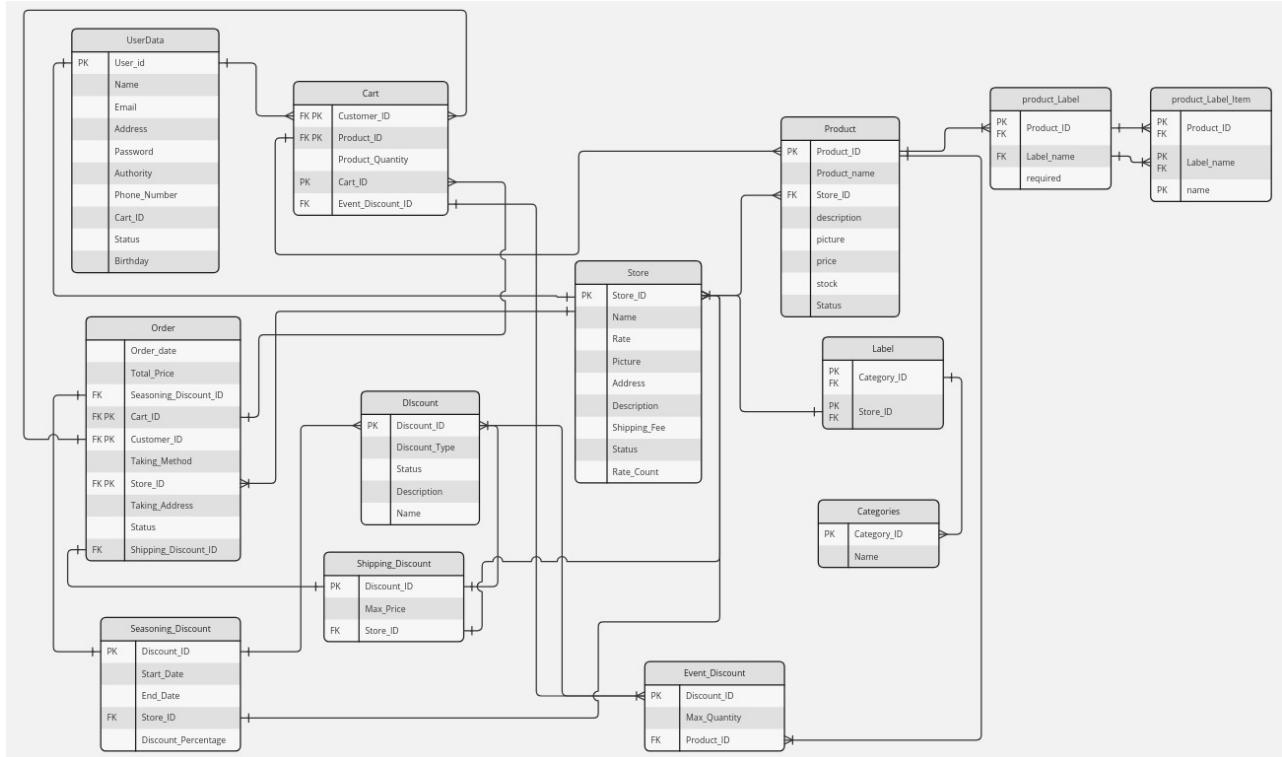


Figure 2: database schema

4.2 Snapshot of Tables

		* user_id	* name	* password	* email	* address	* phone_number	* birthday	* authority	* current_cart_id	* status
		integer	varchar(32)	varchar(32)	varchar(64)	varchar(64)	varchar(16)	date	integer	integer	integer
1	1	admin	admin	admin@gmail.com	address_admin	9000-0000	2021-10-10	111	1	1	1
2	2	alice	pwd1	a@gmail.com	address_1	9000-1111	2021-10-10	1	1	1	1
3	3	bob	pwd2	b@gmail.com	address_2	9000-2222	2022-11-11	10	2	1	1
4	4	tcp	pwd3	c@gmail.com	address_3	9000-3333	2023-12-12	1	1	1	1
5	5	udp	pwd4	d@gmail.com	address_4	9000-4444	2024-01-01	11	1	1	1
6	6	icmp	pwd5	e@gmail.com	address_5	9000-5555	2025-02-02	1	1	1	1
7	7	dns	pwd6	f@gmail.com	address_6	9000-6666	2026-03-03	1	1	1	1
8	8	dhcp	pwd7	g@gmail.com	address_7	9000-7777	2027-04-04	1	1	1	1

Figure 3: user_data

		* store_id	* name	* rate	* rate_count	* address	* picture	* description	* shipping_fee	* status
		integer	varchar(255)	double precis	integer	varchar(255)	varchar(255)	text	integer	integer
1	1	im pasta	4	100	pun street	https://i.imgur.com/3i3tyXJ.	you are not pasta	35	1	1
2	3	number five	3.5	5	pun street	https://i.imgur.com/3i3tyXJ.	special meal	15	1	1
3	2	mos burger	4	123	NTUT	https://i.imgur.com/3i3tyXJ.	mooooooooooooos	55	1	1
4	7	trash noodle	4.9	9876	pun street	https://i.imgur.com/3i3tyXJ.	trash	192	1	1

Figure 4: stores

		* customer_id	* product_id	* store_id	* cart_id	* product_quantity	* event_discount_id	
		integer	integer	integer	integer	integer	integer	
1	1	2	1	1	1	100	8	
2	1	1	1	1	1	100	9	
3	1	2	1	2	2	33	10	
4	3	3	2	1	1	100	1	
5	1	1	1	3	3	7	1	

Figure 5: carts

category

	category_id	name
1	1	Spices
2	2	Sustainable
3	3	Fresh
4	4	Bakery
5	5	Local
6	6	sus thing

Figure 6: categories

products

	product_id	store_id	name	description	picture	price	stock	status	
1	1	1	pasta	tasty pasta	https://i.imgur.com/3i3tyXJ.	150	100	1	
2	2	1	black tea	student card get free	https://i.imgur.com/3i3tyXJ.	0	100	1	
3	3	2	special1	special	https://i.imgur.com/3i3tyXJ.	250	2	1	
4	4	2	special2	special	https://i.imgur.com/3i3tyXJ.	350	2	0	
5	5	2	special3	special	https://i.imgur.com/3i3tyXJ.	450	2	1	
6	6	3	black tea	most famous	https://i.imgur.com/3i3tyXJ.	300	3	1	
7	7	3	rice burger	we have rice	https://i.imgur.com/3i3tyXJ.	500	3	2	
8	8	7	big trash	can add kimchi for free	https://i.imgur.com/3i3tyXJ.	80	3	1	
9	9	7	small trash	can add kimchi for free	https://i.imgur.com/3i3tyXJ.	60	3	1	

Figure 7: products

discounts

	discount_id	discount_type	status	description	name
1	1	0	1	no discount	default
2	2	1	1	all products 90% off	spring discount
3	3	1	1	all products 80% off	spring discount
4	4	1	1	all products 70% off	spring discount
5	5	2	1	free shipping when price over 500	spring discount
6	6	2	1	free shipping when price over 300	spring discount
7	7	2	1	free shipping when price over 100	spring discount
8	8	3	1	buy 3 get 1 free	halloween
9	9	3	1	buy 2 get 1 free	halloween
10	10	3	1	buy 1 get 1 free	halloween

Figure 8: discounts

shipping_discount

	discount_id	max_price	store_id
1	5	1000	1
2	6	500	1
3	7	300	1

Figure 9: shipping_discount

event_discount

	discount_id	max_quantity	product_id
1	8	3	1
2	9	3	2
3	10	3	3

Figure 10: event_discount

seasoning_discount

	discount_id	discount_poco	start_date	end_date
1	2	90	2021-01-01	2021-01-02
2	3	80	2021-01-03	2021-01-04
3	4	70	2021-01-05	2021-01-05

Figure 11: seasoning_discount

orders

	cart_id	store_id	user_id	seasoning_discount_id	shipping_discount_id	status	total_price	order_date	taking_address
1	1	1	1	1	1	1	100	2020-01-01 00:00:00	台北市
2	2	1	1	2	5	1	100	2020-02-01 00:00:00	台北市
3	1	2	3	3	6	1	100	2020-03-01 00:00:00	台北市
4	3	1	1	4	7	1	100	2020-04-01 00:00:00	台北市

Figure 12: orders

product_label

	product_id	label_name	required
1	2	sugar	true
2	7	rice	true
3	8	kimchi	false
4	9	kimchi	false

Figure 13: product_label

label_item

	product_id	label_name	item_na
1	2	sugar	more
2	2	sugar	less
3	2	sugar	no
4	7	rice	yes
5	7	rice	no
6	8	kimchi	yes
7	8	kimchi	no
8	9	kimchi	yes
9	9	kimchi	no

Figure 14: label_item

The screenshot shows a PostgreSQL pgAdmin interface with a dark theme. The title bar says "labels". The main area displays a table with two columns: "category_id" and "store_id", both of type integer. The table contains 6 rows:

	category_id	store_id
1	1	1
2	2	1
3	3	1
4	2	2
5	5	7
6	6	7

Figure 15: labels

4.3 SQL Statements for Database Construction

4.3.1 Create Table

```
-- user_data
CREATE TABLE
    IF NOT EXISTS user_data(
        user_id SERIAL PRIMARY KEY,
        name VARCHAR(32) NOT NULL,
        password VARCHAR(32) NOT NULL,
        email VARCHAR(64) UNIQUE NOT NULL,
        address VARCHAR(64) NOT NULL,
        phone_number VARCHAR(16) NOT NULL,
        birthday DATE NOT NULL,
        authority INTEGER NOT NULL,
        current_cart_id INTEGER NOT NULL,
        status INTEGER NOT NULL
    );
```

```
-- stores
CREATE TABLE
    IF NOT EXISTS stores (
        store_id SERIAL PRIMARY KEY REFERENCES user_data(user_id),
        name VARCHAR(255) NOT NULL,
        rate FLOAT NOT NULL,
        rate_count INTEGER NOT NULL,
        address VARCHAR(255) NOT NULL,
        picture VARCHAR(255) NOT NULL,
        description TEXT NOT NULL,
        shipping_fee INTEGER NOT NULL,
        status INTEGER NOT NULL
    );
```

```
-- products
CREATE TABLE
    IF NOT EXISTS products (
        product_id SERIAL PRIMARY KEY,
        store_id SERIAL REFERENCES stores(store_id),
        name VARCHAR(255) NOT NULL,
        description TEXT NOT NULL,
        picture VARCHAR(255) NOT NULL,
        price INTEGER NOT NULL,
        stock INTEGER NOT NULL,
        status INTEGER NOT NULL
    );
```

```
-- discounts
CREATE TABLE
    IF NOT EXISTS discounts (
        discount_id SERIAL PRIMARY KEY UNIQUE,
```

```
    discount_type INT NOT NULL,  
    status INT NOT NULL,  
    description TEXT,  
    name VARCHAR(255) NOT NULL  
);
```

```
-- seasoning_discount  
CREATE TABLE  
    IF NOT EXISTS seasoning_discount (  
        discount_id SERIAL UNIQUE REFERENCES discounts(discount_id),  
        discount_percentage INT NOT NULL,  
        start_date DATE NOT NULL,  
        end_date DATE NOT NULL  
);
```

```
-- shopping_discount  
CREATE TABLE  
    IF NOT EXISTS shipping_discount (  
        discount_id SERIAL UNIQUE REFERENCES discounts(discount_id),  
        max_price INT NOT NULL,  
        store_id SERIAL REFERENCES stores(store_id)  
);
```

```
-- event_discount  
CREATE TABLE  
    IF NOT EXISTS event_discount (  
        discount_id SERIAL UNIQUE REFERENCES discounts(discount_id),  
        max_quantity INT NOT NULL,  
        product_id SERIAL REFERENCES products(product_id)  
);
```

```
-- carts  
CREATE TABLE  
    IF NOT EXISTS carts (  
        customer_id SERIAL REFERENCES user_data(user_id),  
        product_id SERIAL REFERENCES products(product_id),  
        store_id SERIAL REFERENCES stores(store_id),  
        cart_id INTEGER NOT NULL,  
        product_quantity INT NOT NULL,  
        event_discount_id SERIAL REFERENCES discounts(discount_id),  
        PRIMARY KEY (  
            customer_id,  
            product_id,  
            store_id,  
            cart_id  
        )  
    );
```

```
-- orders
CREATE TABLE
orders (
    cart_id INTEGER,
    store_id INTEGER REFERENCES stores(store_id),
    user_id INTEGER REFERENCES user_data(user_id),
    seasoning_discount_id INTEGER REFERENCES discounts(discount_id),
    shipping_discount_id INTEGER REFERENCES discounts(discount_id),
    status INTEGER,
    total_price INTEGER,
    Order_date TIMESTAMP,
    taking_address VARCHAR(255),
    taking_method INTEGER,
    PRIMARY KEY (cart_id, store_id, user_id)
);
```

```
-- categories
CREATE TABLE
IF NOT EXISTS categories (
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE NOT NULL
);
```

```
-- product_label
CREATE TABLE
IF NOT EXISTS product_label (
    product_id SERIAL REFERENCES products(product_id),
    label_name VARCHAR(255) NOT NULL,
    required BOOLEAN NOT NULL,
    PRIMARY KEY (product_id, label_name)
);
```

```
-- label_item
CREATE TABLE
IF NOT EXISTS label_item (
    product_id SERIAL NOT NULL,
    label_name VARCHAR(255) NOT NULL,
    item_name VARCHAR(255) NOT NULL,
    FOREIGN KEY (product_id, label_name) REFERENCES
        product_label(product_id, label_name),
    PRIMARY KEY (product_id, label_name, item_name)
);
```

4.3.2 Insert Data

```
-- user_data
INSERT INTO user_data (name, password, email, address, phone_number,
    birthday, authority, current_cart_id, status) VALUES
```

```

('admin', 'admin', 'admin@gmail.com', 'address_admin', '0900-0000',
 '2021-10-10 11:30:30', 111, 1, 1),
('alice', 'pwd1', 'a@gmail.com', 'address_1', '0900-1111',
 '2021-10-10 11:30:30', 001, 1, 1),
('bob', 'pwd2', 'b@gmail.com', 'address_2', '0900-2222', '2022-11-11
 11:30:30', 010, 2, 1),
('tcp', 'pwd3', 'c@gmail.com', 'address_3', '0900-3333', '2023-12-12
 11:30:30', 001, 1, 1),
('udp', 'pwd4', 'd@gmail.com', 'address_4', '0900-4444', '2024-01-01
 11:30:30', 011, 1, 1),
('icmp', 'pwd5', 'e@gmail.com', 'address_5', '0900-5555',
 '2025-02-02 11:30:30', 001, 1, 1),
('dns', 'pwd6', 'f@gmail.com', 'address_6', '0900-6666', '2026-03-03
 11:30:30', 001, 1, 1),
('dhcp', 'pwd7', 'g@gmail.com', 'address_7', '0900-7777',
 '2027-04-04 11:30:30', 001, 1, 1);

```

```

-- stores
INSERT INTO stores (store_id, name, rate, rate_count, address, picture,
description, shipping_fee, status) VALUES
(1, 'im pasta', 4, 100, 'pun street',
'https://i.imgur.com/3i3tyXJ.gif', 'you are not pasta', 35, 1),
(3, 'number five', 3.5, 5, 'pun street',
'https://i.imgur.com/3i3tyXJ.gif', 'special meal', 15, 1),
(2, 'mos burger', 4, 123, 'NTUT', 'https://i.imgur.com/3i3tyXJ.gif',
'moooooooooooooos', 55, 1),
(7, 'trash noodle', 4.9, 9876, 'pun street',
'https://i.imgur.com/3i3tyXJ.gif', 'trash', 192, 1);

```

```

-- products
INSERT INTO products (store_id, name, description, picture, price,
stock, status) VALUES
(1, 'pasta', 'tasty pasta', 'https://i.imgur.com/3i3tyXJ.gif', 150,
100, 1),
(1, 'black tea', 'student card get free',
'https://i.imgur.com/3i3tyXJ.gif', 0, 100, 1),
(2, 'special1', 'special', 'https://i.imgur.com/3i3tyXJ.gif', 250,
2, 1),
(2, 'special2', 'special', 'https://i.imgur.com/3i3tyXJ.gif', 350,
2, 0),
(2, 'special3', 'special', 'https://i.imgur.com/3i3tyXJ.gif', 450,
2, 1),
(3, 'black tea', 'most famous', 'https://i.imgur.com/3i3tyXJ.gif',
300, 3, 1),
(3, 'rice burger', 'we have rice',
'https://i.imgur.com/3i3tyXJ.gif', 500, 3, 2),
(7, 'big trash', 'can add kimchi for free',
'https://i.imgur.com/3i3tyXJ.gif', 80, 3, 1),
(7, 'small trash', 'can add kimchi for free',
'https://i.imgur.com/3i3tyXJ.gif', 60, 3, 1);

```

```
-- discounts
INSERT INTO discounts (discount_type, description, name, status) VALUES
(0, 'no discount', 'default', 1),
(1, 'all products 90% off', 'spring discount', 1),
(1, 'all products 80% off', 'spring discount', 1),
(1, 'all products 70% off', 'spring discount', 1),
(2, 'free shipping when price over 1000', 'spring discount', 1),
(2, 'free shipping when price over 500', 'spring discount', 1),
(2, 'free shipping when price over 300', 'spring discount', 1),
(3, 'buy 3 get 1 free', 'halloween', 1),
(3, 'buy 2 get 1 free', 'halloween', 1),
(3, 'buy 1 get 1 free', 'halloween', 1);
```

```
-- seasoning_discount
INSERT INTO seasoning_discount (discount_id, discount_percentage,
start_date, end_date) VALUES
(2,90,'2021-01-01 00:00:00','2021-01-02 00:00:00'),
(3,80,'2021-01-03 00:00:00','2021-01-04 00:00:00'),
(4,70,'2021-01-05 00:00:00','2021-01-05 00:00:00');
```

```
-- shipping_discount
INSERT INTO shipping_discount (discount_id, max_price, store_id) VALUES
(5,1000,1),
(6,500,1),
(7,300,1);
```

```
-- event_discount
INSERT INTO event_discount (discount_id, max_quantity, product_id)
VALUES
(8,3,1),
(9,3,2),
(10,3,3);
```

```
-- carts
INSERT INTO carts (customer_id, cart_id, store_id, product_id,
product_quantity, event_discount_id) VALUES
(1, 1, 1, 2, 100, 8),
(1, 1, 1, 1, 100, 9),
(1, 2, 1, 2, 33, 10),
(3, 1, 2, 3, 100, 1),
(1, 3, 1, 1, 7, 1);
```

```
-- orders
INSERT INTO orders (user_id, cart_id, store_id, seasoning_discount_id,
shipping_discount_id, status, total_price, Order_date,
taking_address, taking_method) VALUES
(1, 1, 1, 1, 1, 100, '2020-01-01 00:00:00', '台北市', 1),
```

```
(1, 2, 1, 2, 5, 1, 100, '2020-02-01 00:00:00', '台北市', 1),
(3, 1, 2, 3, 6, 1, 100, '2020-03-01 00:00:00', '台北市', 1),
(1, 3, 1, 4, 7, 1, 100, '2020-04-01 00:00:00', '台北市', 1);
```

```
-- categories
INSERT INTO categories (name) VALUES
('Spices'),
('Sustainable'),
('Fresh'),
('Bakery'),
('Local'),
('sus thing');
```

```
-- labels
INSERT INTO labels (category_id, store_id) VALUES
(1, 1),
(2, 1),
(3, 1),
(2, 2),
(5, 7),
(6, 7);
```

```
-- product_label
INSERT INTO product_label (product_id, label_name, required) VALUES
(2, 'sugar', true),
(7, 'rice', true),
(8, 'kimchi', false),
(9, 'kimchi', false);
```

```
-- label_item
INSERT INTO label_item (product_id, label_name, item_name) VALUES
(2, 'sugar', 'more'),
(2, 'sugar', 'less'),
(2, 'sugar', 'no'),
(7, 'rice', 'yes'),
(7, 'rice', 'no'),
(8, 'kimchi', 'yes'),
(8, 'kimchi', 'no'),
(9, 'kimchi', 'yes'),
(9, 'kimchi', 'no');
```

4.4 Estimation Size of Each table

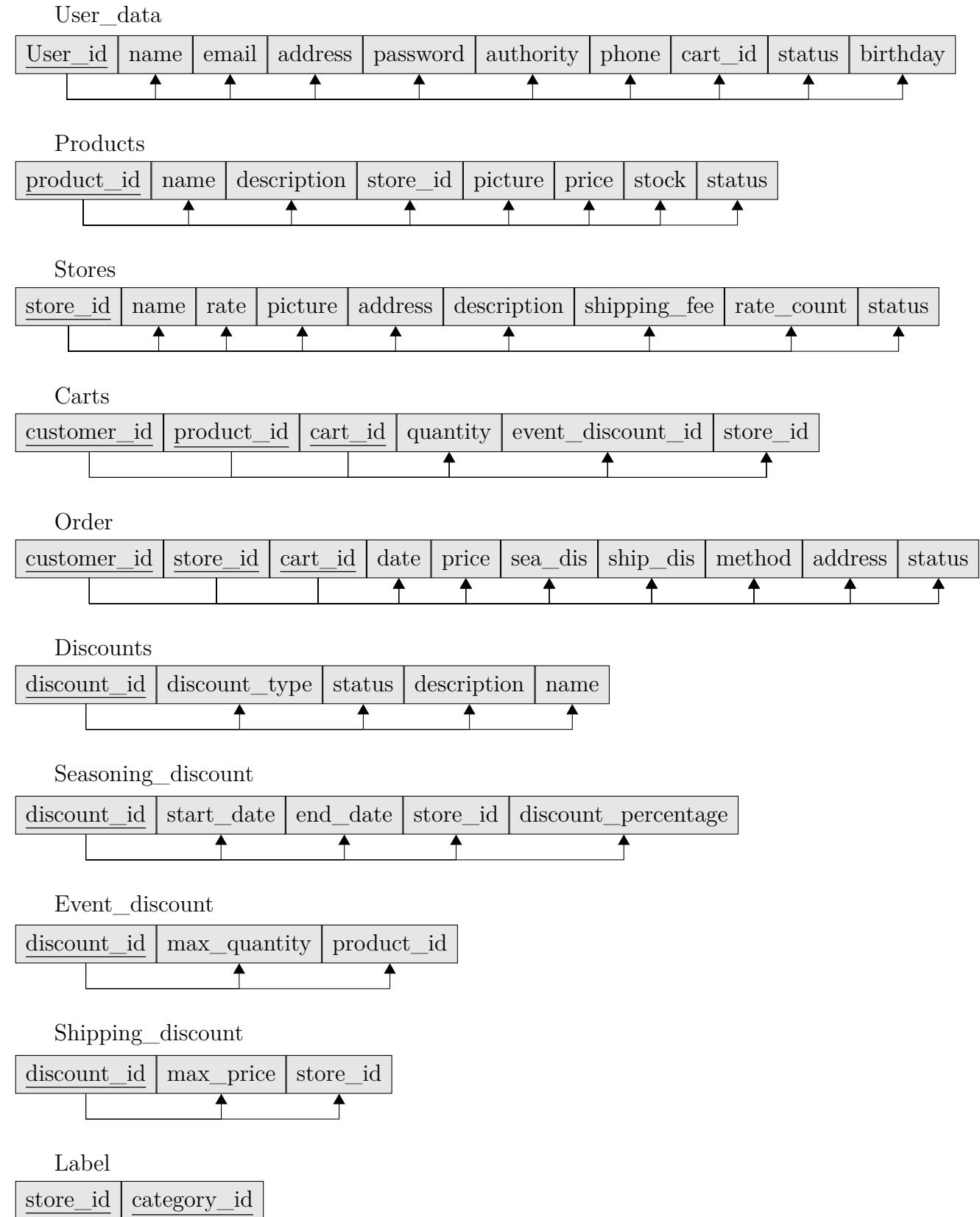
Table	Tuple Size (bytes)	Average Tuple Count	Estimate Size (kb)
user_data	92	300	27.6
stores	120	30	3.6
products	110	600	66
carts	50	4000	200
discounts	70	250	17.5
seasoning_discount	40	20	0.8
shipping_discount	40	30	1.2
event_discount	40	200	8
orders	80	1500	120
categories	40	10	0.4
labels	35	120	42
product_label	35	400	14
label_item	40	1200	48

4.5 Expected Database Operation

Table	可能操作	頻率 (次數/天)
user_data	登入	150
	註冊	5
stores	新增商店	2
	查看商店資料	200
	為商店評分	20
	編輯商店	2
products	新增商品	2
	查看商品資料	60
	編輯商品	10
carts	加入購物車	250
	結帳	90
	查看購物車	100
discounts	新增折扣	9
	編輯折扣	3
	查看折扣	100
seasoning_discounts	新增折扣	3
	編輯折扣	1
shipping_discount	新增折扣	3
	編輯折扣	1
event_discount	新增折扣	3
	編輯折扣	1
orders	查看訂單	100
	更新訂單	400
categories	查看類別	50
labels	編輯店家類別	2
product_label	新增商品類別	10
	編輯商品類別	2
label_item	新增商品類別品項	30
	編輯商品類別品項	10

5 Functional Dependencies and Database Normalization

5.1 Functional Dependencies of the Database Schema



Categories

category_id	name

Product_label

product_id	label_name	required

Product_label_item

product_id	label_name	name

All relations are in BCNF.

6 The User Guide of the System

6.1 System Installation Description

6.1.1 Environment

1. Windows, Linux
2. Docker

6.1.2 Installation

1. clone project repository [github](#)
2. run command `cp frontend/.example.env frontend/.env`
3. run command `docker-compose up --build`
4. go to `http://localhost:5050`

6.2 The User Manual of the System

6.2.1 user

user can register seller or customer

A screenshot of a mobile application interface showing a form for basic user information. The form is divided into two main sections: "Choose User Type" on the left and "Complete basic information" on the right. The "Complete basic information" section contains the following fields:

- Name: jesus
- Email: god@judge.me
- Password: (redacted)
- Password Check: (redacted)
- Phone Number: 090042069
- Address: paradise

A checkbox labeled "Do you be a good PUA user?" is checked. At the bottom of the form is a red "Next Step" button.

Choose User Type

Complete basic information

Complete Store Info

Complete!



Store Name
Rabbit House

Store Description
coffee and cute rabbit

Address
法國上萊茵省科爾馬

Shipping Fee
260

Do you want to be a good PUA store?

Choose User Type

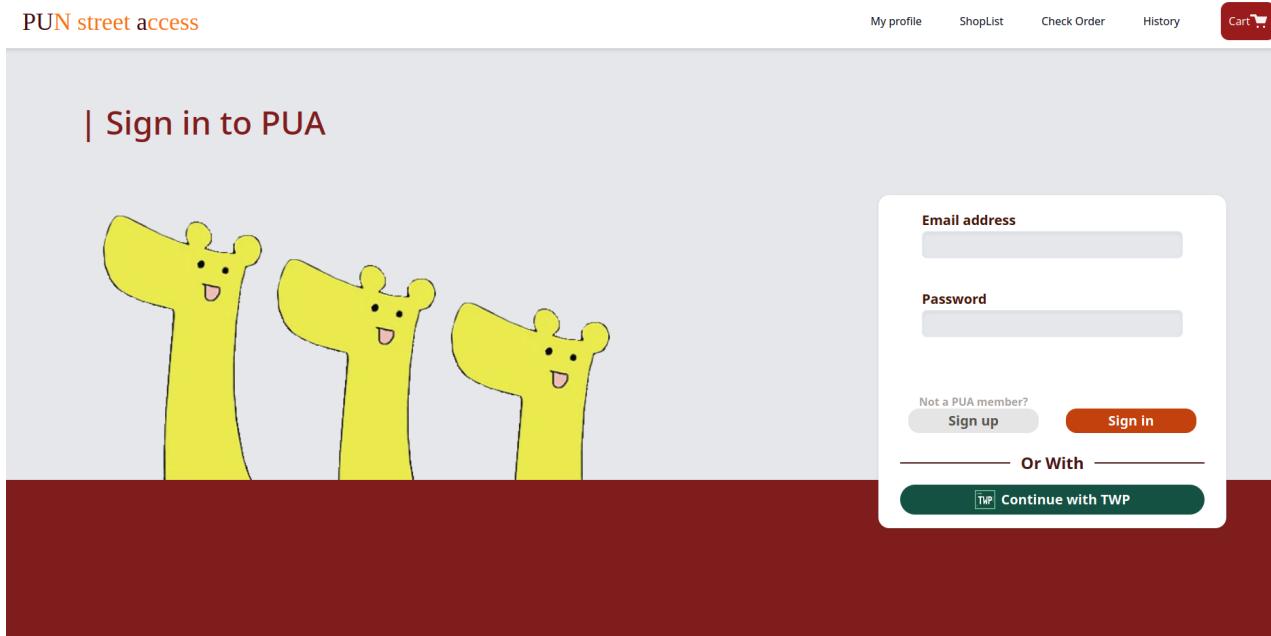
Complete basic information

Complete!

Complete!
you are a PUA member now

[Go to Login](#)

user can login (with PUA account or with TWP[1] account)



6.2.2 customer

customer can browse shops

PUN street access

My profile ShopList Check Order History Cart

customer can search shops with price, tag or string

飯

customer can browse products

菜名	描述	價格
雙倍雞肉飯	雙倍雞肉的份量，低卡高蛋白 要健身想健美，選我就對。💡此餐點優惠價，內容物不得拆裝唷！	NT\$ 225
U-獅子頭雞肉飯	地表最強雙組合!讓您吃的梗滿足	NT\$ 185
好吃雞肉飯	每日現煮 手工去骨的雞腿肉 搭配鹹甜古早味油膏	NT\$ 140
咖哩雞肉飯	大人、小孩都愛的濃郁咖哩配上去骨雞腿肉根本絕配	NT\$ 120
好吃雞肉飯(無配菜)	簡單卻很美味。	NT\$ 115

customer can buy products

雙倍雞肉飯



UE 口味選項 [\[編輯\]](#)

UE 加手工蔥醬
UE 加手工蔥薑、辣椒醬(小辣)
UE 加手工辣椒醬(小辣)
蔥、辣都不加

Add Discount [\[No Event Discount\]](#)

NT\$ 225

雙倍雞肉的份量，低卡高蛋白 要健
身想健美，選我就對。 ❤️此餐點優
惠價，內容物不得拆裝唷！

3 Add Cart

customer can view cart (cart can apply discount)

五號料理 no.5 cuisine Shipping Fee NTS 15



厚切咖哩豬排

NTS 210 x6

+ More Product

雞肉本家 Shipping Fee NTS 44



雙倍雞肉飯

NTS 225 x2



果漾果醋-蘋果

NTS 59 x2

+ More Product

Discount

五號料理 no.5 cuisine	雞肉本家
Shipping Discount NTS 300 free shipping when price over 300	Shipping Discount NTS 0 no discount
year discount 95% 2023-01-05~2024-01-05	

Total Price NTS 1778 [Check Out](#)

customer can view order

PUN street access

The screenshot shows a user interface for tracking an order. At the top, there are tabs: 'Accept order' (disabled), 'Making order', 'Deliver Order', and 'Arrival Address'. The main area displays a message '運送中...' (In transit...) above a section for '雞肉本家' (Chicken Family). It includes three icons of delivery drones and a placeholder address 'Taking Address a street a street a street bbbbbbbbbbbbbb'. On the left, there's a thumbnail image of a restaurant interior.

customer can view history

PUN street access

The screenshot shows a 'History' section with two entries. The first entry is for 'mos burger' with a 4-star rating, dated 2020-03-01T00:00:02Z, cost NT\$20000, and a 'Detail' button. The second entry is for '雞肉本家' with a 3.8-star rating, dated 2023-12-30T15:36:10Z, cost NT\$174, and a 'Detail' button. Each entry includes a small thumbnail image of the restaurant and a 'Rate store' button with five stars.

6.2.3 seller

seller can edit shop (edit tag)

The screenshot shows a mobile application interface for a restaurant named "雞肉本家". At the top, there is a small image of the restaurant's exterior. Below it, the shop name is displayed in large, bold, brown text. Underneath the name is the address: "100台灣台北市中正區八德路一段82巷9弄22號". There are three buttons above the product list: "★ 3.8" (rating), "Fresh" (category), and "Add Tag" (button). The product list contains four items:

- 雙倍雞飯** (Double Chicken Rice) - NTS 225. Description: 想健美，選我就對。此餐點優惠價，內容物不得拆裝唷！ Tags: Spices, Sustainable, Bakery, Local, sus thing.
- U-獅子頭雞肉飯** (U-Shishitou Chicken Rice) - NTS 185. Description: 地表最強雙組合!讓您吃的梗滿足 Tags: Spices, Sustainable, Bakery, Local, sus thing.
- 好吃雞肉飯** (Delicious Chicken Rice) - NTS 140. Description: 每日現煮 手工去骨的雞腿肉 搭配鹹甜古早味油膏 Tags: Spices, Sustainable, Bakery, Local, sus thing.
- 咖哩雞肉飯** (Curry Chicken Rice) - NTS 120. Description: 大人、小孩都愛的濃郁咖哩配上去骨雞腿肉根本絕配 Tags: Spices, Sustainable, Bakery, Local, sus thing.

seller can edit (create) products

The screenshot shows a product creation or editing interface. At the top, the product name "fresh leaf" is entered in a search bar. The main area displays a photograph of several potted plants. To the right of the image, there are several input fields and controls:

- quality**: A radio button group with "Yes" (checked) and "No".
- Add Discount**: A text input field with a "+" button to add a discount.
- Set Status**: A radio button group with "上架中" (On Shelves) (checked) and "已售完" (Sold Out).

At the bottom right is a large orange "Save" button. On the far right, there is a vertical column of status indicators: "good" (green dot), "bad" (red dot), and two other unlabelled red dots.

seller can edit (create) discounts

Add A Discount

X

Event Discount | Get **3** FOR FREE ONE

Max Quantity

Discount Name

Description

Delete **Save**

seller can view and update order

PUN street access

My profile ShopList Check Order History Cart

2023-12-30T15:35:34Z NTS 581 Order user : alice **Accept**

2023-12-30T15:36:10Z NTS 174 Order user : bob **Deliver**

seller can get statistics of the shop



6.2.4 admin

admin can ban/unban users

The screenshot shows a user interface for managing users. At the top, there's a header bar with a profile icon, the name "admin", the email "admin@gmail.com", and two buttons: "process order" and "go my store". Below the header, there are two tabs: "User List" and "Order List". The "User List" tab is active, displaying a list of users with their order details and a "Ban" button. The users listed are: alice (Order user: a@gmail.com), bob (Order user: b@gmail.com), tcp (Order user: c@gmail.com), udp (Order user: d@gmail.com), icmp (Order user: e@gmail.com), dns (Order user: f@gmail.com), and dhcp (Order user: g@gmail.com). The "tcp" entry is highlighted with a red border around its entire row.

User	Order User	Action
alice	a@gmail.com	Ban
bob	b@gmail.com	Ban
tcp	c@gmail.com	UnBan
udp	d@gmail.com	Ban
icmp	e@gmail.com	Ban
dns	f@gmail.com	Ban
dhcp	g@gmail.com	Ban

admin can view all orders

The screenshot shows a mobile application interface. At the top, there is a header bar with a profile icon, the text "admin", "admin@gmail.com", and "address_admin". On the right side of the header are two buttons: "process order" and "go my store". Below the header, there are two tabs: "User List" and "Order List", with "Order List" being the active tab. The main content area displays a list of seven orders, each represented by a card. Each card contains a small icon of a folded cloth, a timestamp, and the order user's name. To the right of each card is a "Detail" button. The orders are as follows:

Order ID	Date	User	Action
1	2020-01-01T00:00:00Z	admin	Detail
2	2020-01-01T00:00:00Z	admin	Detail
3	2020-02-01T00:00:00Z	admin	Detail
4	2020-02-02T00:00:00Z	admin	Detail
5	2020-03-01T00:00:00Z	bob	Detail
6	2020-04-01T00:00:02Z	alice	Detail
7	2020-04-01T00:00:03Z	admin	Detail

7 Suggestions on Database Turning

7.1 View

在取得商店或商品資料時，常常需要 join 許多不同的表格，若建立良好的 view，可使得開發人員不需寫過於複雜的 query，同時也增加資料可讀性。

7.2 Index

在做搜尋功能時，其中有用到 store 中的 category 欄位，若能將 category 做為 index，就可以避免線性搜尋所造成的效能損失。

8 Additional Queries and Views

8.1 First Query

Get shops by search string: use nest queries, aggregate functions, order by clause

```
SELECT
store_id,
name,
rate,
picture,
category_array
FROM (
    SELECT
        store_id,
        name,
        rate,
        picture,
        (
            SELECT
                jsonb_agg(
                    jsonb_build_object(
                        'category_id',
                        categories.category_id,
                        'category_name',
                        categories.name
                    )
                ) as categories_item
            FROM categories
            NATURAL JOIN (
                SELECT labels.category_id
                FROM labels
                WHERE
                    labels.store_id = stores.store_id
            ) AS labels
        ) AS category_array,
        (
            SELECT COUNT(*) > 0
            FROM products
            LEFT JOIN stores as s ON products.store_id =
                stores.store_id
            WHERE
                products.store_id = stores.store_id
                AND(
                    products.name LIKE '%' || '飯' || '%'
                    OR stores.name LIKE '%' || '飯' || '%'
                    OR products.description LIKE '%' || '飯' || '%'
                )
            ) AS is_string_exits
        FROM stores
    ) as store
WHERE is_string_exits = true
ORDER BY store.rate DESC
```

query result

	store_id	name	rate	picture	category_array
1	11	佐藤精肉店 豚丼専門	4.7	https://cdn.discordapp.com.	(NULL)
2	15	山本軒YAMAMOTOKEN	4.7	https://cdn.discordapp.com.	(NULL)
3	3	五號料理 no.5 cuisine	4.4	https://cdn.discordapp.com.	(NULL)
4	13	金湘	4.3	https://cdn.discordapp.com.	(NULL)
5	14	粢坡BonBox	4.3	https://cdn.discordapp.com.	(NULL)
6	12	咖食堂	4.1	https://cdn.discordapp.com.	(NULL)
7	2	mos burger	4	https://cdn.discordapp.com.	[{"category_id": 2, "category_name": "ファミリーレストラン"}]
8	9	雞肉本家	3.8	https://iili.io/JADCTGt.png	[{"category_id": 3, "category_name": "中華"}]

8.2 Second Query

Get products selling statistics by date: use group by clause

```
SELECT
    carts.product_id,
    products.name,
    SUM(carts.product_quantity)
FROM carts
    LEFT JOIN products ON carts.product_id = products.product_id
    LEFT JOIN orders ON carts.store_id = orders.store_id AND
        carts.cart_id = orders.cart_id AND carts.customer_id =
        orders.user_id
WHERE
    orders.store_id = 1
    AND orders.status >= 6
    AND EXTRACT(
        year
        FROM
            order_date
    ) = 2020
    AND EXTRACT(
        month
        FROM order_date
    ) = 1
GROUP BY
    carts.product_id,
    products.name
```

query result

		product_id	name	sum
	1	1	pasta	100
	2	2	black tea	100

8.3 Third Query

Get all Product's information: use nest query, order by clause

```
SELECT
    product_id,
    store_id,
    name,
    description,
    picture,
    stock,
    price,
    status, (
        SELECT
            coalesce(
                jsonb_agg(
                    jsonb_build_object(
                        'discount_max_quantity',
                        max_quantity,
                        'product_id',
                        product_id,
                        'discount_name',
                        name,
                        'discount_description',
                        description,
                        'discount_id',
                        discount_id,
                        'status',
                        status
                    )
                ),
                []
            ) AS event_discount
        FROM discounts
        NATURAL JOIN (
            SELECT
                event_discount.discount_id,
                event_discount.max_quantity,
                event_discount.product_id
            FROM event_discount
            WHERE
                event_discount.product_id = products.product_id
        ) AS discountInfo
    ) AS event_discount_array, (
        SELECT
            coalesce(
                jsonb_agg(
                    jsonb_build_object(
                        'product_id',
                        product_id,
                        'label_name',
                        label_name,
                        'label_value',
                        label_value
                    )
                ),
                []
            ) AS event_label
        FROM event_labels
        WHERE
            event_labels.product_id = products.product_id
    ) AS event_label_array
) AS event_info, (
    SELECT
        coalesce(
            jsonb_agg(
                jsonb_build_object(
                    'product_id',
                    product_id,
                    'label_name',
                    label_name,
                    'label_value',
                    label_value
                )
            ),
            []
        ) AS event_info
    FROM event_info
    WHERE
        event_info.product_id = products.product_id
) AS event_info_array
) AS event_info_all
FROM products
ORDER BY product_id;
```

```

'required',
required,
'item_array', (
    SELECT (
        jsonb_agg(
            jsonb_build_object('name',
                label_item.item_name)
        )
    ) AS item_array
FROM label_item
WHERE
    label_item.label_name =
        product_label.label_name
    AND label_item.product_id =
        product_label.product_id
)
),
'[]'
) AS product_label_array
FROM product_label
WHERE
    product_label.product_id = products.product_id
)
FROM products
WHERE store_id = 1 AND status != 0
ORDER BY product_id;

```

query result

	Q	product_id	store_id	name	description	picture	stock	price	status	event_discount_array	product_label_array
	1	1	1	pasta	tasty pasta	https://i.imgur.com/3i3tyXJ.	100	150	1	[{"status": 1, "product_id": 1}]	[]
	2	2	1	black tea	student card get free	https://i.imgur.com/3i3tyXJ.	100	0	1	[{"status": 1, "product_id": 2}]	[{"required": true, "item_arra
	3	25	1	義式肉醬焗烤	義式肉醬、高級起司絲、美	https://tb-static.uber.com/pr	100	155	1	[]	[{"required": true, "item_arra
	4	26	1	奶油青醬雞肉	獨家特製青醬奶油、香炒薑	https://tb-static.uber.com/pr	100	150	1	[]	[{"required": true, "item_arra
	5	27	1	香蒜白酒蛤蜊	自製蒜醬加濃醇高湯與特級	https://tb-static.uber.com/pr	100	155	1	[]	[{"required": true, "item_arra
	6	28	1	佛羅倫斯奶油	香濃義式白醬、蒜醬、奶油	https://tb-static.uber.com/pr	100	175	1	[]	[{"required": true, "item_arra
	7	29	1	義式肉醬麵	義式肉醬、高級起司粉	https://tb-static.uber.com/pr	100	130	1	[]	[{"required": true, "item_arra
	8	30	1	墨魚醬海鮮麵	濃郁墨魚墨囊熬煮黑醬、新	https://tb-static.uber.com/pr	100	175	1	[]	[{"required": true, "item_arra

9 Conclusions and Future Work

9.1 Conclusions

郭丞軒 這次專案的開發涉及許多領域，從團隊管理、資料庫設計、到文件撰寫及前後端架構等，很多東西都是我第一次接觸。因此，常常在開發過程中搞砸許多事情，造成開發上的困難以及不斷的重構。但也很幸運地受到組員們及組外顧問的幫助，得以讓這次的專案順利進行，也讓我從中得到許多寶貴的經驗。

王漢竑 這次的專案開發是我第一次接觸前. 後. 後. 資料庫端網頁程式設計，也是我第一次接近管理專案，導致需要看很多 code review，進而使用 github action 與 npm lint 幫助我管理專案，在中途有很多 PR 都因為我身兼拆除炸彈的使命與另外一個專案管理的事情還有與 Dr.Smell 對抗的學分競爭，導致拖延很多個禮拜才進行 code review，幸好有生產力爆發的組員才得以順利完成專案。在前後端溝通的問題或是組內資訊流通的問題，我使用了每個禮拜發佈一個週報來解決，他可以順便當成一個 Todo-List，雖然最後證實他沒有很大幫助，但是他取代了每次都不知道要開甚麼的會議，節省了時間。

在資料庫設計的部分有很多綱要的決策常被忘記，導致又花了很多時間得到一摸一樣的結論，需要 hyjl(會議記錄，出自本地顧問) 來解決這個問題。

前端的部分使用的 svelte 是非常有趣的框架，他擁有魔法的語法，加速我們的開發進度，他的缺點大概就是網路上的資料常常會過期，需要另外解決。Tailwind 也是另外一個非常好框架，有邏輯的解決了 css 問題，網頁跑版的問題才可以縮到很小，但是礙於時間問題沒有處理 RWD。

陸芳寰 透過這次資料庫專案，我學到了資料庫除了只有儲存資料，還有它背後儲存資料時所需的複雜邏輯與規則、及使其運行所需的軟/硬體。在專案中我主要接觸到了 UI 製作，並學習到了製作及實現他人設計的困難之處與其解決方法。

專案製作中還有許多其他關鍵的技術。如設計資料庫架構、實現資料庫邏輯、銜接資料庫與 UI 功能... 等。這些技術的製作過程讓我學習到各功能製作者互相溝通是非常重要的事情，必須充分交流、協調才能避免製作遇上瓶頸。

最後各階段報告時老師的指點也讓我受益良多，以淺顯易懂的話語指出設計或實作的瑕疵/不足，讓我們有機會回去修改，拿出更好的成果。

陳姿安 這次專案幾乎所有事情都是初次體驗。

在專案負責後端部份，初次與 golang 相處一段時間後發現寫起來覺得蠻舒適的，且我們採用的框架將後端切得很乾淨寫起來非常順手。

此次也獲得了一些文件撰寫的經驗、也經歷的重構文件的經驗，相信下次設計 api 的時候會更順利。透過這個專案我也獲得了設計網頁界面的經驗，從來沒想過自己會有設計網頁的一天，雖然實做不是由自己完成，但是看到自己設計的 UI 被慢慢實踐出來很開心，每天看到都想修改但這對負責實做的人並不是很道德，所以我放棄了這樣的想法。

也因為負責了後端所以有機會寫到不少 sql，起初以為 sql 只是 insert select update 一些簡單的操作，後來上課後才發現原來 sql 可以寫的相當複雜，且好的魔法 sql 可以將網頁的效率提高，sql 想寫多長就有多長。親切的組員讓我當慣老闆，我非常的開心，希望以後還有機會當慣老闆。

劉承翰 資料庫這門課讓連後端都沒碰過的我第一次知道後端的概念與做法，在實作的過程中還發現有從計算機網路學到的概念可以結合使用，發現還挺有趣的。雖然整體沒辦法貢獻太多 (sorry)，開 PR 還常常被拒絕，但看其他 coding 實力強的人寫 code 也學到蠻多東西的。這次花最久的時間應該就是看 code 了吧，每次有新的 PR 就會嘗試理解它的原理，也趁機會學習一下架構與別人的 coding style:D。

歐佳昀 經過資料庫系統的專案後，我對於資料庫系統的應用有了更深一層的認知，並且也有實際的感覺，原來他不僅僅是作為一門學科，更是資工系學生在開發一個電商專案時，實際會用到的知識。而在專案中，除了資料庫的基礎知識運用外，我也獲得相當多的經驗，像是前端的開發、ui 的設計、與人交流、溝通的技巧等等，或者是分析一個網站的需求，諸如使用的情境，細節等，更重要的是去實際體會了投入心血的感覺，並且在樂在其中，這才是最重要的。

我堅信，一份作業需要帶給學生的，不僅僅是對知識運用的掌握度，更是需要帶給學生新的體驗與認識，而團體的作業更是能體會到不同思維間的碰撞。而我在最後深刻去了解老師說的話，為何這份作業，卻希望我們能放在自己未來的履歷上？更是因為這是我們從頭開始去作出的專案，因此才具有理解和掌握，與認知和成就感。

這份專案的完成，而掌握到的知識，不管是討論各項系統與 api，或者是邏輯性的去提出自己的需求，甚至是自我學習的能力，都是我在這份專案有所學、有所悟的地方，也希望在未來可以充分運用這些經驗，來補足自己的能力。

組外顧問 我的專業在這個組別得到比原本組別更多的尊重，10 / 10 推薦擔任顧問

9.2 Future Work

PUA 目前已經是個非常完善的系統，未來的目標是將其統推廣到全校以及北科周圍的店家，並引入無人機作為外送載體，不只節省人力成本，也使得狹小的美食街不再充滿汽、機車，帶給商家及學生更美好的環境。

我們也計畫在之後與『太白粉專賣店 too white powder (TWP)』合作，讓商家可以透過 PUA 向 TWP 訂購原物料，使供應鏈管理更為便利。

10 References

- [1] jykuo-love-shiritori. *Too White Powder (TWP)*. URL: <https://twp.noobdy.com>.

11 Appendix

Demo video link [link](#)

The link to the system deployed on a public domain [link](#)