CCC Shop

系統需求規格書

Software Requirements Specification (SRS)

Version: 1.0

姓名	學號	E-mail
周宥霄	108820028	t108820028@ntut.org.tw
羅羽軒	108820001	t108820001@ntut.org.tw
胡紹宇	108820008	t108820008@ntut.org.tw
周雨柔	108820015	t108820015@ntut.org.tw
郭梓琳	108820016	t108820016@ntut.org.tw
周世昊	108820027	t108820027@ntut.org.tw
簡上博	108820031	t108820031@ntut.org.tw

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

10/05/2021

目錄 (Table of Contents)

Section 1簡介(Introduction)

1.1 目的 (Purpose)

為了更加理解資料庫的架構設計、操作方式與實際應用，我們透過本學期資料庫系統課程，開發一個完整的 3C 產品線上購物系統「CCC Shop」，旨在完成一個由商家為分類架構的系統，並提供商家(本專案中之工作人員)可以管理與販賣自家商品的空間，使用者則可以以品牌為基礎來購買喜愛的產品。本專案結合前端網頁設計、後端網路技術，並嘗試使用良好的架構來學習資料庫系統。

本專案可完成下列功能：

- 系統管理者 (administrator/admin)
  - ✓ 擁有最高權限
  - ✓ 創建、刪除帳戶
- 工作人員 (staff)
  - ✓ 上架/下架自家商品、編輯商品資訊
  - ✓ 處理訂單、訂定折扣
  - ✓ 產生統計報表
- 會員 (customer)
  - ✓ 登入/登出系統
  - ✓ 購買商品、查看購物車
  - ✓ 查看訂單狀態
  - ✓ 訂單歷史紀錄查詢
  - ✓ 給予評價
- 訪客 (visitor)
  - ✓ 註冊帳號
  - ✓ 瀏覽、搜尋商品

1.2 系統名稱 (Identification)

本專案名稱為：

- 電子商務系統 (E-Commerce System, **ECS**)

各子系統名稱為：

- 帳戶管理子系統 (Account Management Subsystem, **AMS**)
- 商品管理子系統 (Product Management Subsystem, **PMS**)
- 訂單管理子系統 (Order Management Subsystem, **OMS**)
- 購物車管理子系統 (Shopping Cart Management Subsystem, **SCMS**)
- 報表統計子系統 (Report Statistics Subsystem, **RSS**)
- 商品搜尋子系統 (Product Searching Subsystem, **PSS**)
- 商品評價子系統 (Product Rating Subsystem, **PRS**)
- 資料庫子系統 (Database Subsystem, **DBS**)

1.3 概觀 (Overview)

本專案開發的系統主要包含前端、後端、和資料庫三個部分。資料庫我們使用 MariaDB，因為是 MySQL 的復刻所以和 MySQL 有高度相容性，除了有商業支援以外，在開源社群也有足夠的網路資源。後端我們選擇 Java 語言，並搭配 Spring Boot 實作，使用控制反轉和依賴注入等原則管理程式，使應用程式能更簡易的組建與開發。至於前端我們選擇使用 Vue 框架，身為最主流的前端框架之一，網路上能找到大量的學習資源和社群討論，能減輕前端開發的成本，我們也使用 Tailwind 等 CSS Library 來增加整個系統的開發效率和完整性。

在本專案開發，我們參考 Clean Architecture 的概念，大致將我們的程式分為 Use case 及 Adapter 兩層，分別為實作功能及資料連接前端的部分，讓我們在開發時更清楚每隻程式所負責的行為。開發中我們也會遵循 SOLID 原則，讓程式碼乾淨、更好維護。開發過程也會以 TDD 的方式進行開發，以確保程式和系統的每個環節能正確執行。

1.4 符號描述 (Notation Description)

- 系統符號

ECS 1.0.0	The ECS 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.
PMS 1.2.n	The PMS components will be labeled with the number 1.2.n.
OMS 1.3.n	The OMS components will be labeled with the number 1.3.n.
SCMS 1.4.n	The SCMS components will be labeled with the number 1.4.n.
RSS 1.5.n	The RSS components will be labeled with the number 1.5.n.
PSS 1.6.n	The PSS components will be labeled with the number 1.6.n.
PRS 1.7.n	The PRS components will be labeled with the number 1.7.n.
DBS 1.8.n	The DBS components will be labeled with the number 1.8.n.

- 功能系統符號

ECS-F-nnn	ECS 功能性需求(Functional Requirements)
AMS-F-nnn	AMS 功能性需求(Functional Requirements)
PMS-F-nnn	PMS 功能性需求(Functional Requirements)
OMS-F-nnn	OMS 功能性需求(Functional Requirements)
SCMS-F-nnn	SCMS 功能性需求(Functional Requirements)
RSS-F-nnn	RSS 功能性需求(Functional Requirements)
PSS-F-nnn	PSS 功能性需求(Functional Requirements)
PRS-F-nnn	PRS 功能性需求(Functional Requirements)
DBS-F-nnn	DBS 功能性需求(Functional Requirements)

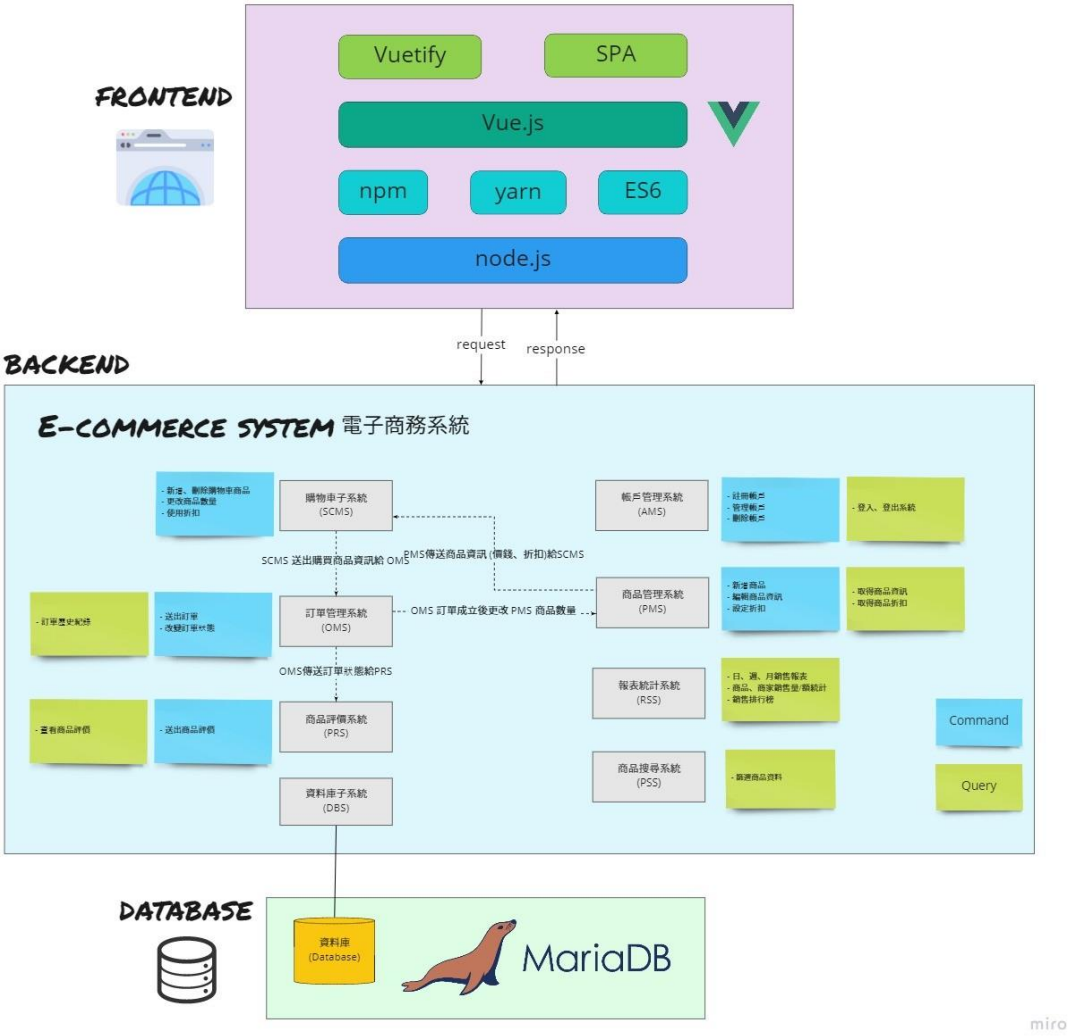
Section 2 系統(System)

2.1 系統描述 (System Description)

本專案電子商務系統 (E-Commerce System, ECS) 主要分為八個子系統，分別為: 帳戶管理子系統 (Account Management Subsystem, AMS)、商品管理子系統 (Product Management Subsystem, PMS)、訂單管理子系統 (Order Management Subsystem, OMS)、購物車管理子系統 (Shopping Cart Management Subsystem, SCMS)、報表統計子系統 (Report Statistics Subsystem, RSS)、商品搜尋子系統 (Product Searching Subsystem, PSS)、商品評價子系統 (Product Rating Subsystem, PRS)。

以下系統結構圖採 CQRS 讀寫分離架構，下圖藍色便條紙為資料庫修改（Command）之系統需求，綠色便條紙則為資料庫查詢（Query）之系統需求。本系統程式碼實作並未套用 CQRS，只是以此來畫架構圖，使之能更清楚的呈現。

2.1.1 系統架構圖 (System Architecture Diagram)



2.2 操作概念 (Operational Concepts or User Stories)

Scenario 1：訪客操作概念 (Visitor Operational Concepts)

訪客經由電子商務系統 (ECS) 瀏覽商品，可以透過商品搜尋子系統 (PSS) 查詢商品，並可以前往帳戶管理子系統 (AMS) 註冊會員帳號。

Scenario 2：會員操作概念 (Customer Operational Concepts)

會員透過帳戶管理子系統 (AMS) 登入會員身分，除了擁有訪客的功能以外，可以透過購物車管理子系統 (SCMS) 將選購的商品加入購物車及隨時查看購物車內的商品。結帳後，可以經由訂單管理子系統 (OMS) 查看訂單狀和訂單的歷史紀錄，並利用商品評價子系統 (PRS) 為訂購的商品給予評價。

Scenario 3：工作人員操作概念 (Staff Operational Concepts)

工作人員透過帳戶管理子系統 (AMS) 登入工作人員身分，除了擁有會員的功能以外，可以透過商品管理子系統 (PMS) 上、下架自家商品、編輯商品相關資訊及訂定優惠方案，可以經由訂單管理子系統 (OMS) 處理訂單，並可以使用報表統計子系統 (RSS) 產生統計報表。

Scenario 4：系統管理者操作概念 (Administrator Operational Concepts)

系統管理者擁有最高的權限，除了擁有工作人員的功能以外，可以透過帳戶管理子系統管理 (AMS) 創建、刪除帳戶，並可對資料庫子系統 (DBS) 進行維護。

2.3 功能性需求 (Functional Requirements)

需求編號	需求描述
AMS-F-001	訪客可以註冊會員
ECS-F-001	訪客可以查看商品
PSS-F-001	訪客可以搜尋商品
AMS-F-002	會員可以登入/登出系統
OMS-F-001	會員可以將購物車中的商品結帳
SCMS-F-001	會員可以查看購物車
OMS-F-002	會員可以查詢訂單歷史紀錄
PRS-F-001	購買商品後，會員可以給予評價
PMS-F-001	工作人員可以上架/下架自家商品
PMS-F-002	工作人員可以編輯自家商品資訊
OMS-F-003	工作人員可以處理自家商品的訂單
PMS-F-003	工作人員可以訂定自家商品的折扣
RSS-F-001	工作人員可以產生統計報表

AMS-F-003	系統管理者可以創建任一種帳戶
AMS-F-004	系統管理者可以刪除任一種帳戶

2.4 資料需求 (Data Requirements)

需求編號	需求描述
AMS-DR-001	有效的使用者需包含: login id, name, password, email, address
DBS-DR-001	資料庫的初始狀態需包含全體工作人員 (staff) 和系統管理者 (administrator)
PMS-DR-001	商品資訊需包含：product id, name, description, other attributes (price, quantity…)
AMS-DR-002	商店資訊需包含： store id, name, address, contact email, phone
OMS-DR-001	訂單資訊需包含： order id, time, customer, name, price and quantity of each purchased item, total amount of the purchase, current status of the order (received, processing, shipping, closed).
RSS-DR-001	銷售報告應包含： date, store name, number of orders, total amount of sales
DBS-DR-002	Login id, product id, store id, order id, discount code 必須唯一

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
DBS-PR-001	資料庫的設計應正規化，減少重複 (Redundancy)
ECS-PR-001	使用者瀏覽時，頁面讀取應小於 5 秒
ECS-PR-002	使用者搜尋時，搜尋時間應小於 3 秒

2.5.2 資安需求 (Security Requirements)

需求編號	需求描述
ECS-SR-001	預防 SQL injection 攻擊行為
AMS-SR-001	建立強密碼策略，以建構合理的身分驗證
AMS-SR-002	資料庫之密碼不可以明文儲存

2.6 介面需求 (Interface Requirements)

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

需求編號	需求描述
ECS-UI-001	商品瀏覽介面
SCMS-UI-001	購物車介面
PSS-UI-001	搜尋介面
AMS-UI-001	會員登入及註冊介面
OMS-UI-001	訂單狀態介面

PRS-UI-001	商品評價介面
OMS-UI-002	訂單歷史紀錄介面
PMS-UI-001	商品管理介面
RSS-UI-001	財務報表介面
AMS-UI-002	帳號管理介面

2.6.2 外部介面需求 (External Interface Requirements)

需求編號	需求描述
ECS-EI-001	使用者操作瀏覽器透過 HTTP 與 ECS 網頁伺服器通訊
ECS-EI-002	前端透過 HTTP 發送 API 請求給後端伺服器
ECS-EI-003	後端伺服器通過 nginx 反向代理 Web API 存取本系統

2.6.3 內部介面需求 (Internal Interface Requirements)

需求編號	需求描述
ECS-II-001	SCMS 送出購買商品資訊給 OMS
ECS-II-002	OMS 傳送訂單狀態給 PRS
ECS-II-003	PMS 傳送商品資訊 (價錢、折扣)給 SCMS
ECS-II-004	OMS 訂單成立後更改 PMS 商品數量

2.7 其他需求 (Other Requirements)

2.7.1 環境需求 (Environmental Requirement)

需求編號	需求描述
ECS-ER-001	需要在有網路的環境

2.7.2 安裝需求 (Installation Requirement)

需求編號	需求描述
ECS-IR-001	前端建置使用 Node.js 16.10.0
ECS-IR-002	後端建置使用 Java JDK 16
ECS-IR-003	資料庫使用 MariaDB 10.6.4

2.7.3 測試需求 (Test Requirements)

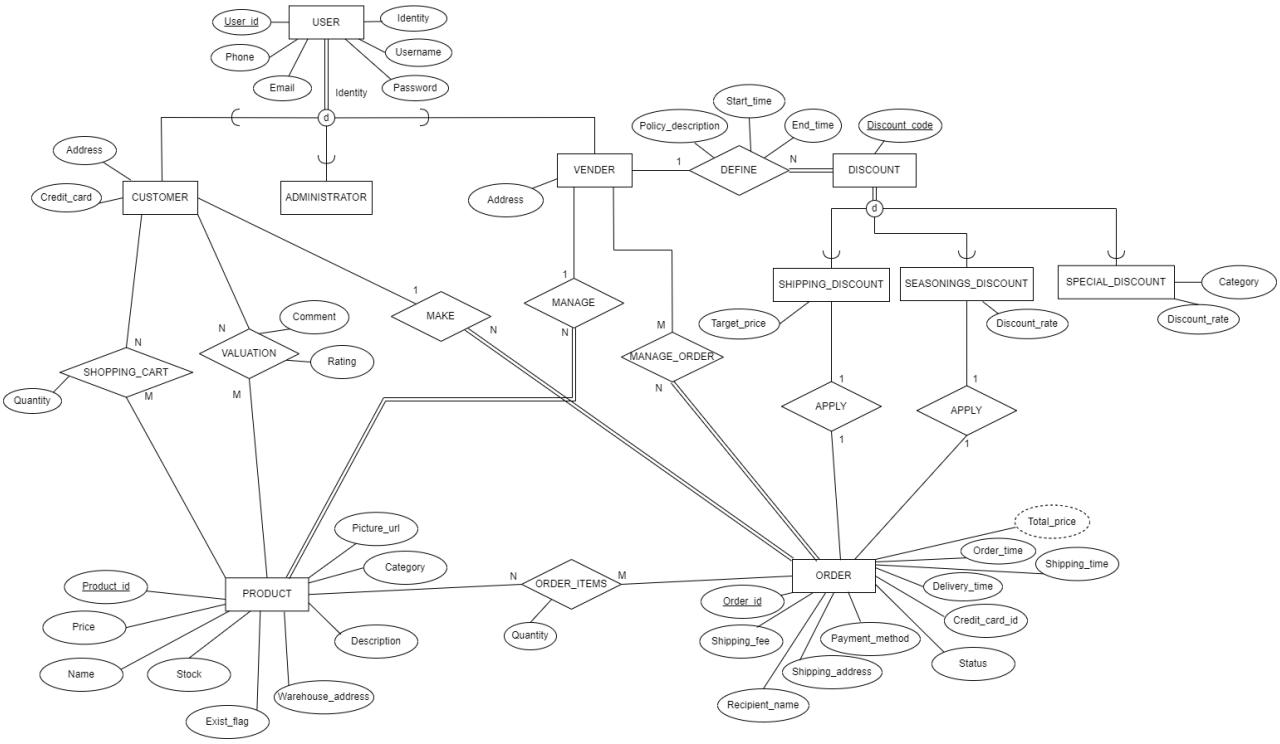
需求編號	需求描述
ECS-TR-001	每個子系統中的所有函式都必須通過測試
ECS-TR-002	在 Chrome、Firefox 測試 UI
ECS-TR-003	使用 wrk 進行壓力測試

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

- 一筆訂單可同時使用三種折扣：運費 (Shipping)、季節性 (Seasonings) 及特殊活動 (Special event) 折扣
- 同一商品在一個時段內只會有一種特殊活動折扣
- 兩個(含)以上特殊活動折扣若對同一商品都有效，則兩個特殊活動的時段不能有重疊，必須錯開
- 商品數量需以 > 0 的整數為單位
- 顧客可選擇購買的商品數量應 ≤ 庫存數量

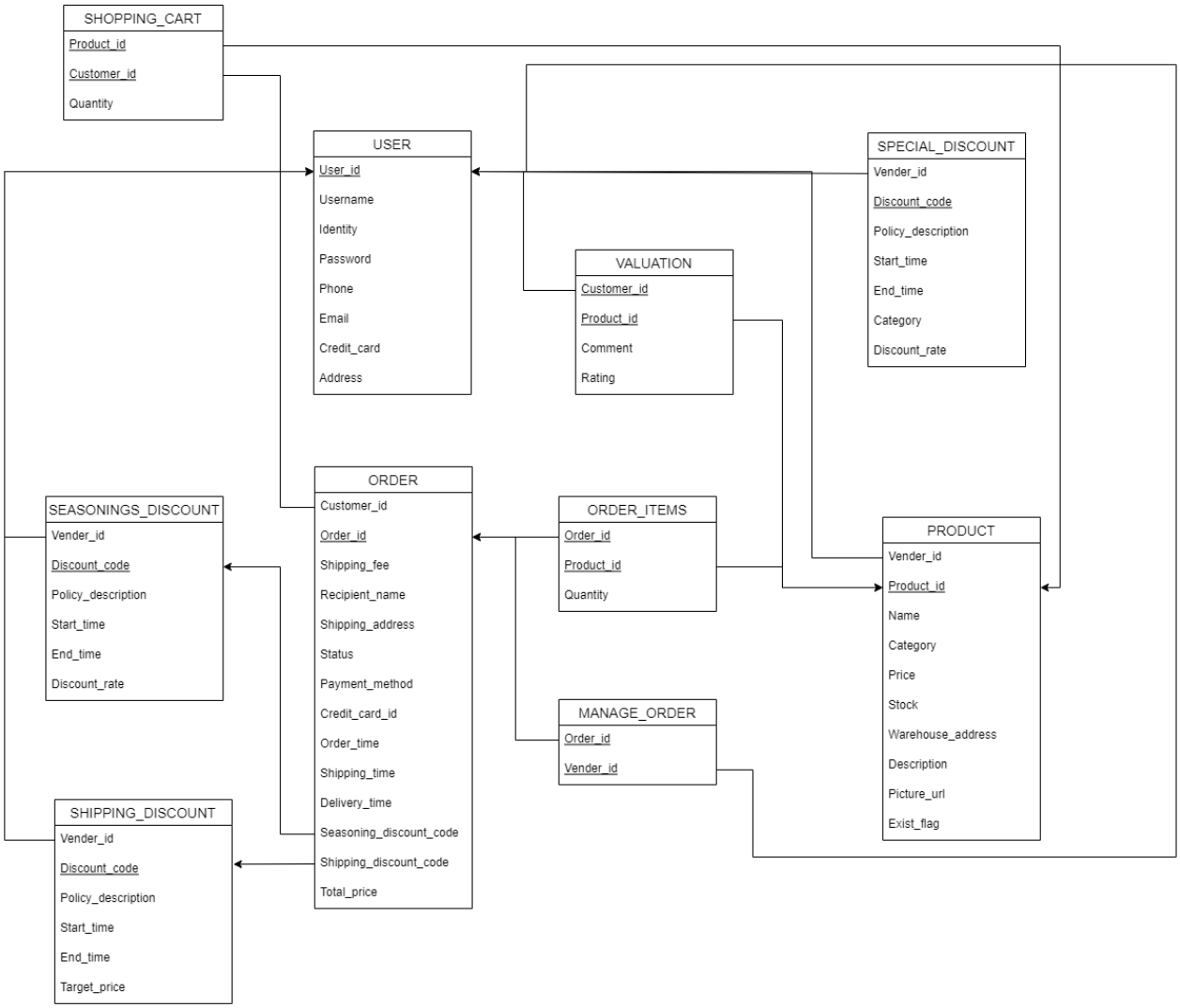
Section 3 資料庫概念設計（Conceptual Design of the Database）

3.1 Entity Relationship ER Model



Section 4 邏輯資料庫綱要（ Logic Database Schema）

4.1 Schema of the Database



USER				
Description: 存放會員的相關資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**
User_id	int	primary	not null	會員編號
Identity	varchar		not null	身分類別
Username	varchar		not null	會員姓名

Password	varchar		not null	會員密碼
Phone	varchar		not null	會員手機號碼
Email	varchar		not null	會員電子信箱
Credit_card	varchar			信用卡號
Address	varchar			地址

SHOPPING_CART				
Description: 存放會員與商品的關係 (會員購物車的商品)				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Product_id	int	primary foreign	not null	商品編號 reference（PRODUCT）
Customer_id	int	primary foreign	not null	會員編號 reference（USER）
Quantity	int		not null	商品數量

ORDER				
Description: 存放訂單資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Order_id	int	primary	not null	訂單編號
Customer_id	int	foreign	not null	會員編號 reference（USER）
Total_price	int		not null	商品總價格
Shipping_fee	int		not null	運費
Recipient_name	varchar		not null	收件者名字
Shipping_address	varchar		not null	收貨地址
Status	varchar		not null	訂單狀態： received, processing, shipping, closed

Payment_method	varchar		not null	付款方式
Credit_card_id	varchar			信用卡號
Order_time	timestamp		not null	訂單下定時間
Shipping_time	timestamp			出貨時間
Delivery_time	timestamp			送達時間
Seasonings_discount	int	foreign		季節性折扣
Shipping_discount	int	foreign		運費折扣

ORDER_ITEMS				
Description:存放訂單與商品的關係 (訂單所含的商品)				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Order_id	int	primary foreign	not null	訂單編號 reference（ORDER）
Product_id	int	primary foreign	not null	商品編號 reference（PRODUCT）
Quantity	int		not null	商品數量

MANAGE_ORDER				
Description:存放商家與訂單的關係				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Order_id	int	primary foreign	not null	訂單編號 reference（ORDER）
Vender_id	int	primary foreign	not null	會員(商家)編號 reference（USER）

VALUATION

Description:存放會員與商品的關係 (會員對商品的評價)				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Customer_id	int	primary foreign	not null	會員編號 reference（USER）
Product_id	int	primary foreign	not null	商品編號 reference（PRODUCT）
Comment	varchar		not null	評語
Rating	int		not null	評價星等

PRODUCT				
Description:存放商品資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Product_id	int	primary	not null	商品編號
Vender_id	varchar	foreign	not null	會員(商家)編號 reference（USER）
Category	varchar		not null	商品種類
Price	integer		not null	商品價格
Name	varchar		not null	商品名稱
Stock	integer		not null	商品庫存量
Warehouse_address	varchar		not null	出貨地址
Picture_url	varchar			商品照片路徑
Description	varchar			商品細部資訊
Exist_flag	boolean		not null	商品是否上架 (0: 下架, 1: 上架)

SHIPPING_DISCOUNT				
Description:存放運費折扣資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**

Discount_code	int	primary	not null	折扣編號
Vender_id	int	foreign	not null	會員(商家)編號 reference（USER）
Policy_description	varchar		not null	折扣規則描述
Start_time	timestamp		not null	折扣開始時間
End_time	timestamp		not null	折扣結束時間
Target_price	int		not null	目標金額

SEASONINGS_DISCOUNT				
Description:存放季節性折扣資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Discount_code	int	primary	not null	折扣編號
Vender_id	int	foreign	not null	會員(商家)編號 reference（USER）
Policy_description	varchar		not null	折扣規則描述
Start_time	timestamp		not null	折扣開始時間
End_time	timestamp		not null	折扣結束時間
Discount_rate	double		not null	折扣比例

SPECIAL_DISCOUNT				
Description:存放特殊折扣資料				
Attribute	**Type**	**Key**	**Nullable**	**Description**
Discount_code	int	primary	not null	折扣編號
Vender_id	int	foreign	not null	會員(商家)編號 reference（USER）
Policy_description	varchar		not null	折扣規則描述
Start_time	timestamp		not null	折扣開始時間
End_time	timestamp		not null	折扣結束時間

Category	varchar		not null	折扣適用商品類別
Discount_rate	double		not null	折扣比例

4.2 Expectation of the possible DB operations, frequencies and data volumes

Table	可能操作	預估使用頻率 (per day)	表格資料量	系統負擔
User	使用者登入驗證	100	1000	100000 (Query/Day)
User	新增使用者資料	5	1000	5 (Insert/Day)
User	更新使用者資料	5	1000	5000 (Query/Day) 5 (Update/Day)
User	刪除使用者資料	1	1000	1000 (Query/Day) 1 (Delete/Day)
Shopping_cart	加入商品	150	500	150 (Insert/Day)
Shopping_cart	查詢購物車	50	500	25000 (Query/Day)
Shopping_cart	變更購買數量	100	500	50000 (Query/Day) 100 (Update/Day)
Shopping_cart	移除購買商品	150	500	50000 (Query/Day) 100 (Update/Day)
Order	新增訂單	10	5000	10 (Insert/Day)
Order	修改訂單狀態	10	5000	50000 (Query/Day) 10 (Update/Day)
Order	取消訂單	1	5000	5000 (Query/Day) 1 (Update/Day)
Order	查詢訂單紀錄	10	5000	50000 (Query/Day)
Order_items	新增訂單	10	10000	10 (Insert/Day)
Order_item	查詢訂單內容	10	10000	100000 (Query/Day)
Order_items	刪除訂單	1	10000	10000 (Query/Day) 1 (Delete/Day)
Manage_order	新增訂單	10	5000	10 (Insert/Day)
Manage_order	取消訂單	1	5000	5000 (Query/Day) 1 (Delete/Day)
Valuation	新增評價	10	1000	10 (Insert/Day)
Product	新增商品	Not often	1000	Don’t care
Product	更新商品資訊	5	1000	5000 (Query/Day) 5 (Update/Day)
Product	移除商品	Not often	1000	Don’t care
Product	查詢商品	500	1000	50000 (Query/Day)
Shipping_discount	新增折扣	Not often	50	Don’t care

Shipping_discount	更新折扣資訊	1	50	50 (Query/Day) 1 (Update/Day)
Shipping_discount	移除折扣	Not often	50	Don’t care
Seasonings_discount	新增折扣	Not often	30	Don’t care
Seasonongs_discount	更新折扣資訊	1	30	30 (Query/Day) 1 (Update/Day)
Seasonings_discount	移除折扣	Not often	30	Don’t care
Special_discount	新增折扣	Not often	50	Don’t care
Special_discount	更新折扣資訊	1	50	50 (Query/Day) 1 (Update/Day)
Special_discount	移除折扣	Not often	50	Don’t care

4.3 Expectation of the Database size

Table	Size (in terms of Kbyte)
User	16
Shopping_cart	32
Order	64
Order_items	32
Manage_order	32
Valuation	32
Product	32
Shipping_discount	32
Seasonings_discount	32
Special_discount	48

4.4 SQL Statements Used to Construct the Schema

```sql
-- Table structure for table `user`
CREATE TABLE IF NOT EXISTS `user`
(
    `id`          int(10) PRIMARY KEY AUTO_INCREMENT,
    `username`    varchar(255) NOT NULL,
    `identity`    varchar(30)  NOT NULL,
    `password`    varchar(255) NOT NULL,
    `phone`       varchar(30)  NOT NULL,
    `email`       varchar(255) NOT NULL,
    `credit_card` varchar(20)  DEFAULT NULL,
    `address`     varchar(255) DEFAULT NULL
);
```

```sql
-- Table structure for table `product`
CREATE TABLE IF NOT EXISTS `product`
(
    `id`                int(10) PRIMARY KEY AUTO_INCREMENT,
    `vender_id`         int(10)      NOT NULL,
    `name`              varchar(255) NOT NULL,
    `category`          varchar(255) NOT NULL,
    `price`             int(20)      NOT NULL,
    `stock`             int(6)       NOT NULL,
    `warehouse_address` varchar(100) NOT NULL,
    `description`       varchar(500) DEFAULT NULL,
    `pictureURL`        varchar(500) DEFAULT NULL,
    `exist_flag`        boolean         NOT NULL,
    FOREIGN KEY (vender_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `shipping_discount`
CREATE TABLE IF NOT EXISTS `shipping_discount`
(
    `discount_code`      int(10) PRIMARY KEY AUTO_INCREMENT,
    `vender_id`          int(10)      NOT NULL,
    `policy_description` varchar(500) NOT NULL,
    `start_time`         TIMESTAMP    NOT NULL,
    `end_time`           TIMESTAMP    NOT NULL,
    `target_price`       int(20)      NOT NULL,
    FOREIGN KEY (vender_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `seasonings_discount`
CREATE TABLE IF NOT EXISTS `seasonings_discount`
(
    `discount_code`      int(10) PRIMARY KEY AUTO_INCREMENT,
    `vender_id`          int(10)      NOT NULL,
    `policy_description` varchar(500) NOT NULL,
    `start_time`         TIMESTAMP    NOT NULL,
    `end_time`           TIMESTAMP    NOT NULL,
    `discount_rate`      double       NOT NULL,
    FOREIGN KEY (vender_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `special_discount`
CREATE TABLE IF NOT EXISTS `special_discount`
(
    `discount_code`      int(10) PRIMARY KEY AUTO_INCREMENT,
    `vender_id`          int(10)      NOT NULL,
    `policy_description` varchar(500) NOT NULL,
    `start_time`         TIMESTAMP    NOT NULL,
    `end_time`           TIMESTAMP    NOT NULL,
    `category`           varchar(255) NOT NULL,
    FOREIGN KEY (vender_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `order`
CREATE TABLE IF NOT EXISTS `order`
(
    `id`                    int(10) PRIMARY KEY AUTO_INCREMENT,
    `customer_id`           int(10)      NOT NULL,
    `shipping_fee`          int(5)       NOT NULL,
    `recipient_name`        varchar(20)  NOT NULL,
    `shipping_address`      varchar(255) NOT NULL,
    `status`                varchar(20)  NOT NULL,
    `payment_method`        varchar(20)  NOT NULL,
    `credit_card_id`        varchar(20)          DEFAULT NULL,
    `order_time`            TIMESTAMP    NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `shipping_time`         TIMESTAMP    NULL     DEFAULT NULL,
    `delivery_time`         TIMESTAMP    NULL     DEFAULT NULL,
    `seasoning_discount_code` int(6)              DEFAULT NULL,
    `shipping_discount_code` int(6)               DEFAULT NULL,
    `total_price`           int(6)               NOT NULL,
    FOREIGN KEY (seasoning_discount_code) REFERENCES `seasonings_discount` (`discount_code`) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (shipping_discount_code) REFERENCES `shipping_discount` (`discount_code`) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (customer_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `order_items`
CREATE TABLE IF NOT EXISTS `order_items`
(
    `order_id`   int(10),
    `product_id` int(10),
    `quantity`   int(6) NOT NULL,
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES `order` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES `product` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `shopping_cart`
CREATE TABLE IF NOT EXISTS `shopping_cart`
(
    `product_id`  int(10),
    `customer_id` int(10),
    `quantity`    int(6) NOT NULL,
    PRIMARY KEY (product_id, customer_id),
    FOREIGN KEY (product_id) REFERENCES `product` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `manage_order`
CREATE TABLE IF NOT EXISTS `manage_order`
(
    `order_id`  int(10),
    `vender_id` int(10),
    PRIMARY KEY (order_id, vender_id),
    FOREIGN KEY (order_id) REFERENCES `order` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (vender_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Table structure for table `valuation`
CREATE TABLE IF NOT EXISTS `valuation`
(
    `customer_id` int(10),
    `product_id`  int(10),
    `comment`     varchar(500) NOT NULL,
    `rating`      int(1)       NOT NULL,
    PRIMARY KEY (customer_id, product_id),
    FOREIGN KEY (customer_id) REFERENCES `user` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES `product` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.5 SQL Statements Used to Insert the data - data population

```sql
INSERT `user` (`username`, `identity`, `password`, `phone`, `email`, `credit_card`, `address`)
VALUES ('Admin', 'admin', 'Admin123', '0912341234', 'admin@gmail.com', '0000111-2222333', 'home'),
       ('Apple', 'staff', 'Apple123', '0912345678', 'apple@gmail.com', '1111222-3333444', '台北市大安區忠孝東路xxx號5F'),
       ('ASUS', 'staff', 'ASUS123', '0900112233', 'google@gmail.com', NULL, '台北市大安區忠孝東路xxx號5F'),
       ('Samsung', 'staff', 'Samsung123', '0909090909', 'acer@gmail.com', NULL, '台北市大安區忠孝東路xxx號5F'),
       ('Zachary', 'customer', 'Zachary123', '0943214321', 'zachary@gmail.com', '2222333-4444555', '台北市大安區忠孝東路xxx號5F'),
       ('Mandy', 'customer', 'Mandy123', '0987654321', 'mandy@gmail.com', NULL, '台北市大安區忠孝東路xxx號5F'),
       ('Patrick', 'customer', 'Patrick123', '0943214321', 'patrick@gmail.com', '1122333-4444555', '台北市大安區忠孝東路xxx號5F'),
       ('Sandy', 'customer', 'Sandy123', '0943214321', 'sandy@gmail.com', '8822333-4444555', '台北市大安區忠孝東路xxx號5F'),
       ('David', 'customer', 'David123', '0987654321', 'david@gmail.com', NULL, '台北市大安區忠孝東路xxx號5F'),
       ('Teddy', 'customer', 'Teddy123', '0987654321', 'teddy@gmail.com', '8822333-4444555', '台北市大安區忠孝東路xxx號5F');
```

```sql
INSERT `shipping_discount` (`vender_id`, `policy_description`, `start_time`, `end_time`, `target_price`)
VALUES (2, '2021/12/18當日結帳金額超過10000免運費', '2021-12-18 00:00:00', '2021-12-18 23:59:59', 10000),
       (3, '雙十一結帳金額超過15000免運', '2021-11-07 00:00:00', '2021-11-13 23:59:59', 15000),
       (4, '聖誕節結帳金額超過30000運費折抵', '2021-12-20 00:00:00', '2021-12-31 23:59:59', 30000);
```

```sql
INSERT `seasonings_discount` (`vender_id`, `policy_description`, `start_time`, `end_time`, `discount_rate`)
VALUES (2, '開學季全面9折', '2020-08-01 00:00:00', '2020-09-30 23:59:59', 0.9),
       (3, '春節特賣全商品享79折優惠', '2021-02-01 00:00:00', '2021-02-28 23:59:59', 0.79),
       (4, '母親節活動全店88折', '2021-05-01 00:00:00', '2021-05-31 23:59:59', 0.88);
```

```sql
INSERT `special_discount` (`vender_id`, `policy_description`, `start_time`, `end_time`, `category`)
VALUES (2, 'iPhone 13 Pro特惠出清', '2021-08-01 00:00:00', '2021-09-30 23:59:59', 'PHONE'),
       (3, 'TUF Dash F15折扣', '2021-02-01 00:00:00', '2021-02-28 23:59:59', 'NOTEBOOK'),
       (4, 'Galaxy Z Fold3 5G上市優惠', '2021-05-01 00:00:00', '2021-05-31 23:59:59', 'TABLET');
```

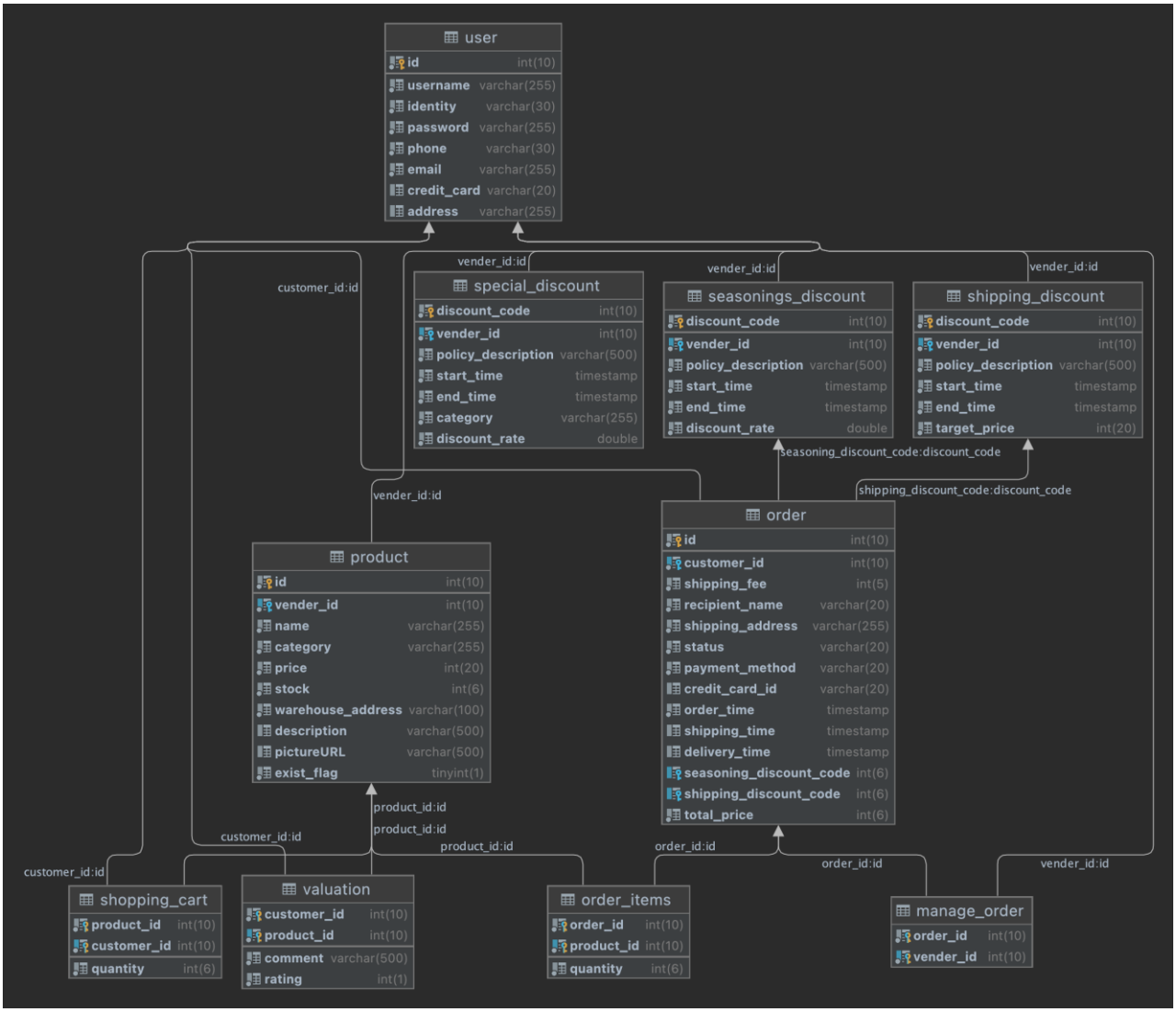
```sql
INSERT `order_items` (`order_id`, `product_id`, `quantity`)
VALUES (1, 7, 1),
       (2, 13, 2),
       (3, 9, 1),
       (4, 4, 1),
       (5, 11, 3);

INSERT `shopping_cart` (`product_id`, `customer_id`, `quantity`)
VALUES (3, 5, 3),
       (8, 6, 1),
       (12, 5, 1);

INSERT `manage_order` (`order_id`, `vender_id`)
VALUES (1, 3),
       (2, 4),
       (3, 3),
       (4, 2),
       (5, 4);
```

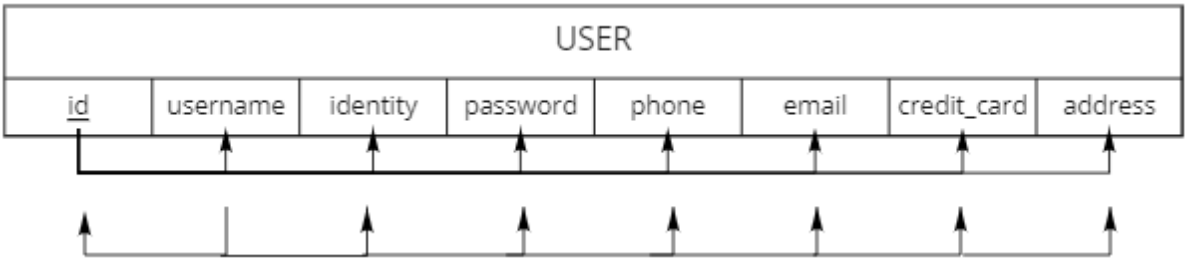
```
INSERT `valuation` (`customer_id`, `product_id`, `comment`, `rating`)
VALUES (4, 1, 'Not really well.', 2),
       (5, 1, 'Not really well.', 3),
       (5, 2, 'Good.', 4),
       (6, 2, 'Excellent!', 5),
       (7, 4, 'Good.', 4),
       (7, 3, 'Excellent!', 5),
       (8, 4, 'Good.', 4),
       (8, 1, 'Gooood!', 4),
       (8, 5, 'Excellent!', 5),
       (9, 6, 'very good!', 5),
       (4, 10, 'very good!', 4),
       (9, 11, 'Excellent!', 5),
       (9, 12, 'so good!', 5),
       (9, 13, 'very good!', 4);
```

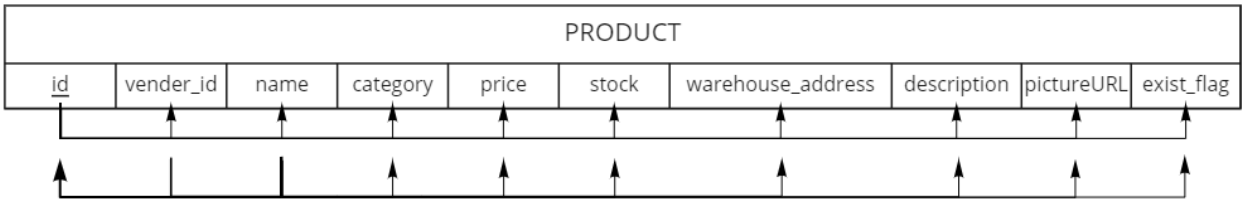
4.6 The implementation of tables in target DBMS

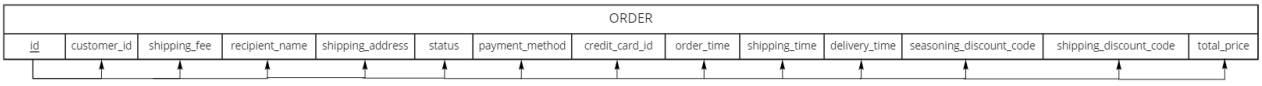


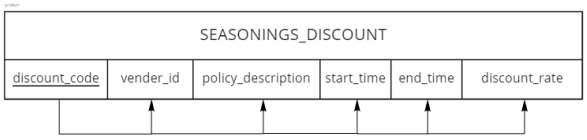
Section 5 功能性依賴(Functional Dependencies and Database Normalization)

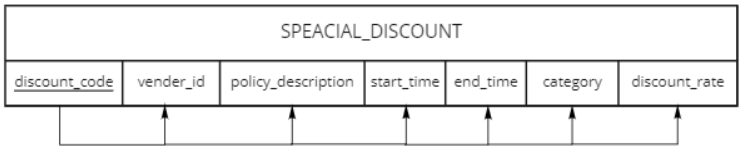
5.1 Functional Dependencies

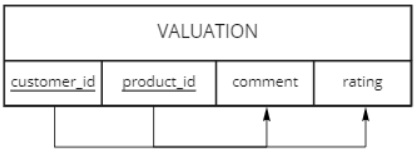












Section 6 系統使用者指南 (The User Guide of the System)

6.1 System Installation Description

Clone our source code from the github site below, and follow the instructions on README to install our system.

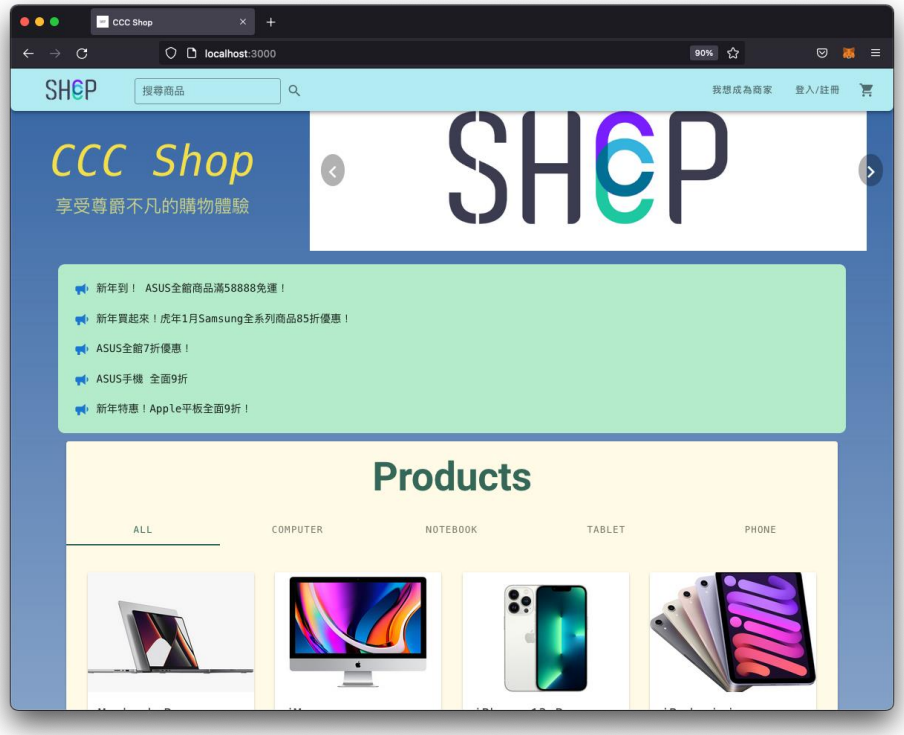
Frontend: https://github.com/CCC-Shop/ccc-shop-frontend.git

Backend: https://github.com/CCC-Shop/ccc-shop-backend.git

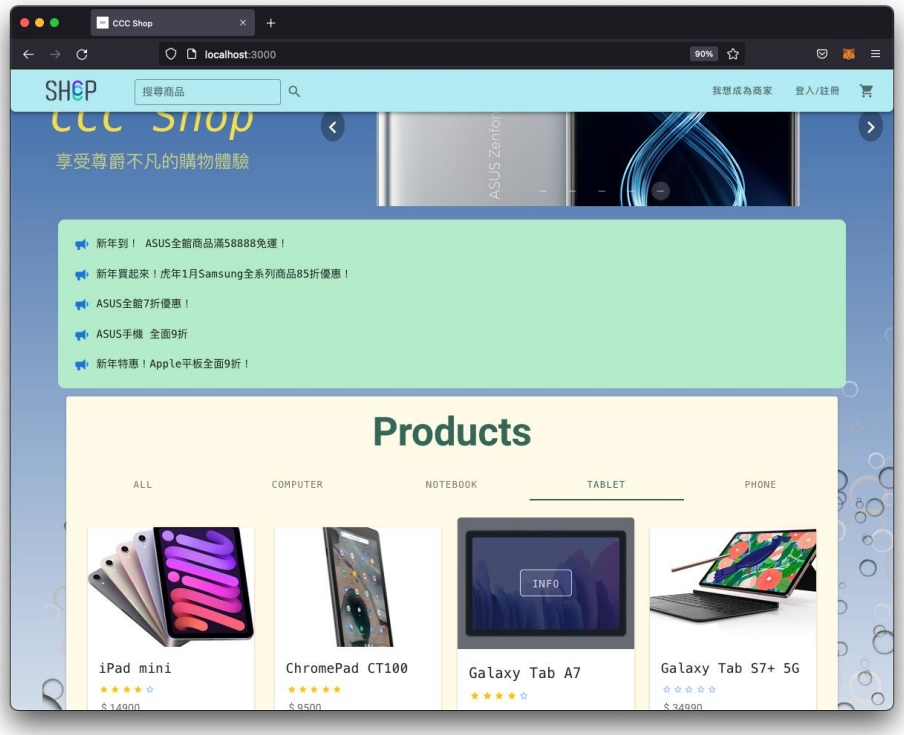
6.2 The Use of the System

6.2.1 訪客使用者介面 (Guest)

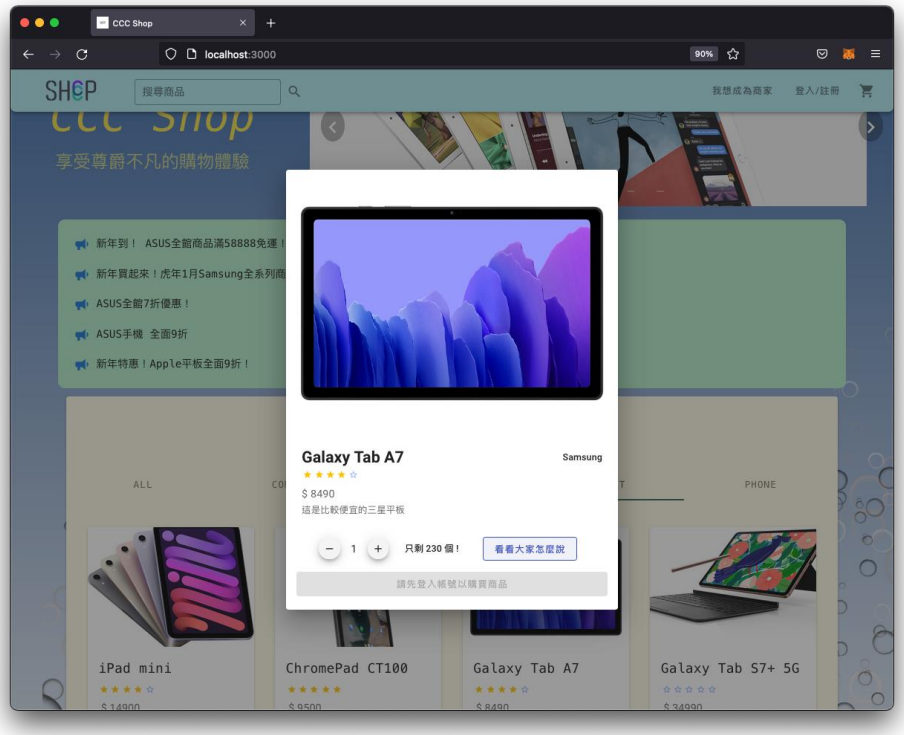
A. View All Products



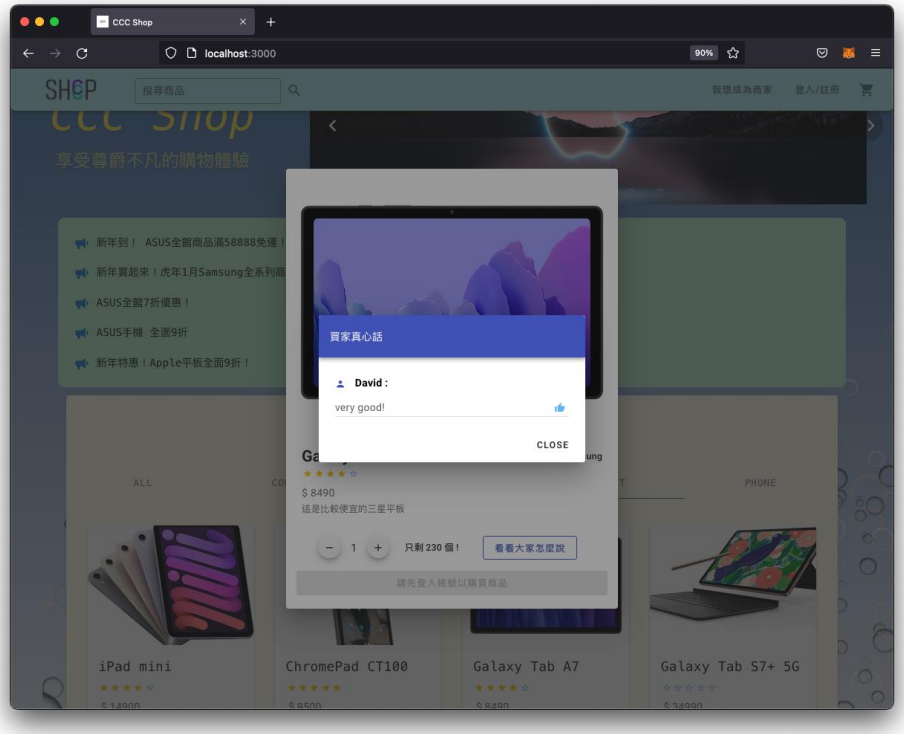
B. View Category Products



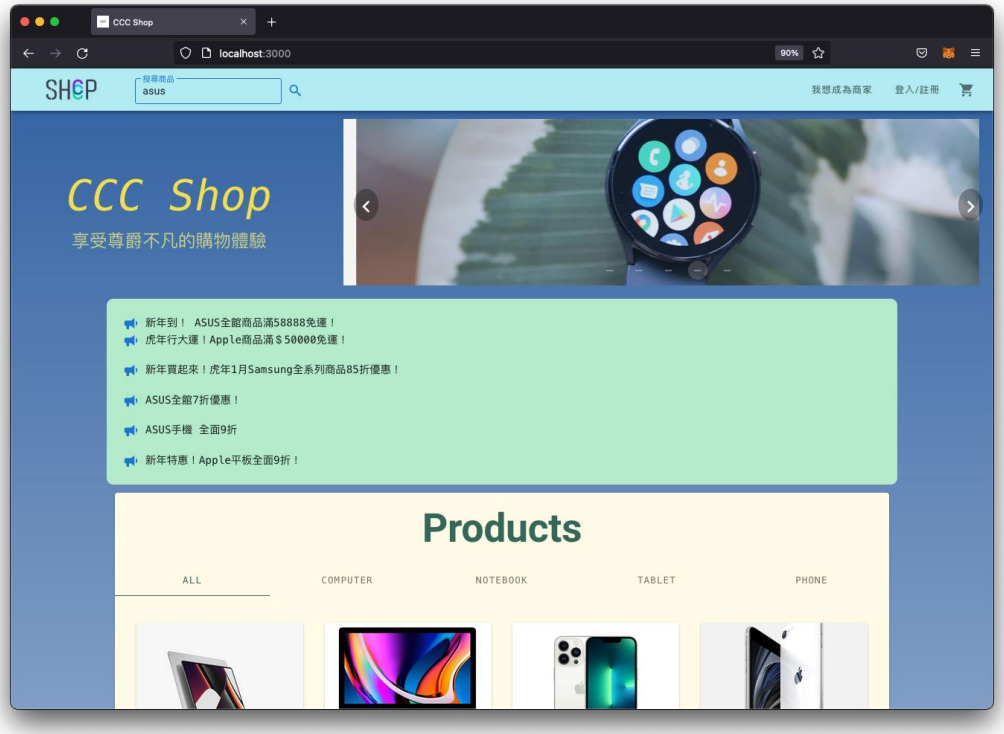
C. View Product Info



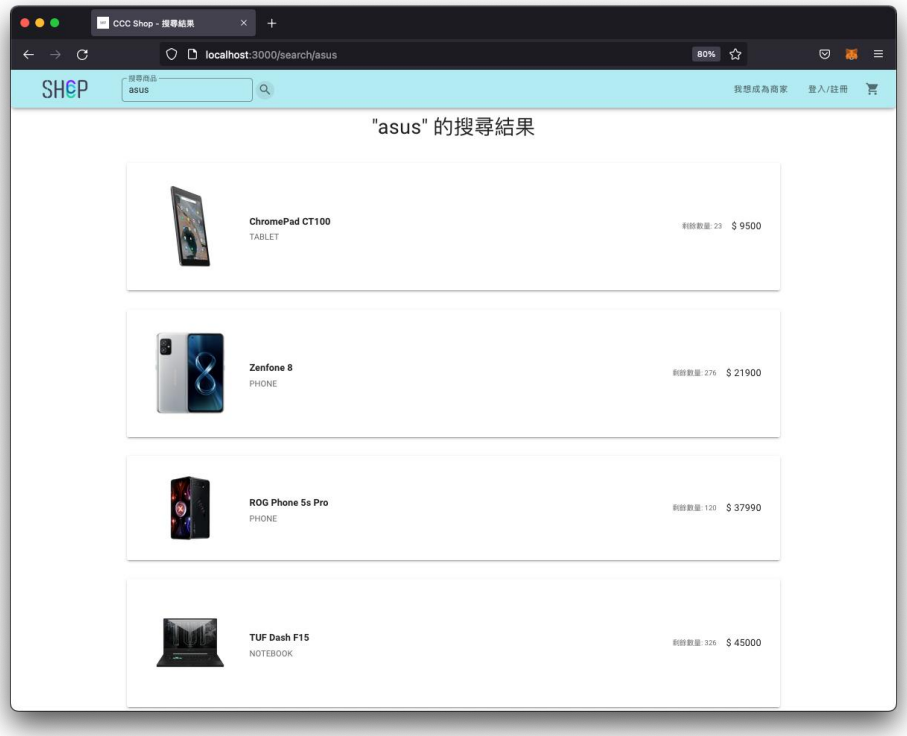
D. View Product Valuation



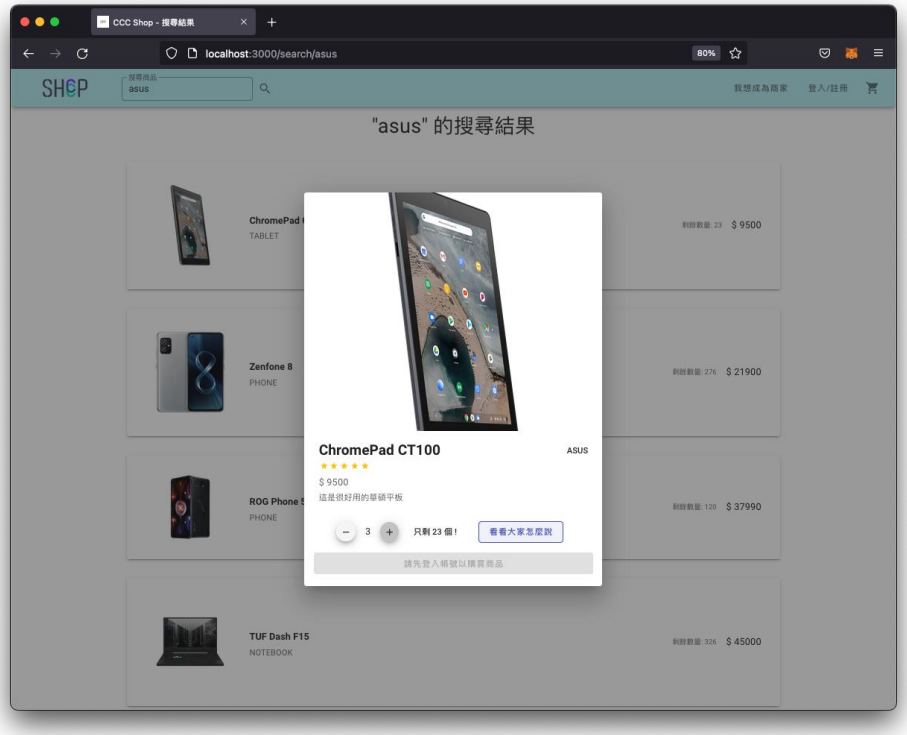
E. Search Product



F. Search Product Result

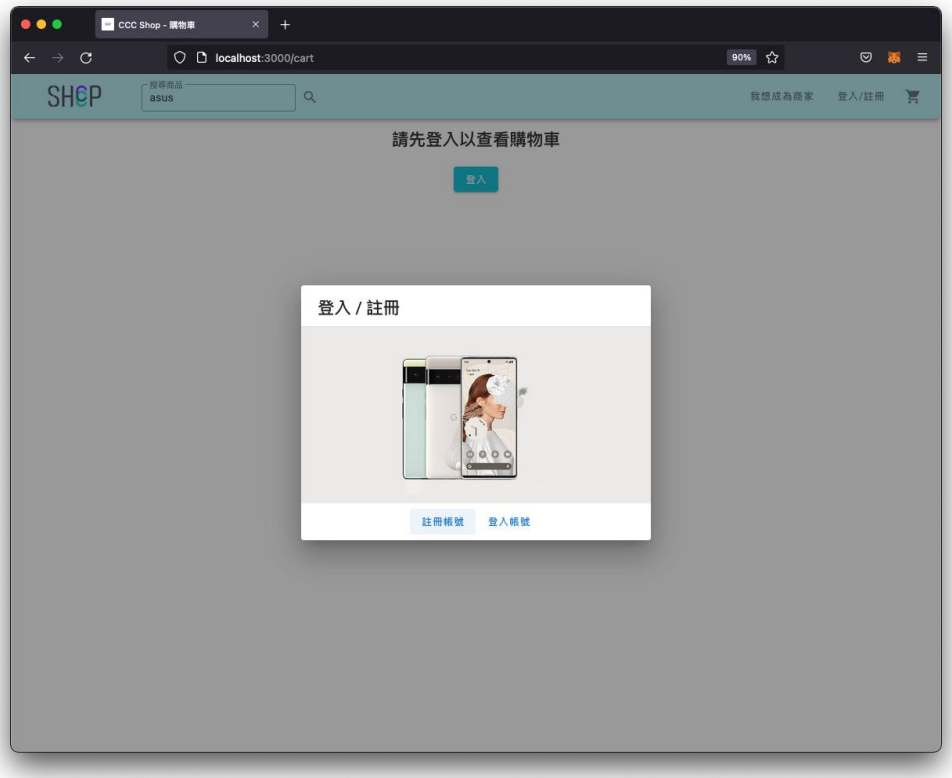


G. Search Product Info

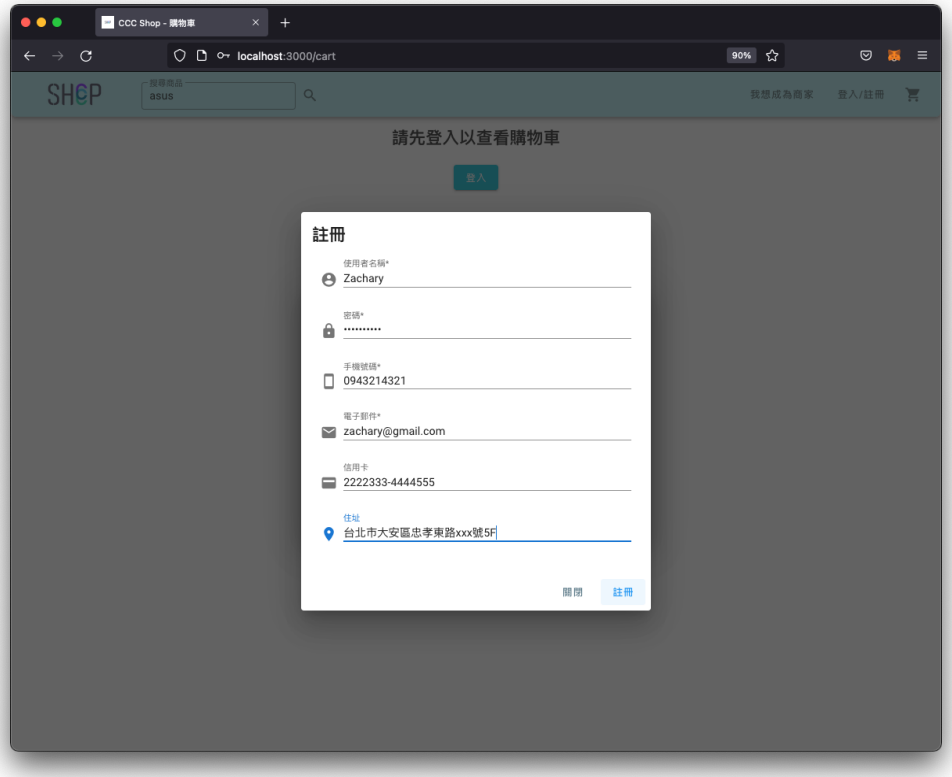


6.2.2 顧客使用者介面 (Customer)

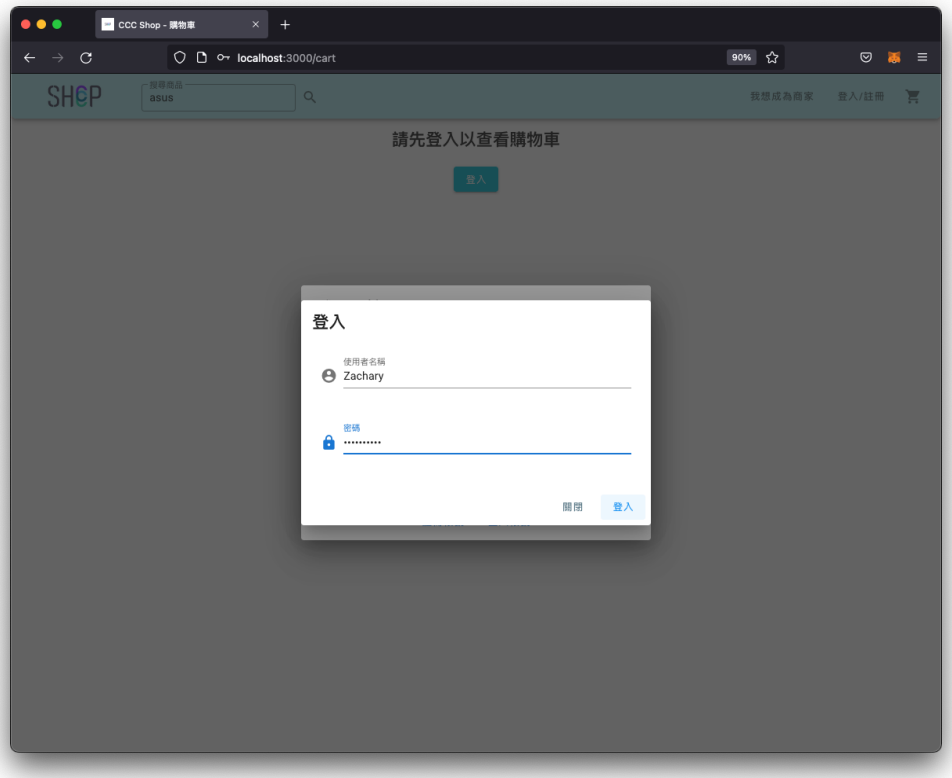
A. Register



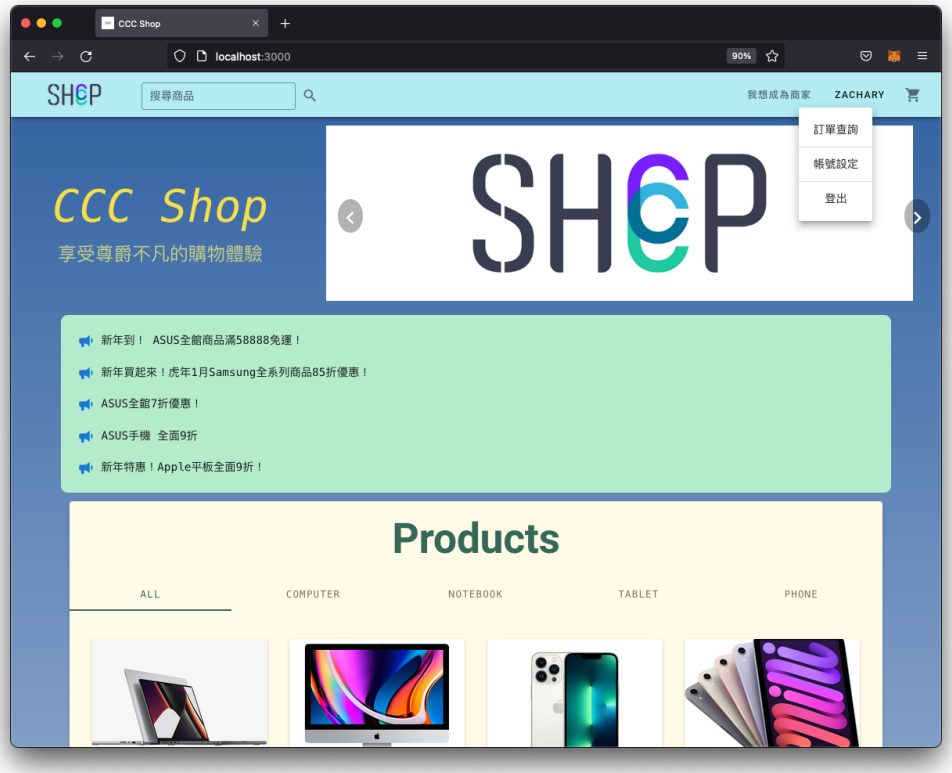
B. Register Customer



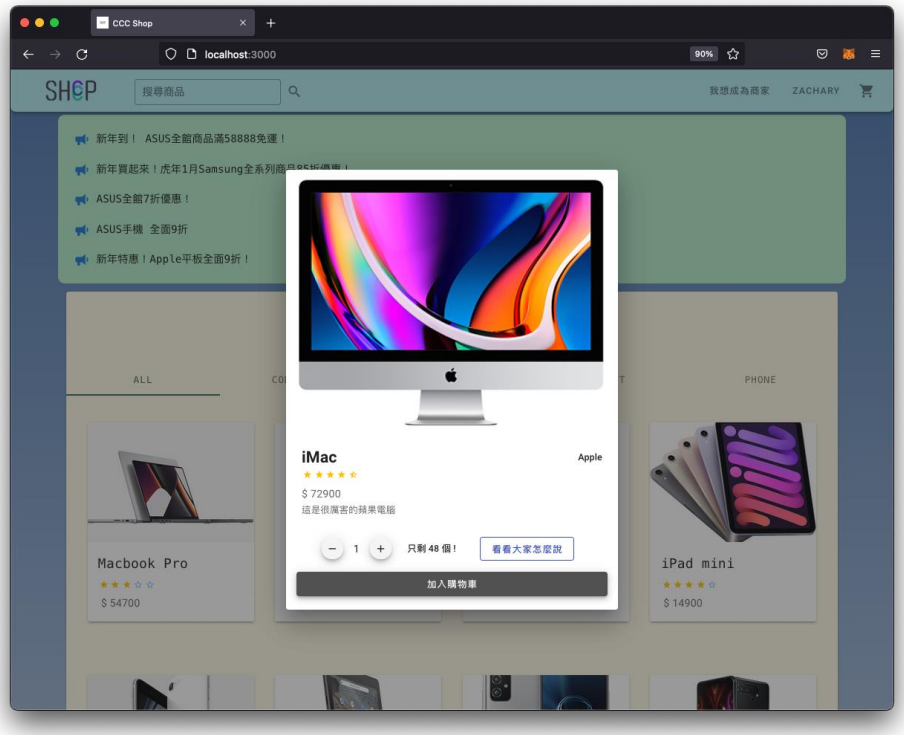
C. Login



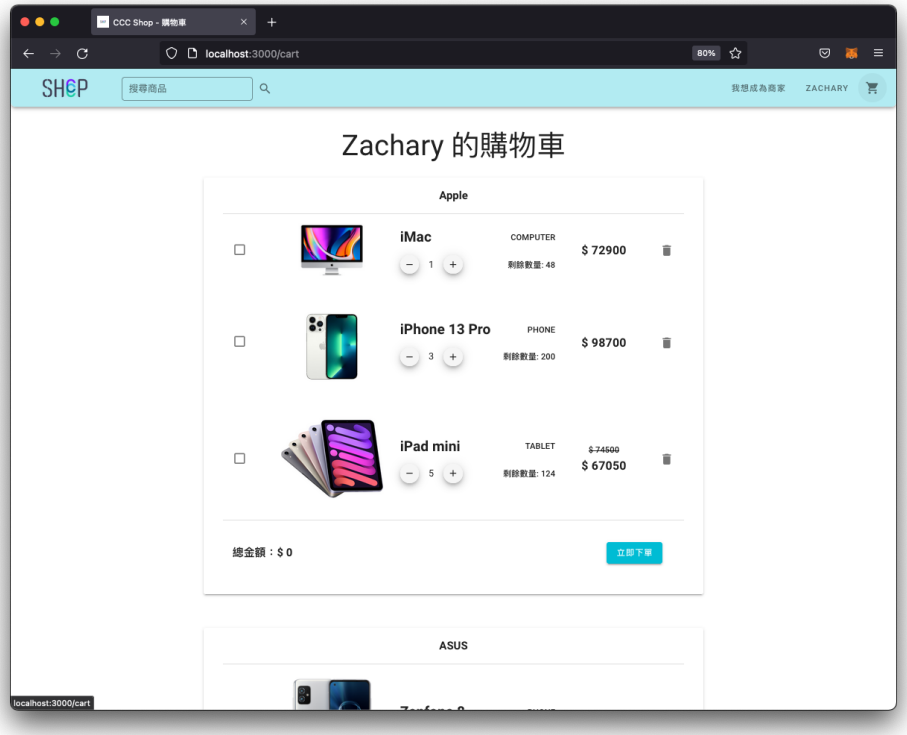
D. Login Page



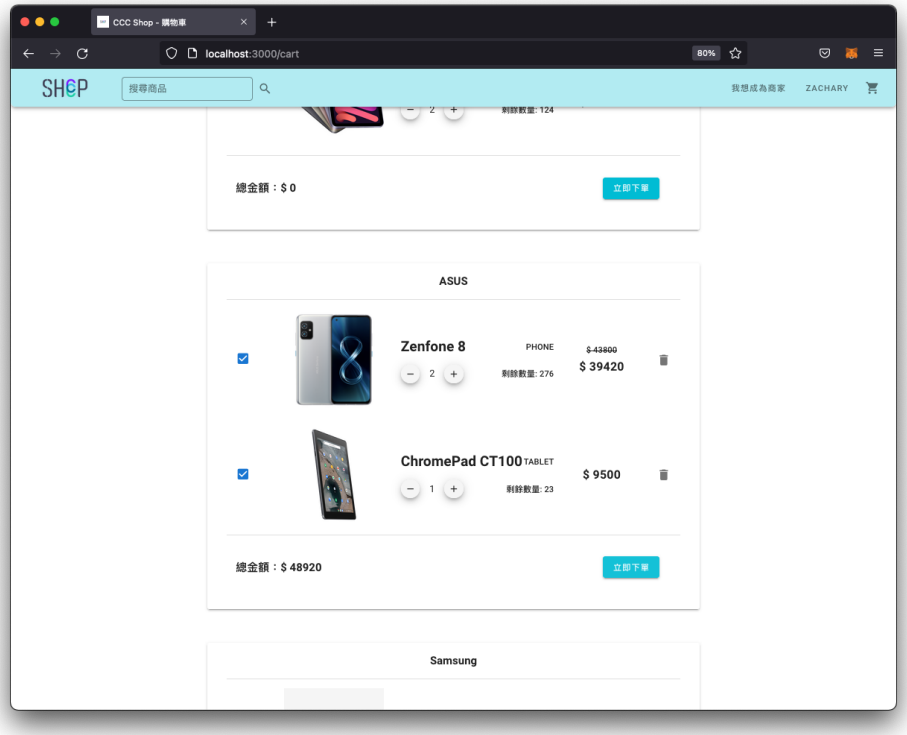
E. Add Product To Cart



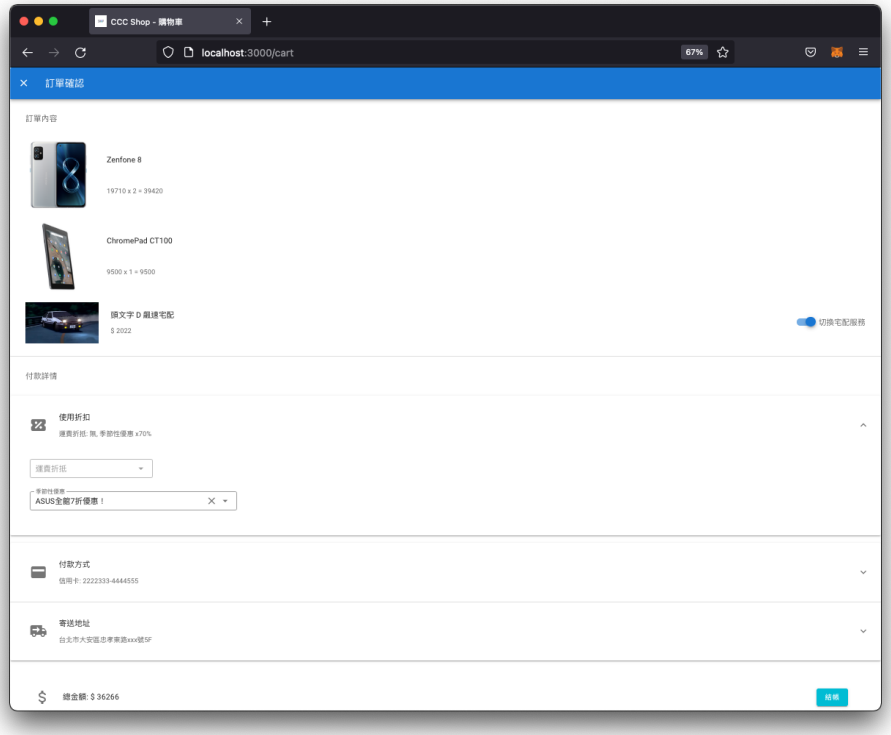
F. Shopping Cart Page



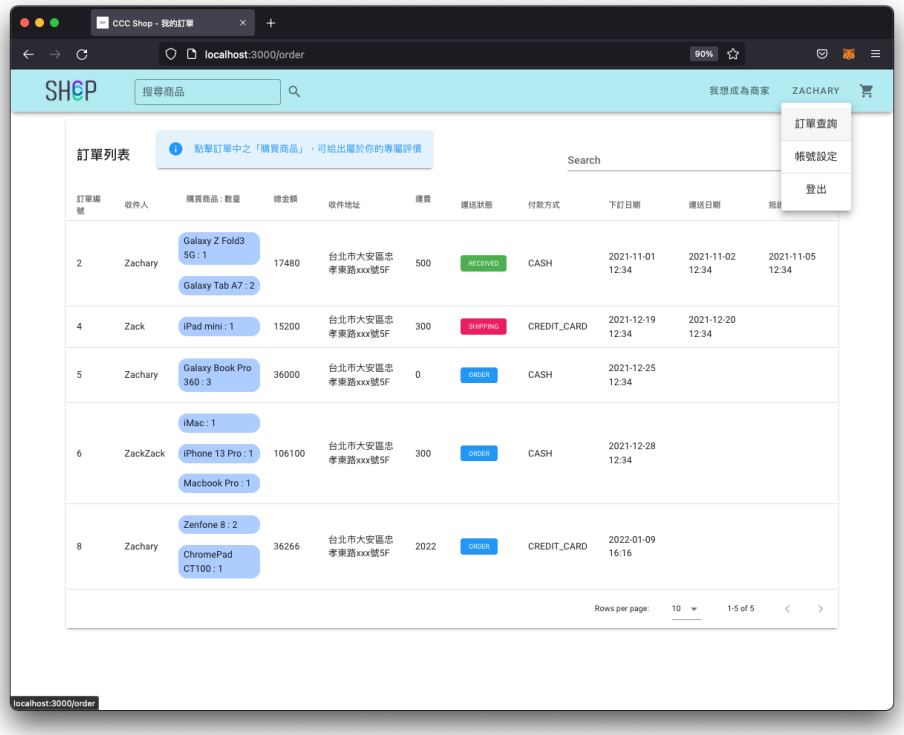
G. Make Order



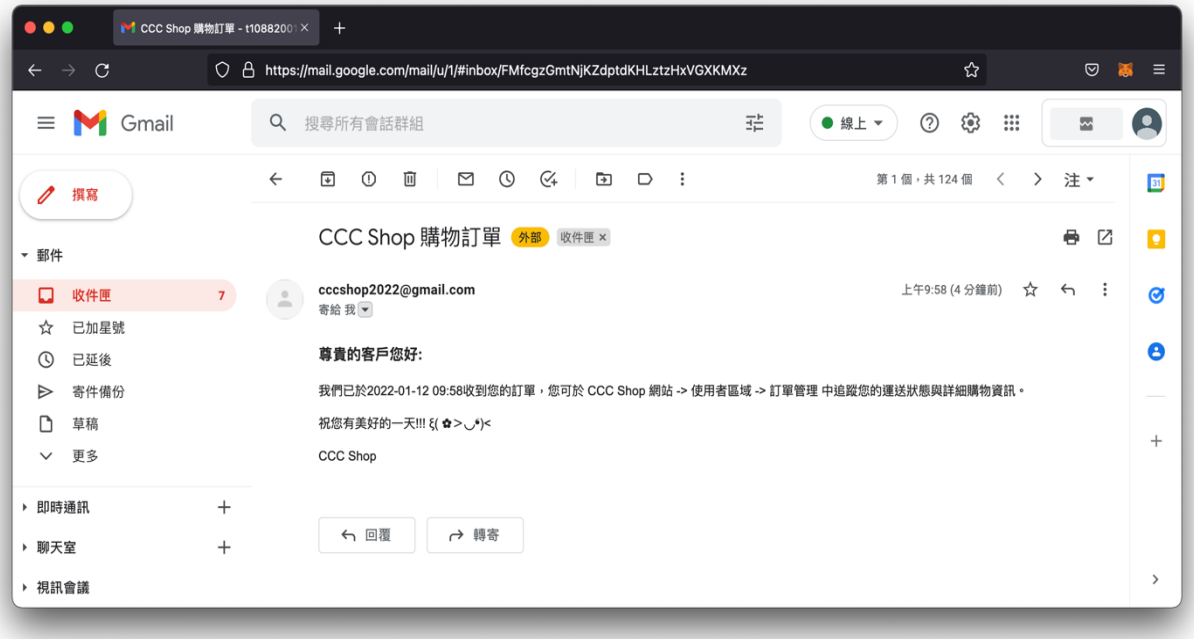
H. Checkout Order



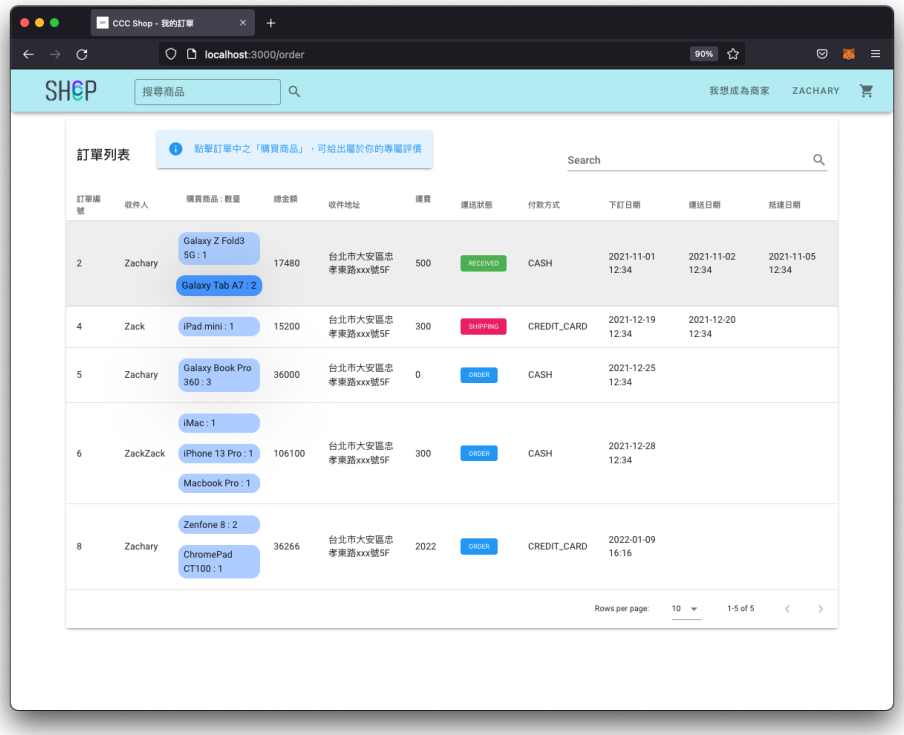
I. View Order History



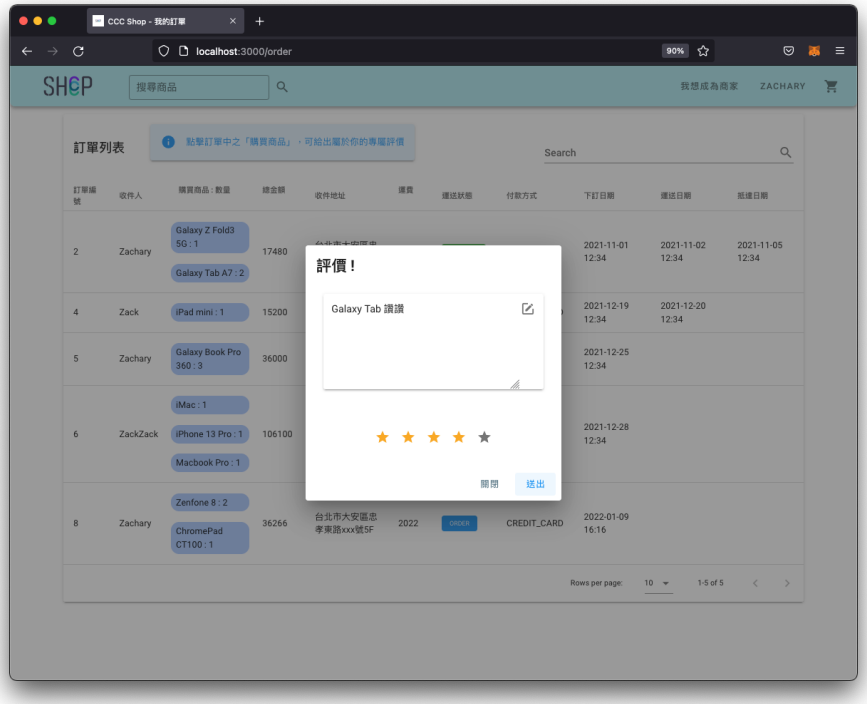
J. Order Created Message Sent to Customer



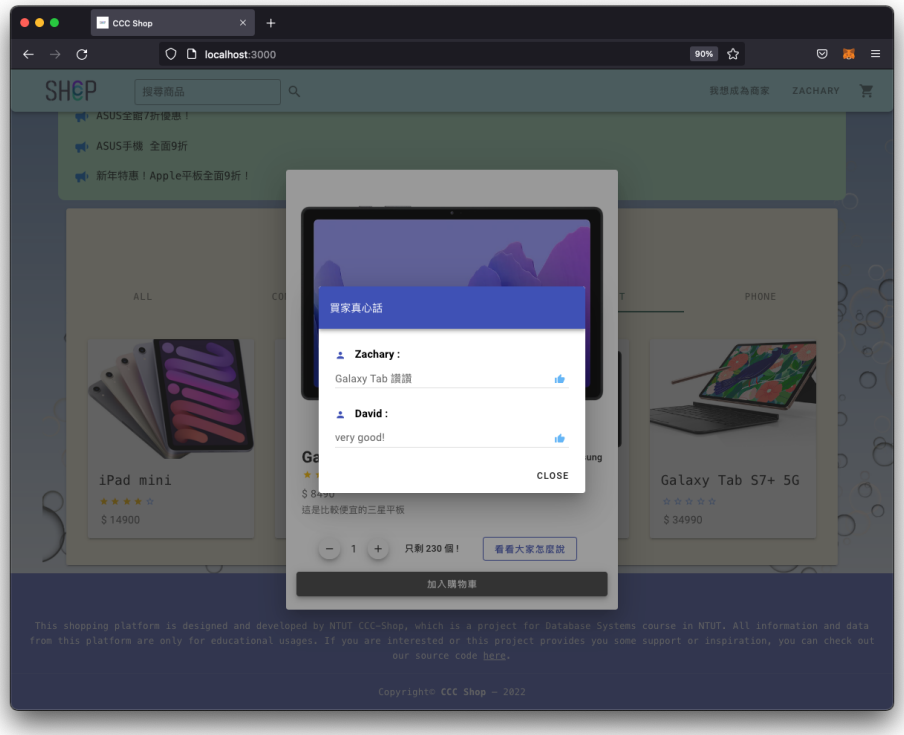
K. Give Valuation



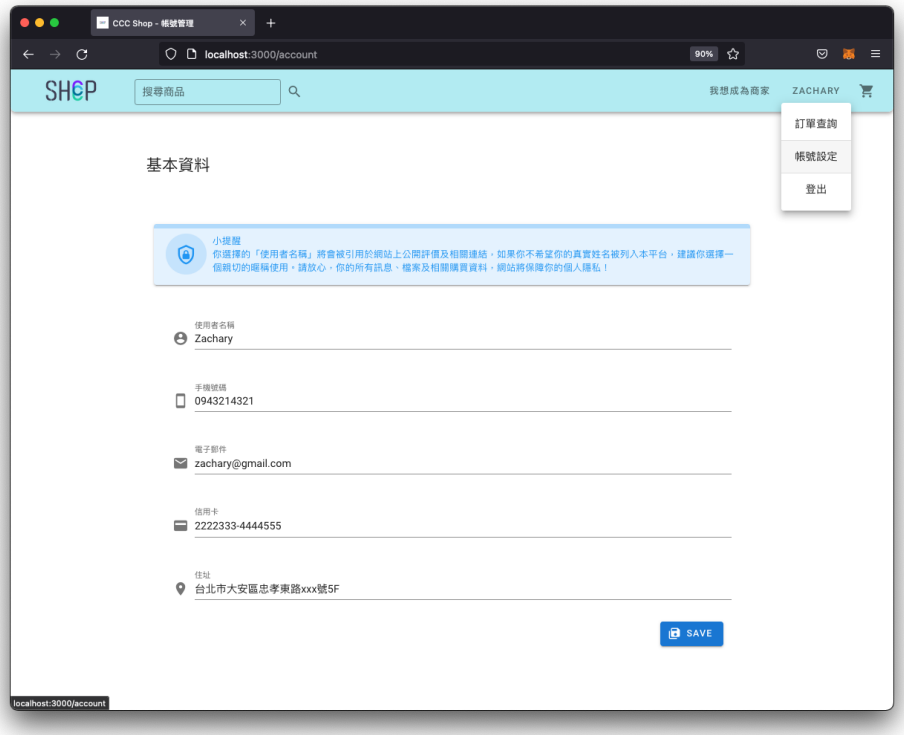
L. Write Comment



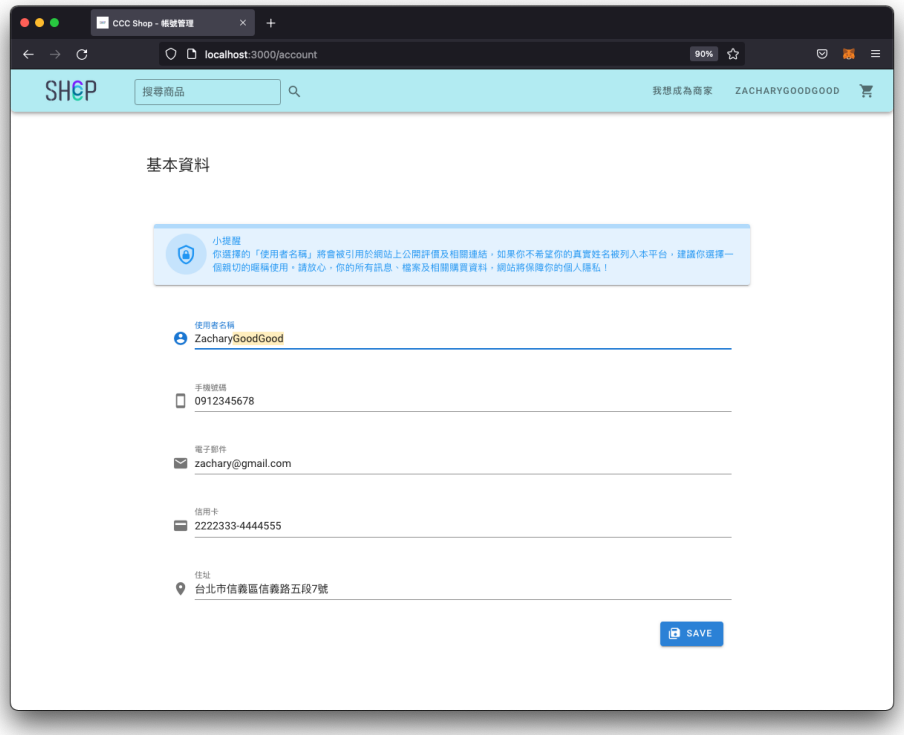
M. Valuation Result



N. Edit Profile Page

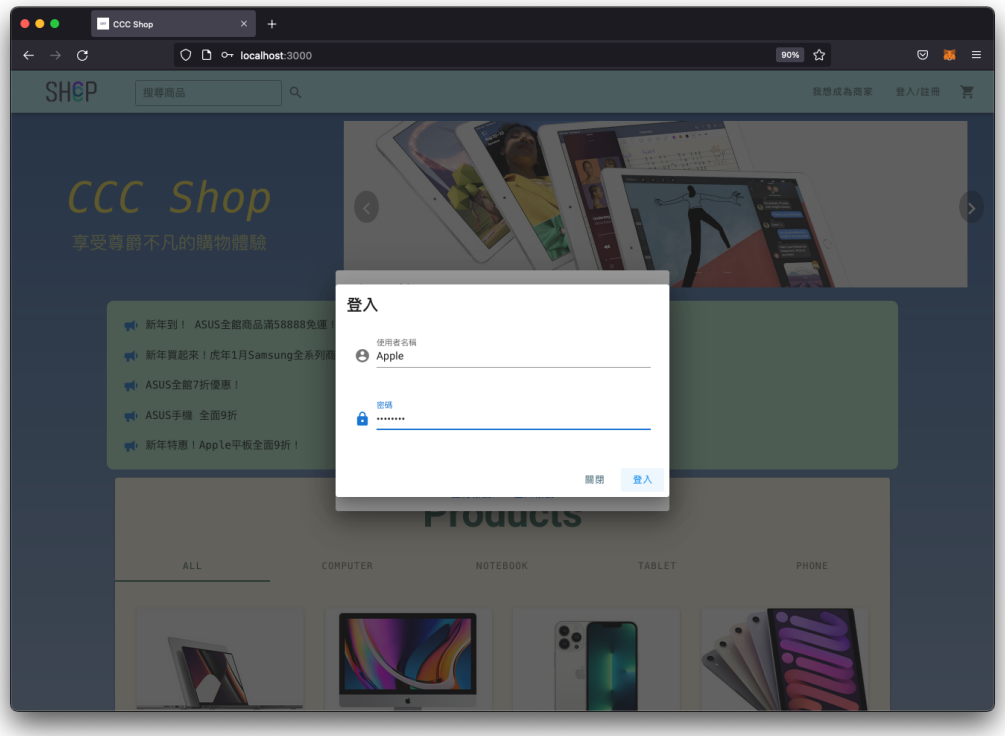


O. Change User Info

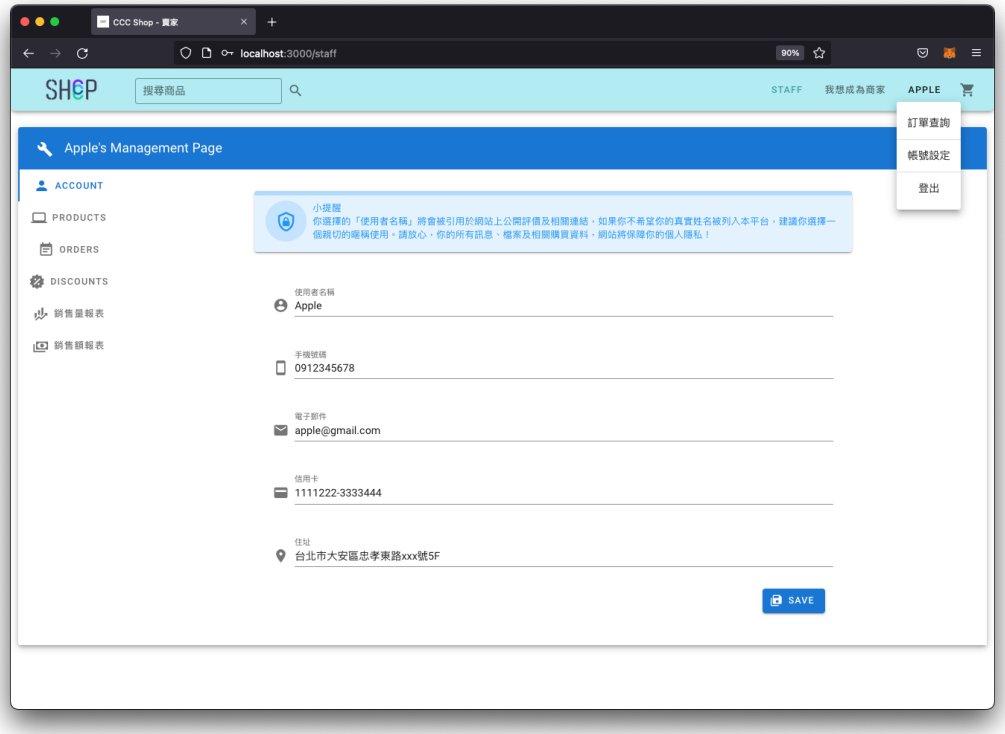


6.2.3 商家使用者介面 (Staff)

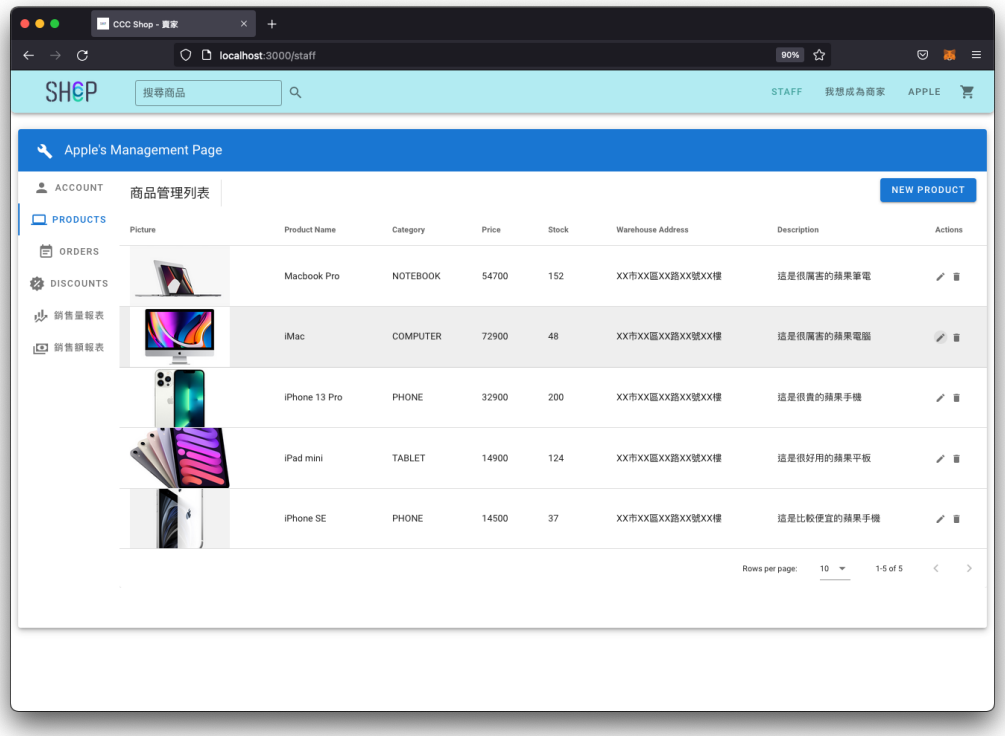
A. Login



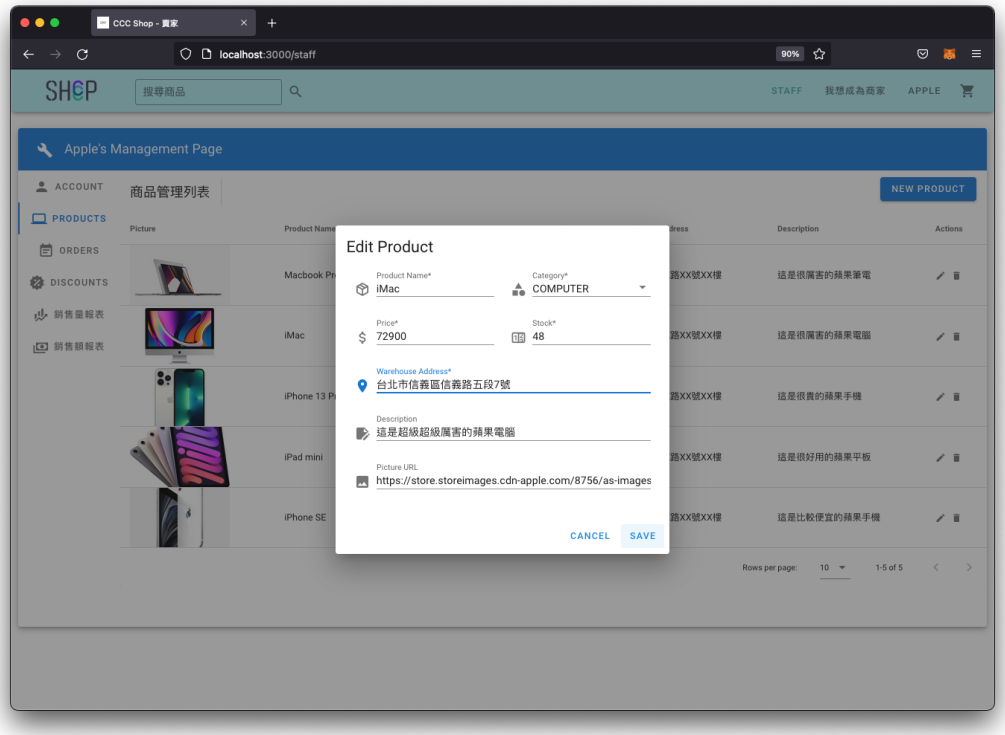
B. Staff Page



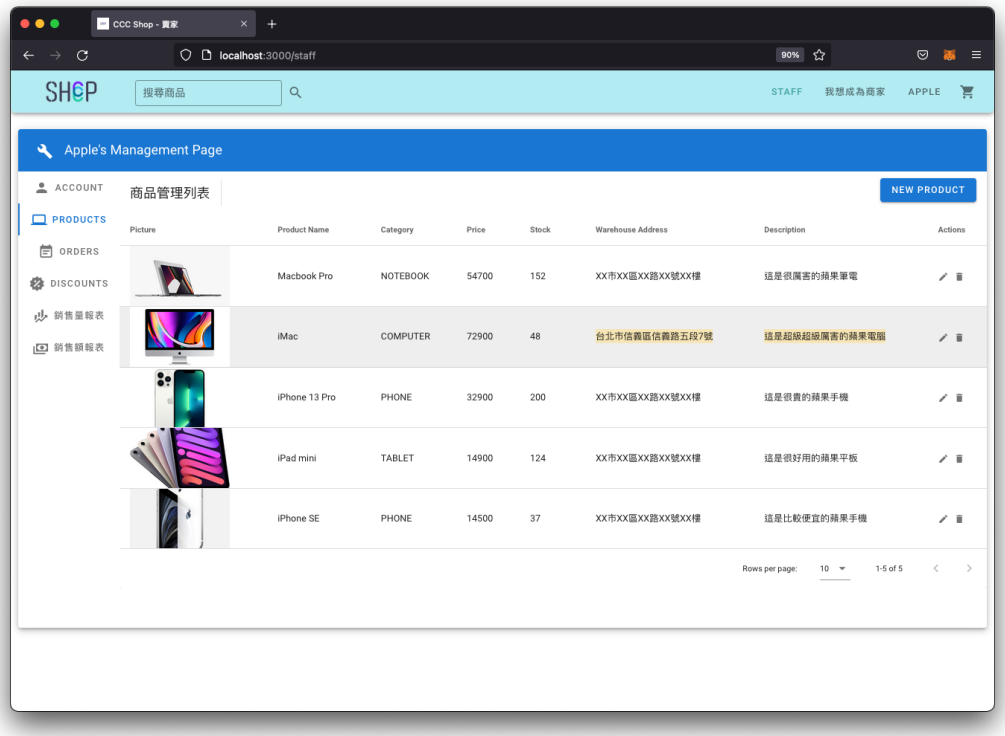
C. Edit Product Info



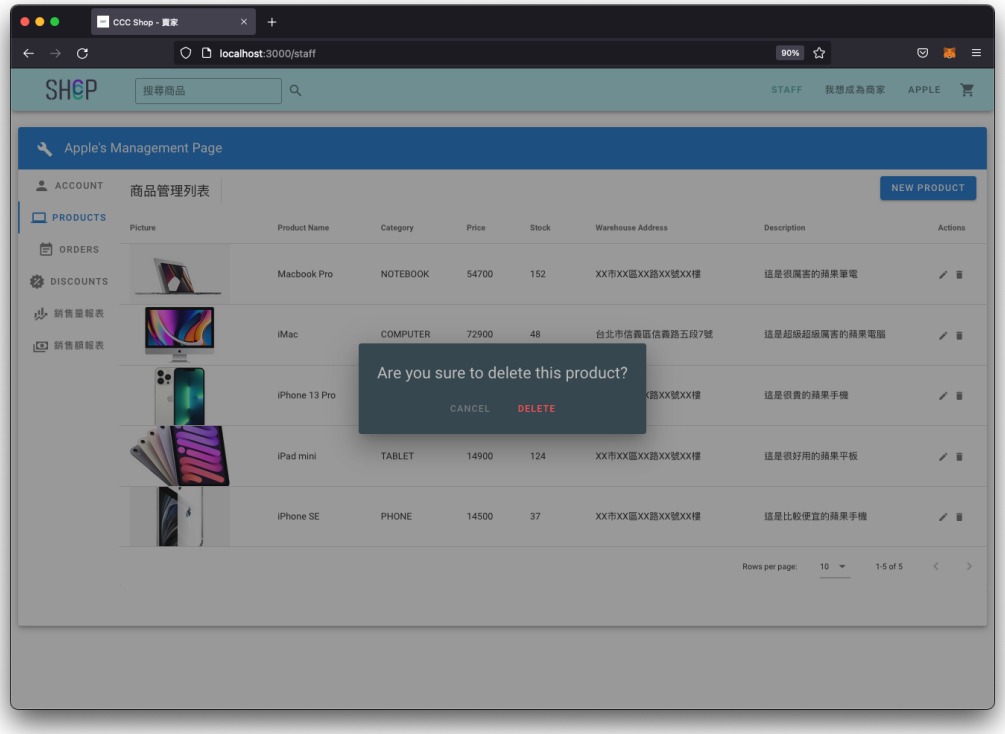
D. Change Description



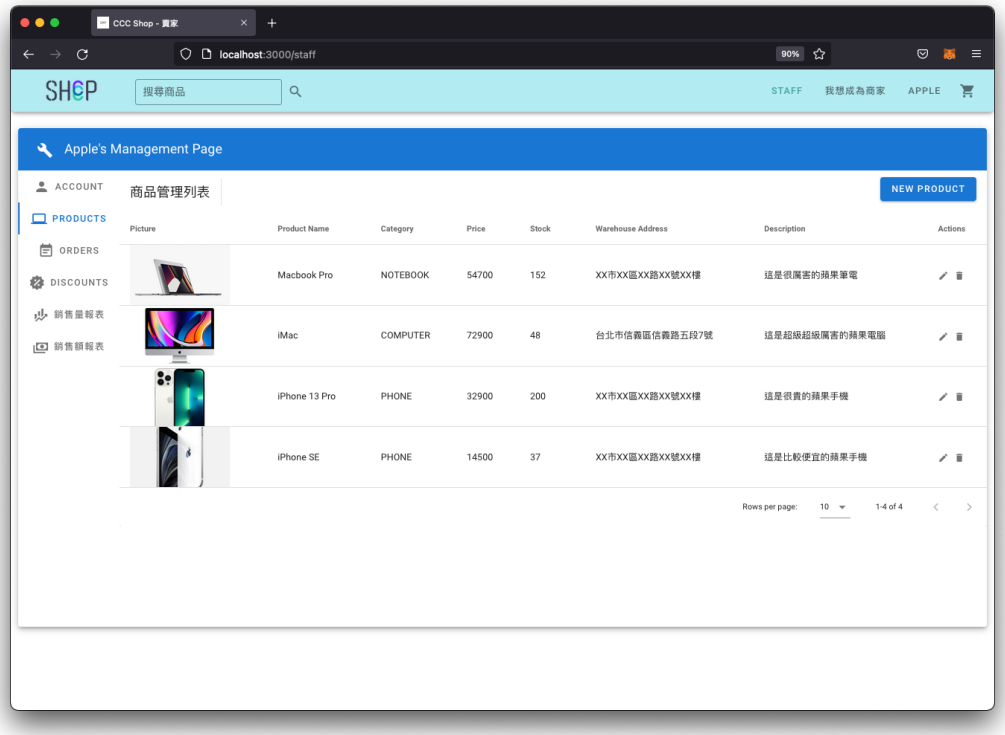
E. Change Product Result



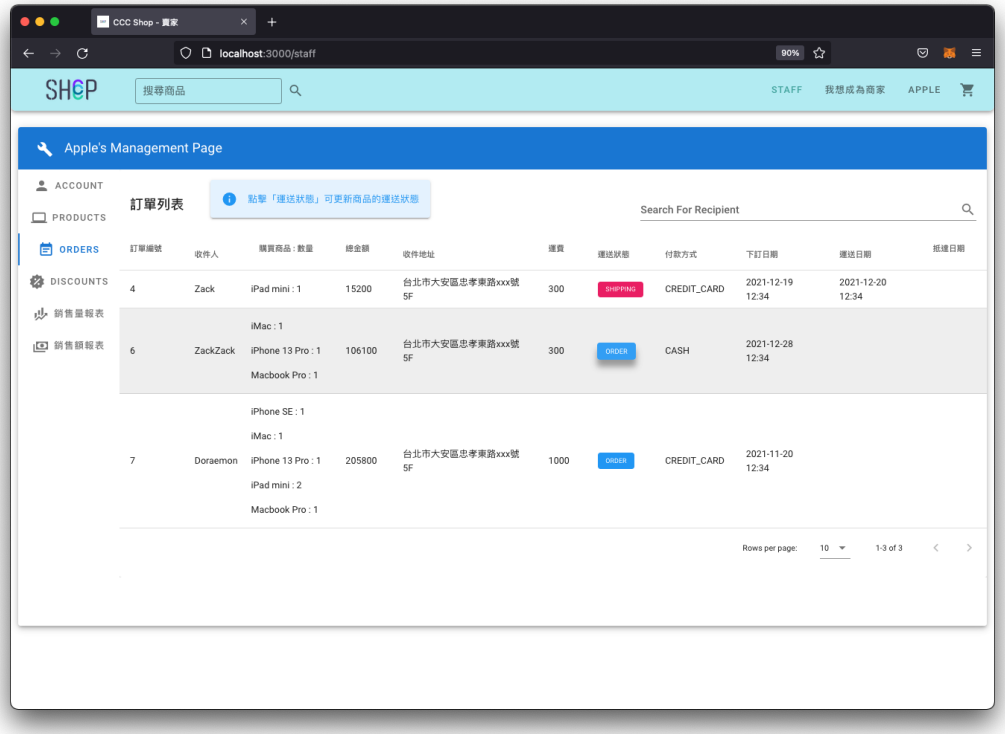
F. Delete Product



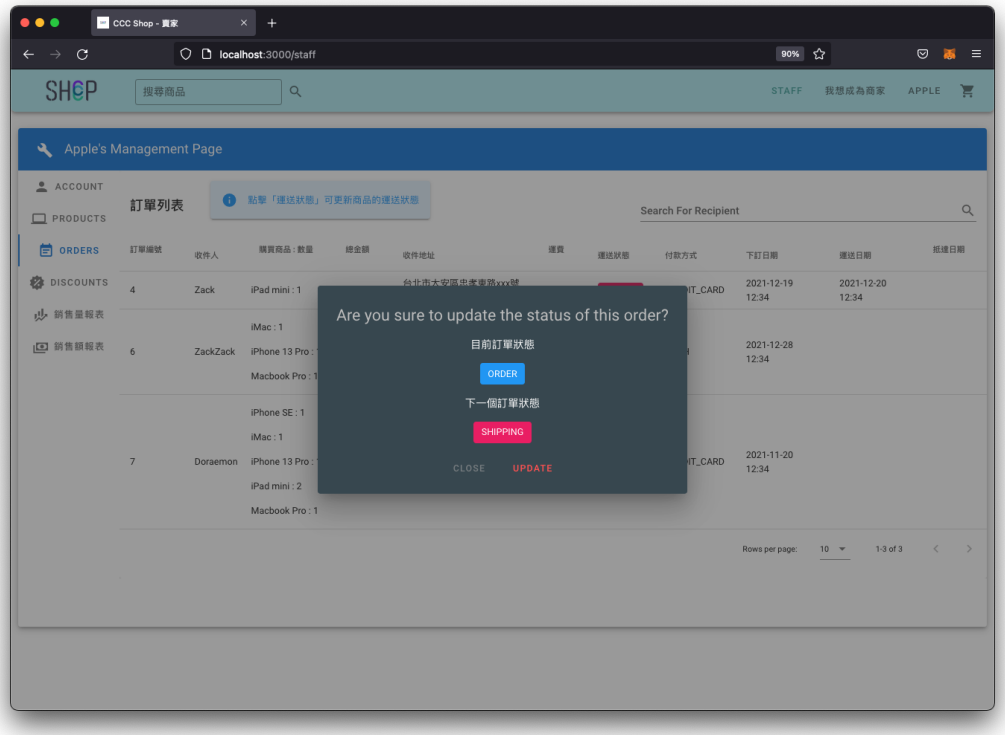
G. Delete Product Result



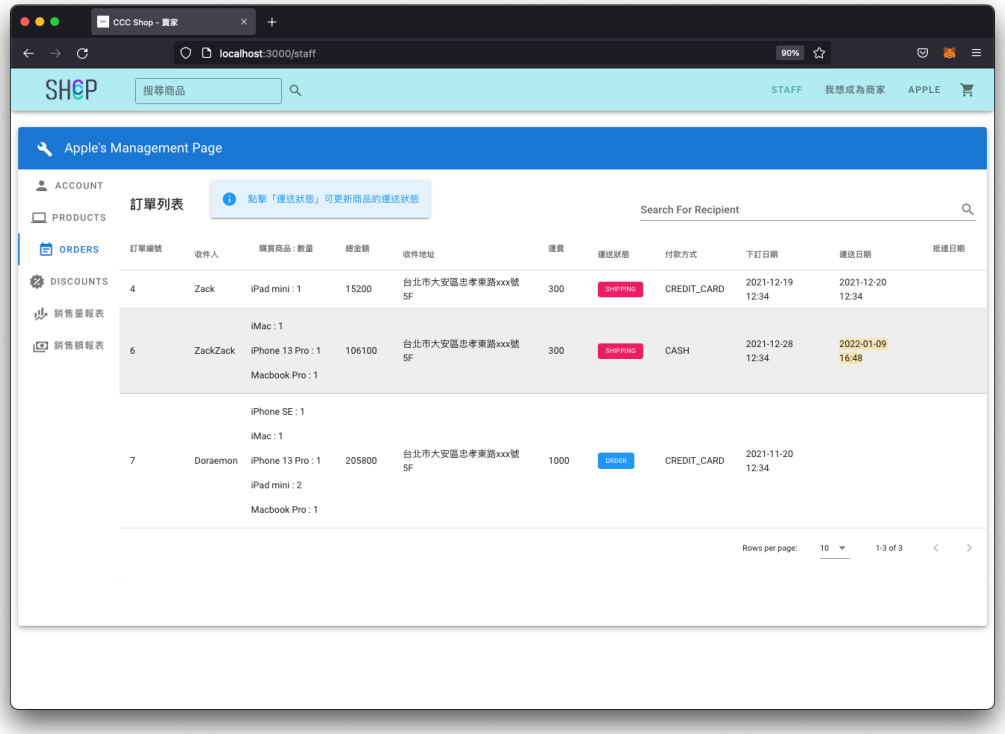
H. Manage Order



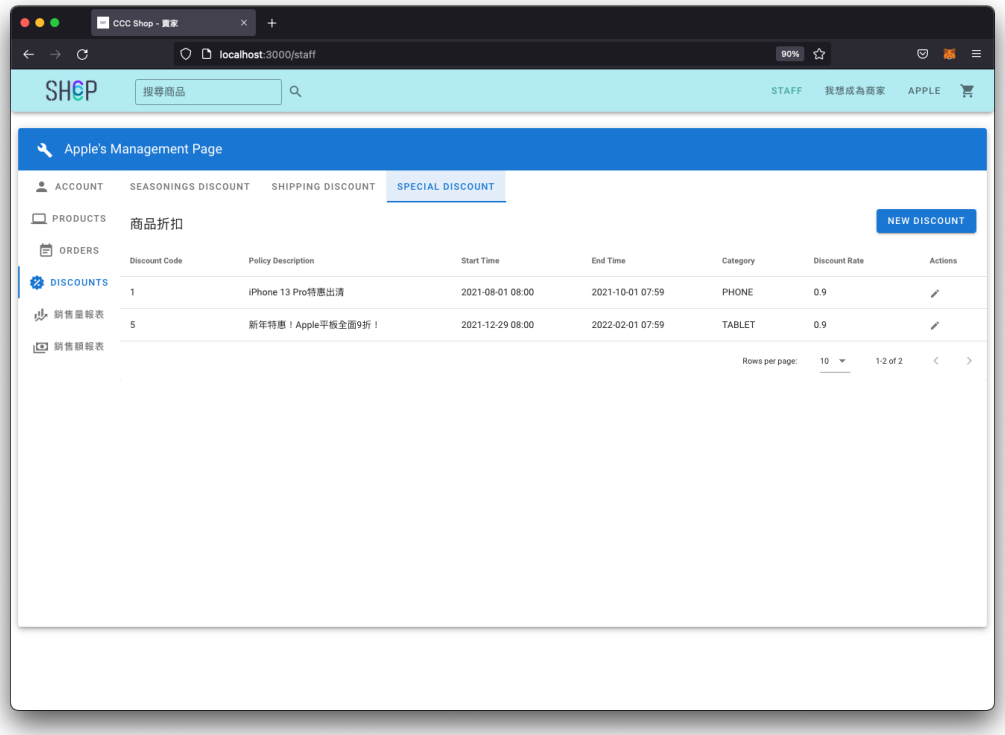
I. Change Status



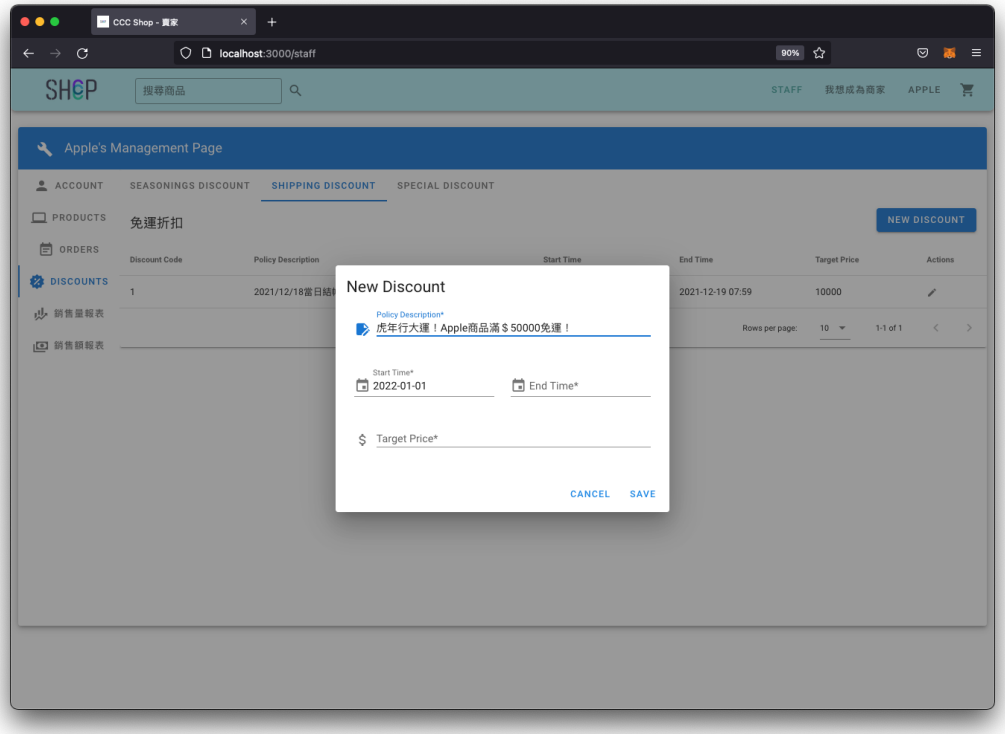
J. Order Changed Result



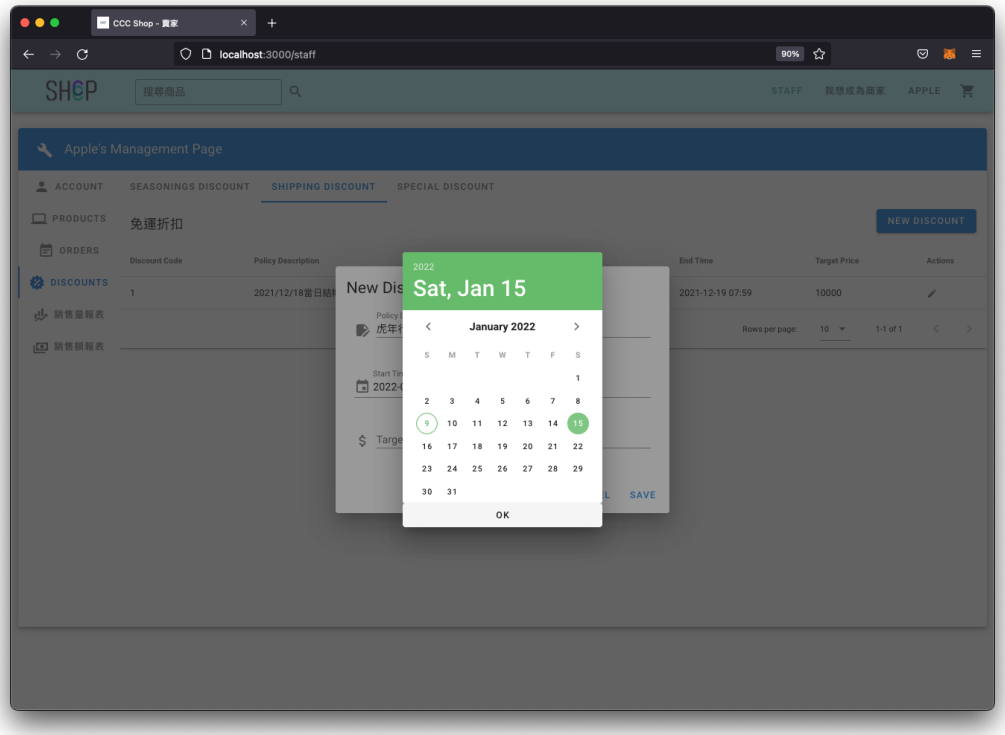
K. Manage Discount



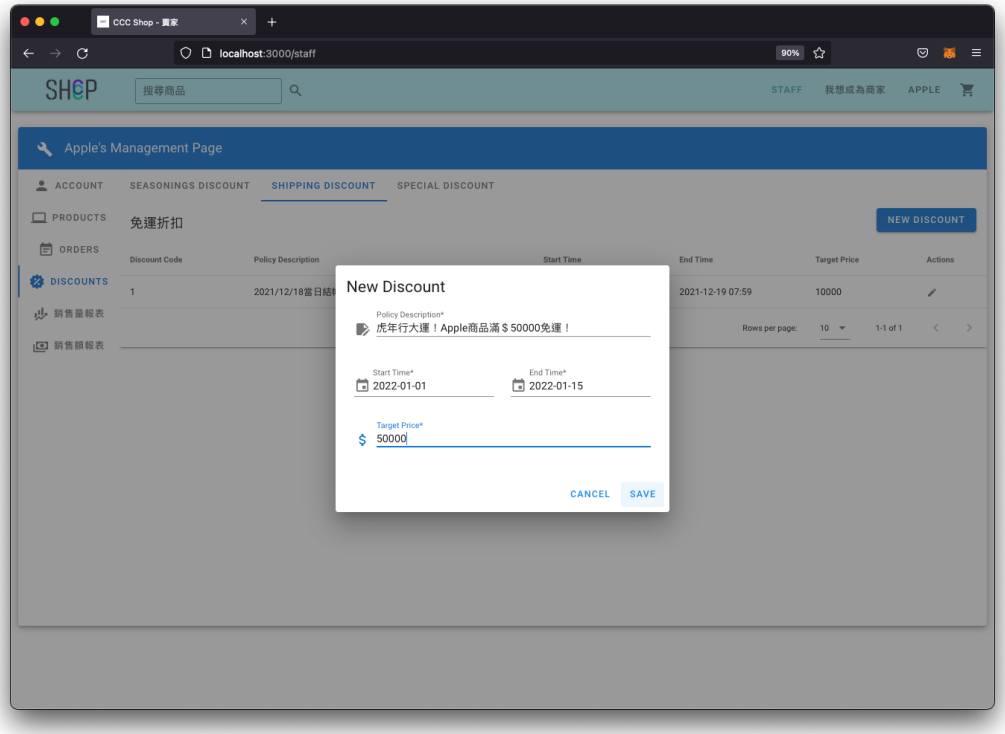
L. New Discount



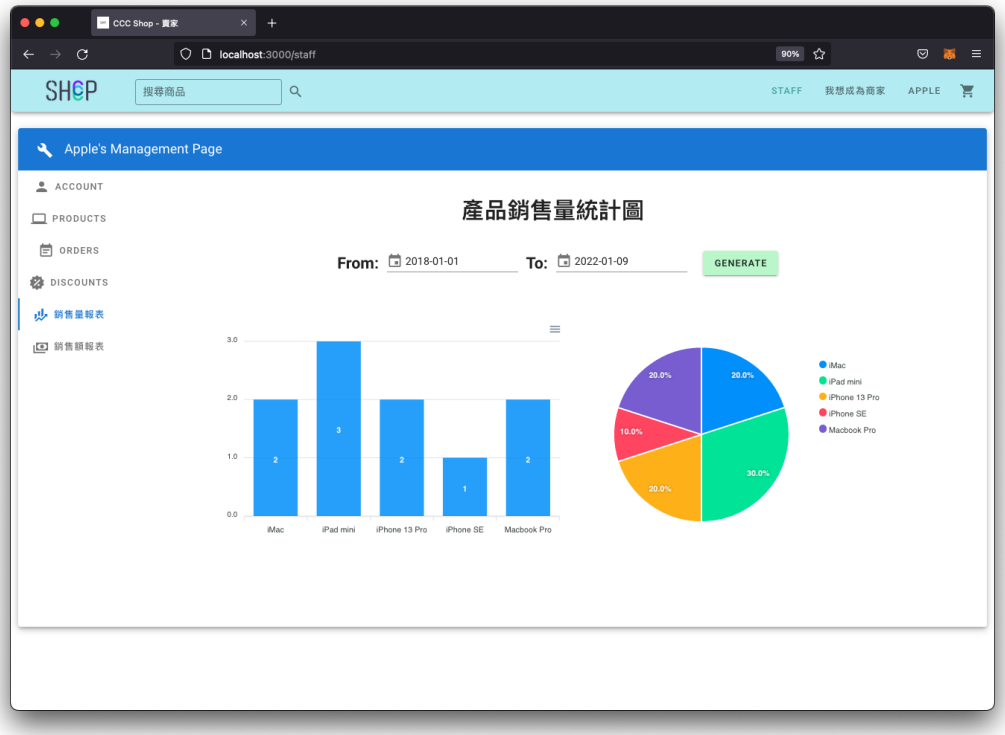
M. Choose Date



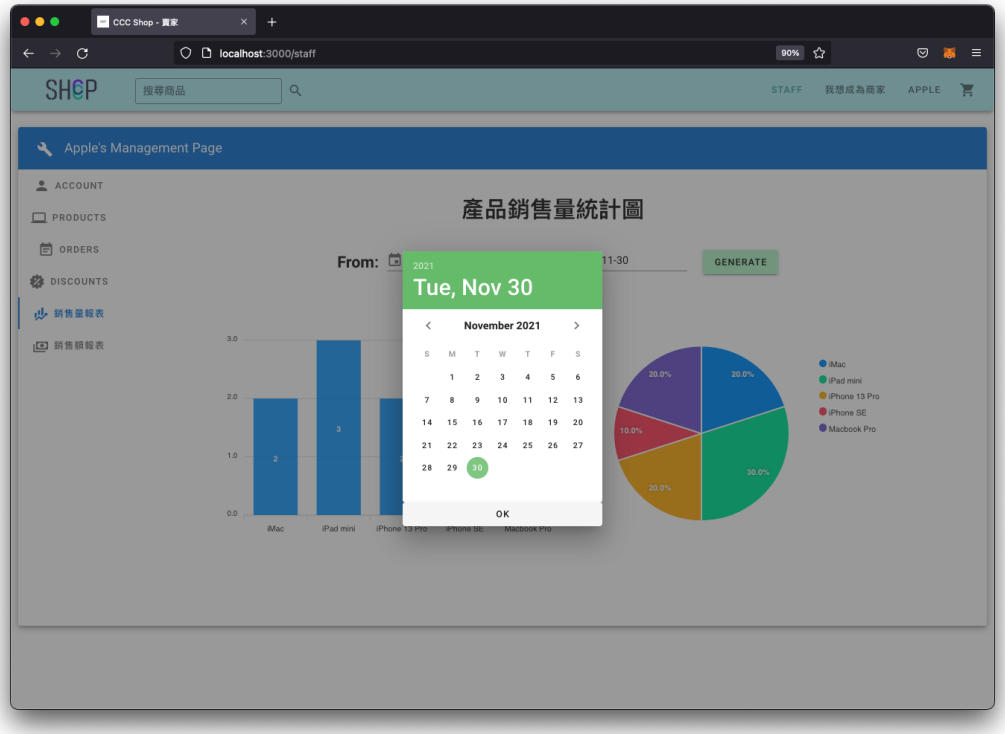
N. Save Discount



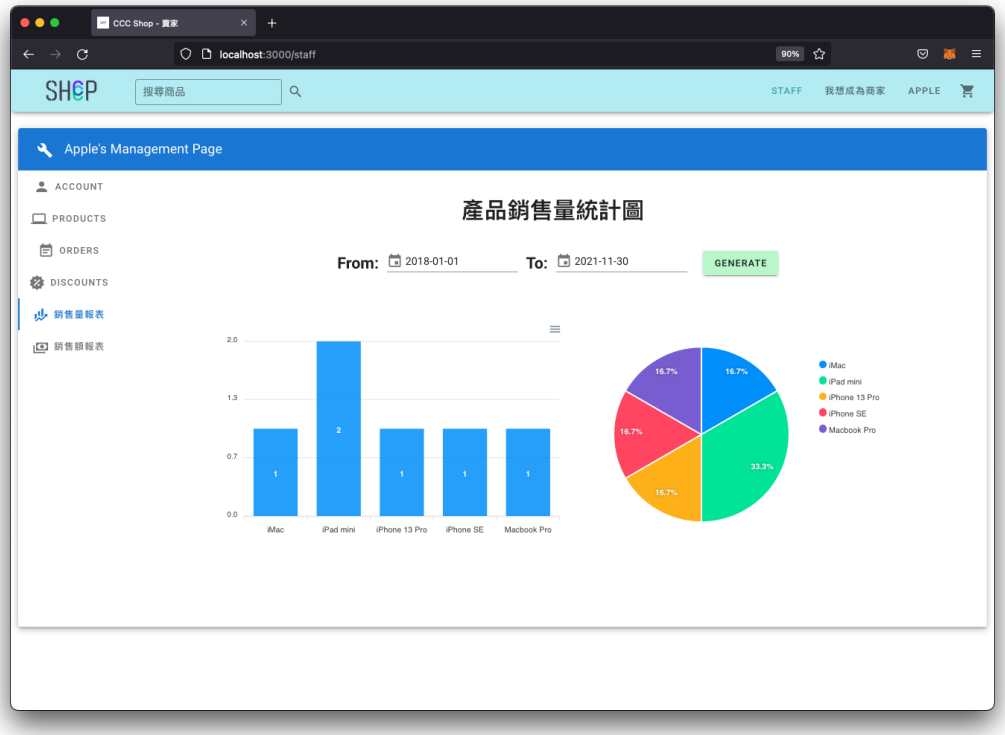
O. Sales Report



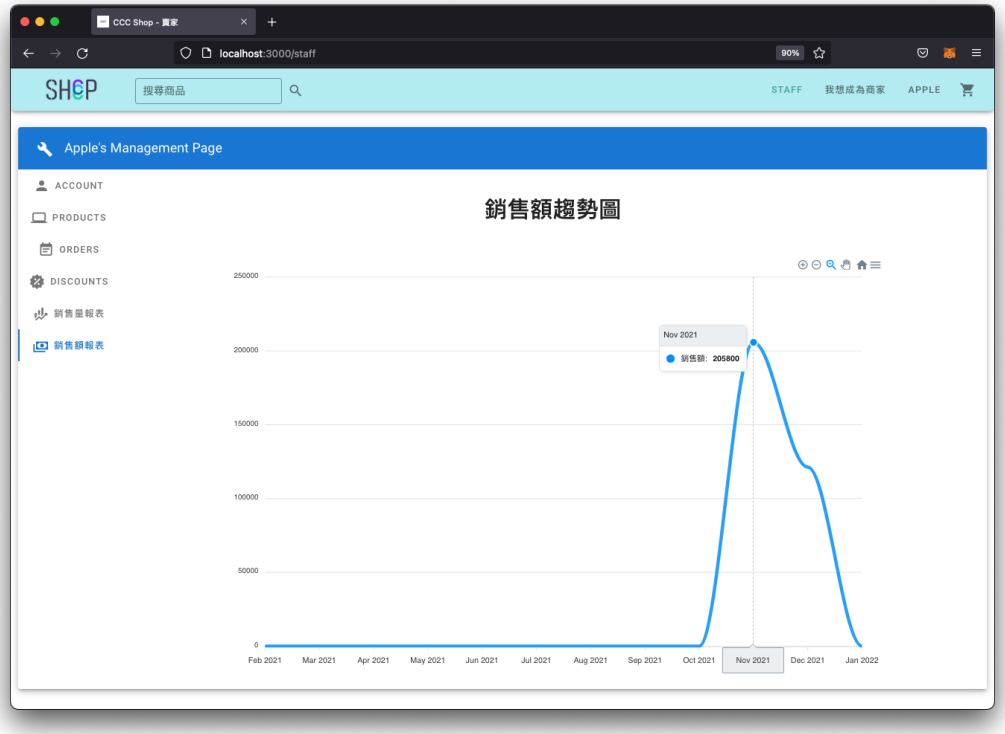
P. Choose The Range Of Sales Report



Q. Sales Report New Result

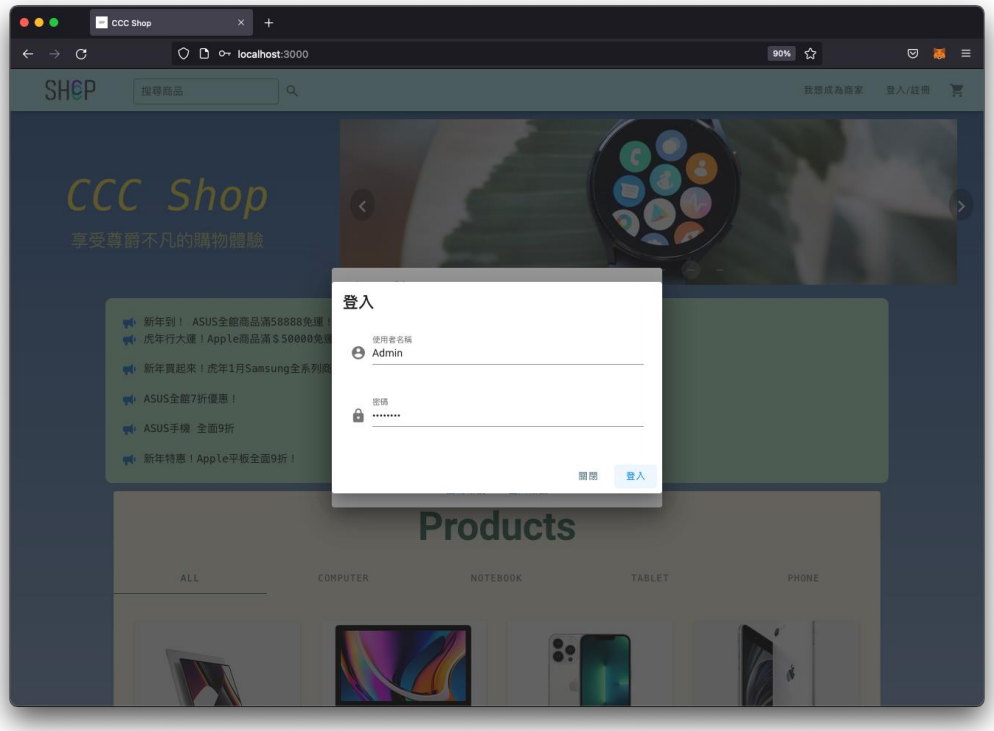


R. Sales Trend Report

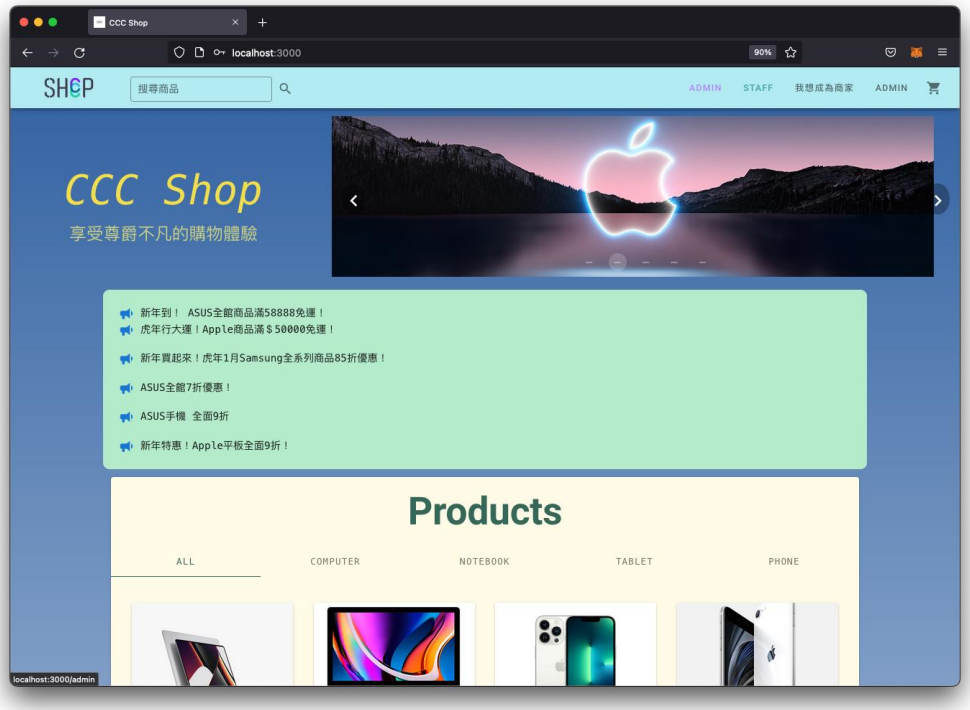


6.2.4 管理員使用者介面 (Admin)

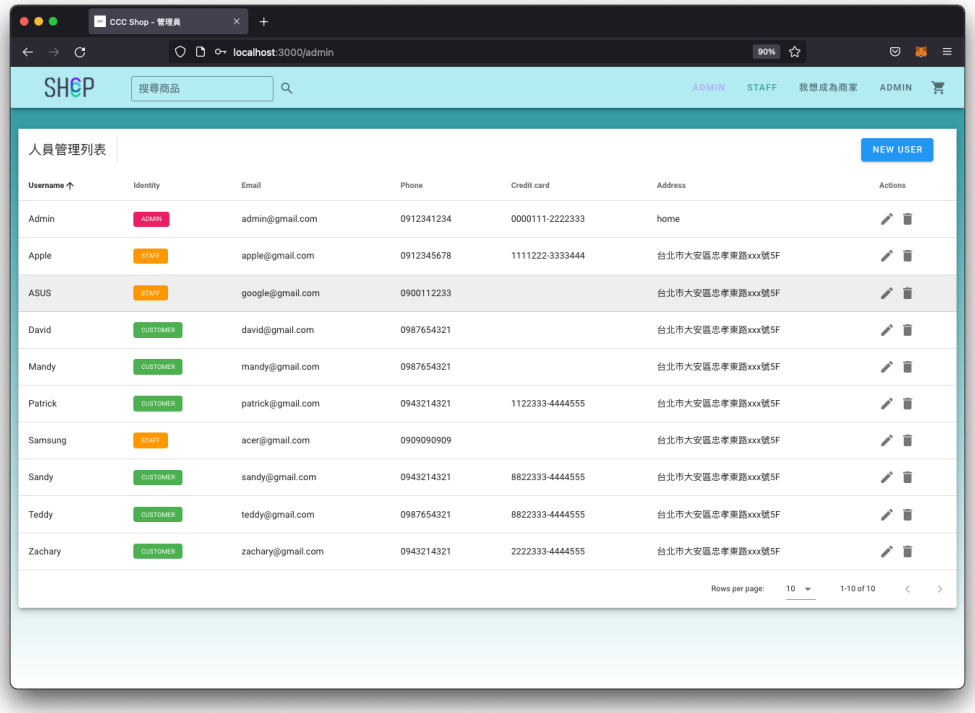
A. Login



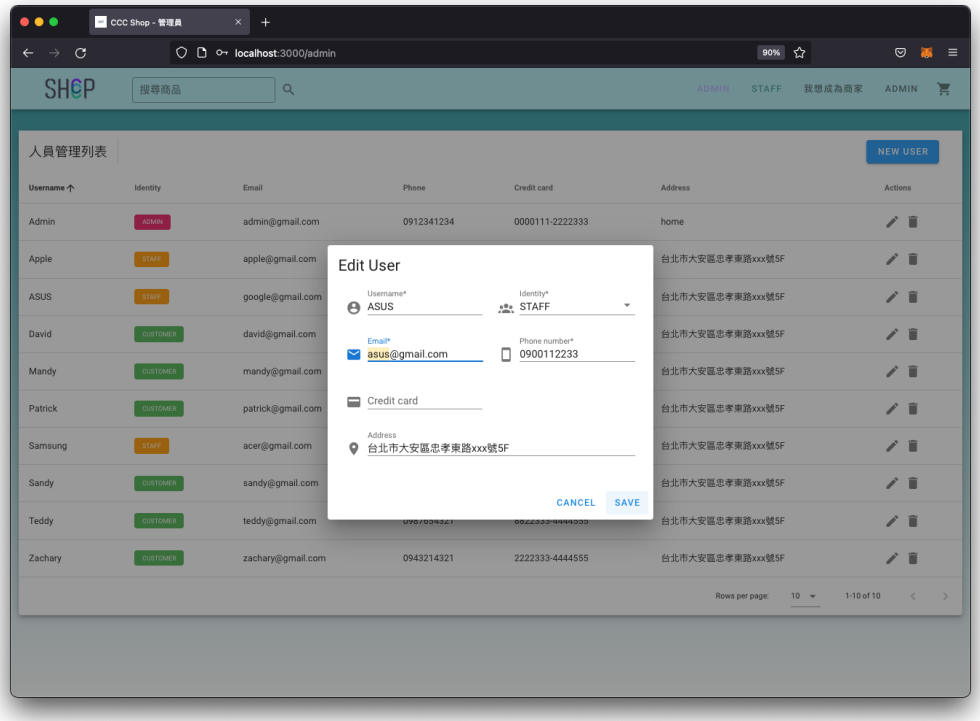
B. Login Page



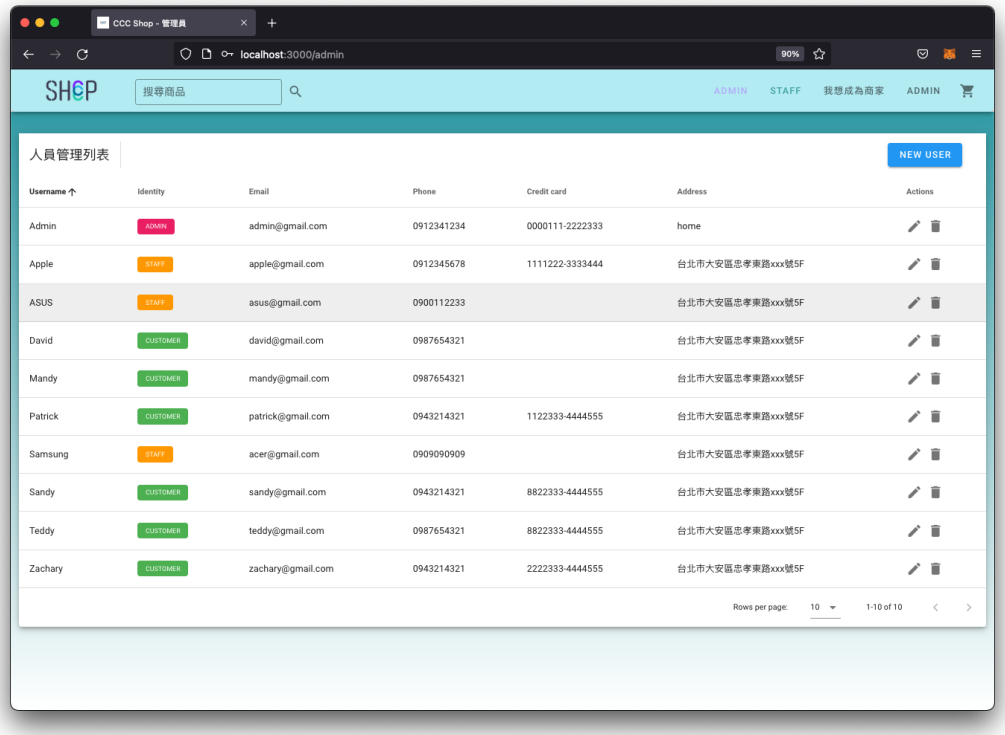
C. Manage Users



D. Edit User Info

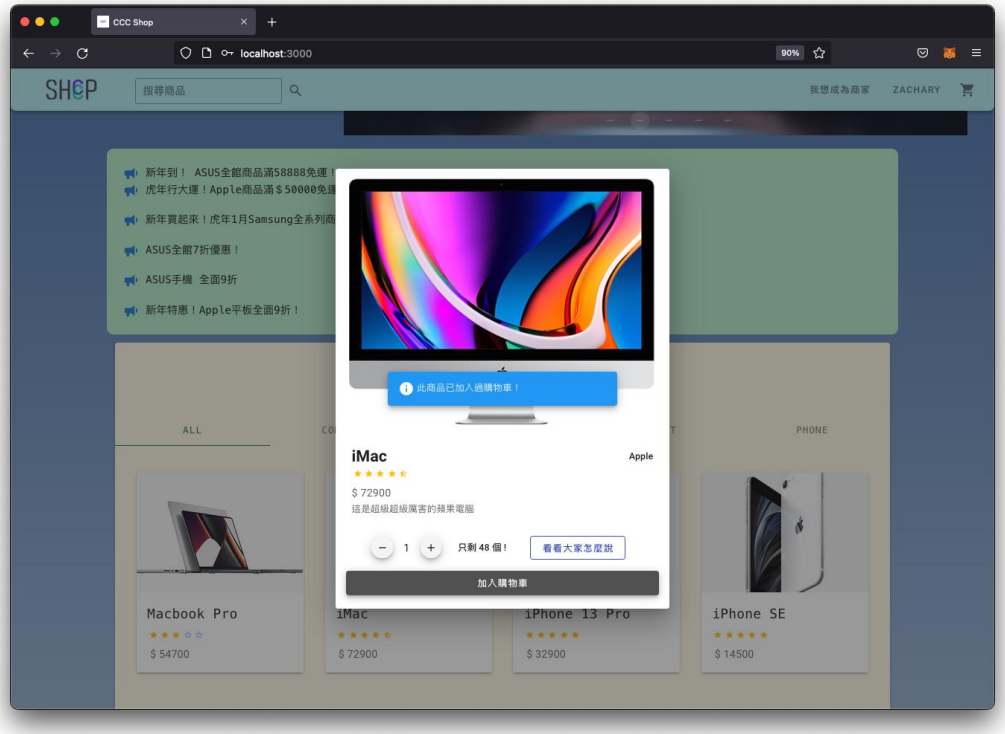


E. Edit Result

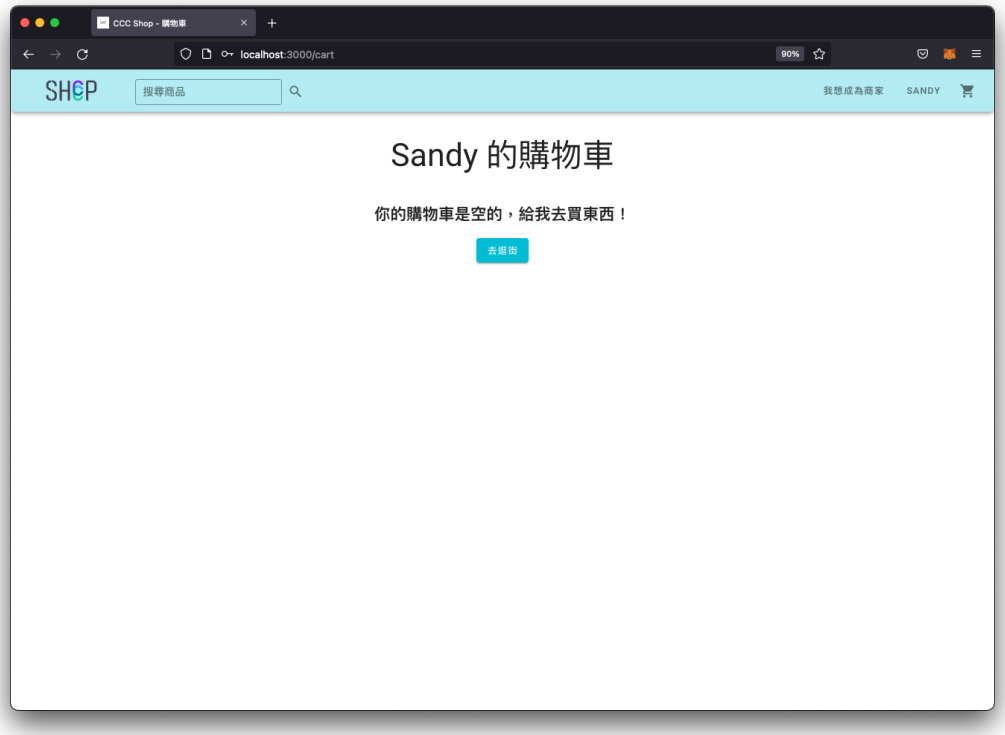


6.2.5 More Details

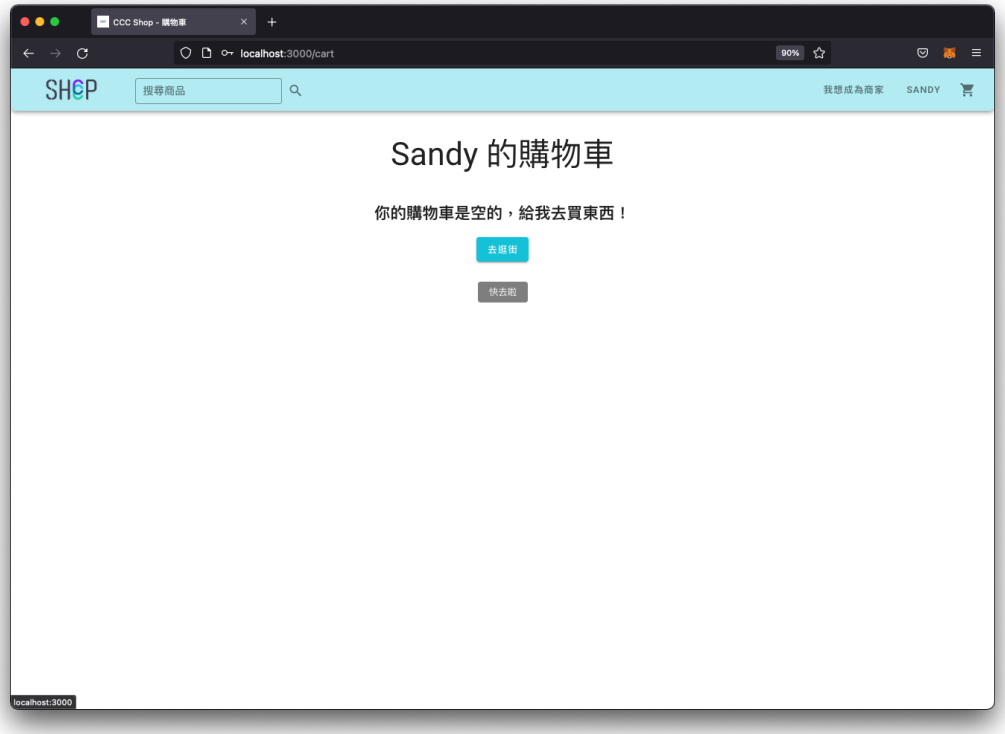
A. Cannot Add Duplicate Product To Cart



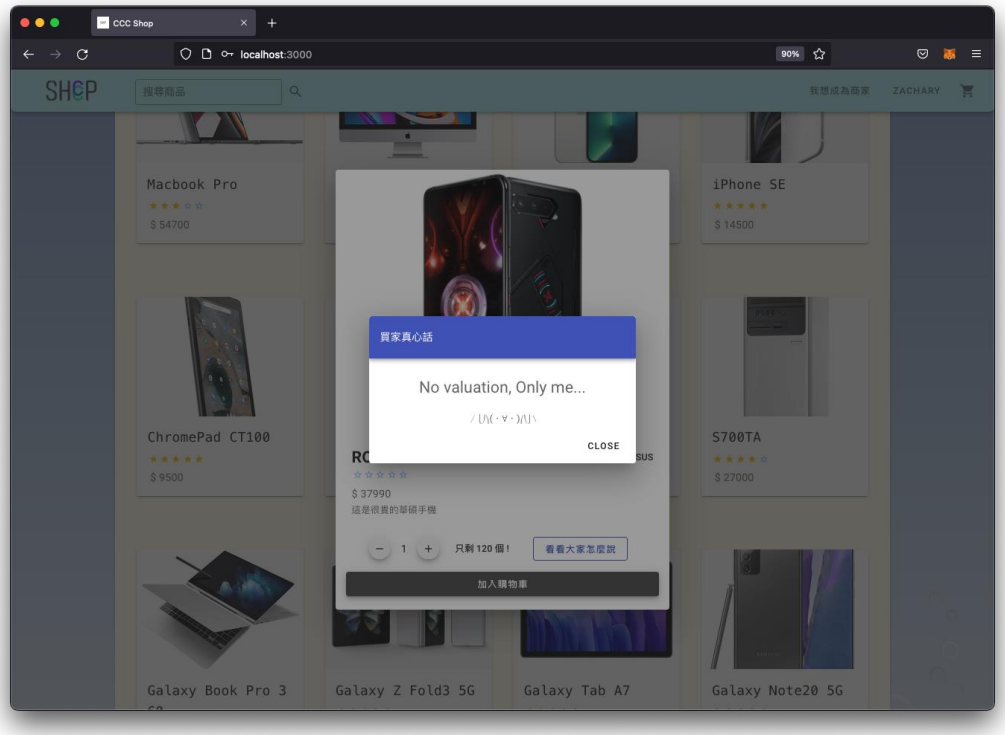
B. Empty Cart



C. Empty Cart Message



D. No Valuation



Section 7 資料庫優化建議 (Suggestions on Database Tuning)

7.1 索引結構 (index structures)

為資料庫建立索引（index），好的綱要設計，與巢狀查詢（nest query）的使用等能有效優化資料庫的查詢速度，若系統之資料數量過大時可從以上幾個方面著手，並針對查詢頻率較高之使用情景優先採用，減少查詢中的 Join 來達成資料庫優化。

Section 8 Additional Queries and Views

8.1 使用者管理

A. 新增使用者

```
public void execute(CreateUserInput input, CreateUserOutput output) {

    try(Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `user` (`username`, `identity`, `password`, `phone`, `email`, `credit_card`, `address` )" +
                        " VALUES (?,?,?,?,?,?,?)");

        stmt.setString( parameterIndex: 1, input.getUsername());
        stmt.setString( parameterIndex: 2, input.getIdentity().toString());
        stmt.setString( parameterIndex: 3, input.getPassword());
        stmt.setString( parameterIndex: 4, input.getPhone());
        stmt.setString( parameterIndex: 5, input.getEmail());
        stmt.setString( parameterIndex: 6, input.getCreditCard());
        stmt.setString( parameterIndex: 7, input.getAddress());

        stmt.executeUpdate();

        output.setUsername(input.getUsername());

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 刪除使用者

```
public void execute(DeleteUserInput input, DeleteUserOutput output) {

    try (Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement( sql: "SELECT `id` FROM `user` WHERE id = ?");
        stmt.setInt( parameterIndex: 1, input.getId());
        ResultSet rs = stmt.executeQuery();

        stmt = connection.prepareStatement( sql: "DELETE FROM `user` WHERE id = ?");
        stmt.setInt( parameterIndex: 1, input.getId());
        stmt.executeUpdate();

        if (!rs.next()){
            output.setWorkCheck(false);
            throw new RuntimeException("delete user failed");
        } else {
            output.setWorkCheck(true);
        }

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 登入(回傳此使用者資訊)

```
public void execute(LoginUserInput input, LoginUserOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * FROM `user` WHERE `username` = ? and `password` = ?");

        stmt.setString( parameterIndex: 1, input.getUsername());
        stmt.setString( parameterIndex: 2, input.getPassword());

        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            output.setId(rs.getInt( columnLabel: "id"));
            output.setUsername(rs.getString( columnLabel: "username"));
            output.setIdentity(rs.getString( columnLabel: "identity"));
            output.setPhone(rs.getString( columnLabel: "phone"));
            output.setEmail(rs.getString( columnLabel: "email"));
            output.setCreditCard(rs.getString( columnLabel: "credit_card"));
            output.setAddress(rs.getString( columnLabel: "address"));
        } else {
            output.setUsername("fail");
            throw new RuntimeException("Login failed");
        }

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 列出所有使用者

```
public void execute(GetAllUserOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<User> users = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * FROM `user`;");

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                User user = new User();
                user.setId(rs.getInt( columnLabel: "id"));
                user.setUsername(rs.getString( columnLabel: "username"));
                user.setIdentity(Identity.valueOf(rs.getString( columnLabel: "identity")));
                user.setPhone(rs.getString( columnLabel: "phone"));
                user.setEmail(rs.getString( columnLabel: "email"));
                user.setCreditCard(rs.getString( columnLabel: "credit_card"));
                user.setAddress(rs.getString( columnLabel: "address"));

                users.add(user);
            }
        }

        output.setUserList(users);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

E. 編輯(更新)使用者

```
public void execute(UpdateUserInput input, UpdateUserOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `user` SET " +
                        "`username`=?, " +
                        "`identity`=?, " +
                        "`phone`=?, " +
                        "`email`=?, " +
                        "`credit_card`=?, " +
                        "`address`=? " +
                        "WHERE `id`=?;");

        stmt.setString( parameterIndex: 1, input.getUsername());
        stmt.setString( parameterIndex: 2, input.getIdentity().toString());
        stmt.setString( parameterIndex: 3, input.getPhone());
        stmt.setString( parameterIndex: 4, input.getEmail());
        stmt.setString( parameterIndex: 5, input.getCreditCard());
        stmt.setString( parameterIndex: 6, input.getAddress());
        stmt.setInt( parameterIndex: 7, input.getId());

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.2 商品管理

A. 新增商品

```
public void execute(CreateProductInput input, CreateProductOutput output) {

    try(Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `product` (`name`, " +
                        "`vender_id`, " +
                        "`category`, " +
                        "`price`, " +
                        "`stock`, " +
                        "`warehouse_address`, " +
                        "`description`, " +
                        "`pictureURL`, " +
                        "`exist_flag`)" +
                        " VALUES (?,?,?,?,?,?,?,?,?)");

        stmt.setString( parameterIndex: 1, input.getName());
        stmt.setInt( parameterIndex: 2, input.getVenderId());
        stmt.setString( parameterIndex: 3, input.getCategory().toString());
        stmt.setInt( parameterIndex: 4, input.getPrice());
        stmt.setInt( parameterIndex: 5, input.getStock());
        stmt.setString( parameterIndex: 6, input.getWarehouseAddress());
        stmt.setString( parameterIndex: 7, input.getDescription());
        stmt.setString( parameterIndex: 8, input.getPictureURL());
        stmt.setBoolean( parameterIndex: 9,  x: true);

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 刪除(下架)商品

```
public void execute(DeleteProductInput input, DeleteProductOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        if (productExists(connection, input.getId())) {
            PreparedStatement stmt = connection.prepareStatement(
                    sql: "UPDATE `product` SET `exist_flag` = ? WHERE id = ?");
            stmt.setBoolean( parameterIndex: 1,  x: false);
            stmt.setInt( parameterIndex: 2, input.getId());

            stmt.executeUpdate();
        }
        else {
            throw new RuntimeException("product doesn't exist, where product id = " + input.getId());
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 列出某商品資訊

```
public void execute(GetProductInput input, GetProductOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * FROM `product` WHERE `id`=?");
        stmt.setInt( parameterIndex: 1, input.getId());

        try (ResultSet rs = stmt.executeQuery()){
            if (rs.next()){
                output.setName(rs.getString( columnLabel: "name"));
                output.setVenderId(rs.getInt( columnLabel: "vender_id"));
                output.setCategory(Category.valueOf(rs.getString( columnLabel: "category")));
                output.setPrice(rs.getInt( columnLabel: "price"));
                output.setStock(rs.getInt( columnLabel: "stock"));
                output.setWarehouseAddress(rs.getString( columnLabel: "warehouse_address"));
                output.setDescription(rs.getString( columnLabel: "description"));
                output.setPictureURL(rs.getString( columnLabel: "pictureURL"));
                output.setExistFlag(rs.getBoolean( columnLabel: "exist_flag"));
            }
            else {
                throw new RuntimeException("product doesn't exist, where product id = " + input.getId());
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 列出所有商品

```
public void execute(GetAllProductOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<Product> productList = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT p.id, p.vender_id, p.name, p.category, p.price, p.stock, p.warehouse_address, p.description, " +
                        "p.pictureURL, user.username, AVG(rating), p.exist_flag " +
                        "FROM `user`, `product` AS p LEFT OUTER JOIN `valuation` AS v " +
                        "ON p.id = v.product_id " +
                        "WHERE user.id = p.vender_id " +
                        "GROUP BY p.id");

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                Product product = new Product();
                product.setId(rs.getInt( columnLabel: "id"));
                product.setVenderId(rs.getInt( columnLabel: "vender_id"));
                product.setName(rs.getString( columnLabel: "name"));
                product.setCategory(Category.valueOf(rs.getString( columnLabel: "category")));
                product.setPrice(rs.getInt( columnLabel: "price"));
                product.setStock(rs.getInt( columnLabel: "stock"));
                product.setWarehouseAddress(rs.getString( columnLabel: "warehouse_address"));
                product.setDescription(rs.getString( columnLabel: "description"));
                product.setPictureURL(rs.getString( columnLabel: "pictureURL"));
                product.setVenderName(rs.getString( columnLabel: "username"));
                product.setRate(rs.getDouble( columnLabel: "avg(rating)"));
                product.setExistFlag(rs.getBoolean( columnLabel: "exist_flag"));

                productList.add(product);
            }
        }
        output.setProductList(productList);

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

E. 更新某商品庫存數量

```
private void updateProductStock(Connection connection, int productId, int quantity) {
    try (PreparedStatement stmt = connection.prepareStatement(
            sql: "UPDATE `product` " +
                    "SET `stock`= `stock` - ? " +
                    "WHERE `id`=?")) {
        stmt.setInt( parameterIndex: 1, quantity);
        stmt.setInt( parameterIndex: 2, productId);

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

F. 編輯商品

```
public void execute(UpdateProductInput input, UpdateProductOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        if (productExists(connection, input.getId())) {
            PreparedStatement stmt = connection.prepareStatement(
                    sql: "UPDATE `product` " +
                            "SET `name`=?, " +
                            "`vender_id`=?, " +
                            "`category`=?, " +
                            "`price`=?, " +
                            "`stock`=?, " +
                            "`warehouse_address`=?, " +
                            "`description`=?, " +
                            "`pictureURL`=?, " +
                            "`exist_flag`=? " +
                            "WHERE `id`=?");

            stmt.setString( parameterIndex: 1, input.getName());
            stmt.setInt( parameterIndex: 2, input.getVenderId());
            stmt.setString( parameterIndex: 3, input.getCategory().toString());
            stmt.setInt( parameterIndex: 4, input.getPrice());
            stmt.setInt( parameterIndex: 5, input.getStock());
            stmt.setString( parameterIndex: 6, input.getWarehouseAddress());
            stmt.setString( parameterIndex: 7, input.getDescription());
            stmt.setString( parameterIndex: 8, input.getPictureURL());
            stmt.setBoolean( parameterIndex: 9, input.getExistFlag());
            stmt.setInt( parameterIndex: 10, input.getId());

            stmt.executeUpdate();
        } else {
            throw new RuntimeException("product doesn't exist, where product id = " + input.getId());
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.3 訂單管理

A. 新增訂單

```
public void execute(CreateOrderInput input, CreateOrderOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `order` (`customer_id`, " +
                        "`shipping_fee`, " +
                        "`recipient_name`, " +
                        "`shipping_address`, " +
                        "`status`, " +
                        "`payment_method`, " +
                        "`credit_card_id`, " +
                        "`order_time`, " +
                        "`seasoning_discount_code`, " +
                        "`shipping_discount_code`, " +
                        "`total_price`)" +
                        " VALUES (?,?,?,?,?,?,?,?,?,?,?)");

        stmt.setInt( parameterIndex: 1, input.getCustomerId());
        stmt.setInt( parameterIndex: 2, input.getShippingFee());
        stmt.setString( parameterIndex: 3, input.getRecipientName());
        stmt.setString( parameterIndex: 4, input.getShippingAddress());
        stmt.setString( parameterIndex: 5, input.getStatus().toString());
        stmt.setString( parameterIndex: 6, input.getPaymentMethod().toString());
        stmt.setString( parameterIndex: 7, input.getCreditCardId());
        Calendar calendar = Calendar.getInstance(TimeZone.getTimeZone("Asia/Taipei"));
        stmt.setTimestamp( parameterIndex: 8, input.getOrderTime(), calendar);
        if (input.getSeasoningDiscountCode() == 0) {
            stmt.setNull( parameterIndex: 9, NULL);
        } else {
            stmt.setInt( parameterIndex: 9, input.getSeasoningDiscountCode());
        }
        if (input.getShippingDiscountCode() == 0) {
            stmt.setNull( parameterIndex: 10, NULL);
        } else {
            stmt.setInt( parameterIndex: 10, input.getShippingDiscountCode());
        }
        stmt.setInt( parameterIndex: 11, input.getTotalPrice());

        stmt.executeUpdate();

        int orderId = getOrderId(connection);
        createOrderItems(connection, orderId, input.getOrderItems(), input.getCustomerId());
        output.setOrderTime(getOrderTime(connection, orderId));
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 將訂單與商品關係計入資料庫

```java
private void createOrderItems(Connection connection, int orderId, Map<Integer, Integer> orderItems, int customerId) {
    if (orderItems.isEmpty()) {
        return;
    }
    List<Integer> productIdList = new ArrayList<>();
    orderItems.keySet().forEach(productId -> {
        productIdList.add(productId);
        try (PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `order_items` (`order_id`, " +
                        "`product_id`, " +
                        "`quantity`)" +
                        " VALUES (?,?,?)")) {
            stmt.setInt( parameterIndex: 1, orderId);
            stmt.setInt( parameterIndex: 2, productId);
            stmt.setInt( parameterIndex: 3, orderItems.get(productId));

            stmt.executeUpdate();
            updateProductStock(connection, productId, orderItems.get(productId));
            deleteItemsInShoppingCart(connection, productId, customerId, orderItems.get(productId));
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    });
    generateOrderAndVenderRelation(connection, orderId, productIdList.get(0));
}
```

C. 將訂單和商家關係計入資料庫

```java
private void insertValueToManageOrder(Connection connection, int orderId, int venderId) {
    try (PreparedStatement stmt = connection.prepareStatement(
            sql: "INSERT `manage_order` (`order_id`, " +
                    "`vender_id`)" +
                    " VALUES (?,?)")) {
        stmt.setInt( parameterIndex: 1, orderId);
        stmt.setInt( parameterIndex: 2, venderId);

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 列出某顧客所有訂單

```java
public void execute(GetCustomerOrderInput input, GetCustomerOrderOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<Order> orderList = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * FROM `order` WHERE `customer_id`=?;");
        stmt.setInt( parameterIndex: 1, input.getCustomerId());

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                Order order = new Order();
                order.setId(rs.getInt( columnLabel: "id"));
                order.setCustomerId(rs.getInt( columnLabel: "customer_id"));
                order.setShippingFee(rs.getInt( columnLabel: "shipping_fee"));
                order.setRecipientName(rs.getString( columnLabel: "recipient_name"));
                order.setShippingAddress(rs.getString( columnLabel: "shipping_address"));
                order.setStatus(Status.valueOf(rs.getString( columnLabel: "status")));
                order.setPaymentMethod(Payment.valueOf(rs.getString( columnLabel: "payment_method")));
                order.setCreditCardId(rs.getString( columnLabel: "credit_card_id"));
                order.setOrderTime(rs.getTimestamp( columnLabel: "order_time"));
                order.setShippingTime(rs.getTimestamp( columnLabel: "shipping_time"));
                order.setDeliveryTime(rs.getTimestamp( columnLabel: "delivery_time"));
                order.setSeasoningDiscountCode(rs.getInt( columnLabel: "seasoning_discount_code"));
                order.setShippingDiscountCode(rs.getInt( columnLabel: "shipping_discount_code"));
                order.setTotalPrice(rs.getInt( columnLabel: "total_price"));
                addOrderItems(connection, order);

                orderList.add(order);
            }
        }

        output.setOrderList(orderList);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

E. 列出某商家所有訂單

```
public void execute(GetVenderOrderInput input, GetVenderOrderOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<Order> orderList = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * FROM `manage_order` WHERE `vender_id`=?;");
        stmt.setInt( parameterIndex: 1, input.getVenderId());
        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                addOrderToList(connection, rs.getInt( columnLabel: "order_id"), orderList);
            }
        }

        output.setOrderList(orderList);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

F. 更新訂單狀態

```
public void execute(UpdateOrderInput input, UpdateOrderOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `order` SET `status`= ? WHERE `id`= ?");

        stmt.setString( parameterIndex: 1, input.getStatus().toString());
        stmt.setInt( parameterIndex: 2, input.getOrderId());

        stmt.executeUpdate();

        updateTime(connection, input.getStatus(), input.getTime(), input.getOrderId());

        output.setTime(input.getTime());
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.4 評論管理

A. 新增評論

```
public void execute(CreateValuationInput input, CreateValuationOutput output) {
    try(Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `valuation` (`customer_id`, `product_id`, `comment`, `rating`)" +
                        " VALUES (?,?,?,?)");

        stmt.setInt( parameterIndex: 1, input.getCustomerId());
        stmt.setInt( parameterIndex: 2, input.getProductId());
        stmt.setString( parameterIndex: 3, input.getComment().toString());
        stmt.setInt( parameterIndex: 4, input.getRating());

        stmt.executeUpdate();

        output.setCustomerId(input.getCustomerId());
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 列出某商品所有評論

```
public void execute(GetValuationInput input, GetValuationOutput output) {
    List<Valuation> valuationList = new ArrayList<>();
    try(Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT `valuation`.*, `user`.username FROM `valuation`, `user` " +
                        "WHERE `product_id` = ? " +
                        "AND `valuation`.`customer_id` = `user`.`id`");

        stmt.setInt( parameterIndex: 1, input.getProductId());
        ResultSet rs = stmt.executeQuery();

        while(rs.next()){
            String customerName = rs.getString( columnLabel: "username");
            String comment = rs.getString( columnLabel: "comment");
            int rating = rs.getInt( columnLabel: "rating");
            Valuation valuation = new Valuation(customerName, comment, rating);
            valuationList.add(valuation);
        }

        output.setValuationList(valuationList);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.5 季節折扣管理

A. 新增季節折扣

```
public void execute(CreateSeasoningsDiscountInput input, CreateSeasoningsDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `seasonings_discount` (`vender_id`, " +
                        "`policy_description`, " +
                        "`start_time`, " +
                        "`end_time`, " +
                        "`discount_rate`)" +
                        " VALUES (?,?,?,?,?)");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setString( parameterIndex: 2, input.getPolicyDescription());
        stmt.setTimestamp( parameterIndex: 3, input.getStartTime());
        stmt.setTimestamp( parameterIndex: 4, input.getEndTime());
        stmt.setDouble( parameterIndex: 5, input.getDiscountRate());

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 列出目前有效的季節折扣

```
public void execute(GetCurrentSeasoningsDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<SeasoningsDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `seasonings_discount` " +
                        "WHERE ? > `start_time` " +
                        "AND ? < `end_time`;");

        Timestamp currentTime = Timestamp.from(Instant.now());
        Calendar calendar = Calendar.getInstance(TimeZone.getTimeZone("Asia/Taipei"));  // set timezone
        stmt.setTimestamp( parameterIndex: 1, currentTime, calendar);
        stmt.setTimestamp( parameterIndex: 2, currentTime, calendar);

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                SeasoningsDiscount seasoningsDiscount = new SeasoningsDiscount();
                seasoningsDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                seasoningsDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                seasoningsDiscount.setVenderName(queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                seasoningsDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                seasoningsDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                seasoningsDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                seasoningsDiscount.setDiscountRate(rs.getDouble( columnLabel: "discount_rate"));

                discounts.add(seasoningsDiscount);
            }
        }

        output.setSeasoningsDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 列出該商家的季節折扣

```
public void execute(GetVenderSeasoningsDiscountInput input, GetVenderSeasoningsDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<SeasoningsDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `seasonings_discount` " +
                        "WHERE `vender_id` = ?;");

        stmt.setInt( parameterIndex: 1, input.getVenderId());

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                SeasoningsDiscount seasoningsDiscount = new SeasoningsDiscount();
                seasoningsDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                seasoningsDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                seasoningsDiscount.setVenderName(GetCurrentSeasoningsDiscountUseCase.queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                seasoningsDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                seasoningsDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                seasoningsDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                seasoningsDiscount.setDiscountRate(rs.getDouble( columnLabel: "discount_rate"));

                discounts.add(seasoningsDiscount);
            }
        }

        output.setSeasoningsDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 編輯季節折扣

```
public void execute(EditSeasoningsDiscountInput input, EditSeasoningsDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `seasonings_discount` SET `vender_id`= ? WHERE `discount_code`= ?");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setInt( parameterIndex: 2, input.getDiscountCode());

        stmt.executeUpdate();

        updatePolicyDescription(connection, input.getPolicyDescription(), input.getDiscountCode()

        updateStartTime(connection, input.getStartTime(), input.getDiscountCode());
        updateEndTime(connection, input.getEndTime(), input.getDiscountCode());
        updateDiscountRate(connection, input.getDiscountRate(), input.getDiscountCode());

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.6 特別折扣管理

A. 新增特別折扣

```
public void execute(CreateSpecialDiscountInput input, CreateSpecialDiscountOutput output) {
    try(Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `special_discount` (`vender_id`, " +
                        "`policy_description`, " +
                        "`start_time`, " +
                        "`end_time`, " +
                        "`category`, " +
                        "`discount_rate`)" +
                        " VALUES (?,?,?,?,?,?)");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setString( parameterIndex: 2, input.getPolicyDescription());
        stmt.setTimestamp( parameterIndex: 3, input.getStartTime());
        stmt.setTimestamp( parameterIndex: 4, input.getEndTime());
        stmt.setString( parameterIndex: 5, input.getCategory().toString());
        stmt.setDouble( parameterIndex: 6, input.getDiscountRate());

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 列出目前有效的特別折扣

```
public void execute(GetCurrentSpecialDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<SpecialDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `special_discount` " +
                        "WHERE ? > `start_time` " +
                        "AND ? < `end_time`;");

        Timestamp currentTime = Timestamp.from(Instant.now());
        Calendar calendar = Calendar.getInstance(TimeZone.getTimeZone("Asia/Taipei"));  // set timezone
        stmt.setTimestamp( parameterIndex: 1, currentTime, calendar);
        stmt.setTimestamp( parameterIndex: 2, currentTime, calendar);

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                SpecialDiscount specialDiscount = new SpecialDiscount();
                specialDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                specialDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                specialDiscount.setVenderName(GetCurrentSeasoningsDiscountUseCase.queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                specialDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                specialDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                specialDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                specialDiscount.setCategory(Category.valueOf(rs.getString( columnLabel: "category")));
                specialDiscount.setDiscountRate(rs.getDouble( columnLabel: "discount_rate"));

                discounts.add(specialDiscount);
            }
        }

        output.setSpecialDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 列出該商家的特別折扣

```
public void execute(GetVenderSpecialDiscountInput input, GetVenderSpecialDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<SpecialDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `special_discount` " +
                        "WHERE `vender_id` = ?;");

        stmt.setInt( parameterIndex: 1, input.getVenderId());

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                SpecialDiscount specialDiscount = new SpecialDiscount();
                specialDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                specialDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                specialDiscount.setVenderName(GetCurrentSeasoningsDiscountUseCase.queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                specialDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                specialDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                specialDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                specialDiscount.setCategory(Category.valueOf(rs.getString( columnLabel: "category")));
                specialDiscount.setDiscountRate(rs.getDouble( columnLabel: "discount_rate"));

                discounts.add(specialDiscount);
            }
        }

        output.setSpecialDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 編輯特別折扣

```
public void execute(EditSpecialDiscountInput input, EditSpecialDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `special_discount` SET `vender_id`= ? WHERE `discount_code`= ?");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setInt( parameterIndex: 2, input.getDiscountCode());

        stmt.executeUpdate();

        updatePolicyDescription(connection, input.getPolicyDescription(), input.getDiscountCode());
        updateStartTime(connection, input.getStartTime(), input.getDiscountCode());
        updateEndTime(connection, input.getEndTime(), input.getDiscountCode());
        updateCategory(connection, input.getCategory().toString(), input.getDiscountCode());
        updateDiscountRate(connection, input.getDiscountRate(), input.getDiscountCode());

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.7 運費折扣管理

A. 新增運費折扣

```
public void execute(CreateShippingDiscountInput input, CreateShippingDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "INSERT `shipping_discount` (`vender_id`, " +
                        "`policy_description`, " +
                        "`start_time`, " +
                        "`end_time`, " +
                        "`target_price`)" +
                        " VALUES (?,?,?,?,?)");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setString( parameterIndex: 2, input.getPolicyDescription());
        stmt.setTimestamp( parameterIndex: 3, input.getStartTime());
        stmt.setTimestamp( parameterIndex: 4, input.getEndTime());
        stmt.setInt( parameterIndex: 5, input.getTargetPrice());

        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 列出目前有效的運費折扣

```java
public void execute(GetCurrentShippingDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<ShippingDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `shipping_discount` " +
                        "WHERE ? > `start_time` " +
                        "AND ? < `end_time`;");

        Timestamp currentTime = Timestamp.from(Instant.now());
        Calendar calendar = Calendar.getInstance(TimeZone.getTimeZone("Asia/Taipei"));  // set timezone
        stmt.setTimestamp( parameterIndex: 1, currentTime, calendar);
        stmt.setTimestamp( parameterIndex: 2, currentTime, calendar);

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                ShippingDiscount shippingDiscount = new ShippingDiscount();
                shippingDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                shippingDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                shippingDiscount.setVenderName(GetCurrentSeasoningsDiscountUseCase.queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                shippingDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                shippingDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                shippingDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                shippingDiscount.setTargetPrice(rs.getInt( columnLabel: "target_price"));

                discounts.add(shippingDiscount);
            }
        }

        output.setShippingDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 列出該商家的運費折扣

```
public void execute(GetVenderShippingDiscountInput input, GetVenderShippingDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        List<ShippingDiscount> discounts = new ArrayList<>();
        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT * " +
                        "FROM `shipping_discount` " +
                        "WHERE `vender_id` = ?;");

        stmt.setInt( parameterIndex: 1, input.getVenderId());

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                ShippingDiscount shippingDiscount = new ShippingDiscount();
                shippingDiscount.setDiscountCode(rs.getInt( columnLabel: "discount_code"));
                shippingDiscount.setVenderId(rs.getInt( columnLabel: "vender_id"));
                shippingDiscount.setVenderName(GetCurrentSeasoningsDiscountUseCase.queryVenderName(connection, rs.getInt( columnLabel: "vender_id")));
                shippingDiscount.setPolicyDescription(rs.getString( columnLabel: "policy_description"));
                shippingDiscount.setStartTime(rs.getTimestamp( columnLabel: "start_time"));
                shippingDiscount.setEndTime(rs.getTimestamp( columnLabel: "end_time"));
                shippingDiscount.setTargetPrice(rs.getInt( columnLabel: "target_price"));

                discounts.add(shippingDiscount);
            }
        }

        output.setShippingDiscountList(discounts);
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 編輯運費折扣

```
public void execute(EditShippingDiscountInput input, EditShippingDiscountOutput output) {
    try (Connection connection = this.mySQLDriver.getConnection()) {
        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `shipping_discount` SET `vender_id`= ? WHERE `discount_code`= ?");

        stmt.setInt( parameterIndex: 1, input.getVenderId());
        stmt.setInt( parameterIndex: 2, input.getDiscountCode());

        stmt.executeUpdate();

        updatePolicyDescription(connection, input.getPolicyDescription(), input.getDiscountCode());
        updateStartTime(connection, input.getStartTime(), input.getDiscountCode());
        updateEndTime(connection, input.getEndTime(), input.getDiscountCode());
        updateTargetPrice(connection, input.getTargetPrice(), input.getDiscountCode());

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

8.8 購物車管理

A. 新增商品至購物車

```
public void execute(UpdateShoppingCartInput input, UpdateShoppingCartOutput output) {

    try (Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `shopping_cart` SET `quantity` = ? WHERE `product_id`= ? AND `customer_id` = ?");

        stmt.setInt( parameterIndex: 1, input.getQuantity());
        stmt.setInt( parameterIndex: 2, input.getProductId());
        stmt.setInt( parameterIndex: 3, input.getCustomerId());

        stmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

B. 刪除購物車內商品

```
public void execute(DeleteShoppingCartItemInput input, DeleteShoppingCartItemOutput output) {

    try (Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement( sql: "SELECT `product_id`, `customer_id` FROM `shopping_cart` WHERE product_id = ? AND customer_id = ?;");
        stmt.setInt( parameterIndex: 1, input.getProductId());
        stmt.setInt( parameterIndex: 2, input.getCustomerId());
        ResultSet rs = stmt.executeQuery();

        stmt = connection.prepareStatement( sql: "DELETE FROM `shopping_cart` WHERE product_id = ? AND customer_id = ?");
        stmt.setInt( parameterIndex: 1, input.getProductId());
        stmt.setInt( parameterIndex: 2, input.getCustomerId());
        stmt.executeUpdate();

        if (!rs.next()) {
            output.setWorkCheck(false);
            throw new RuntimeException("delete shopping cart item failed");
        } else {
            output.setWorkCheck(true);
        }

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

C. 列出所有購物車內商品

```java
public void execute(GetShoppingCartItemsInput input, GetShoppingCartItemsOutput output) {

    try (Connection connection = this.mySQLDriver.getConnection()) {

        List<ShoppingCartItemsForOneVender> shoppingCartItems = new ArrayList<>();
        List<Item> itemList = new ArrayList<>();
        int lastVenderId = 0;
        String lastVenderName = "";
        int rowAmount;

        PreparedStatement stmt = connection.prepareStatement(
                sql: "SELECT `v`.`username`, sc.quantity, `p`.* " +
                        "FROM ((`shopping_cart` AS sc LEFT OUTER JOIN `product` AS p ON p.id = sc.product_id) LEFT OUTER JOIN `user` AS v ON v.id = p.vender_id) " +
                        "WHERE sc.customer_id = ? " +
                        "ORDER BY p.vender_id;");

        stmt.setInt( parameterIndex: 1, input.getId());

        try (ResultSet rs = stmt.executeQuery()) {
            rs.last();
            rowAmount = rs.getRow();
            rs.beforeFirst();
            while (rs.next()) {

                int id = rs.getInt( columnLabel: "id");
                int venderId = rs.getInt( columnLabel: "vender_id");
                String name = rs.getString( columnLabel: "name");
                String category = rs.getString( columnLabel: "category");
                int price = rs.getInt( columnLabel: "price");
                int stock = rs.getInt( columnLabel: "stock");
                String warehouseAddress = rs.getString( columnLabel: "warehouse_address");
                String description = rs.getString( columnLabel: "description");
                String pictureURL = rs.getString( columnLabel: "pictureURL");
                String venderName = rs.getString( columnLabel: "username");
                int quantity = rs.getInt( columnLabel: "quantity");

                Item item = new Item(id, venderId, name, category, price, stock, warehouseAddress, description, pictureURL, venderName, quantity);

                if (rs.getRow() == 1) {
                    lastVenderId = venderId;
                    lastVenderName = venderName;
                    itemList.add(item);
                } else if (lastVenderId == venderId) {
                    itemList.add(item);
                } else {
                    ShoppingCartItemsForOneVender shoppingCartItemsForOneVender = new ShoppingCartItemsForOneVender(lastVenderName, itemList);
                    shoppingCartItems.add(shoppingCartItemsForOneVender);
                    itemList = new ArrayList<>();
                    lastVenderId = venderId;
                    lastVenderName = venderName;
                    itemList.add(item);
                }
                if (rs.getRow() == rowAmount) {
                    ShoppingCartItemsForOneVender shoppingCartItemsForOneVender = new ShoppingCartItemsForOneVender(venderName, itemList);
                    shoppingCartItems.add(shoppingCartItemsForOneVender);
                }
            }
        }
        output.setShoppingCartItems(shoppingCartItems);

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

D. 更新購物車內商品數量

```java
public void execute(UpdateShoppingCartInput input, UpdateShoppingCartOutput output) {

    try (Connection connection = this.mySQLDriver.getConnection()) {

        PreparedStatement stmt = connection.prepareStatement(
                sql: "UPDATE `shopping_cart` SET `quantity` = ? WHERE `product_id`= ? AND `customer_id` = ?");

        stmt.setInt( parameterIndex: 1, input.getQuantity());
        stmt.setInt( parameterIndex: 2, input.getProductId());
        stmt.setInt( parameterIndex: 3, input.getCustomerId());

        stmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

Section 9 Conclusions and Future Work

9.1 Conclusions

藉著修習資料庫系統，我們規劃實作出 3C 產品線上購物系統「CCC Shop」。小組固定於每周三晚上一同開發，從資料的規劃，資料庫的建立，後端系統的控制，一直到前端的互動以及前端與後端的整合，團隊的積極合作使我們專案如期完成。

然而，沒有相關經驗的我們，在過程中遇到了種種問題，以下是比較重大的收穫以及我們的應對問題方式：

- 對 Vue 不熟悉：
  組員大部分並沒接觸過 Vue 這個語言。因此，我們分工作業，一部分的人研究前端，剩下的人維持開發進度，編撰 SRS 以及後端開發等。先請前端組把前端架構建立出來後，再教會所有組員，使大家都有幫忙開發前端的能力和經驗。

- VueX 的使用：
  前端開發時，發現部分資料應暫存於前端以便各種操作，同時能優化程式效能。像 User 的相關資料，若沒有存於前端，則每次需要時都要向後端發送 API，也無法做到前端傳遞資料的行為。所以我們搭配 Vue，學習使用了 VueX 的架構、技術。

- sessionStorage：
  使用者登入後，因為資料並沒有存在網頁中，重整頁面使用者就會自動被登出。因此我們用了 sessionStorage 來解決此問題，而且通過 sessionStorage 的內建函式，也使其他前端的操作更順利。

- 跨來源資源共用(CORS)：
  前端在向後端發送請求時，發生的非同源資料導致地跨來源請求問題(CORS policy)也是一大難題，嘗試許多方法後，最後我們將前端的 port 固定成 3000，並在後端加了白名單，使 web server 接受前端傳來的請求才順利解決這道難題。

- 後端架構：
  為了開發的順暢，以及考慮到可能由不同人員來擴充功能的情況，我們統一了程式架構。第一層根據物件做了分類；第二層參考 Clean Architecture 使用 adapter、usecase、entity 來分層，而在 usecase 中以 CRUD 為基礎進行撰寫。開發模式則採 TDD 的方式，根據需求寫完測試後，再實現 code。清楚的架構和統一的開發流程使得我們對彼此的開發進度掌握度高，在需要時能夠即時提供幫助，也讓我們在出錯時，能較輕鬆地找出錯誤原因。

- 時區：
  java 程式拿取時間時，因時區問題會拿到與預期不相同的資料。經過團隊的討論，為了操作上的直觀和便利，我們決定使用台灣的時間，並利用 calendar 的語法進行時間上的調整。

- Email 自動寄送功能：
  我們希望在送出訂單時，系統會自動寄送 Email 通知使用者，藉此給予反饋和確認。所以，我們替 CCC SHOP 建立了一個 google 的商家帳戶，在嘗試的過程遇到了許多安全性認證的問題，最後藉由 google 內部認證解決。

- 前後端的結合：
  我們使用 JSON 的格式來傳輸前後端的資料。在編寫傳輸的資料結構時，深深體悟到為別人著想的重要。即使分工，但整個程式環環相扣，若傳出的資料不便進行加工就容易導致程式的 bug 或拖延開發進度。所以在實作程式前，應該先講好彼此所需要的資料以及格式，以免需要不斷重複修改同樣的 code。

經由這次專案，我們體認到分工合作的益處。當面臨問題時，不必孤身一人埋頭苦戰，透過和同伴們討論，能更快速地找到解決辦法，不僅可以互相學習精進自我能力，也使開發的效率大幅提升。而固定的開會時間，除了讓我們的進度持續推進外，更能聽取大家的意見，不斷增進網站的品質。

製作這個專案，雖然花了大量的時間與精力，但我們深知自己受益匪淺，相信這些經驗的累積會成為我們的墊腳石，幫助我們在未來的開發更加順利，也希望留下這份紀錄，給往後遇到相關問題的人們一點啟發。

9.2 Future Work

針對未來的規劃，我們希望能把現有的程式碼做更好的整理與規劃，並且完善平台的功能與規範，提升使用者體驗，讓我們的平台能夠被實際運用。

為了平台之後的擴大發展，需要再次評估目前的軟體架構，與更仔細地檢查細節，例如：針對使用者的輸入進行限制、丟例外的訊息要明確易懂等。有些狀況即使目前不會發生（像是目前要刪除物件的函式不存在並不會丟出例外提示，因為從前端選取要刪除的物件不可能選到不存在的物件），但若為不正常的運作，就應盡可能地預防，以防未來程式變得十分複雜時，問題的修復難度會大幅提升，而造成修復時間過久，使用者的大量流失的窘狀。

未來，我們計畫在平台中新增更多功能，像是增加客戶與商家的聊天室、平台客服管

道、相關注意事項及法規的提示，還有建立更多元的報表方便商家進行各種分析，甚至直接寫入相應的分析方法，讓商家更輕鬆地看出各商品的熱門程度與自身經營狀況。如果有經費的話，希望能設計專業人員幫忙優化平台的介面、設計。

除了以上適宜的設計與充足的功能外，明訂使用者的相關規範也至關重要。針對非正常使用的消費者以及商家做出相應的處置，像是不斷退貨或亂給評價的消費者以及販賣瑕疵品甚至進行詐騙的店家應該有明確的規範使其受到禁止並接受制裁，給予相應的賠償。

在我們把平台開發完整後，希望能與外部機構簽約合作。例如：與物流公司談妥運費以及相關保障、與銀行推出相應的優惠、請律所協助處理法律的相關事務、請行銷公司宣傳平台等。為此，找尋願意支持平台的投資者是必要的，相對的，我們願意給予初始的贊助者相應的收益分成或使用平台的福利等。

在軟體正式上市之前，會做足測試保障我們系統的穩定性，例如先讓一小部分用戶實際使用軟體幾週，並回報 bug 或相關建議給開發人員，以及委託資安公司進行弱點掃描與滲透測試等。

由於課程時間有限，即使我們完成了課程的需求，還是覺得這個平台並不完整，希望未來我們精進自己的能力後，能讓這個平台上市，期許它成為家喻戶曉，市占率高的 3C 交易平台。

Glossary

Vue	一套以視圖層為基礎發展的 JavaScript 漸進式框架。
VueX	Vuex 是 Vue.js 應用程序的狀態管理模式。它充當應用程序中所有組件的集中存儲，其規則確保狀態只能以可預測的方式發生變化。
Vuetify	建立在 Vue 的 CSS 架構上，裡面提供了一些關於 UI 設計的主流方向與設計經驗來幫助使用者開發。
Apexcharts	一個現代化 JavaScript 圖表庫，用於使用簡單的 API 視覺化交互式圖表。
MariaDB	一個關聯式資料庫管理系統。
HTTP	全球資訊網上進行檔案交換的一套規則，用來界定 HomePage 與 HTTP 伺服器之間的互動。
Nginx	非同步框架的網頁伺服器，也可以用作反向代理、負載平衡器和 HTTP 快取。
SOLID	物件導向程式設計和物件導向設計的五個基本原則，使得程式設計師開發一個容易進行軟體維護和擴充的系統變得更加可能。
TDD	一種軟體開發過程中的應用方法，倡導先寫測試程序，再編碼實現。
Clean Architecture	由 Robert C. Martin 提出，嚴格遵守關注點分離設計原則的模型化系統藍圖。目標是創建簡單、可擴展且易於維護的分層架構。

References

Clean architecture	https://www.oncehub.com/blog/explaining-clean-architecture
TDD	https://martinfowler.com/bliki/TestDrivenDevelopment.html
SOLID	https://www.bmc.com/blogs/solid-design-principles/
Vue	https://vuejs.org/
VueX	https://vuex.vuejs.org/
Vuetify	https://vuetifyjs.com/en/
Apexcharts	https://apexcharts.com/
MariaDB	https://mariadb.org/
Diagrams.net	https://app.diagrams.net/

Appendix

CCC Shop zip file:

https://drive.google.com/drive/folders/1c6d85NOAnsM28ZsjS-UdZYV1cO7Hh6Ar?usp=sharing

Demo Video:

visitor scenario: https://youtu.be/_eldVc57jGM

customer scenario: https://youtu.be/QkzACw5h_xQ

staff scenario: https://youtu.be/pY14iRMhgj4

admin scenario: https://youtu.be/El9QJEijaSo

(youtube channel: https://www.youtube.com/playlist?list=PLuIs3VbIw0SmLry-_GpOQhUIGmaYZIKfI)

CCC SHOP Github:

https://github.com/CCC-Shop

final SRS and PPT: https://github.com/CCC-Shop/ccc-shop-report

source code and databasefile: https://github.com/CCC-Shop/ccc-shop-frontend

https://github.com/CCC-Shop/ccc-shop-backend