Read It Again

系統需求規格書

Software Requirements Specification (SRS)

Version: 1.0

姓名	學號	E-mail
陳文晟	110AC1005	t110AC1005@ntut.org.tw
林君曆	110810006	t110810006@ntut.org.tw
詹採晴	110590032	t110590032@ntut.org.tw
葉芸茜	B812110004	b812110004@tmu.edu.tw
湯青秀	B812110011	b812110011@tmu.edu.tw

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

01/03/2024

目錄 (Table of Contents)

Section 1 簡介 (Introduction)

1.1 目的 (Purpose)

大學生普遍都有尋找二手書的需求，不過並無較單一簡單使用的二手書交易平台，又或者是需要受到平台抽成。各自學校的 Facebook 社團雖然方便，但仍有許多不足，如書的買賣狀態、搜尋排序等，Facebook 較無相關的功能，缺乏便利性，因此我們希望藉由本學期資料庫系統課程所學，開發二手書交易系統，以提升二手書買賣體驗，同時增加實務開發的能力。

本系統希望達成以下目標:

1. 管理員：
   - 維護平台公正交易
   - 維護、更新平台系統
2. 賣家：
   - 管理商品、賣場
   - 交易訂單管理
   - 交易時聯繫雙方
   - 商品推廣
   - 瀏覽找尋商品需求
3. 買家：
   - 瀏覽、購買商品
   - 查詢交易訂單、購物車管理
   - 評論該次交易訂單、整體商場評分
   - 提出找尋商品公告

1.2 系統名稱 (Identification)

主系統：

線上二手書交易系統 (Online Secondhand Book Trading System, **OSBTS**)

子系統：

1. 後臺管理子系統 (Backend Management System, **BMS**)
2. 用戶管理子系統(User Management Subsystem, **UMS**)
3. 商品管理子系統 (Product Management Subsystem, **PMS**)
4. 交易管理子系統 (Transaction Management Subsystem, **TMS**)
5. 購物車管理子系統(Shopping Cart Management Subsystem, **SCMS**)
6. 商品搜尋推薦管理子系統(Search and Recommend Management Subsystem, **SRMS**)
7. 評價和評論子系統 (Review and Rating Subsystem, **RRS**)
8. 消息和通知子系統 (Messaging and Notification System, **MNS**)
9. 訂單配送管理子系統(Order and Logistics Management Subsystem, **OLMS**)
10. 報告和分析子系統 (Reporting and Analytics Subsystem, **RAS**)
11. 資料庫子系統(Database System, **DBS**)

1.3 概觀 (Overview)

在本系統開發項目中，我們精心選擇了一套先進的工具和框架，以實現高效、穩定且易於維護的開發流程。後端框架選用了 FastAPI，由於它的現代性、速度和易用性，能夠有效提升開發效率並保證系統性能。資料庫方面，我們選擇了 PostgreSQL，它不僅成熟、可靠，且提供強大的資料管理和索引功能，使得它成為了關聯式資料庫的理想選擇。此外，為了滿足可能的高讀取需求和即時功能，我們引入了 Redis 作為緩存和消息處理的解決方案，它能有效減輕 PostgreSQL 的讀取壓力並提升系統的回應速度。

在前端開發方面，React.js 被選中，其豐富的組件庫和活躍的社區支援，能快速構建出美觀且功能強大的用戶界面。對於容器化和部署，我們選用了 Docker 和 Docker Compose，它們提供了一個一致且可控的環境，降低了從開發到生產的配置複雜度。

版本控制是開發的重要環節，因此我們選擇了 Git 和 GitHub，以保證代碼的版本管理並提供一個協作的平台。綜上所述，這些選定的工具和框架不僅滿足了我們當前的開發需求，也為未來的擴展和維護打下了堅實的基礎。

1.4 符號描述 (Notation Description)

OSBTS 1.0.0	The OSBTS system will be labeled with the number 1.0.0.
BMS 1.1.0	The BMS components will be labeled with the number 1.1.n.
UMS 1.2.0	The UMS components will be labeled with the number 1.2.n.
PMS 1.3.0	The PMS components will be labeled with the number 1.3.n.
TMS 1.4.0	The TMS components will be labeled with the number 1.4.n.
SCMS 1.5.0	The SCMS components will be labeled with the number 1.5.n.
SRMS 1.6.0	The SRMS components will be labeled with the number 1.6.n.
RRS 1.7.0	The RRS components will be labeled with the number 1.7.n.
MNS 1.8.0	The MNS components will be labeled with the number 1.8.n.
OLMS 1.9.0	The OLMS components will be labeled with the number 1.9.n.
RAS 1.10.0	The RAS components will be labeled with the number 1.10.n.
DBS 1.11.0	The DBS components will be labeled with the number 1.11.n.

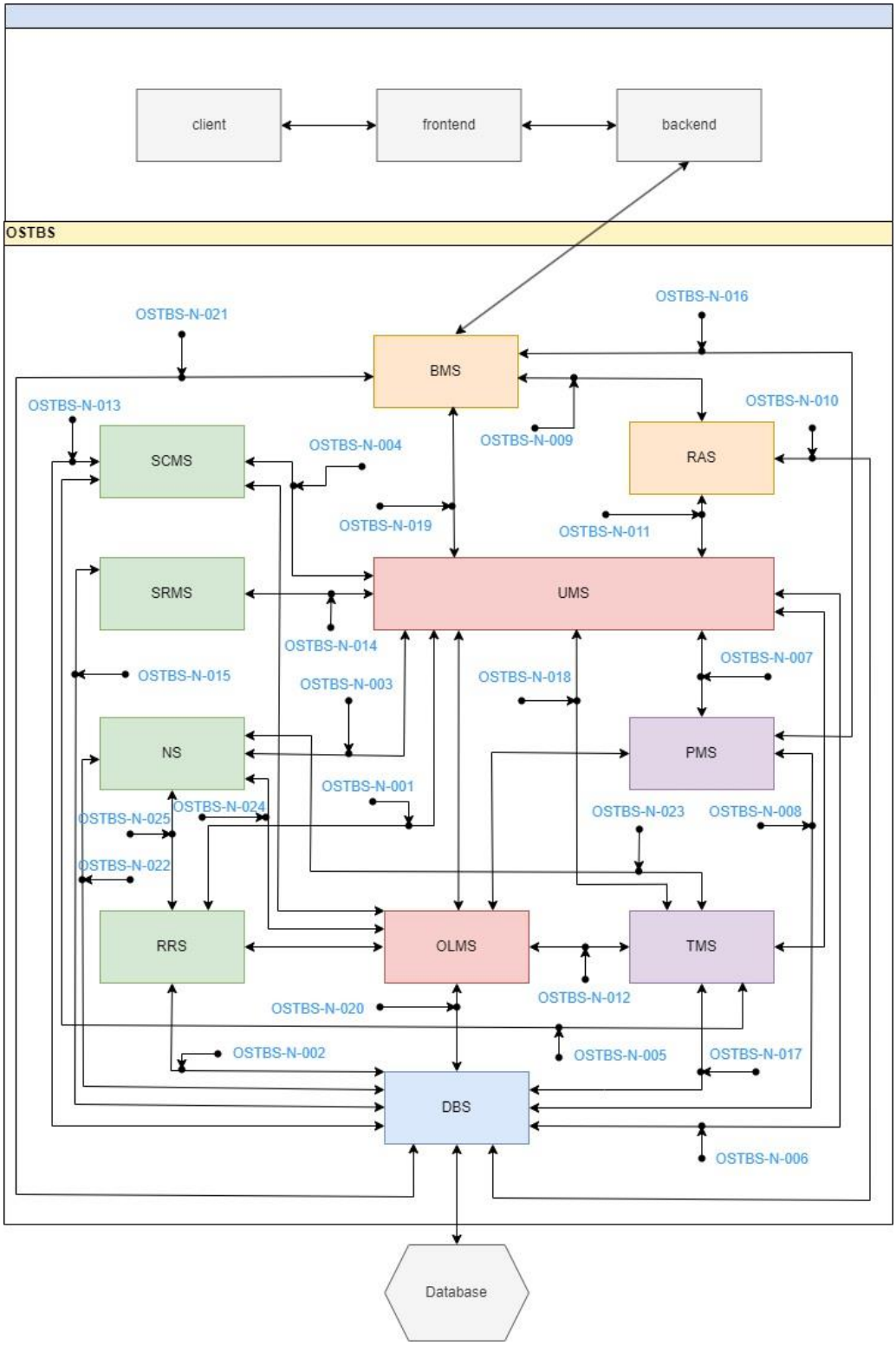
OSBTS-F-nnn	OSBTS 功能性需求(Functional Requirements)
OSBTS-N-nnn	OSBTS 非功能性需求(Non-Functional Requirements)
BMS-F-nnn	BMS 功能性需求(Functional Requirements)
BMS-N-nnn	BMS 非功能性需求(Non-Functional Requirements)
UMS-F-nnn	UMS 功能性需求(Functional Requirements)
UMS-N-nnn	UMS 非功能性需求(Non-Functional Requirements)
PMS-F-nnn	PMS 功能性需求(Functional Requirements)

PMS-N-nnn	PMS 非功能性需求(Non-Functional Requirements)
TMS-F-nnn	TMS 功能性需求(Functional Requirements)
TMS-N-nnn	TMS 非功能性需求(Non-Functional Requirements)
SCMS-F-nnn	SCMS 功能性需求(Functional Requirements)
SCMS-N-nnn	SCMS 非功能性需求(Non-Functional Requirements)
SRMS-F-nnn	SRMS 功能性需求(Functional Requirements)
SRMS-N-nnn	SRMS 非功能性需求(Non-Functional Requirements)
RRS-F-nnn	RRS 功能性需求(Functional Requirements)
RRS-N-nnn	RRS 非功能性需求(Non-Functional Requirements)
MNS-F-nnn	MNS 功能性需求(Functional Requirements)
MNS-N-nnn	MNS 非功能性需求(Non-Functional Requirements)
OLMS-F-nnn	OLMS 功能性需求(Functional Requirements)
OLMS-N-nnn	OLMS 非功能性需求(Non-Functional Requirements)
RAS-F-nnn	RAS 功能性需求(Functional Requirements)
RAS-N-nnn	RAS 非功能性需求(Non-Functional Requirements)
DBS-F-nnn	DBS 功能性需求(Functional Requirements)
DBS-N-nnn	DBS 非功能性需求(Non-Functional Requirements)

Section 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)

Scenario 1 ： 訪客操作概念

可於平台上主頁經由搜尋列查詢並瀏覽商品，在書本列表頁面可以排序、篩選商品，可以將點擊書本查看商品詳情，將喜歡的書本加入購物車，加入前必須先註冊並登入。

1. 瀏覽商品詳情
2. 查詢、排序、篩選商品
3. 將商品加入購物車前須登入或註冊為用戶

Scenario 2：買家用戶操作概念

可先登入或註冊為用戶，登入帳號後，用戶可到用戶中心查看、修改個人資料及密碼，亦可查看、修改、刪除運送地址。

用戶(買家)可於平台上主頁經由搜尋列查詢並瀏覽商品，在書本列表頁面可以排序、篩選商品，可以將點擊書本查看商品詳情，將喜歡的書本加入購物車。

到購物車的頁面後，可以刪除購物車內的商品，也可選擇要結帳的賣家，並查看有哪些書本，總價格為多少等資訊，可以選擇要運送的方式以及優惠券，最後提交訂單。

用戶(買家)可以根據訂單配送狀態查看訂單列表，若想要取消訂單也可以在「我的訂單」處理。另外，用戶(買家)可查看每一筆訂單的訂單狀態及配送進度，在收到書本完成交易過後，可以更改訂單狀態至完成並且針對訂單留下評論。

1. 登入或註冊為用戶
2. 個人檔案查看、編輯
3. 修改密碼
4. 個人運送地址的查看、編輯、刪除
5. 瀏覽商品詳情
6. 查詢、排序、篩選商品
7. 將商品加入、刪除購物車
8. 查看購物車
9. 對商品結帳，包含選擇運送方式和優惠券，並送出訂單
10. 更新或取消訂單，需要雙方同意
11. 根據訂單配送狀態查看訂單列表
12. 查看訂單狀態及配送進度
13. 確認訂單完成
14. 對訂單做評論及評價，須先完成訂單才可做評價

Scenario 3：賣家用戶操作概念

可先登入或註冊為用戶，登入帳號後，用戶可到用戶中心查看、修改個人資料及密碼，亦可查看、修改、刪除運送地址。

用戶(賣家)登入過後，可以進入書籍管理頁面，根據商品狀態查看賣場商品列表，並可新增、修改、刪除賣場商品並管理書本的上下架。

到優惠券管理頁面可根據優惠券狀態查看賣場優惠券列表，並可新增、修改、刪除優惠券。優惠券的部分分為免運、折價、特殊活動，可以設定優惠券的內容。

在訂單管理列表中可根據訂單狀態查看訂單列表，管理訂單狀態(還在進行中的訂單可以修改其狀態)。點擊訂單可以觀看訂單詳情及配送進度，已完成的訂單可以觀看買家的評論。

1. 登入或註冊為用戶
2. 個人檔案查看、編輯
3. 修改密碼
4. 個人運送地址的查看、編輯、刪除
5. 根據商品狀態查看賣場商品列表
6. 查看、新增、編輯、刪除賣場商品
7. 管理商品上下架
8. 根據訂單配送狀態查看訂單列表
9. 管理訂單(確認買家訂單)
10. 查看訂單配送進度
11. 查看訂單的評論及評價
12. 根據優惠券狀態查看賣場優惠券列表，分為免運、折價、特殊活動
13. 查看、新增、編輯、刪除優惠券

Scenario 4：平台管理者操作概念

用戶(平台管理者)登入後，可進行用戶權限操作，也可以對用戶的資料進行審查，並產生每期銷售報表，並對資料庫進行維護等相關作業。

2.3 功能性需求 (Functional Requirements)

需求編號	需求描述
OSBTS-F-001	用戶身分驗證、註冊
OSBTS-F-002	資料新增與修改
OSBTS-F-003	搜尋、篩檢、排序、推薦商品
OSBTS-F-004	將商品放入購物車
OSBTS-F-005	所選商品結帳
OSBTS-F-006	訪客結帳前需加入會員
OSBTS-F-007	用戶(買家)訂單追蹤
OSBTS-F-008	用戶(買、賣家)互評
OSBTS-F-009	用戶(賣家)有上下架與編輯商品資訊權限
OSBTS-F-010	平台管理者可建立商品分類標籤
OSBTS-F-011	平台管理員有關閉用戶之權限

OSBTS-F-012	平台管理員有封鎖用戶之權限
OSBTS-F-013	平台管理員可查看用戶(賣家)營業額
OSBTS-F-014	平台管理員可對資料庫子系統進行維護等相關作業

2.4 資料需求 (Data Requirements)

需求編號	需求描述
UMS-N-001	帳號
UMS-N-002	密碼
UMS-N-003	用戶名稱
UMS-N-004	性別
UMS-N-005	電話
UMS-N-006	電子郵件
UMS-N-007	用戶角色(買家/賣家/兩者)
BMS-N-001	商店名稱
BMS-N-002	商店標誌
PMS-N-001	上架狀態
PMS-N-002	商品資訊
PMS-N-003	照片
PMS-N-004	書籍分類
PMS-N-005	庫存量
TMS-N-001	交易賣家
TMS-N-002	交易買家
TMS-N-003	付款狀態
TMS-N-004	物流狀態
TMS-N-005	訂單狀態
TMS-N-006	訂單號碼
SCMS-N-001	商品資訊
SRMS-N-001	用戶搜索記錄
SRMS-N-002	關注店鋪
RRS-N-001	買家評論分數、評論內容
RRS-N-002	賣家評論分數、評論內容
NS-N-001	消息內容
NS-N-002	通知用戶資訊
OLMS-N-001	訂單編號
OLMS-N-002	物流狀態
OLMS-N-003	收件位置資訊

OLMS-N-004	金流交易資訊
RAS-N-001	賣家數據
RAS-N-002	分析結果
DBS-N-001	網站資訊

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
OSBTS-N-001	資料表設計須簡潔
OSBTS-N-002	使用者瀏覽時之頁面加載時間不超過 5 秒
OSBTS-N-003	使用者搜尋之加載應於 5 秒內結束

2.5.2 資安需求 (Security Requirements)

需求編號	需求描述
OSBTS-N-004	確保用戶帳戶有足夠複雜的密碼要求，減少風險。
OSBTS-N-005	確保只有需要的人員可以訪問系統的敏感部分。
OSBTS-N-006	定期更新和修補操作系統、應用程式和安全軟體，以防範已知的安全漏洞。
OSBTS-N-007	定期備份數據，建立復原計劃，以應對數據損失或系統中斷。

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
OSBTS-N-008	用戶登入/註冊介面
OSBTS-N-009	用戶中心介面
OSBTS-N-010	書籍瀏覽介面
OSBTS-N-011	書籍詳細資訊介面
OSBTS-N-012	搜尋介面
OSBTS-N-013	購物車介面
OSBTS-N-014	訂單資料介面
OSBTS-N-015	商品管理介面
OSBTS-N-016	設定商品優惠介面
OSBTS-N-017	銷售報表介面

2.6.2 外部介面需求 (External Interface Requirements)

需求編號	需求描述
OSBTS-N-018	使用者透過瀏覽器通過 HTTP 通訊協定與 OSBTS Web 主機前端通訊

OSBTS-N-019	前端透過 HTTP 發送 API 請求給後端 Server

2.6.3 內部介面需求 (Internal Interface Requirements)

需求編號	需求描述
OSBTS-N-020	用戶管理子系統(買、賣家) 和評價和評論子系統間傳送、接收評論
OSBTS-N-021	評價和評論子系統和資料庫子系統傳送、接收商品評價的相關資訊
OSBTS-N-022	用戶管理子系統(買、賣家) 和消息和通知子系統間傳送、接收訊息
OSBTS-N-023	用戶管理子系統(買家) 和購物車管理子系統傳送、接收選取商品資訊
OSBTS-N-024	購物車管理子系統和交易管理子系統間傳送、接收購買商品、結帳和支付資訊
OSBTS-N-025	用戶管理子系統(買、賣家)和資料庫子系統間傳送、接收各用戶登入資訊
OSBTS-N-026	用戶管理子系統(賣家)和商品管理子系統間傳送、接收用戶欲上下架商品價格、庫存、分類標籤、時間戳記等資訊
OSBTS-N-027	商品管理子系統和資料庫子系統間傳送、接收產品分類等資訊
OSBTS-N-028	後臺管理子系統和報告和分析子系統間傳送、接收商店結帳及所得利潤的資訊
OSBTS-N-029	報告和分析子系統和資料庫子系統間傳送、接收財務收支及銷售數量的資訊
OSBTS-N-030	用戶管理子系統和報告和分析子系統間傳送、接收用戶交易金流及驗證資訊
OSBTS-N-031	交易管理子系統和訂單配送管理子系統間傳送、接收訂購人、訂單編號、配送方式與地點等資訊。
OSBTS-N-032	購物車管理子系統和資料庫子系統間傳送、接收用戶購物車資訊
OSBTS-N-033	用戶管理子系統(買家)和商品搜尋推薦管理子系統間傳送、接收用戶追蹤的商店資訊
OSBTS-N-034	商品搜尋推薦管理子系統和資料庫子系統間傳送、接收用戶追蹤的商店資訊
OSBTS-N-035	後臺管理子系統和商品管理子系統間傳送、接收優惠券的資訊
OSBTS-N-036	交易管理子系統和資料庫子系統間傳送、接收各筆交易的詳細資訊。
OSBTS-N-037	用戶管理子系統和交易管理子系統傳送、接收訂購人身分認證、支付信息資訊。
OSBTS-N-038	後臺管理子系統和用戶管理子系統(賣家)間傳送、接收商店詳細

	資訊及權限設定。
OSBTS-N-039	訂單配送管理子系統和資料庫子系統間傳送、接收寄送地址、物流進度等資訊。
OSBTS-N-040	後臺管理子系統和資料庫子系統間傳送、接收折扣及權限資訊。
OSBTS-N-041	消息和通知子系統和資料庫子系統間傳送、接收個人訊息、推薦通知、訂單收/付款通知。
OSBTS-N-042	交易管理子系統和消息和通知子系統間傳送、接收付款確認等通知。
OSBTS-N-043	訂單配送管理子系統和消息和通知子系統間傳送、接收訂單物流進度通知。
OSBTS-N-044	評論及評價子系統和消息和通知子系統間傳送、接收商品評分及評論內容。

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
OSBTS-N-045	使用 React.js 作為前端框架
OSBTS-N-046	使用 FastAPI 作為後端框架
OSBTS-N-047	使用 PostgreSQL 作為 DBMS ，使用 Redis 作為快取資料
OSBTS-N-048	使用 Docker 作為專案部屬環境
OSBTS-N-049	使用 Git 和 GitHub 作為版本控制工具和 Repo

2.7.2 安裝需求 (Installation Requirement)

需求編號	需求描述
OSBTS-N-050	使用 Docker Compose 自動建置服務

2.7.3 測試需求 (Test Requirements)

需求編號	需求描述
OSBTS-N-051	使用 CI/CD 相關軟體自動部屬測試

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

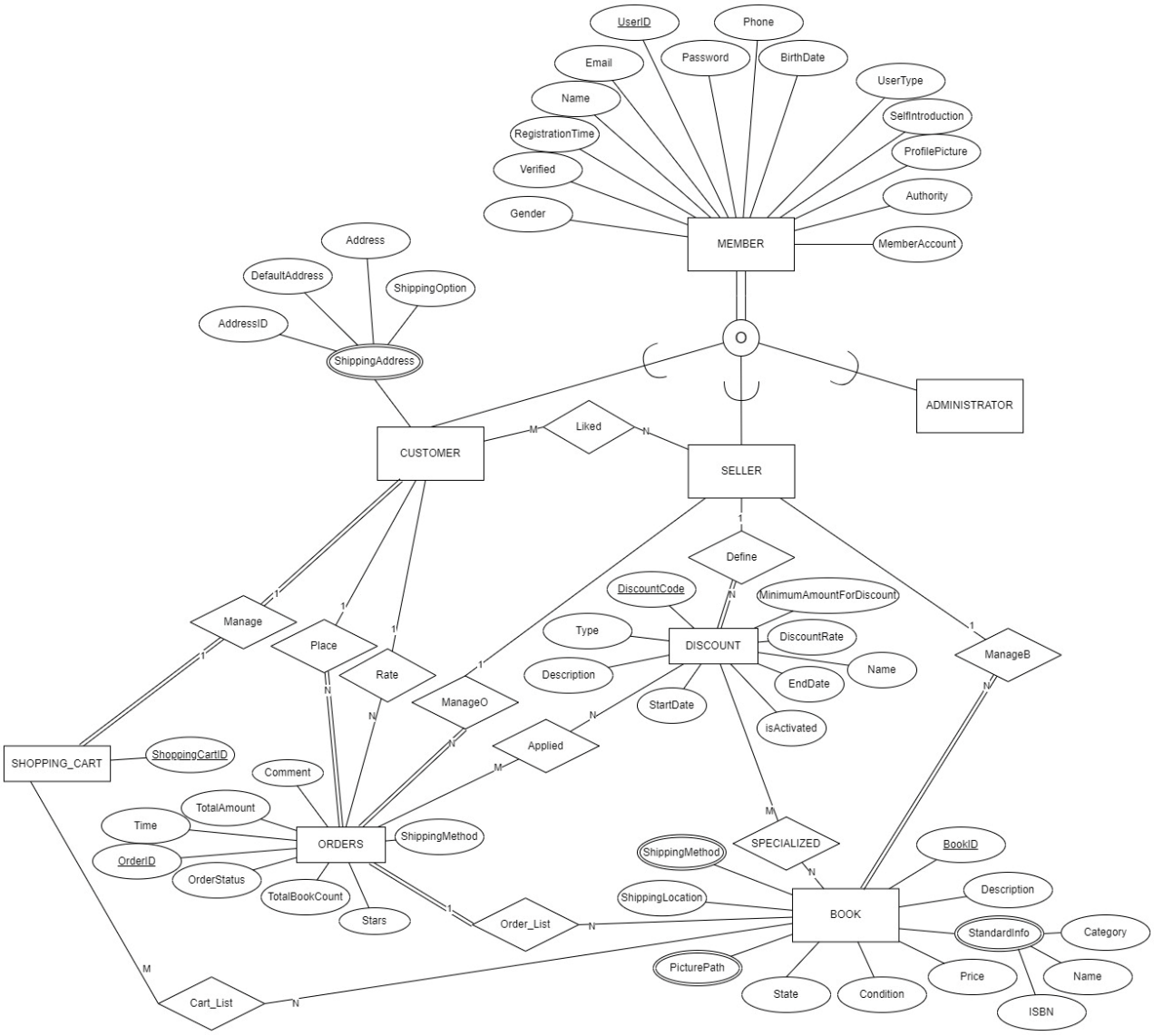
- 三種類型的折扣可以應用於同一次購買。
- 產品不能在同一時期內關聯到多個特殊活動折扣。
- 對於給定產品的特殊活動折扣，其時間段不能重疊。
- 每個購買的產品數量必須是大於零的整數。
- 購買產品的數量必須小於或等於該產品的庫存數量。
- 系統應該限制用戶對於未登錄的訪問，只有已註冊且已登錄的用戶才能執行購買、查

詢訂單等操作。

- 系統應該處理產品價格和庫存的實時更新，以確保庫存不會出現負數，並及時反映價格的變化。
- 如果用戶在結帳前中斷了購物過程，系統應該保留其購物車內容，以便他們在以後重新登錄時能夠繼續購物。
- 對於取消或退款的訂單，系統應該處理相應的庫存和資金返還操作，並通知相關方。
- 系統應該有日誌記錄功能，以追蹤用戶和管理員的操作，以及用於故障排除和安全性監控。
- 系統應該有數據備份和恢復機制，以確保數據不會因故障或意外損失而丟失。
- 任何有關支付的信息都應該按照相應的支付行業標準進行處理，並符合數據安全要求。

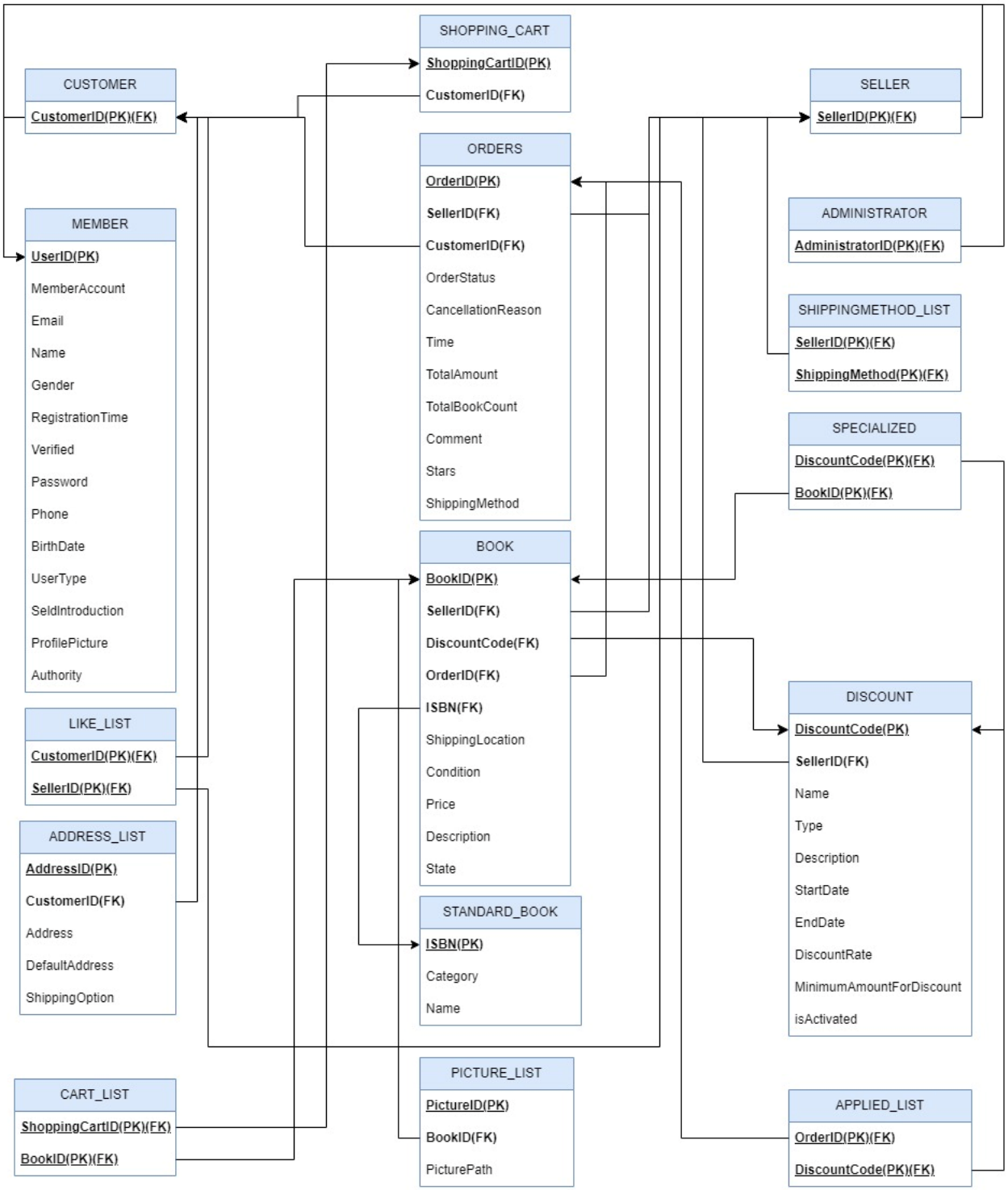
Section 3 資料庫概念設計（Conceptual Design of the Database）

3.1 Entity Relationship ER Model



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

4.1 Schema of the Database



MEMBER				
Description：存放各類與使用者有關的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
UserID	varchar	Primary	Not null	會員編號(id>0)
Email	varchar	Unique	Not null	會員郵件
Name	varchar		Not null	會員名稱
Gender	boolean			會員性別
RegistrationTime	timestamp		Not null	會員註冊時間
Verified	varchar		Not null	會員驗證狀態
Password	varchar		Not null	會員密碼
Phone	interger	Unique	Not null	會員手機號碼
BirthDate	timestamp		Not null	會員生日
Address	varchar			會員地址
UserType	varchar		Not null	會員類型
Seldlntroduction	varchar			會員自介
ProfilePicture	varchar			會員圖片
Authority	varchar			Token 授權碼

CUSTOMER				
Description：存放顧客的列表				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
CustomerID	varchar	Primary foreign	Not null	顧客 ID reference (MEMBER)

SELLER				
Description：存放賣家的列表				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
SellerID	varchar	Primary foreign	Not null	賣家 ID reference (MEMBER)

ADMINISTRATOR				
Description：存放管理員的列表				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
AdministratorID	varchar	Primary foreign	Not null	管理員 ID reference (MEMBER)

BOOK				
Description：存放書本有關的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
BookID	varchar	primary	Not null	書本 ID
SellerID	varchar	foreign	Not null	由哪一賣家上架
OrderID	varchar	foreign		被賣出的訂單編號
DiscountCode	varchar	foreign		Special event 的編號
ISBN	varchar	foreign	Not null	ISBN
ShippingLocation	varchar		Not null	書本出售地點
State	varchar		Not null	書本狀態
Condition	varchar		Not null	書本書況
Price	Integer		Not null	書本價格
Description	varchar		Not null	書本描述

STANDARD_BOOK				
Description：國際書籍標準資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
ISBN	varchar	primary	Not null	ISBN
Name	varchar		Not null	書本名稱
Category	varchar	foreign	Not null	書本分類

STANDARD_BOOK						
Description：購物車顧客對應關係的資料						
Attribute				**Domain Type**	**Key**	**Nullable**	**Description**
ISBN				varchar	primary	Not null	購物車 ID
Name	varchar	Unique	Not null	書本名稱			
Category				varchar		Not null	書本分類

SHOPPING_CART				
Description：購物車顧客對應關係的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**

ShoppingCartID	varchar	primary	Not null	購物車 ID
CustomerID	varchar	foreign	Not null	顧客 ID reference (MEMBER)

ORDERS				
Description：訂單相關的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
OrderID	varchar	primary	Not null	訂單 ID
SellerID	varchar	foreign	Not null	賣家 ID reference (MEMBER)
CustomerID	varchar	foreign	Not null	買家 ID reference (MEMBER)
OrderStatus	varchar		Not null	訂單狀態
CancellationReason	varchar			訂單取消原因
Time	timestamp		Not null	下單時間
TotalAmount	Integer		Not null	訂單金額
TotalBookCount	Integer		Not null	訂單書本數量
Comment	varchar			訂單評論
Stars	INT			訂單評價(1-5)
ShippingMethod	varchar		Not null	訂單運送方式

DISCOUNT				
Description：折扣列表相關的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
DiscountCode	varchar	primary	Not null	優惠券 Code
SellerID	varchar	foreign	Not null	賣家 ID reference (MEMBER)
Name	varchar		Not null	優惠券名稱
Type	varchar		Not null	折扣類型
Description	Varchar		Not null	折扣描述
StartDate	Timestamp		Not null	折扣起始時間
EndDate	Timestamp		Not null	折扣結束時間
DiscountRate	Float		Not null	折扣率
TotalAmountForDiscount	Integer		Not null	折扣的總金額
Applied	bool		Not null	是否被應用(賣家可以中途停止優惠)

LIKE_LIST

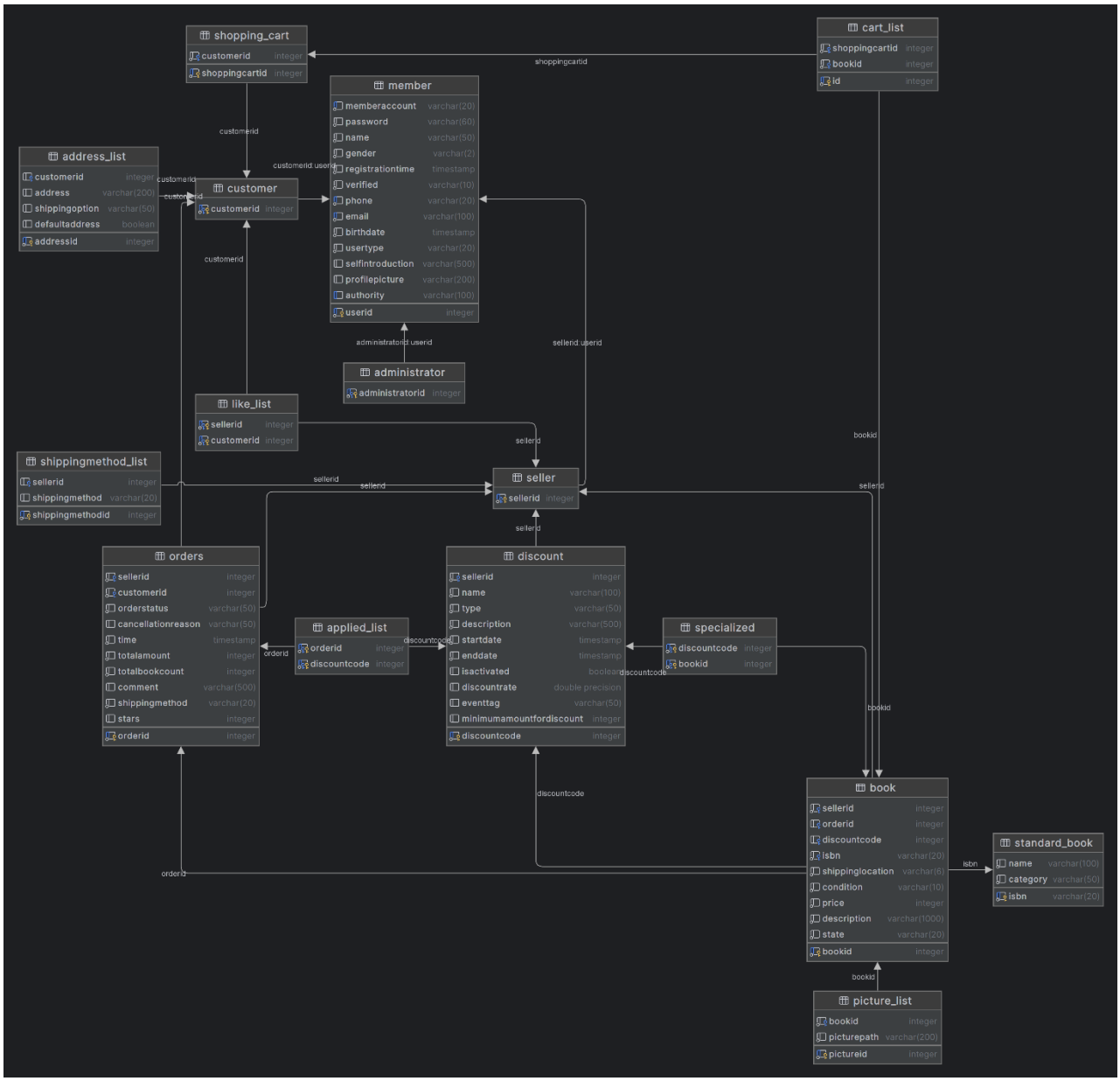
Description：喜歡賣家列表對應關係的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
SellerID	varchar	primary foreign	Not null	賣家 ID reference (MEMBER)
CustomerID	varchar	primary foreign	Not null	買家 ID reference (MEMBER)

APPLIED_LIST				
Description：已折價之訂單對應關係的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
OrderID	varchar	primary foreign	Not null	訂單 ID reference (ORDER)
DiscountCode	varchar	primary foreign	Not null	優惠券 Code reference (DISCOUNT)

Cart_List				
Description：購物車對應書籍關係的資料				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
ShoppingCartID	varchar	foreign	Not null	購物車 ID reference (SHOPPING_CART)
BookID	varchar	foreign	Not null	書本 ID reference (BOOK)

SPECIALIZED				
Description：折扣與書籍對應關係的資料(只有 special event 的折價會綁定書籍)				
Attribute	**Domain Type**	**Key**	**Nullable**	**Description**
DiscountCode	varchar	primary foreign	Not null	折價券 Code reference (Discount)
BookID	varchar	primary foreign	Not null	書籍 ID reference (Book)

4.2 The implementation of tables in target DBMS



4.3 SQL Statements Used to Construct the Schema

```sql
-- Create MEMBER table
CREATE TABLE MEMBER (
    UserID SERIAL PRIMARY KEY,
    MemberAccount VARCHAR(20) UNIQUE NOT NULL,
    Password VARCHAR(60) NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Gender VARCHAR(2) NOT NULL,
    RegistrationTime TIMESTAMP NOT NULL,
    Verified VARCHAR(10) NOT NULL,
    Phone VARCHAR(20) UNIQUE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    BirthDate TIMESTAMP NOT NULL,
    UserType VARCHAR(20) NOT NULL,
    SelfIntroduction VARCHAR(500),
    ProfilePicture VARCHAR(200),
    Authority VARCHAR(100) UNIQUE
);
```

```sql
-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
    CustomerID INTEGER PRIMARY KEY REFERENCES MEMBER(UserID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Create SELLER table
CREATE TABLE SELLER (
    SellerID INTEGER PRIMARY KEY REFERENCES MEMBER(UserID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Create ADMINISTRATOR table
CREATE TABLE ADMINISTRATOR (
    AdministratorID INTEGER PRIMARY KEY REFERENCES MEMBER(UserID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Create ORDERS table
CREATE TABLE ORDERS (
    OrderID SERIAL PRIMARY KEY,
    SellerID INTEGER NOT NULL REFERENCES SELLER(SellerID) ON UPDATE CASCADE ON DELETE CASCADE,
    CustomerID INTEGER NOT NULL REFERENCES CUSTOMER(CustomerID) ON UPDATE CASCADE ON DELETE CASCADE,
    OrderStatus VARCHAR(50) NOT NULL,
    CancellationReason VARCHAR(50),
    Time TIMESTAMP NOT NULL,
    TotalAmount INTEGER NOT NULL,
    TotalBookCount INTEGER NOT NULL,
    Comment VARCHAR(500),
    ShippingMethod VARCHAR(20) NOT NULL,
    Stars INTEGER
);
```

```sql
-- Create DISCOUNT table
CREATE TABLE DISCOUNT (
    DiscountCode SERIAL PRIMARY KEY,
    SellerID INTEGER NOT NULL REFERENCES SELLER(SellerID) ON UPDATE CASCADE ON DELETE CASCADE,
    Name VARCHAR(100) NOT NULL,
    Type VARCHAR(50) NOT NULL CHECK (Type IN ('shipping fee', 'seasoning', 'special event')),
    Description VARCHAR(500) NOT NULL,
    StartDate TIMESTAMP NOT NULL,
    EndDate TIMESTAMP NOT NULL,
    IsActivated BOOLEAN,
    DiscountRate FLOAT,
    EventTag VARCHAR(50),
    MinimumAmountForDiscount INTEGER
    CONSTRAINT check_shipping_fee_type CHECK (Type <> 'shipping fee' OR (Type = 'shipping fee' AND MinimumAmountForDiscount IS NOT NULL)), --shipping fee 一定要
    CONSTRAINT check_special_event_type CHECK (Type <> 'special event' OR (Type = 'special event' AND EventTag IS NOT NULL)), --special event一定要有event tag
    CONSTRAINT check_seasoning_type CHECK (Type <> 'seasoning' OR (Type = 'seasoning' AND DiscountRate IS NOT NULL AND MinimumAmountForDiscount IS NOT NULL))
);
```

```sql
CREATE TABLE STANDARD_BOOK (
    ISBN VARCHAR(20) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Category VARCHAR(50) NOT NULL
);
```

```sql
-- Create BOOK table
CREATE TABLE BOOK (
    BookID SERIAL PRIMARY KEY,
    SellerID INTEGER NOT NULL REFERENCES SELLER(SellerID) ON UPDATE CASCADE ON DELETE CASCADE,
    OrderID INTEGER REFERENCES ORDERS(OrderID) ON UPDATE CASCADE ON DELETE CASCADE,
    DiscountCode INTEGER REFERENCES DISCOUNT(DiscountCode) ON UPDATE CASCADE ON DELETE CASCADE,
    ISBN VARCHAR(20) NOT NULL REFERENCES STANDARD_BOOK(ISBN) ON UPDATE CASCADE ON DELETE CASCADE,
    ShippingLocation VARCHAR(6) NOT NULL,
    Condition VARCHAR(10) NOT NULL,
    Price INTEGER NOT NULL,
    Description VARCHAR(1000) NOT NULL,
    State VARCHAR(20) NOT NULL
);
```

```sql
-- Create SHOPPING_CART table
CREATE TABLE PICTURE_LIST (
    PictureID SERIAL PRIMARY KEY,
    BookID INTEGER NOT NULL REFERENCES BOOK(BookID) ON UPDATE CASCADE ON DELETE CASCADE,
    PicturePath VARCHAR(200) NOT NULL
);
```

```sql
-- Create SHOPPING_CART table
CREATE TABLE SHOPPING_CART (
    ShoppingCartID SERIAL PRIMARY KEY,
    CustomerID INTEGER NOT NULL REFERENCES CUSTOMER(CustomerID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql
-- Create LIKE_LIST TABLE
CREATE TABLE LIKE_LIST (
    SellerID INTEGER REFERENCES SELLER(SellerID) ON UPDATE CASCADE ON DELETE CASCADE,
    CustomerID INTEGER REFERENCES CUSTOMER(CustomerID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (SellerID, CustomerID)
);
```

```sql
-- Create APPLIED_LIST TABLE
CREATE TABLE APPLIED_LIST (
    OrderID INTEGER REFERENCES ORDERS(OrderID) ON UPDATE CASCADE ON DELETE CASCADE,
    DiscountCode INTEGER REFERENCES DISCOUNT(DiscountCode) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (OrderID, DiscountCode)
);
```

```sql
-- Create Cart_List TABLE
CREATE TABLE CART_LIST (
    ShoppingCartID INTEGER NOT NULL REFERENCES SHOPPING_CART(ShoppingCartID) ON UPDATE CASCADE ON DELETE CASCADE,
    BookID INTEGER NOT NULL REFERENCES BOOK(BookID) ON UPDATE CASCADE ON DELETE CASCADE,
    id SERIAL PRIMARY KEY
);
```

```sql
CREATE TABLE SHIPPINGMETHOD_LIST (
    ShippingMethodID SERIAL PRIMARY KEY,
    SellerID INTEGER REFERENCES SELLER(SellerID) ON UPDATE CASCADE ON DELETE CASCADE,
    ShippingMethod VARCHAR(20)
);
```

```sql
-- Create SPECIALIZED TABLE
CREATE TABLE SPECIALIZED (
    DiscountCode INTEGER REFERENCES DISCOUNT(DiscountCode) ON UPDATE CASCADE ON DELETE CASCADE,
    BookID INTEGER REFERENCES BOOK(BookID) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (DiscountCode, BookID)
);
```

4.4 SQL Statements Used to Insert the data - data population

```
-- Insert fake data into MEMBER table
INSERT INTO MEMBER (MemberAccount, Password, Name, Gender, RegistrationTime, Verified, Phone, Email, BirthDate, Address, UserType, SelfIntroduction, ProfilePicture, Authority) VALUES
('gang15', '_iXMFY*dt2', '錢建宏', '女性', '2020-10-05T22:40:14', '已認證', '022 54277143', 'qinli@example.net', '1989-10-17', '686 蘆洲昆陽街716號之9', 'Standard', '我是錢建宏，我愛大家', null, null),
('yli', 'Z3lWKBVr)4', '高志豪', '女性', '2022-08-13T11:47:34', '已認證', '0943-631687', 'weikang@example.net', '2001-03-17', '402 新竹市松山巷1段57號5樓', 'Standard', '我是高志豪，我愛大家', null, null),
('zzhu', '@@X6GKzb_L', '潘依婷', '保密', '2020-07-20T01:18:19', '已認證', '087 67390042', 'hcao@example.com', '1994-07-10', '67169 台中和街巷4號9樓', 'Standard', '我是潘依婷，我愛大家', null, null),
('zhangjing', '6)3&4RabfD', '吳詩婷', '女性', '2020-01-21T14:30:38', '已認證', '007 10731504', 'gang62@example.com', '1971-05-21', '88151 中壢縣文化路510號之8', 'Standard', '我是吳詩婷，我愛大家', null, null),
('guiying01', 'O^z2SwJdXQ', '董慧君', '男性', '2022-11-08T18:17:29', '已認證', '04-43614189', 'guiyingcao@example.com', '1959-11-05', '10115 北港學府街9號9樓', 'Standard', '我是董慧君，我愛大家', null, null),
('mingguo', '5^#E8IDo8c', '施雅文', '中性', '2022-01-17T07:49:27', '已認證', '09 8380281', 'fang23@example.net', '1977-01-05', '58671 竹北太平路7段88號3樓', 'Standard', '我是施雅文，我愛大家', null, null),
('leihuang', 'Dx%!kKJ1)0', '鄧志豪', '女性', '2020-03-19T19:46:23', '已認證', '07 1944666', 'xiulan58@example.com', '1994-06-12', '453 板橋大橋頭街7段686號之2', 'Standard', '我是鄧志豪，我愛大家', null, null),
('wei57', 'JC#22oUfI_', '林淑芬', '男性', '2020-12-01T07:33:25', '已認證', '099 69654723', 'kongchao@example.com', '1986-07-16', '12790 金門縣自立路762號之4', 'Standard', '我是林淑芬，我愛大家', null, null),
('luxiuying', 'E!7R@H%dO@', '翟哲瑋', '中性', '2021-10-06T06:14:05', '已認證', '03-1507567', 'bgong@example.net', '1945-04-18', '524 連江市民享巷4段449號3樓', 'Admin', '我是翟哲瑋，我愛大家', null, null),
('min97', 'cm@^Mdet(1', '沈婷婷', '保密', '2020-12-20T06:25:52', '已認證', '0980325724', 'min24@example.com', '1953-02-25', '65658 蘆竹新埔路338號1樓', 'Admin', '我是沈婷婷，我愛大家', null, null),
('min39', '$+TrCcIzo5', '張惠雯', '女性', '2022-05-30T09:28:59', '已認證', '(08) 93150254', 'jun26@example.com', '1952-12-16', '175 中和象山巷904號0樓', 'Admin', '我是張惠雯，我愛大家', null, null),
('xueguiying', '&a_5PcisRZ', '蔡佳蓉', '中性', '2022-10-07T15:50:30', '未認證', '006 62325613', 'li63@example.net', '1972-10-25', '691 竹北市雙連巷4段5號1樓', 'Standard', '我是蔡佳蓉，我愛大家', null, null),
('nmeng', 'dbR7LQbY$7', '巫佳蓉', '男性', '2022-10-24T06:37:39', '未認證', '02 9757313', 'wgao@example.com', '1933-03-22', '26427 花蓮民族路5號1樓', 'Standard', '我是巫佳蓉，我愛大家', null, null),
('li23', 'Uv@j9Kydo)', '張馨儀', '保密', '2022-01-12T20:19:55', '未認證', '0980-276511', 'yanjia@example.com', '1949-10-03', '182 大里勝利街9號1樓', 'Standard', '我是張馨儀，我愛大家', null, null),
('wei77', '%(VwzWFgF8', '田被涵', '保密', '2021-11-30T19:12:53', '未認證', '03-2953474', 'guiying89@example.org', '1971-01-17', '333 花蓮市古亭路9號之7', 'Standard', '我是田被涵，我愛大家', null, null),
('mengjuan', '$1Tf#_Di5T', '劉惠雯', '女性', '2022-05-15T05:41:16', '未認證', '(04) 29291402', 'xiulan69@example.org', '1998-12-17', '560 古坑天母街97號之6', 'Admin', '我是劉惠雯，我愛大家', null, null),
('minfang', 'k*4smUxj7u', '何雅萍', '女性', '2022-12-02T20:51:46', '未認證', '0919642301', 'ehe@example.com', '1974-10-21', '96965 板橋長安巷367號3樓', 'Admin', '我是何雅萍，我愛大家', null, null),
('dailei', 'ux0NWg+5%2', '王美琳', '女性', '2020-01-04T18:04:09', '未認證', '087 35604269', 'qzhang@example.org', '1993-03-20', '313 中壢永寧巷543號4樓', 'Admin', '我是王美琳，我愛大家', null, null),
('yan91', '%%KyL3pD$7', '張佳蓉', '男性', '2023-08-25T14:34:34', '未認證', '01-87605207', 'okong@example.net', '1987-11-20', '55790 阿里山新生街621號4樓', 'Admin', '我是張佳蓉，我愛大家', null, null),
('gongna', 'yYB61&Gp)1', '鄺佳穎', '男性', '2023-06-09T21:37:48', '未認證', '06-50953265', 'fang76@example.net', '1988-06-28', '457 金門芝山巷16號之5', 'Admin', '我是鄺佳穎，我愛大家', null, null);
```

```
-- Insert fake data into CUSTOMER table
INSERT INTO CUSTOMER (CustomerID) VALUES
(1),
(2),
(3),
(4),
(5),
(6),
(7),
(8);
```

```
-- Insert fake data into SELLER table
INSERT INTO SELLER (SellerID) VALUES
(1),
(2),
(3),
(4),
(5),
(6),
(7),
(8);
```

```
-- Insert fake data into ADMINISTRATOR table
INSERT INTO ADMINISTRATOR (AdministratorID) VALUES
(9),
(10),
(11);
```

```
-- Insert fake data into ORDER table
INSERT INTO ORDERS (SellerID, CustomerID, OrderStatus, CancellationReason, Time, TotalAmount, TotalBookCount, Comment, ShippingMethod, Stars) VALUES
(2, 1, 'Completed', null, '2021-01-02 15:00:00', 400, 2, '人很NICE!', '全家', 5);
```

```
-- Insert fake data into DISCOUNT table
INSERT INTO DISCOUNT (SellerID, Name, Type, Description, StartDate, EndDate, IsActivated, DiscountRate, EventTag, MinimumAmountForDiscount)
VALUES
(2, '暑期促銷', 'seasoning', '所有書籍9折', '2023-07-01 00:00:00', '2023-10-01 23:59:59', true, 0.9, null, 200),
(2, '免運券', 'shipping fee', '滿200免運費', '2024-01-01 00:00:00', '2024-02-01 23:59:59', true, null, null, 200),
(2, '好禮額外送', 'special event', '現在買小王子就額外送親子繪本', '2023-07-01 00:00:00', '2024-02-01 23:59:59', true, null, '額外好禮', null),
(3, '50折價券', 'seasoning', '只要滿200額外折價50元', '2023-07-01 00:00:00', '2024-02-01 23:59:59', true, 50.0, null, 200),
(3, '買一送一', 'special event', '現在哈利波特買一送一', '2023-07-01 00:00:00', '2024-02-01 23:59:59', true, null, '買一送一', null);
```

```
-- Insert fake data into STANDARD_BOOK table
INSERT INTO STANDARD_BOOK (ISBN, Name, Category) VALUES
('978-3-16-148410-0', '哈利波特', 'Fantasy'),
('978-3-16-148411-7', '小王子', 'Children'),
('978-3-16-148411-8', '微積分', 'Math');
```

```
-- Insert fake data into BOOK table
INSERT INTO BOOK (SellerID, OrderID, DiscountCode, ISBN, ShippingLocation , Condition, Price, Description, State) VALUES
(3, null, 5, '978-3-16-148410-0', '台北市', '新', 500, '這是一本關於魔法的書。', 'on sale'),
(2, null, 3, '978-3-16-148411-7', '高雄市', '二手', 300, '一本經典的兒童文學作品。', 'on sale'),
(2, 1, null, '978-3-16-148411-8', '高雄市', '二手', 200, '北科大一微積分', 'sold'),
(3, null, null, '978-3-16-148411-7', '高雄市', '二手', 200, '一本經典的兒童文學作品。', 'on sale');
```

```sql
-- Insert fake data into SHOPPING_CART table
INSERT INTO PICTURE_LIST (BookID, PicturePath) VALUES
(1, 'book01.jpg'),
(2, 'book02_01.jpg'),
(2, 'book02_02.jpg'),
(3, 'book03_01.jpg'),
(3, 'book03_02.jpg'),
(4, 'book04.jpg');
```

```sql
-- Insert fake data into SHOPPING_CART table
INSERT INTO SHOPPING_CART (CustomerID) VALUES
(1);
```

```sql
-- Insert fake data into LIKE_LIST table
INSERT INTO LIKE_LIST (SellerID, CustomerID) VALUES
(2, 1);
```

```sql
-- Insert fake data into APPLIED_LIST table
INSERT INTO APPLIED_LIST (OrderID, DiscountCode) VALUES
(1, 2);
```

```sql
-- Insert fake data into Cart_List table
INSERT INTO CART_LIST VALUES
(1, 1),
(1, 2);
```

```sql
INSERT INTO ADDRESS_LIST (CustomerID, Address, ShippingOption, DefaultAddress) VALUES
(1, '67169 台中和街巷4號9樓', 'home', true),
(1, '台北市大安區基隆路二段142之1號及142之2號', '7-ELEVEN', true),
(1, '台北市信義區吳興街156巷2弄2號4號1樓', '7-ELEVEN', false);
```

```
INSERT INTO SHIPPINGMETHOD_LIST(SellerID, ShippingMethod) VALUES
(1, '7-ELEVEN'),
(2, '7-ELEVEN'),
(3, '7-ELEVEN'),
(4, '7-ELEVEN'),
(5, '7-ELEVEN'),
(6, '7-ELEVEN'),
(7, '7-ELEVEN'),
(8, '7-ELEVEN'),
(1, '全家'),
(2, '全家'),
(3, '全家'),
(4, '全家'),
(5, '全家'),
(6, '全家'),
(7, '全家'),
(8, '全家'),
(1, '快遞'),
(2, '快遞'),
(3, '快遞'),
(4, '快遞'),
(5, '快遞'),
(6, '快遞'),
(7, '快遞'),
(8, '快遞');
```

```
-- Insert fake data into SPECIALIZED table
INSERT INTO SPECIALIZED VALUES
(3, 2),
(5, 1);
```

4.5 Estimation of Database Operations Frequency in Database

表格名稱	可能操作情境	預估頻率 (per day)	預估資料量 (tuples)	系統負擔 (worst case)
MEMBER	使用者登入驗證	200	400	80000 Query / Day
MEMBER	會員註冊資料	100	400	40000 Insert / Day
MEMBER	會員編輯資料	2	400	800 Update / Day
CUSTOMER	新增為買家	100	400	40000 Insert / Day
SELLER	新增為賣家	100	400	40000 Insert / Day
ADMINISTRATOR	新增為管理員	1	1	1 Insert / Day
BOOK	使用者查看書本	400	100	40000 Query / Day
BOOK	賣家上架書本	20	2	40 Insert / Day

BOOK	賣家修改書本	15	1	15 Update / Day
BOOK	賣家刪除書本	10	5	50 Delete / Day
DISCOUNT	賣家發布優惠券	10	5	50 Insert / Day
DISCOUNT	賣家查看優惠券	10	5	50 Query / Day
DISCOUNT	買家使用優惠券	10	1	10 Query / Day
DISCOUNT	賣家修改優惠券	10	2	20 Update / Day
DISCOUNT	賣家刪除優惠券	5	2	10 Delete / Day
SHOPPING_CART	買家新增購物車	100	100	10000 Insert / Day
SHOPPING_CART	買家查看購物車	100	50	5000 Query / Day
SHOPPING_CART	買家清空購物車	100	5	500 Delete / Day
ORDERS	買家下訂單	20	20	400 Insert / Day
ORDERS	買賣家更改訂單	20	15	300 Update / Day
ORDERS	買賣家查看訂單	20	20	400 Query / Day
ORDERS	買家刪除訂單	20	10	200 Delete / Day
LIKE_LIST	買家新增賣家為喜愛賣家	20	30	600 Insert / Day
LIKE_LIST	買家移除賣家為喜愛賣家	20	10	200 Delete / Day
LIKE_LIST	買家查看喜愛賣家	10	30	300 Query / Day
APPLIED_LIST	標記買家已使用的優惠券	10	1	10 Insert / Day
APPLIED_LIST	賣家查看已使用的優惠券	10	1	10 Query / Day
Cart_List	買家新增書本至購物車	100	200	20000 Insert / Day
Cart_List	買家查看購物車內的書本	100	50	5000 Query / Day
Cart_List	買家增減購物車內的書本	100	20	2000 Delete / Day
SPECIALIZED	商場管理者新增活動優惠券	10	2	20 Insert / Day
SPECIALIZED	買家使用活動優惠券	10	1	10 Query / Day
SPECIALIZED	商場管理者刪除活動優惠券	5	1	5 Delete / Day
SPECIALIZED	商場管理者修改活動優惠券	2	1	2 Update / Day

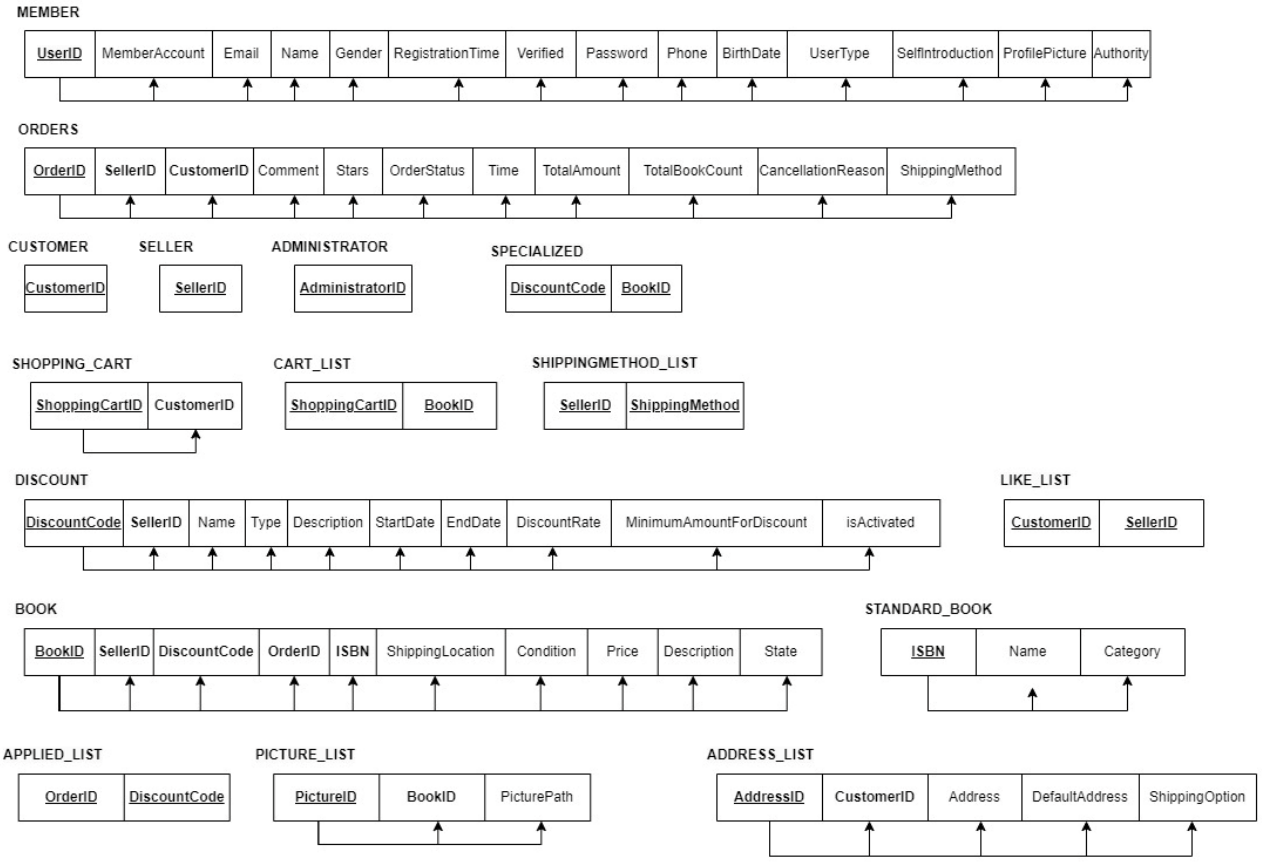
4.6 Estimation of Database Size in Database

表格名稱	欄位及其大小	總欄位大小 (Bytes)	平均元組數 (Rows)	表格大小 (KBytes)
MEMBER	UserID: 4 MemberAccount: 80 (20 * 4) Password: 200 (50 * 4) Name: 200 (50 * 4) Gender: 8 (2 * 4) RegistrationTime: 8 Verified: 40 (10 * 4) Phone: 80 (20 * 4) Email: 400 (100 * 4) BirthDate: 8 UserType: 80 (20 * 4) SelfIntroduction: 2000 (500 * 4) ProfilePicture: 800 (200 * 4) Authority: 400 (100 * 4)	5308	假設:1000	(5308*1000) / 1024 ≈ 5183 KB
CUSTOMER	CustomerID: 4	4	假設:500	(4*500) / 1024 ≈ 2 KB
SELLER	SellerID: 4	4	假設:500	(4*500) / 1024 ≈ 2 KB
ADMINISTRATOR	AdministratorID: 4	4	假設:500	(4*500) / 1024 ≈ 2 KB
ORDERS	OrderID: 4 SellerID: 4 CustomerID: 4 OrderStatus: 200 (50 * 4) Time: 8 TotalAmount: 4 TotalBookCount: 4 Comment: 2000 (500 * 4) Stars: 4 CancellationReason: 200(50 * 4) ShippingMethod: 80 (20 * 4)	2512	假設:2000	(2512*2000) / 1024 ≈ 4906 KB
BOOK	BookID: 4 SellerID: 4 OrderID: 4 DiscountCode: 4	4536	假設:1000	(4536*1000) / 1024 ≈ 4430 KB

	ISBN: 80 (20 * 4) ShippingLocation: 24 (6 * 4) Name: 400 (100 * 4) Condition: 12 (3 * 4) Price: 4 Description: 4000 (1000 * 4)			
STANDARD_BOOK	ISBN: 80 (20 * 4) Name: 400 (100 * 4) Category: 200 (50 * 4)	680	假設:1000	(680*1000) / 1024 ≈ 664 KB
DISCOUNT	DiscountCode: 4 SellerID: 4 Name: 400 (100 * 4) Type: 200 (50 * 4) Description: 2000 (500 * 4) StartDate: 8 EndDate: 8 DiscountRate: 4 MinimumAmountForDiscount: 4 isActivated: 1	2629	假設:300	(2629*300) / 1024 ≈ 770 KB
SHOPPING_CART	ShoppingCartID: 4 CustomerID: 4	8	假設:1000	(8*1000) / 1024 ≈ 8 KB
LIKE_LIST	SellerID: 4 CustomerID: 4	8	假設:2000	(8*2000) / 1024 ≈ 16 KB
APPLIED_LIST	OrderID: 4 DiscountCode: 20	24	假設:500	(24*500) / 1024 ≈ 12 KB
Cart_List	ShoppingCartID: 4 BookID: 4	8	假設:1500	(8*1500) / 1024 ≈ 12 KB
SPECIALIZED	DiscountCode: 4 BookID: 4	8	假設:300	(8*300) / 1024 ≈ 2 KB
PICTURE_LIST	BookID: 4 PicturePath: 800(200*4)	804	假設:2000	(804*2000) / 1024 ≈ 1570 KB
ADDRESS_LIST	AddressID: 4 CustomerID: 4 Address: 800(200 * 4) DefaultAddress: 1 ShippingOption: 200(50 * 4)	1009	假設:2000	(1009*2000) / 1024 ≈ 1966 KB
SHIPPINGMETHOD_LIST	SellerID: 4 ShippingMethod: 80(20 * 4)	84	假設:2000	(84 * 2000) / 1024 ≈ 164 KB

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

5.1 Functional Dependencies

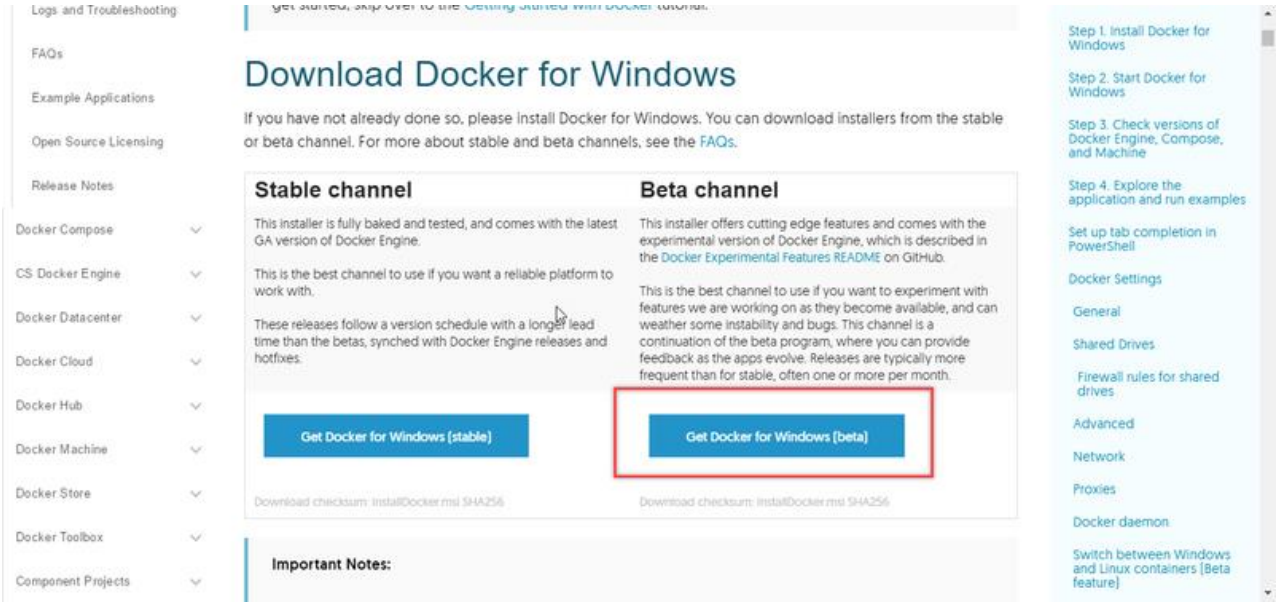


Section 6 數據庫系統的使用(The Use of the Database System)

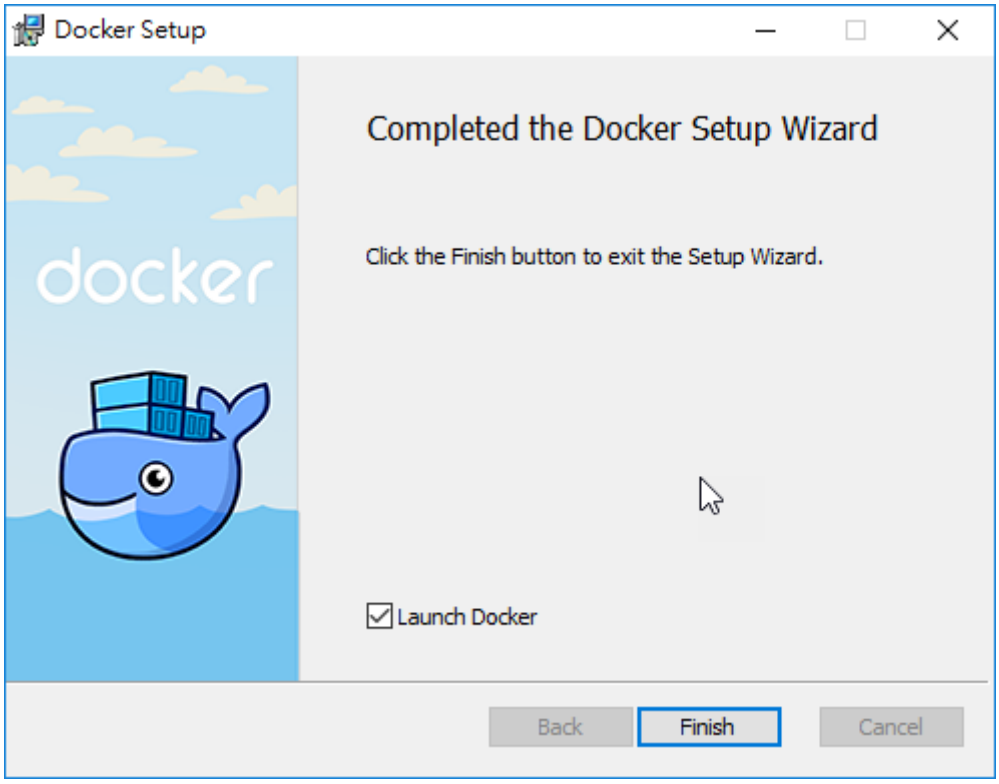
6.1 系統安裝 (System Installation Description)

先開啟 Hyper-V

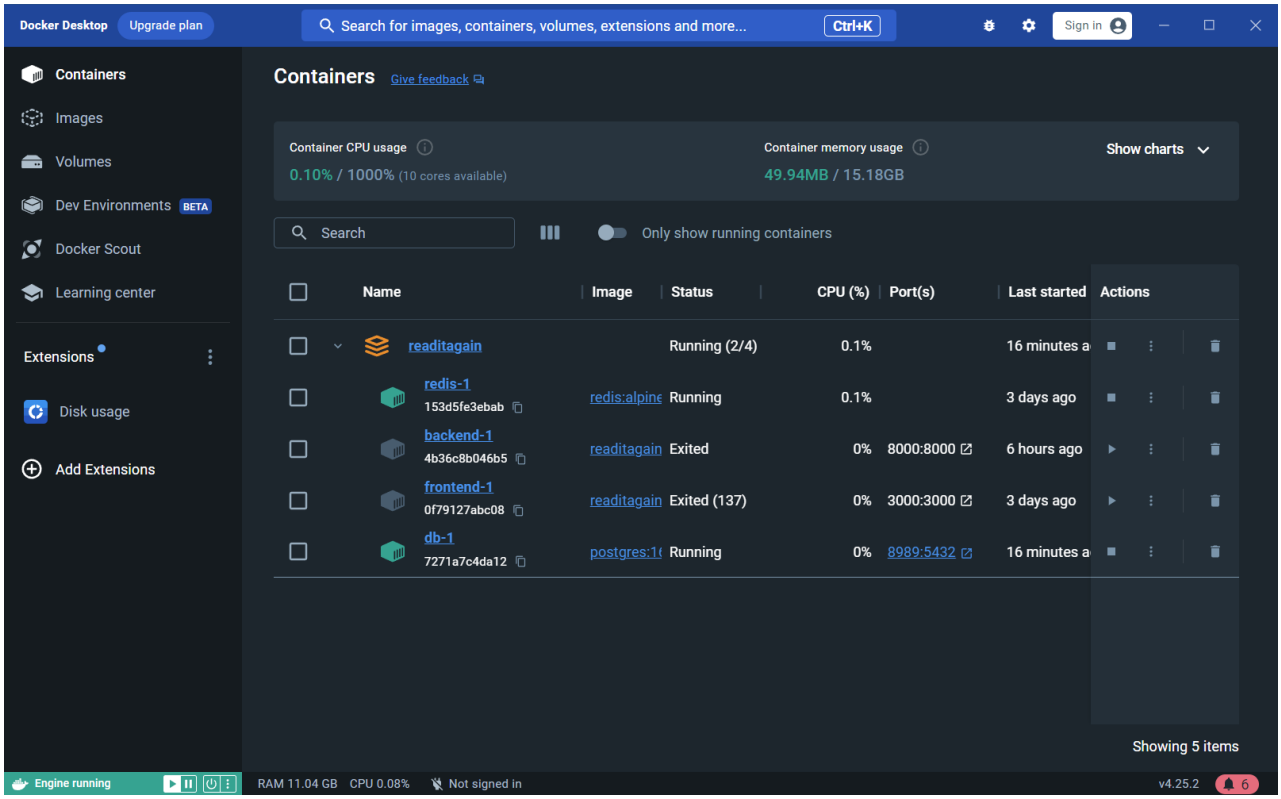
下載 Docker 安裝檔



安裝 Docker



執行打開 Docker 後



進入專案根目錄

執行圖片 中指示的指令

For Docker Deployment

1. **Building the Environment:** To set up your Docker environment, run the following command:

```
docker-compose up --build -d
```

2. **Stopping and Removing Containers:** To stop and remove the Docker containers, execute:

```
docker-compose down
```

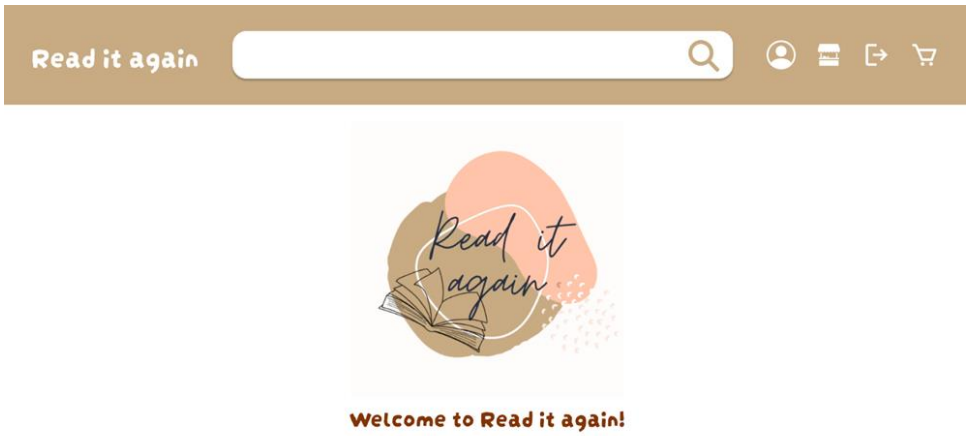
即可進入 127.0.0.1:3000 查看網頁
也可以在控制台執行以下指令來控制 Database

3. **Executing Commands in a Running Container:** For example, to run SQL commands in the database:

```
docker exec -it readitagain-db-1 sh # containerName: readitagain-db-1
psql -U admin readitagain-data       # connect to PostgreSQL db(readitagain-data)
# write some sql commands...
exit                                 # exit db
exit                                 # exit container
```

6.2 系統使用 (The Use of the System)

首頁:



登入註冊介面

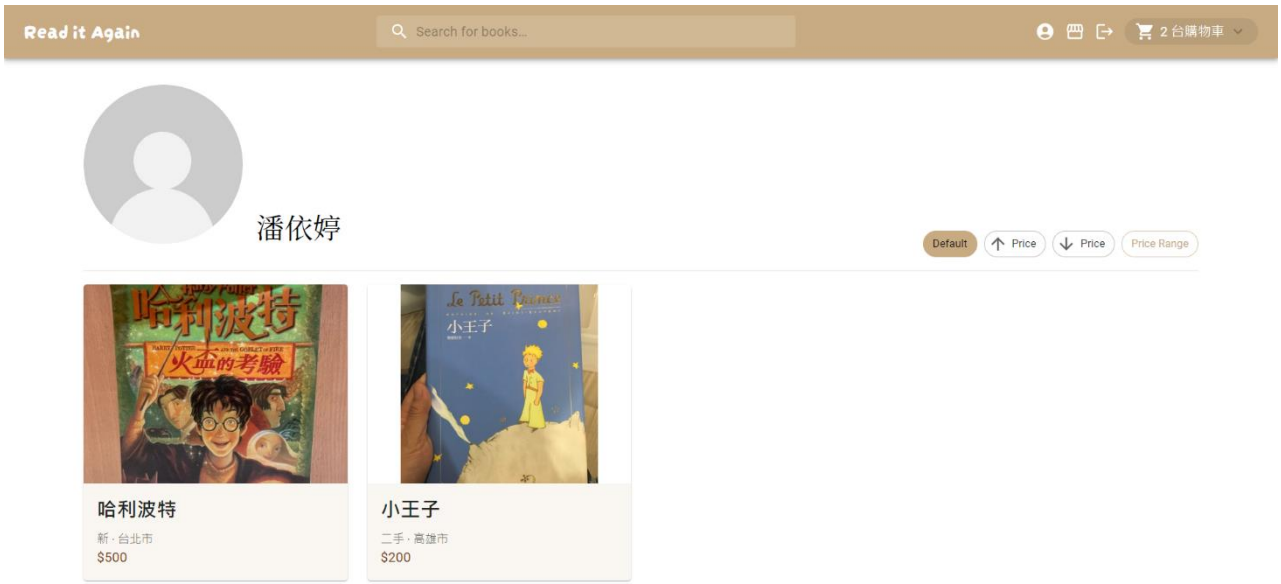

Read it again
Sign up
First Name 姓 *
Last Name 名字 *
Account 帳號 *
Password 密碼 *
Phone 手機號碼 *
Birthday 出生日期 *
Gender 性別 *
Email Address *
SIGN UP
已經有帳號了？登入
Copyright © Read it Again 2024.


Read it again
Sign in
Account *
Password *
Remember me
SIGN IN
沒有帳號？註冊
Copyright © Read it Again 2024.

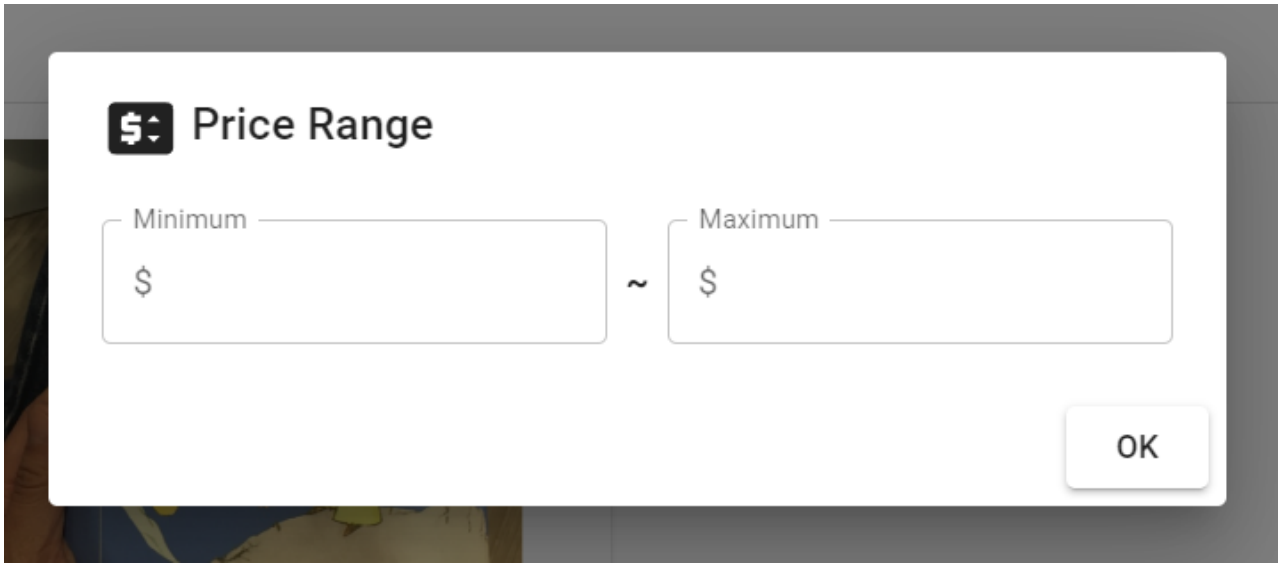
搜尋書本列表

待售書本詳細頁面

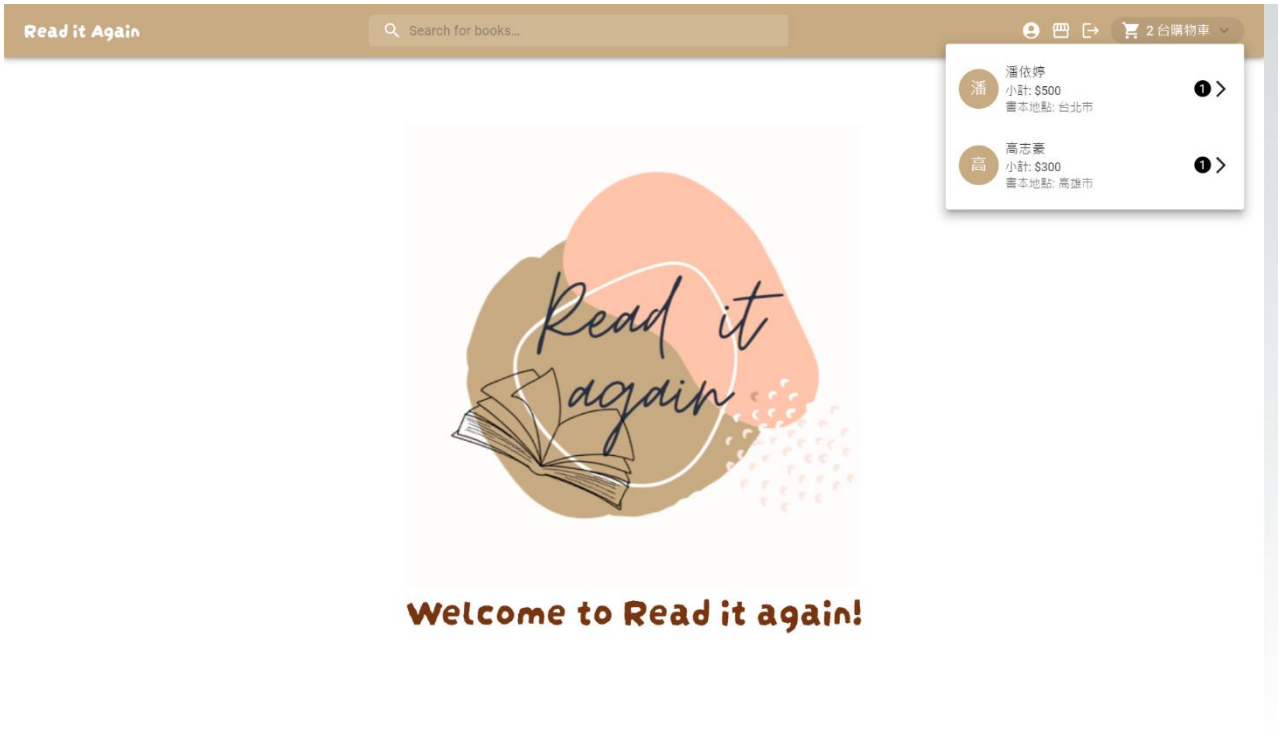
賣家賣場頁面

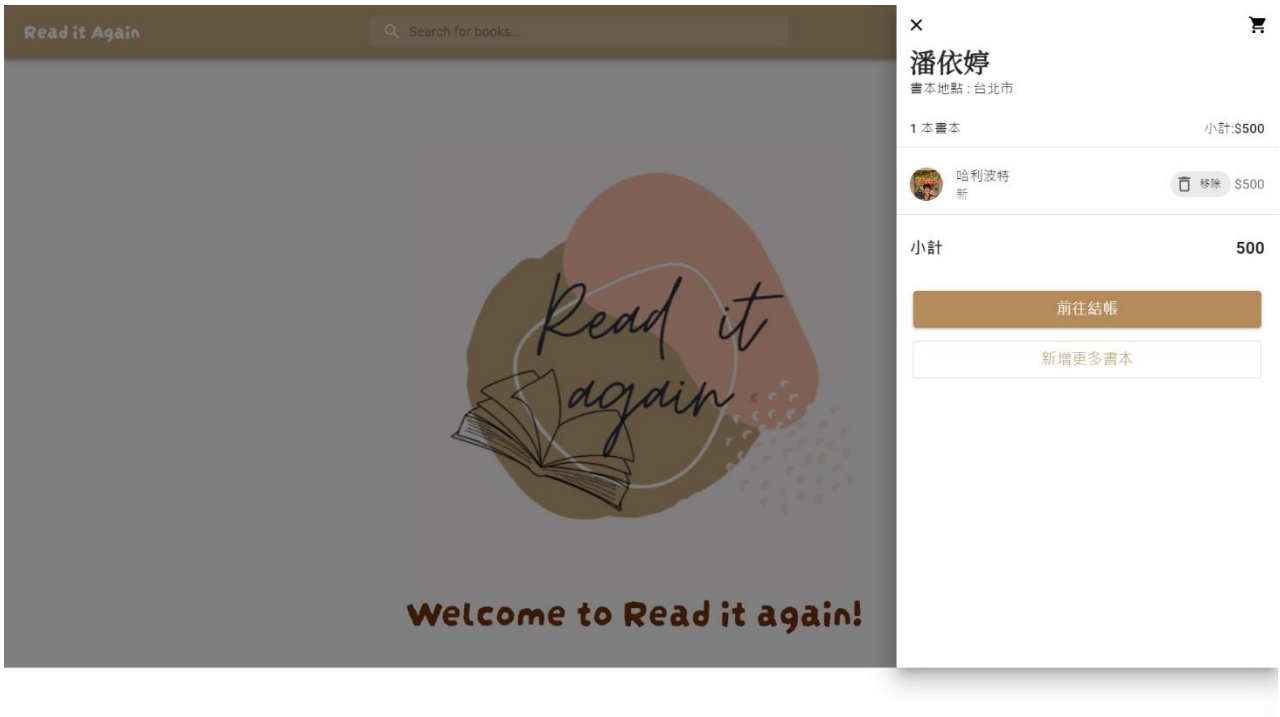


價格篩選彈窗介面



購物車頁面


Read it Again
Search for books...
2 台購物車
潘依婷
小計: $500
書本地點: 台北市
高志豪
小計: $300
書本地點: 高雄市
Read it again
Welcome to Read it again!

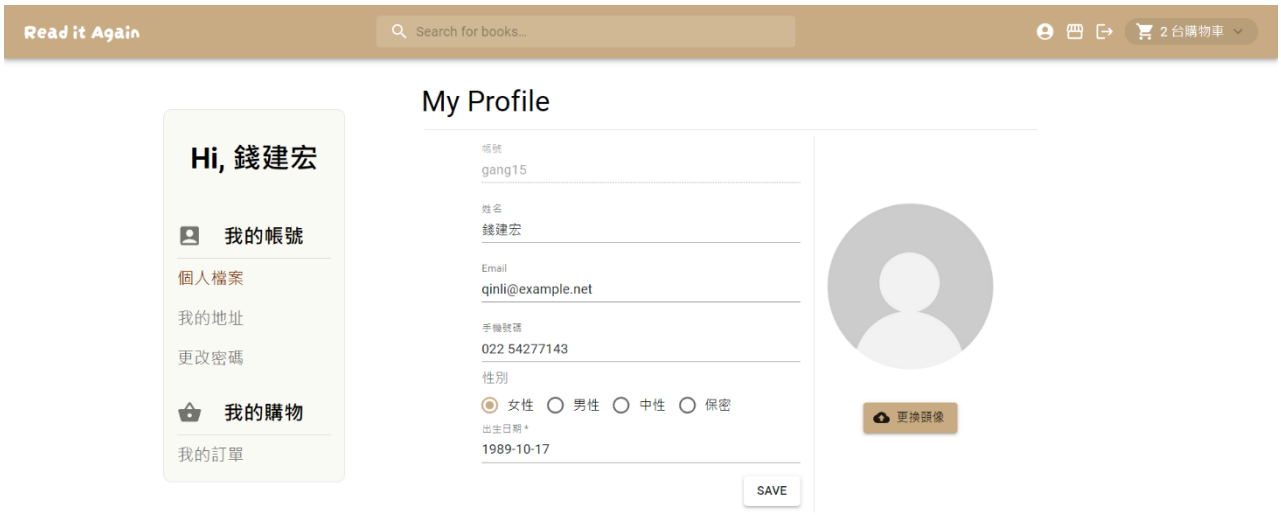

Read it Again
Search for books...
Read it again
Welcome to Read it again!
潘依婷
書本地點 : 台北市
1 本書本
小計:$500
哈利波特
新
移除
$500
小計
500
前往結帳
新增更多書本

結帳頁面

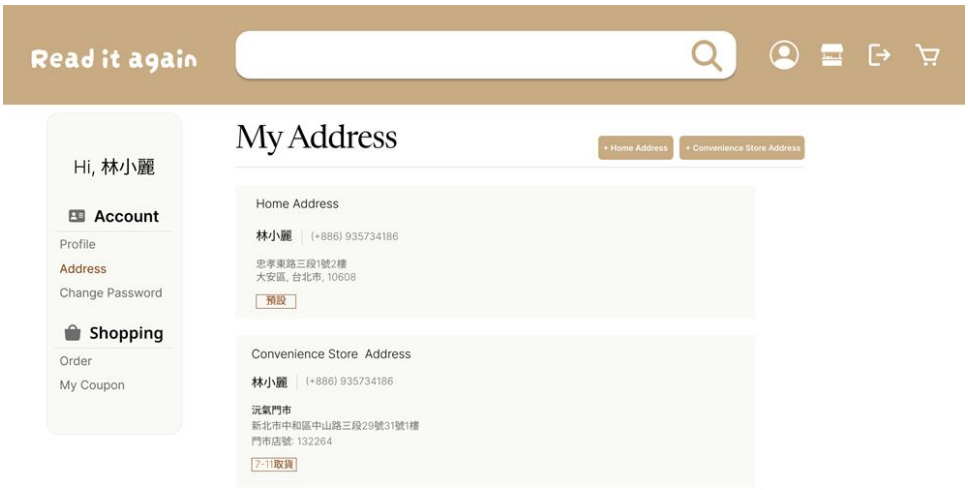
結帳時選擇優惠券之彈窗介面

結帳時選擇運送方式之彈窗介面

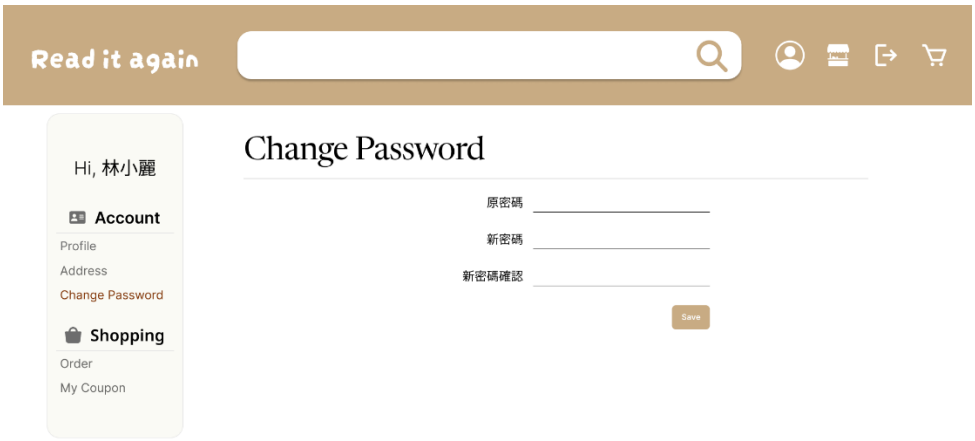
會員資訊頁



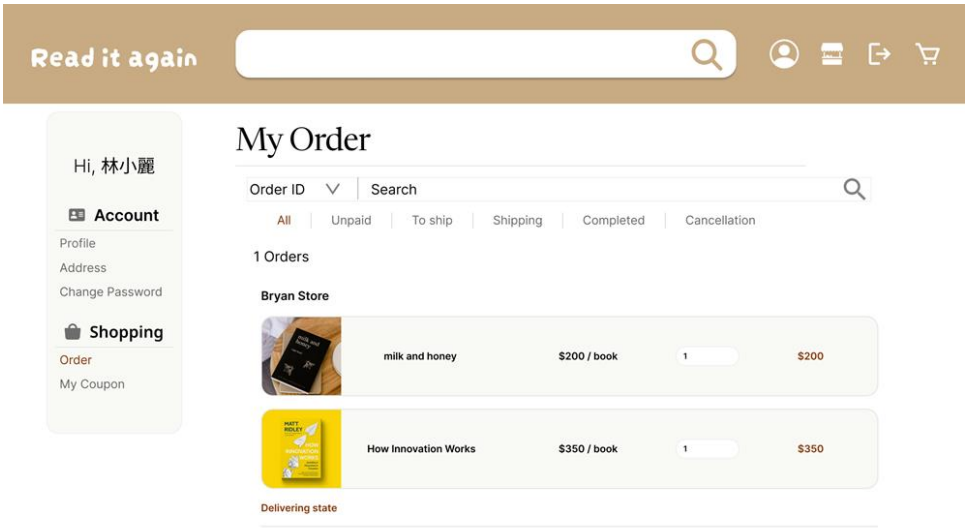
買家運送地址設定頁



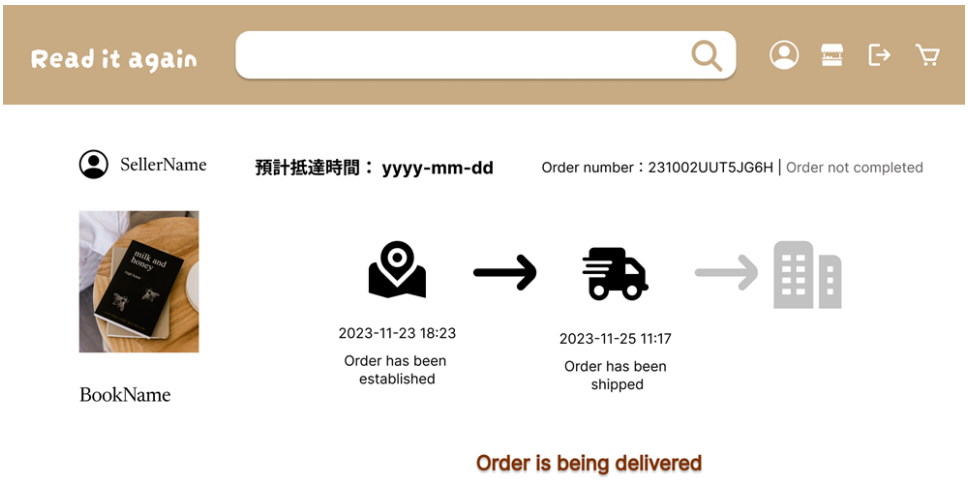
會員更改密碼頁



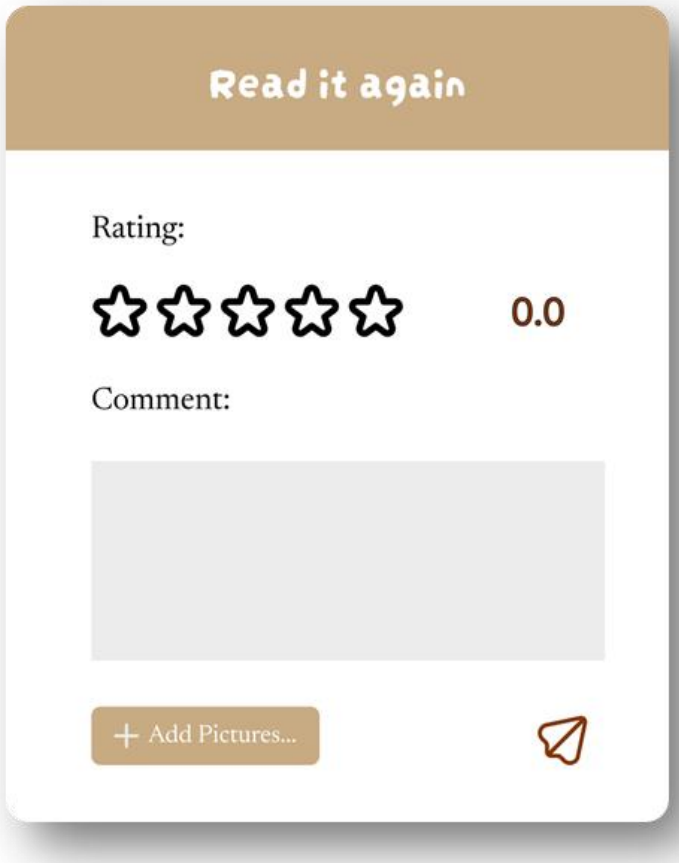
買家訂單頁



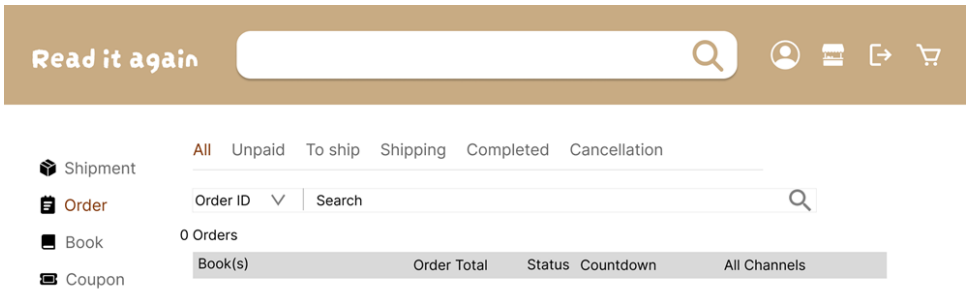
訂單之運送時間



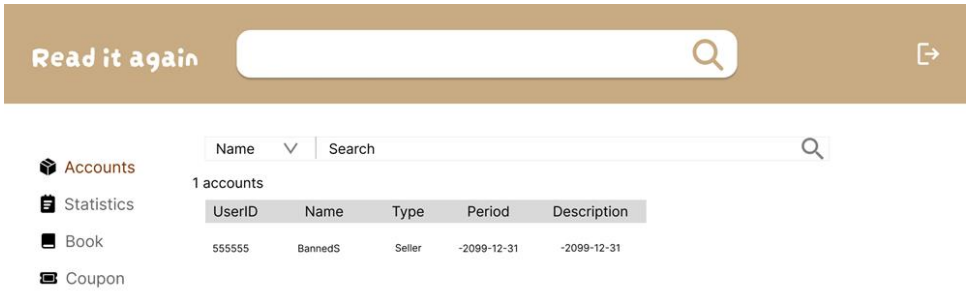
評價訂單時之彈窗介面



賣家中心介面



商場管理員後台頁面



資料庫優化建議(Suggestions of Database Tuning)

1. 資料庫設計的部分:
    a. 因為 **customer, seller, administrator** 繼承了 **member** 這個 **class(entity)**，所以在後端 **query** 用戶相關的資料時，需要在 **customer, seller, administrator** 中找到該用戶是否存在，接著再到 **member** 中進行第二次的搜尋，假設在一個表當中搜尋是線性時間，我們的查詢則會需要 **O(n^2)**的時間複雜度。若要解決這個問題，我們可以創建三個不同使用者身分的 **view**，在查詢使用者資料時，可以直接對指定的視圖查詢，這樣就只需要線性時間就能完成，效能會高許多
    b. 在瀏覽賣家頁面時，會顯示該賣家賣的書籍，然而我們的搜尋方式為迭代 **books** 表格所有的資料，留下 **sellerid** 為該賣家的資料，回傳到前端。不過若是可以使用老師上課後來教的方法，在 **book** 表格建立 **sellerid** 的索引，那在查詢賣家頁面的書籍時的速度會快非常多，同樣這樣的 **indexing** 方式可以應用在有 **1** 對多關係的的表格當中，如一個買家會有多個 **address**，可以在 **address_list** 的 **customerID** 建立索引，其他如 **LIKE_LIST, APPLIED_LIST, CART_LIST** 都可以應用此方法，不過須留意的是在 **indexing** 過後，雖在查詢方面的效能可以提升不少，但在資料的刪除插入是會有代價的，所以須留意該表格的使用情形，資料的變動頻繁與否也會是決定是否要採用此方法的其中考量因素之一。
2. 在本學期有限的時間內，我們未能完成聊天室功能的實作。然而，為了有效應對系統效能問題，引入 **Redis** 快取機制是一個極具合理性且高效的方法。這樣的做法將對系統的整體負載有著重大的幫助與改善。

Section 7 附加查詢和視圖(Additional Queries and Views)

取得用戶資料

```
select(Member).where(Member.memberaccount == token)
```

```
account = await get_current_user(token)
user = await session.scalars(select(Member).where(Member.memberaccount == account))
user = user.first()
```

找到該用戶的購物車

```
shoppingCart = await session.scalars(select(Shopping_Cart).where(Shopping_Cart.customerid == user.userid))
shoppingCart = shoppingCart.first()
```

取得購物車中所有的書籍編號

```
cart_items = await session.scalars(select(Cart_List).where(Cart_List.shoppingcartid == shoppingCart.shoppingcartid))
cart_items = cart_items.all()
```

找出書籍的首圖

```
picture = await session.scalars(select(Picture_List).where(Picture_List.bookid == item.bookid).order_by(Picture_List.pictureid))
picture = picture.first()
```

取得賣家的名稱

```
seller_name = await session.scalars(select(Member.name).where(Member.userid == seller_id))
seller_name = seller_name.first()
```

用戶註冊

```
CREATE OR REPLACE FUNCTION create_member_and_related_tables(
    memberaccount VARCHAR,
    hashed_password VARCHAR,
    member_name VARCHAR,
    member_gender VARCHAR,
    member_registration_time TIMESTAMP WITH TIME ZONE,
    member_verified VARCHAR,
    member_phone VARCHAR,
    member_email VARCHAR,
    member_birthdate TIMESTAMP WITHOUT TIME ZONE,
    member_user_type VARCHAR,
    member_self_introduction VARCHAR,
    member_profile_picture VARCHAR,
    member_authority VARCHAR
)
RETURNS VOID AS $$
DECLARE
    member_userid INT;
BEGIN
    -- 插入新的會員資料
    INSERT INTO MEMBER (MemberAccount, Password, Name, Gender, RegistrationTime, Verified,
                        Phone, Email, BirthDate, UserType, SelfIntroduction, ProfilePicture, Authority)
    VALUES (memberaccount, hashed_password, member_name, member_gender, member_registration_time, member_verified, member_phone,
            member_email, member_birthdate, member_user_type, member_self_introduction, member_profile_picture, member_authority)
    RETURNING UserID INTO member_userid;

    -- 插入相應的 Seller 資料
    INSERT INTO Seller (sellerid) VALUES (member_userid);

    -- 插入相應的 Customer 資料
    INSERT INTO Customer (customerid) VALUES (member_userid);

END;
$$ LANGUAGE plpgsql;
```

```
SELECT create_member_and_related_tables(
    'user123'::VARCHAR,
    'hashed_password'::VARCHAR,
    'John Doe'::VARCHAR,
    'M'::VARCHAR,
    CURRENT_TIMESTAMP::TIMESTAMP WITH TIME ZONE,
    '未認證'::VARCHAR,
    '1234567890'::VARCHAR,
    'john@example.com'::VARCHAR,
    '1990-01-01'::TIMESTAMP WITHOUT TIME ZONE,
    'Standard'::VARCHAR,
    'Hello, I am John Doe.'::VARCHAR,
    'default.jpg'::VARCHAR,
    'normal'::VARCHAR
);
```

搜尋書籍(包含篩選)

```
query = text("""
    SELECT *
    FROM Book
    WHERE Book.name ILIKE :name AND Book.state = 'on sale'
    ORDER BY
        CASE
            WHEN :sort_by = 'price_ascending' THEN Book.price
            WHEN :sort_by = 'price_descending' THEN Book.price DESC
        END
    WHERE
        Book.price BETWEEN :min_price AND :max_price;
""")
```

取得該賣家販售的書本

```
books = await session.scalars(select(Book).where(Book.sellerid == seller.sellerid))
books = books.all()
```

取得指定書籍的資料

```
book = await session.scalars(select(Book).where(Book.bookid == book_id))
book = book.first()
pictures = await session.scalars(select(Picture_List).where(Picture_List.bookid == book_id))
pictures = pictures.all()
```

配給用戶購物車(新增)

```
Cart = Shopping_Cart(customerid=user.userid)
session.add(Cart)
await session.commit()
await session.refresh(Cart)
```

取得購物車列表

```
item_exists = await session.scalars(select(Cart_List).where(Cart_List.shoppingcartid == shoppingCart.shoppingcartid, Cart_List.bookid == book_id))
```

新增購物車清單中的書

```
new_item = Cart_List(
    shoppingcartid=shoppingCart.shoppingcartid, bookid=book.bookid)
session.add(new_item)
await session.commit()
await session.refresh(new_item)
```

刪除購物車中的指定書籍

```
item_exists = await session.scalars(select(Cart_List).where(Cart_List.shoppingcartid == shoppingCart.shoppingcartid, Cart_List.bookid == book_id))
item_exists = item_exists.first()

await session.delete(item_exists)
await session.commit()
```

更改密碼

```
hashed_password = get_password_hash(new_password)
stmt = update(Member).where(Member.userid ==
                            user.userid).values(password=hashed_password)
await session.execute(stmt)
await session.commit()
```

修改個人資料

```
update_values = {
'memberaccount': name_input,
'email': email_input,
'phone': phone_input,
'gender': gender_input,
'birthdate': birthdate_input
}

stmt = (
    update(Member)
    .where(Member.userid == user.userid)
    .values(**update_values)
)

await session.execute(stmt)
await session.commit()
```

上傳大頭貼

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

更新用戶資訊

```
user.profilepicture = f"{user.memberaccount}.{img_type}"
await session.commit()
```

顯示地址

選取用戶資訊、選取該用戶之地址

增加地址

```
user = await get_current_user_data(token, session)
addresses = await session.scalars(select(Address_List).where(Address_List.customerid == user.userid))
addresses = addresses.all()
```

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

加入新地址

```
new_address = Address_List(customerid=user.userid, **address.dict())

# 將新地址新增到資料庫中
session.add(new_address)
await session.commit()
```

編輯地址資訊

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

依輸入更新不同欄位之資訊

```
if address.address:
    stmt = update(Address_List).where(Address_List.customerid == user.userid,
                                      Address_List.addressid == address_id).values(address=address.address)
    await session.execute(stmt)
if address.defaultaddress:
    stmt = update(Address_List).where(Address_List.customerid == user.userid,
                                      Address_List.addressid == address_id).values(defaultaddress=address.defaultaddress)
    await session.execute(stmt)
if address.shippingoption:
    stmt = update(Address_List).where(Address_List.customerid == user.userid,
                                      Address_List.addressid == address_id).values(shippingoption=address.shippingoption)
    await session.execute(stmt)

await session.commit()
```

刪除地址

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

選取指定地址

```
address = await session.scalars(select(Address_List).where(Address_List.customerid == user.userid, Address_List.addressid == address_id))
address = address.first()
```

刪除指定地址

```
await session.delete(address)
await session.commit()
```

刪除指定地址

```
address_still_exists = await session.scalars(select(Address_List).where(Address_List.customerid == user.userid, Address_List.addressid == address_id))
address_still_exists = address_still_exists.first()
```

選取優惠卷

取得該用戶之購物車

```
cart = await show_cart(token, session)
```

選出該賣場有的優惠卷

```
coupon_query = await session.scalars(select(Discount).where(Discount.sellerid == seller_id))
```

結帳

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

選取地址

```
sql_query = f'''
                select ADDRESS_LIST.ShippingOption, ADDRESS_LIST.Address, ADDRESS_LIST.DefaultAddress
                from ADDRESS_LIST
                where ADDRESS_LIST.ShippingOption in ( select SHIPPINGMETHOD_LIST.ShippingMethod
                                                       from SHIPPINGMETHOD_LIST
                                                       where SHIPPINGMETHOD_LIST.SellerID = {seller_id})
                and ADDRESS_LIST.CustomerID = {user.userid}
            '''
result = await session.execute(text(sql_query))
result = result.fetchall()
```

創建訂單

選取用戶資訊

```
user = await get_current_user_data(token, session)
```

插入新訂單資訊

```
stmt = insert(Orders).values(sellerid=seller_id, customerid=user.userid, orderstatus='To ship',
                        time=datetime.now(), totalamount=checkout_data.total_amount, totalbookcount=checkout_data.total_book_count
                        )
await session.execute(stmt)
await session.commit()
```

選取最新加入的訂單

```
orders = await session.scalars(select(Orders).order_by(desc(Orders.orderid)))
orders = orders.first()
```

取得該用戶之購物車

```
cart = await show_cart(token, session)
```

加入訂單資訊

```
for book in book_rows:
    sql_update = f'''
                    update BOOK
                    set OrderID = {orders.orderid}
                    where BookID = {book.bookid}
                  '''
    await session.execute(text(sql_update))
    await session.commit()
for book in book_rows:
    sql_update = f'''
                    update BOOK
                    set State = 'ordered'
                    where BookID = {book.bookid}
                  '''
    await session.execute(text(sql_update))
    await session.commit()
for coupon in selected_coupons:
    stmt = insert(Applied_List).values(
        orderid=orders.orderid, discountcode=coupon.discountcode)
    await session.execute(stmt)
    await session.commit()
```

取得賣家介紹

```
seller = await session.scalars(select(Seller).where(Seller.sellerid == seller_id))
seller = seller.first()
```

取得賣家

取得該賣家之用戶資訊

```
member = await session.scalars(select(Member).where(Member.userid == seller.sellerid))
member = member.first()
```

取得賣家商店頁面

取得賣家簡介

```
seller_info = await get_seller_info(seller_id, session)
```

取得賣家上架的書

```
query = select(Book).where(Book.sellerid ==
                           seller_id, Book.state == 'on sale')
```

將書依要求排序

```
if sort_by == 'price_ascending':
    query = query.order_by(Book.price)
elif sort_by == 'price_descending':
    query = query.order_by(Book.price.desc())

if min_price is not None and max_price is not None:
    query = query.where(Book.price >= min_price, Book.price <= max_price)

books = await session.scalars(query)
```

選取書本的第一張圖片

```
pictures = await session.scalars(select(Picture_List).where(Picture_List.bookid == book.bookid).order_by(Picture_List.pictureid))
picture = pictures.first()
```

選取買家

```
user = await get_current_user_data(token, session)
seller = await session.scalars(select(Seller).where(Seller.sellerid == user.userid))
```

選取賣家

```
user = await get_current_user_data(token, session)
customer = await session.scalars(select(Customer).where(Customer.customerid == user.userid))
```

取得訂單資訊

選取訂單中所有的書

```
books = await session.scalars(select(Book).where(Book.orderid == order_id))
books = books.all()
```

選取書本第一張圖片

```
pictures = await session.scalars(select(Picture_List).where(Picture_List.bookid == book.bookid).order_by(Picture_List.pictureid))
picture = pictures.first()
```

顯示買家訂單列表

選取買家資訊

選取買家所有訂單

```
customer = await get_current_customer(token, session)

orders = await session.scalars(select(Orders).where(Orders.customerid == customer.customerid))
orders = orders.all()
```

選取訂單中書本細節

```
book_details = await get_order_book_details(session, order.orderid)
```

選取賣家所有訂單

選取賣家資訊

選取賣家所有訂單

```
seller = await get_current_seller(token, session)

orders = await session.scalars(select(Orders).where(Orders.sellerid == seller.sellerid))
orders = orders.all()
```

選取訂單中書本細節

```
book_details = await get_order_book_details(session, order.orderid)
```

取得指定訂單細節

依用戶類型選取該用戶訂單

```
order_query = None
if user_type == "customer":
    order_query = select(Orders).where(
        Orders.customerid == user_id, Orders.orderid == order_id)
elif user_type == "seller":
    order_query = select(Orders).where(
        Orders.sellerid == user_id, Orders.orderid == order_id)

order = await session.scalars(order_query)
order = order.first()
```

選取該訂單中全部書本

```
books = await session.scalars(select(Book).where(Book.orderid == order.orderid))
books = books.all()
```

買家看特定訂單細節

取得買家資訊

取得該訂單細節

```
customer = await get_current_customer(token, session)
order_detail = await get_order_details(session, "customer", customer.customerid, order_id)
```

賣家看特定訂單細節

取得賣家資訊

取得該訂單細節

```
seller = await get_current_seller(token, session)
order_detail = await get_order_details(session, "seller", seller.sellerid, order_id)
```

賣家取得特定訂單評價

取得賣家資訊

取得該訂單中評論與分數

```
seller = await get_current_seller(token, session)
comment = await session.scalar(select(Orders.comment).where(Orders.orderid == order_id, Orders.sellerid == seller.sellerid))
stars = await session.scalar(select(Orders.stars).where(Orders.orderid == order_id))
```

更新訂單狀態

```
if next_status == 'Completed':
    book_state = 'sold'
    sql_update = f'''
                update BOOK
                set state = '{book_state}'
                where OrderID = {order_id}
                '''
    await session.execute(text(sql_update))
    await session.commit()

sql_update = f'''
                update ORDERS
                set OrderStatus = '{next_status}
                where OrderID = {order_id}
                '''
await session.execute(text(sql_update))
await session.commit()
sql_query = f'''
                select *
                from ORDERS
                where OrderID = {order_id}
            '''
order = await session.execute(text(sql_query))
order = order.first()
```

取得訂單

```
    if person == 'customer':
        customer = await get_current_customer(token, session)
        order = await session.scalars(select(Orders).where(Orders.orderid == order_id, Orders.customerid == customer.customerid))
    elif person == 'seller':
        seller = await get_current_seller(token, session)
        order = await session.scalars(select(Orders).where(Orders.orderid == order_id, Orders.sellerid == seller.sellerid))
    else:
        raise HTTPException(
            status_code=400, detail="Invalid person role provided")
    order = order.first()
```

送出取消訂單

```
order = await get_order(person, order_id, token, session)
if order:
    order.orderstatus = "Cancelling"
    await session.commit()
```

對方確認是否取消訂單

```
order = await get_order(person, order_id, token, session)
if order:
    if is_accepted:
        order.orderstatus = "Cancelled"
        order.cancellationreason = reason
        await session.commit()
```

訂單完成後買家可評論訂單

```
if order.orderstatus == "Completed":
    stmt = update(Orders).where(Orders.orderid == order.orderid).values(
        comment=comment_input, stars=stars_input)
    await session.execute(stmt)
    await session.commit()
```

賣家查看優惠券

```
seller = await get_current_seller(token, session)
coupons = await session.scalars(select(Discount).where(Discount.sellerid == seller.sellerid))
coupons = coupons.all()
```

賣家創建優惠券

```
new_coupon = Discount(sellerid=seller.sellerid, **coupon.dict())
session.add(new_coupon)
await session.commit()
```

賣家編輯優惠券

```
stmt = (
    update(Discount)
    .where(Discount.discountcode == discount_code)
    .values(**update_data)
)
try:
    await session.execute(stmt)
    await session.commit()
```

賣家刪除優惠券

```
await session.delete(discount)
await session.commit()
```

優惠券生效

```
stmt = (
    update(Discount)
    .where(Discount.discountcode == discount_code)
    .values(isactivated=activate)
)
try:
    await session.execute(stmt)
    await session.commit()
    return f"Coupon {discount_code} is activate={activate} now!"
```

賣家查看賣場書本列表，可查詢特定書籍

```
if book_id is not None:
    books = await session.scalars(select(Book).where(Book.sellerid == seller.sellerid, Book.bookid == book_id))
else:
    books = await session.scalars(select(Book).where(Book.sellerid == seller.sellerid))
    books = books.all()
```

賣家根據書本狀態及篩選關鍵字查看書本列表

```
if bookstate != 'All' and book.state != bookstate:
    continue
if keyword_type == 'Book name' and keyword and keyword.lower() not in book.name.lower():
    continue
pictures = await session.execute(select(Picture_List).where(Picture_List.bookid == book.bookid).order_by(Picture_List.pictureid))
picture = pictures.scalars().first()
```

賣家新增書本

```
book_state = 'no picture'
sql_update = f'''
             insert into BOOK (SellerID, ISBN, ShippingLocation , Name, Condition, Price, Description, Category, State)
             values ({seller.sellerid}, '{isbn}', '{ShippingLocation}', '{Name}', '{Condition}', {Price}, '{Description}', '{Category}', '{book_state}')
             '''
await session.execute(text(sql_update))
await session.commit()
```

```
sql_update = f'''
                insert into PICTURE_LIST (BookID, PicturePath)
                values ({book.bookid}, '{pictureName}')
                '''
await session.execute(text(sql_update))
await session.commit()
```

```
sql_update = update(Book).where(
    book.bookid == Book.bookid).values(state=book_state)
await session.execute(sql_update)
await session.commit()
```

賣家編輯書本內容(不含圖片)，根據對應的欄位進行修改

```
if isbn:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(isbn=isbn)
    await session.execute(stmt)
if ShippingLocation:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(shippinglocation=ShippingLocation)
    await session.execute(stmt)
if Name:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(name=Name)
    await session.execute(stmt)
if Condition:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(condition=Condition)
    await session.execute(stmt)
if Price:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(price=Price)
    await session.execute(stmt)
if Description:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(description=Description)
    await session.execute(stmt)
if Category:
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(category=Category)
    await session.execute(stmt)
if state:
    valid_types = ['on sale', 'removed']
    if state not in valid_types:
        raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST,
                            detail=f"Invalid state type. Supported types are {valid_types}")
    stmt = update(Book).where(Book.bookid ==
                              book_id).values(state=state)
    await session.execute(stmt)
await session.commit()
```

賣家上傳書本圖片，可一次上傳多本

```
sql_update = f'''
                    insert into PICTURE_LIST (BookID, PicturePath)
                    values ({book_id}, '{pictureName}')
                    '''
await session.execute(text(sql_update))
await session.commit()
```

賣家刪除書本圖片

```
await session.delete(pic)
await session.commit()
```

賣家刪除書本及其照片

```
await session.delete(pic_exist)
await session.commit()
```

```
await session.delete(book)
await session.commit()
```

賣家查看賣場書本詳情

```
seller = await get_current_seller(token, session)
book = await session.scalars(select(Book).where(Book.sellerid == seller.sellerid, Book.bookid == book_id))
book = book.first()
```

Glossary

References

Appendix