

RT-Mart

系統需求規格書

Software Requirements Specification (SRS)

Version: 1.1

姓名	學號	E-mail
胡育鈞	112500017	hyujun940529@gmail.com
陳駿逸	112590022	chunichen2005@gmail.com
高翎竣	112590034	tim0401.kao@gmail.com
鄭錦鑫	112590042	jinxintee@gmail.com
葉凱成	112590035	0528seng@gmail.com

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

01/06/2025

目錄 (Table of Contents)

Section 1 簡介 (Introduction)	1
1.1 目的 (Purpose)	1
1.2 系統名稱 (Identification).....	2
1.3 概觀 (Overview).....	3
1.4 符號描述 (Notation Description).....	4
Section 2 系統(System).....	6
2.1 系統描述 (System Description)	6
2.1.1 系統架構圖 (System Architecture Diagram)	6
2.3 功能性需求 (Functional Requirements).....	10
2.4 資料需求 (Data Requirements)	12
2.5 非功能性需求 (Non-Functional Requirements).....	16
2.5.1 效能需求 (Performance Requirements).....	16
2.5.2 資安需求 (Security Requirements)	16
2.6 介面需求 (Interface Requirements)	17
2.6.1 使用者介面需求 (User Interfaces Requirements)	17
2.6.2 外部介面需求 (External Interface Requirements)	18
2.6.3 內部介面需求 (Internal Interface Requirements)	19
2.7 其他需求 (Other Requirements)	20

2.7.1 環境需求 (Environmental Requirement).....	20
2.7.2 安裝需求 (Installation Requirement)	20
2.7.3 測試需求 (Test Requirements).....	20
2.8 商業規則與限制 (Business Rules and Integrity Constraints).....	21
Section 3 資料庫概念設計 (Conceptual Design of the Database).....	22
3.1 Entity Relationship ER Model	22
Section 4 邏輯資料庫綱要 (Logic Database Schema)	23
4.1 Schema of the Database.....	23
4.2 Domain of the Database.....	23
4.3 Expectation of the possible DB operations, frequencies and data volumes	41
4.4 SQL Statements Used to Construct the Schema	43
4.5 SQL Statements Used to Insert the data	52
Section 5 功能性依賴(Functional Dependencies and Database Normalization).....	55
5.1 Functional Dependencies.....	55
Section 6 邏輯資料庫綱要 (Logic Database Schema)	56
6.1 系統安裝 (System Installation Description).....	56
6.2 系統使用 (The Use of the System).....	56
6.2.1 買家使用介面	56
6.2.2 商家使用介面	59
6.2.3 管理員使用介面	60
Section 7 Suggestions of the Database Turning.....	62
Section 8 附加查詢和視圖(Additional Queries and Views)	63
8.1 資料查詢語法(Database Queries)	63

Section 9 Conclusion and Future Work	67
9.1 Conclusions (結論).....	67
9.2 Future Work (未來展望)	67
Glossary.....	69
References	70
Appendix	71

Section 1 簡介 (Introduction)

1.1 目的 (Purpose)

藉由本學期修習的資料庫系統課程，我們希望能更加理解資料庫的運作原理與流程設計，並將過往所學的網頁程式設計、網際網路技術應用及各項程式語言基礎知識融會貫通，透過團隊分工合作完成一個以資料庫為核心的系統實作。

因此，我們決定開發一個線上電商系統「大潤發電商平臺」，以支援買家、賣家與管理員三方角色的操作需求。透過此平台，買家能夠進行商品的搜尋、瀏覽與購買；賣家能夠維護商店資訊並管理商品；管理員則能夠掌控與維護整體系統的正常運作。

透過本專案的建置，我們期望能在實際開發過程中加深對資料庫設計、前後端整合以及系統需求分析的理解，並進一步培養團隊協作與系統規劃能力。

此系統的主要目標如下：

- 買家 (Buyer)
 - 註冊與登入帳號，維護個人資料。
 - 搜尋與瀏覽商品，將商品加入購物車並完成下單。
 - 追蹤訂單進度與查看歷史訂單。
- 賣家 (Seller)
 - 維護商店基本資料。
 - 管理商品資料 (上架、下架、單品編輯)。
 - 處理訂單並產生銷售報表。

- 設定專屬折扣與促銷活動。
- 管理員 (**Admin**)
 - 管理系統內使用者與賣家帳號 (審核、停權)。
 - 檢視與維護訂單總表，協助交易管理。
 - 設定全系統層級的折扣策略。

1.2 系統名稱 (**Identification**)

本專案將實作一個線上電商系統——大潤發電商平臺。系統使用者分為三種身分：買家、賣家與管理員。不同身分在系統中擁有不同的功能與權限。買家可以搜尋、瀏覽並購買商品；賣家可以管理商店、商品與訂單；管理員則負責監督買家、賣家以及整體系統的運作。系統以關聯式資料庫(**MariaDB**)為基礎，前端使用 **React.js** 與 **Bootstrap** 建置，後端則採用 **NestJS** 開發。

1.3 概觀 (Overview)

本系統大潤發電商平臺(RT-Mart E-Commerce Platform, RTM)由三個主系統與其下數個子系統所組成的，說明如下：

- 買家管理系統 (**Buyer Management System, BMS**)
 - BPMS : Buyer Profile Management Subsystem (買家檔案管理子系統)
 - BOTS : Buyer Order Tracking Subsystem (訂單追蹤子系統)
 - BPCS : Buyer Product Cart Subsystem (購物車子系統)
 - BPSS : Buyer Product Searching Subsystem (商品搜尋子系統)
- 賣家管理系統 (**Seller Management System, SMS**)
 - SPfMS : Seller Profile Management Subsystem (商店資料子系統)
 - SOMS : Seller Order Management Subsystem (商店訂單與報表子系統)
 - SPdMS : Seller Product Management Subsystem (商品管理子系統)
 - SDMS : Seller Discount Management Subsystem (折扣與促銷子系統)
- 管理員管理系統 (**Admin Management System, AMS**)
 - AUMS : Admin User Management Subsystem (使用者管理子系統)
 - ASMS : Admin Store Management Subsystem (賣家管理子系統)
 - AOMS : Admin Order Management Subsystem (訂單管理子系統)
 - ADMS : Admin Discount Management Subsystem (系統折扣管理子系統)

1.4 符號描述 (Notation Description)

RTM	RT-Mart E-Commerce Platform
RTM-UI-xxx	User Interface requirements of RTM
RTM-D-xxx	Data requirements of RTM
RTM-O-xxx	Other requirements of RTM
RTM-B-xxx	Business Rules and Integrity Constraints of RTM
RTM-EI-xxx	External Interface requirements of RTM
RTM-II-xxx	Internal Interface requirements of RTM

BMS	Buyer Management System
BPMS	Buyer Profile Management Subsystem
BOTS	Buyer Order Tracking Subsystem
BPCS	Buyer Product Cart Subsystem
BPSS	Buyer Product Searching Subsystem
BMS-F-xxx	Functional requirements of BMS
BMS-N-xxx	Non-functional requirements of BMS

SMS	Seller Management System
SPfMS	Seller Profile Management Subsystem
SOMS	Seller Order Management Subsystem
SPdMS	Seller Product Management Subsystem
SDMS	Seller Discount Management Subsystem
SMS-F-xxx	Functional requirements of SMS

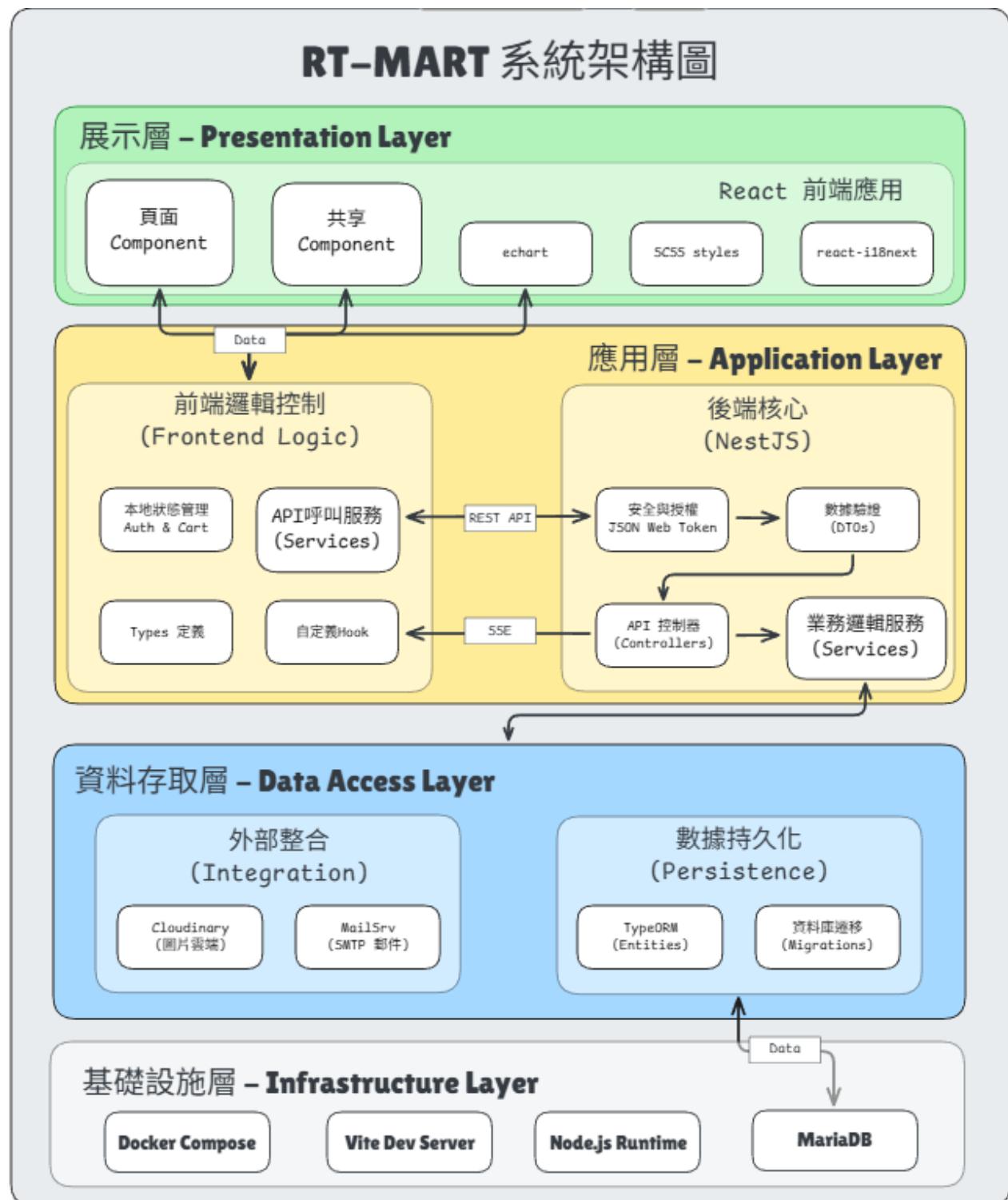
SMS-N-xxx	Non-functional requirements of SMS
-----------	---

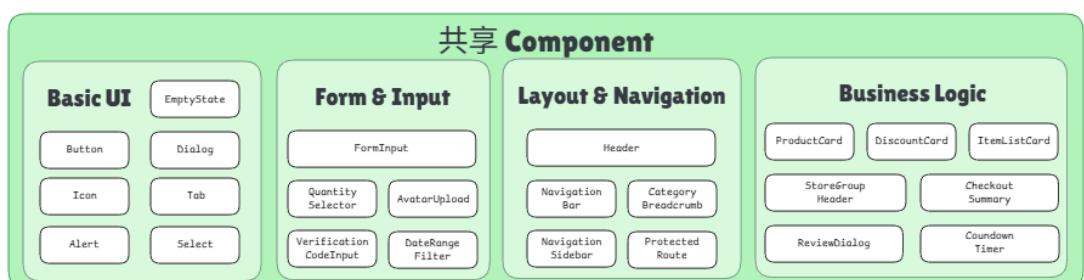
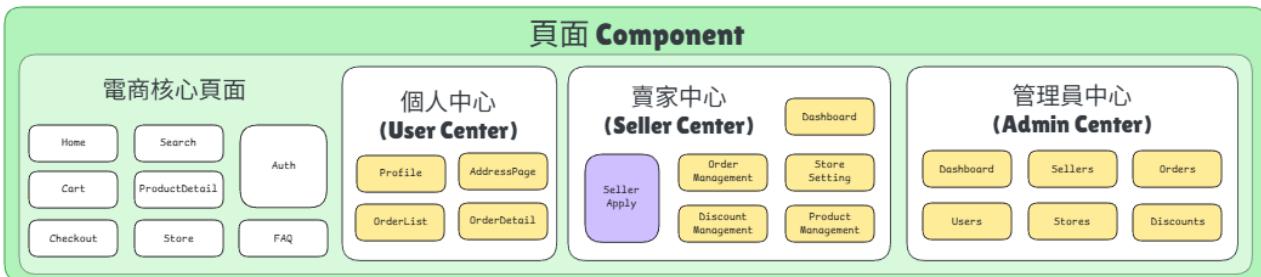
AMS	Admin Management System
AUMS	Admin User Management Subsystem
ASMS	Admin Store Management Subsystem
AOMS	Admin Order Management Subsystem
ADMS	Admin Discount Management Subsystem
AMS-F-xxx	Functional requirements of AMS
AMS-N-xxx	Non-functional requirements of AMS

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)

當使用者進入本網站時，系統將自動導向至「商品搜尋子系統(**Buyer Product Searching Subsystem** · 簡稱 **BPSS**)」。使用者可選擇透過「買家檔案管理子系統 (**Buyer Profile Management Subsystem** · 簡稱 **BPMS**)」登入買家身分，瀏覽網站內容，亦可在未登入的情況下以訪客身分進行瀏覽。

- 買家 (**Buyer**)

在尚未進行身份驗證的情況下，只可在系統主頁面搜尋欄輸入商品名稱並尋找產品。身份驗證與授權後，可於產品資訊頁面加入至購物車，並透過「購物車子系統 (**Buyer Product Cart Subsystem**)」中送出訂單。買家可至「訂單追蹤子系統 (**Buyer Order Tracking Subsystem**)」追蹤並查詢歷史訂單紀錄。

- 賣家 (**Seller**)

在尚未進行身份驗證的情況下，只可在系統主頁面搜尋欄輸入商品名稱並尋找產品。身份驗證與授權後，賣家需至「商店資料子系統 (**Seller Profile Management Subsystem**)」自訂店家相關資訊，如：聯絡資訊、地址與銀行帳戶等。賣家可透過「商品管理子系統 (**Seller Product Management Subsystem**)」依個人喜好上下架產品，也可透過「折扣與促銷子系統 (**Seller Discount Management Subsystem**)」自訂專屬折扣與促銷活動等。賣家可於「商店訂單與報表子系統 (**Seller Order Management Subsystem**)」修改訂單狀態，並產生相關銷售報表。

- 管理員 (**Admin**)

在尚未進行身份驗證的情況下，只可在系統主頁面搜尋欄輸入商品名稱並尋找產品。身份驗證與授權後，可透過「使用者管理子系統 (**Admin User Management Subsystem**)」顯示並審核買家個人資料的真實性，與「賣家管理子系統(**Admin Store Management Subsystem**)」顯示並審核賣家相關資料，如有不法交易、假帳號詐欺或已構成消費者權益損毀，管理員可將買家或賣家用戶實施停權。管理員可透過「訂單管理子系統 (**Admin Order Management Subsystem**)」協助賣家管理交易訂單，並且維護與檢視訂單總表。管理員可透過「系統折扣管理子系統 (**Admin Discount Management Subsystem**)」制定全系統層級的優惠與折扣活動。

2.3 功能性需求 (Functional Requirements)

需求編號	需求描述
BMS-F-001	應允許使用者註冊帳號。
BMS-F-002	應允許使用者登入帳號。
BMS-F-003	應允許使用者搜尋商品。
BMS-F-004	應允許使用者申請成為賣家。
BMS-F-005	應允許買家將商品加至購物車。
BMS-F-006	應允許買家下單購物車。
BMS-F-007	應允許買家檢視其訂單歷史紀錄。
BMS-F-008	應允許買家自訂個人檔案，包括姓名、電子郵件、密碼等資訊。
SMS-F-001	應允許賣家新增、刪除、修改商品及類型。
SMS-F-002	應允許賣家處理買家的訂單請求，包括接受、拒絕訂單。
SMS-F-003	應允許賣家為商品或商品類型設定特殊活動折扣。
SMS-F-004	應允許賣家產生報表或統計資料。
SMS-F-005	應允許賣家自訂其商店檔案，包括聯絡資訊、實體地址與銀行帳號。

AMS-F-001	應允許管理員管理所有買家帳號及其個人檔案。
AMS-F-002	應允許管理員管理所有賣家帳號及其賣家檔案。
AMS-F-003	應允許管理員檢視所有商品訂單。
AMS-F-004	應允許管理員設定季節性折扣。
AMS-F-005	應允許管理員設定運費折扣。

2.4 資料需求 (Data Requirements)

需求編號	需求描述
RTM-D-001	系統應包含使用者的 ID、登入用 ID、名稱、密碼 (hash)、電子信箱、角色 (買家、賣家、管理員)、電話、被創建時間、被更新時間、被刪除時間。
RTM-D-002	系統應包含賣家的 ID、賣家的使用者 ID、銀行帳戶、管理員驗證、驗證時間、驗證管理員的使用者 ID。
RTM-D-003	系統應包含收件地址的 ID、收件地址的使用者 ID、收件人姓名、收件人電話、收件城市、收件區域、郵寄區號、收件地址描述、是否是使用者預設地址。
RTM-D-004	系統應包含 token 的 ID、動硬的使用者 ID、token (hash)、到時時間、是否被撤銷、創建時間。
RTM-D-005	系統應包含商店 ID、對應的賣家 ID、商店名稱、商店敘述、實體地址、電子郵件、電話、平均評價數、總評價數、被刪除時間、被創建時間、被更新時間。
RTM-D-006	系統應紀錄商品類型的 ID、類型代碼、類型名稱、父類型的商品類型 ID、是否正在啟用。

RTM-D-007	系統應包含商品的 ID、對應的商店 ID、類型 ID、名稱、單價、瀏覽次數、評價平均數、評價總數、被刪除時間、被創建時間、被更新時間。
RTM-D-008	系統應包含商品圖片的 ID、對應的商品 ID、圖片網址、圖片順序。
RTM-D-009	系統應包含庫存的 ID、對應的商品 ID、庫存數量、預留數量、被更新時間。
RTM-D-010	系統應包含購物車的 ID、對應的使用者 ID、被創建時間、被更新時間。
RTM-D-011	系統應包含購物車項目的 ID、對應的購物車 ID、對應的商品 ID、數量、被創建時間、被更新時間。
RTM-D-012	系統應包含購物車紀錄的 ID、對應的使用者 ID、購物車的快照、查詢用商品數、轉換後的訂單 ID、被創建時間。
RTM-D-013	系統應包含訂單的 ID、訂單編號、對應的使用者 ID、對應的商店 ID、訂單狀態、費用、運費、折價金額、總金額、付款方式、金流商訂單號、郵次性鍵、寄送地址快照、筆記欄、被創建時間、被更新時間、付費時間、商品寄送時間、商品抵達時間、完成時間、被取消時間。

RTM-D-014	系統應包含訂單項目的 ID、對應的訂單 ID、對應的商品 ID、商品快照、數量、原價、折扣後價格、實際單價、總額。
RTM-D-015	系統應包含折扣的 ID、折扣的編號、名稱、種類(運費、季節、特殊活動)、描述、最低消、啟用時間、結束時間、是否啟用、使用限制數、已使用次數、創建人類型(系統、賣家)、創建人對應的使用者 ID、被創建時間。
RTM-D-016	系統應包含季節折扣的 ID、對應的折扣 ID、折扣比率、折扣上限。
RTM-D-017	系統應包含運費折扣的 ID、對應的折扣 ID、折扣金額。
RTM-D-018	系統應包含特殊活動折扣的 ID、對應的折扣 ID、對應的商店 ID、對應的折扣類型 ID、折扣比率、折扣上限。
RTM-D-019	系統應包含訂單折扣的 ID、對應的訂單 ID、對應的折扣 ID、折扣金額、使用時間。
RTM-D-020	系統應包含審計日誌的 ID、事件識別號、事件發生時間、更動的 table 名稱、紀錄識別碼、執行的操作類型、對應執行操作的使用者 ID、請求識別號、來源 IP 位址、使用者代理字串、執行操作的服務名稱、變更前的資料、變更後的資料、具體變更內容、資料校驗和。

RTM-D-021

使用者 ID、賣家 ID、收件地址 ID、tokenID、商店 ID、商品類型 ID、
商品 ID、商品圖片的 ID、庫存 ID、購物車 ID、購物車項目 ID、購
物車紀錄 ID、訂單 ID、訂單項目 ID、折扣 ID、季節折扣 ID、運費
折扣 ID、特殊活動折扣 ID、訂單折扣 ID 皆須唯一

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
BMS-N-001	網頁回應時間應少於 5 秒。
BMS-N-002	當表單提交過程中發生錯誤時，系統應顯示錯誤訊息，並將使用者重新導向至前一頁。
BMS-N-003	當系統內部發生錯誤時，系統必須顯示錯誤訊息，並將使用者重新導向至錯誤頁面。

2.5.2 資安需求 (Security Requirements)

需求編號	需求描述
BMS-N-004	未經授權之使用者應被限制，僅能存取 BPSS 及 BPMS(以進行登入)。
BMS-N-005	買家應被限制，僅能存取 BMS 之相關內容。
BMS-N-006	使用者的 token 會在一定期間內生效，並定期清理時效外資料。
SMS-N-001	賣家應被限制，僅能存取 BMS 、 SMS 之相關內容。

SMS-N-002	使用者僅在已註冊為買家之情況下，方可註冊為賣家。
-----------	--------------------------

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
RTM-UI-001	支援現代主流瀏覽器，包括 Chrome、Firefox、Safari 及 Edge 。
RTM-UI-002	提供響應式網頁介面 (Responsive Web Interface)，能夠適應桌面、平板及行動裝置螢幕尺寸。
RTM-UI-003	使用者介面應使用 React.js 搭配 HTML、JavaScript 及 CSS 開發，設計風格參考蝦皮 (Shopee) 。
RTM-UI-004	使用對話框 (Modal Dialogs) 顯示錯誤訊息與成功通知，並在 3-5 秒後自動消失。
RTM-UI-005	所有表單應提供即時驗證回饋 (Real-time Validation)，並在對應的輸入欄位旁顯示清楚的錯誤訊息。
RTM-UI-006	導覽列 (Navigation Bar) 應在所有頁面保持可見，並根據使用者角色 (買家、賣家、管理員) 顯示不同的選單項目。
RTM-UI-007	商品詳情頁面的商品圖片應支援放大檢視功能 (Zoom-in

	Functionality)。
RTM-UI-008	購物車圖示應標示目前購物車內的商品數量，並即時更新。
RTM-UI-009	資料載入時使用載入動畫 (Spinner) 或骨架屏 (Skeleton Screen) 指示載入狀態。
RTM-UI-010	支援繁體中文與英文語言介面。

2.6.2 外部介面需求 (External Interface Requirements)

需求編號	需求描述
RTM-EI-001	支援信用卡、ATM 轉帳繳費。
RTM-EI-002	使用 Nodemailer 作為電子郵件發送工具。
RTM-EI-003	於以下事件發送電子郵件通知：帳號註冊確認、密碼重設、訂單確認、訂單狀態更新、訂單完成。
RTM-EI-004	所有外部 API 通訊使用 HTTPS 協定與 SSL/TLS 加密。
RTM-EI-005	妥善處理外部 API 失敗，顯示錯誤訊息並實作重試機制。

2.6.3 內部介面需求 (Internal Interface Requirements)

需求編號	需求描述
RTM-II-001	採用 RESTful API 架構進行前端 (React.js) 與後端 (NestJS) 之間的通訊。
RTM-II-002	API 端點遵循 RESTful 命名慣例 (如 /api/products 、 /api/orders/{orderId})。
RTM-II-003	API 請求與回應應使用 JSON 格式進行資料交換。
RTM-II-004	API 實作標準 HTTP 狀態碼：200 (成功) 、 201 (已建立) 、 400 (錯誤請求) 、 401 (未授權) 、 403 (禁止存取) 、 404 (找不到資源) 、 500 (伺服器內部錯誤) 。
RTM-II-005	使用 JWT 進行身份驗證，透過 HTTP Authorization 標頭傳遞。
RTM-II-006	實作速率限制，每位使用者每分鐘最多 100 次請求。
RTM-II-007	列表 API 回應包含分頁資訊 (頁碼、每頁筆數、總筆數)。
RTM-II-008	後端驗證所有請求，回傳包含錯誤代碼的詳細錯誤訊息。

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
RTM-O-001	處理器須為 64 位元雙核心，時脈速度須超過 2.0GHz。
RTM-O-002	儲存空間須至少具備 200GB 之硬碟容量。
RTM-O-003	記憶體容量須為 2GB 或以上。
RTM-O-004	網路速度須為 10 mbps 或以上。

2.7.2 安裝需求 (Installation Requirement)

需求編號	需求描述
RTM-O-005	Visual Studio Code
RTM-O-006	Docker Desktop (to make the environment consistent to all devs)
RTM-O-007	Node.js & npm

2.7.3 測試需求 (Test Requirements)

需求編號	需求描述
RTM-O-008	系統須以實際資料進行全面測試，且所有模組皆須通過單元測試。
RTM-O-009	系統應通過弱點測試。

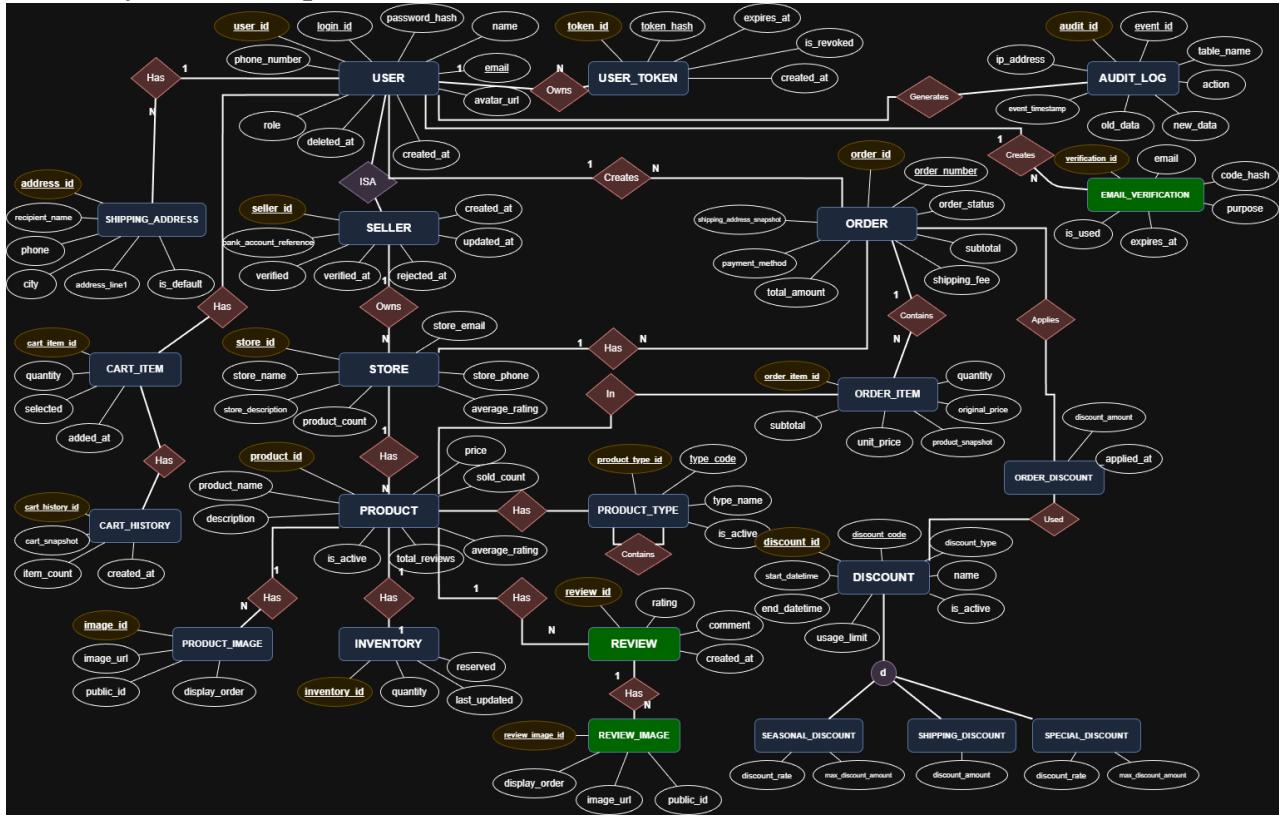
RTM-O-010	系統應通過滲透測試。
-----------	------------

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

需求編號	需求描述
RTM-B-001	針對同一產品或產品類別之同類型折扣不得重疊。
RTM-B-002	每筆產品購買數量須為大於零之整數。
RTM-B-003	單一產品之購買數量須小於或等於該產品目前庫存數量。

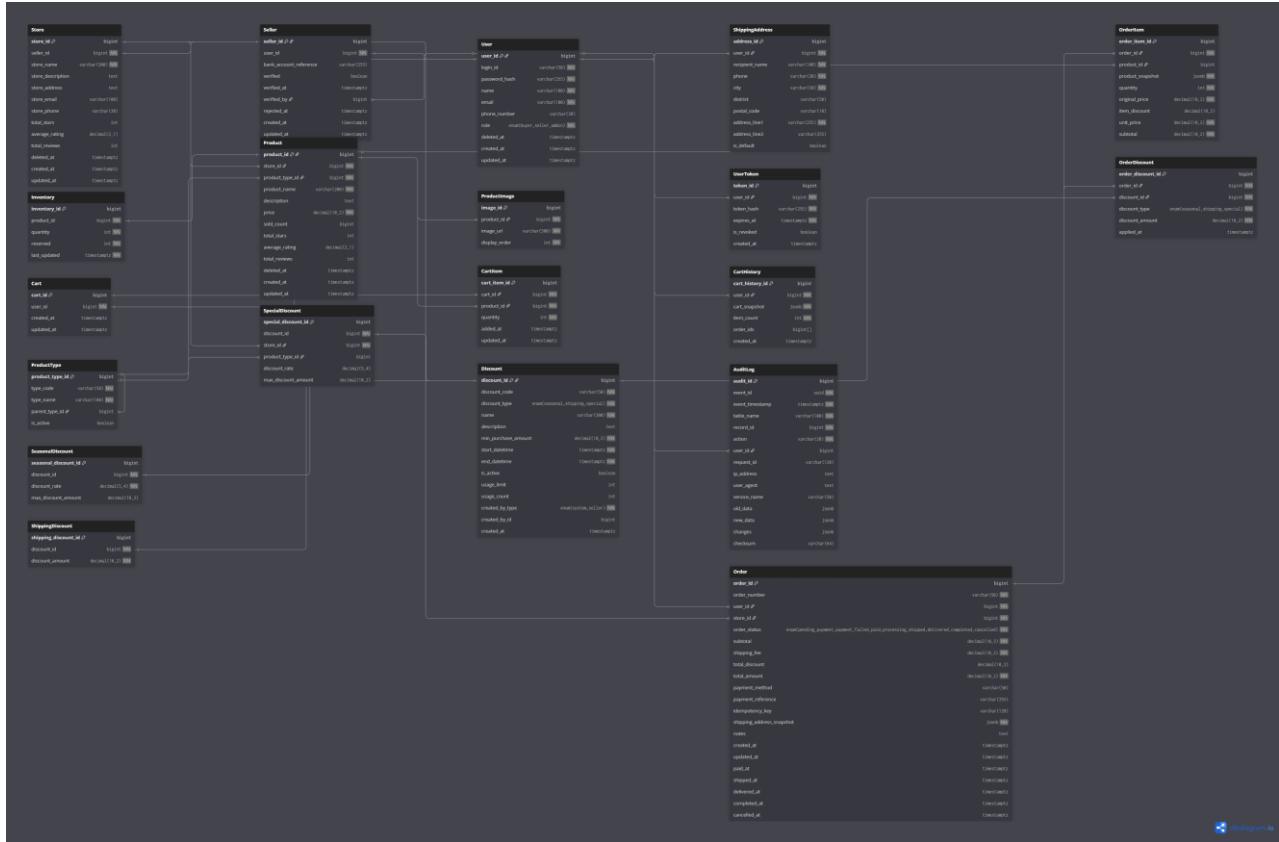
Section 3 資料庫概念設計 (Conceptual Design of the Database)

3.1 Entity Relationship ER Model



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

4.1 Schema of the Database



4.2 Domain of the Database

User				
Description: 存放使用者資料				
Attribute	Type	Key	Nullable	Description
user_id	Integer	Primary	Not null	使用者編碼
login_id	Varchar	Unique	Not null	使用者帳號
password_hash	Varchar		Not null	使用者密碼

name	Nvarchar		Not null	使用者名稱
email	VARCHAR	Unique	Not null	使用者信箱
phone_number	VARCHAR			使用者行動電話
role	Enum		Not null	使用者角色 (buyer/seller/admin)
deleted_at	Timestamptz			使用者被刪除時間
created_at	Timestamptz		Not null	使用者被創建時間
updated_at	Timestamptz		Not null	使用者被更新時間
avatar_url	VARCHAR			使用者頭像

Seller				
Description: 存放賣家資料				
Attribute	Type	Key	Nullable	Description
seller_id	Integer	Primary	Not null	賣家編碼
user_id	Integer	Foreign Primary	Not null	使用者編碼
bank_account _reference	VARCHAR		Not null	賣家銀行帳戶

verified	Boolean		Not null	已經過管理員驗證
verified_at	Timestamptz			經過管理員驗證時間
verified_by	Integer			驗證管理員的使用者編號

ShippingAddress				
Description: 存放收件地址資料				
Attribute	Type	Key	Nullable	Description
address_id	Integer	Primary	Not null	收件地址編碼
user_id	Integer	Foreign	Not null	使用者編碼
recipient_name	Varchar		Not null	收件人姓名
phone	Varchar		Not null	收件人連絡電話
city	Varchar		Not null	收件地址的城市
district	Varchar			收件地址的區域
postal_code	Varchar			收件地址的郵寄區號
address_line1	Nvarchar		Not null	收件地址描述 (block, alley)

address_line2	Nvarchar			收件地址描述 (number, floor)
is_default	Boolean		Not null	是否為使用者的預設收件地址

UserToken				
Description: 使用者登入資訊				
Attribute	Type	Key	Nullable	Description
token_id	Integer	Primary	Not null	token 編碼
user_id	Integer	Foreign	Not null	使用者編碼
token_hash	Varchar	Unique	Not null	token 欄位
expires_at	Timestamptz		Not null	到期時間
is_revoked	Boolean		Not null	是否被撤銷
created_at	Timestamptz		Not null	創建時間

Store				
Description: 商店資訊				
Attribute	Type	Key	Nullable	Description

store_id	Integer	Primary	Not null	商店編碼
seller_id	Integer	Foreign Primary	Not null	賣家編碼
store_name	Nvarchar		Not null	商店名稱
store_description	Text			商店敘述
store_address	Text			商店實體地址
store_email	VARCHAR			商店電子郵件
store_phone	VARCHAR			商店電話
average_rating	Decimal		Not null	平均評價數
total_ratings	Integer		Not null	總評價數
deleted_at	Timestamptz			商店被刪除時間
created_at	Timestamptz		Not null	商店被創建時間
updated_at	Timestamptz		Not null	商店被更新時間

ProductType

Description: 商品類型資訊				
Attribute	Type	Key	Nullable	Description
product_type_id	Integer	Primary	Not null	商品類型編碼
type_code	Varchar	Unique	Not null	商品類型代碼
type_name	Nvarchar		Not null	商品類型名稱
parent_type_id	Integer			父商品類型的商品類型編碼
is_active	Boolean		Not null	商品類型是否正在啟用

Product				
Description: 商品資料				
Attribute	Type	Key	Nullable	Description
product_id	Integer	Primary	Not null	商品編碼
store_id	Integer	Foreign	Not null	商店編碼
product_type_id	Integer	Foreign	Not null	商品類型編碼
product_name	Nvarchar		Not null	商品名稱
description	Text		Not null	商品描述

price	Decimal		Not null	商品單價
view_count	Integer		Not null	瀏覽次數
average_rating	Decimal		Not null	評價平均數
total_reviews	Integer		Not null	總評價數
deleted_at	Timestamptz			商品被刪除時間
created_at	Timestamptz		Not null	商品被創建時間
updated_at	Timestamptz		Not null	商品被更新時間

ProductImage				
Description: 商品圖片資料				
Attribute	Type	Key	Nullable	Description
image_id	Integer	Primary	Not null	商品圖片編碼
product_id	Integer	Foreign	Not null	商品編碼
image_url	Varchar		Not null	商品圖片網址
display_order	Integer		Not null	商品圖片順序

Inventory				
Description: 商品庫存資料				
Attribute	Type	Key	Nullable	Description
Inventory_id	Integer	Primary	Not null	商品庫存編碼
product_id	Integer	Foreign Primary	Not null	商品編碼
quantity	Integer		Not null	商品庫存數量
reserved	Integer		Not null	預留未完成交易的商品數
last_updated	Timestamptz		Not null	商品庫存被更新時間

Cart				
Description: 購物車資料				
Attribute	Type	Key	Nullable	Description
cart_id	Integer	Primary	Not null	購物車編碼
user_id	Integer	Foreign Primary	Not null	使用者編碼
created_at	Timestamptz		Not null	購物車被創建時間

updated_at	Timestamptz		Not null	購物車被更新時間
------------	-------------	--	----------	----------

CartHistory				
Attribute	Type	Key	Nullable	Description
cart_history_id	Integer	Primary	Not null	購物車歷史紀錄編碼
user_id	Integer	Foreign	Not null	使用者編碼
cart_snapshot	Jsonb		Not null	購物車快照紀錄
item_count	Integer		Not null	購物的商品總數 (查詢用)
order_ids	Integer	Foreign		購物車歷史紀錄轉換的訂單編碼
created_at	Timestamptz		Not null	購物車歷史紀錄被創建時間

Order				
Description: 訂單資料				
Attribute	Type	Key	Nullable	Description
order_id	Integer	Primary	Not null	訂單編碼

order_number	Varchar	Unique	Not null	訂單編號
user_id	Integer	Foreign	Not null	使用者編碼
store_id	Integer	Foreign	Not null	商店編碼
order_status	Enum		Not null	訂單狀態 (pending_payment, payment_failed, paid, processing, shipped, delivered, completed, canceled)
subtotal	Decimal		Not null	訂單費用
shipping_fee	Decimal		Not null	訂單運費
total_discount	Decimal		Not null	訂單折價金額
total_amount	Decimal		Not null	訂單總金額
payment_method	Varchar		Not null	訂單付款方式
payment_reference	Varchar			金流商訂單號
idempotency_key	Varchar	Unique		訂單幂次性鍵
shipping_address_snapshot	Jsonb		Not null	訂單寄送地址快照
notes	Text			訂單筆記欄

created_at	Timestamptz		Not null	訂單被創建時間
updates_at	Timestamptz		Not null	訂單被更新時間
paid_at	Timestamptz			訂單付費時間
shipped_at	Timestamptz			訂單商品寄送時間
delivered_at	Timestamptz			訂單商品抵達時間
completed_at	Timestamptz			訂單交易完成時間
cancelled_at	Timestamptz			訂單被取消時間

OrderItem				
Description: 訂單項目資料				
Attribute	Type	Key	Nullable	Description
order_item_id	Integer	Primary	Not null	訂單項目編碼
order_id	Integer	Foreign	Not null	訂單編碼
product_id	Integer	Foreign	Not null	商品編碼
product_snapshot	Jsonb		Not null	商品快照

quantity	Integer		Not null	訂單項目的商品數量
original_price	Decimal		Not null	訂單項目的原價
item_discount	Decimal		Not null	訂單項目折扣後價格
unit_price	Decimal		Not null	訂單項目計算後實際單價
subtotal	Decimal		Not null	訂單項目總額

Discount				
Description: 折扣資料				
Attribute	Type	Key	Nullable	Description
discount_id	Integer	Primary	Not null	折扣編碼
discount_code	Varchar	Unique	Not null	折扣編號
discount_type	Enum		Not null	折扣種類 (seasonal, shipping, special)
discount_name	Nvarchar		Not null	折扣名稱
description	Text			折扣描述
min_purchase_amount	Decimal		Not null	折扣最低消費

start_datetime	Timestamptz		Not null	折扣啟用時間
end_datetime	Timestamptz		Not null	折扣結束時間
is_active	Boolean		Not null	折扣是否正在啟用
usage_limit	Integer			折扣使用限制數
usage_count	Integer		Not null	折扣已使用次數
created_by_type	Enum		Not null	折扣創建人類型 (system, seller)
created_by_id	Integer			折扣創建人的使用者編號
created_at	Timestamptz		Not null	折扣被創造時間

SeasonalDiscount				
Description: 季節折扣資料				
Attribute	Type	Key	Nullable	Description
seasonal_discount_id	Integer	Primary	Not null	季節折扣編碼
discount_id	Integer	Foreign	Not null	折扣編碼
discount_rate	Decimal		Not null	折扣比率

max_discount_amount	Decimal		Not null	折扣最高金額上限
---------------------	---------	--	----------	----------

ShippingDiscount				
Description: 運費折扣資料				
Attribute	Type	Key	Nullable	Description
shipping_discount_id	Integer	Primary	Not null	運費折扣編碼
discount_id	Integer	Foreign	Not null	折扣編碼
discount_amount	Decimal		Not null	折扣金額

SpecialDiscount				
Description: 特殊活動折扣資料				
Attribute	Type	Key	Nullable	Description
special_discount_id	Integer	Primary	Not null	特殊活動折扣編碼
discount_id	Integer	Foreign	Not null	折扣編碼
store_id	Integer	Foreign	Not null	商店編碼
product_type_id	Integer	Foreign	Not null	折扣的商品類型編碼
discount_rate	Decimal		Not null	折扣比率

max_discount_amount	Decimal		Not null	折扣最高金額上限
---------------------	---------	--	----------	----------

OrderDiscount				
Description: 訂單折扣資料				
Attribute	Type	Key	Nullable	Description
order_discount_id	Integer	Primary	Not null	訂單折扣編碼
order_id	Integer	Foreign	Not null	訂單編碼
discount_id	Integer	Foreign	Not null	折扣編碼
discount_type	Enum		Not null	訂單折扣類型 (seasonal, shipping, special)
discount_amount	Decimal		Not null	訂單折扣的金額
applied_at	Timestamptz		Not null	訂單應用此折扣的時間

AuditLog				
Description: 前臺審計日誌資料				
Attribute	Type	Key	Nullable	Description
audit_id	Integer	Primary	Not null	審計日誌編碼

event_id	UUID		Not null	事件識別號
event_timestamp	Timestamptz		Not null	事件發生時間
table_name	Varchar		Not null	更動的 table 名稱
record_id	Integer		Not null	紀錄識別碼
action	Varchar		Not null	執行的操作類型
user_id	Integer			執行操作的使用者編號
request_id	Varchar			請求識別號
ip_address	Inet			來源 IP 位址
user_agent	Text			使用者代理字串
service_name	Varchar			執行操作的服務名稱
old_data	Jsonb			變更前的資料
new_data	Jsonb			變更後的資料
changes	Jsonb			具體變更內容
checksum	Varchar			資料校驗和(用於驗證是否進過後續後臺資料更動)

Review				
Description: 商品評論				
Attribute	Type	Key	Nullable	Description
review_id	Integer	Primary	Not null	評論編碼
product_id	Integer		Not null	商品編碼
user_id	Integer		Not null	使用者編碼
rating	Integer		Not null	評論分數
comment	Varchar		Not null	留言
created_at	Timestamptz			評論撰寫時間
updated_at	Timestamptz			評論更新時間
ReviewImage				
Description: 評論圖片				
Attribute	Type	Key	Nullable	Description
review_image_id	Integer	Primary	Not null	評論圖片編號
review_id	Integer		Not null	評論編號

image_url	Varchar			評論圖片連結
public_id	Integer	Unique		Cloudinary image id
display_order	Integer			顯示順序
EmailVerification				
Description: 電子郵件驗證				
Attribute	Type	Key	Nullable	Description
verification_id	Integer	Primary	Not null	驗證編號
email	Varchar		Not null	電子郵件
code_hash	Varchar		Not null	驗證碼 hash
purpose	Enum		Not null	驗證原因(註冊帳號、忘記密碼)
metadata	longtext			額外資訊 (用戶名、臨時密碼等)
expires_at	Timestamptz		Not null	到期時間
is_used	Boolean		Not null	是否已被使用
created_at	Timestamptz		Not null	創建時間

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

Table	可能操作	頻率 (per day)	資料量	系統負擔
User	使用者登入	200	400	80000
User	使用者註冊	10	400	4000
User	使用者資料更新	5	400	2000
User	帳號停權	2	400	800
UserToken	讀取 Token	200	400	80000
UserToken	新增 Token	10	400	4000
UserToken	Token 撤銷	2	400	800
UserToken	過期清理	1	400	400
ShippingAddress	讀取 address	80	600	48000
ShippingAddress	新增/更新 address	5	600	3000
AuditLog	寫入紀錄操作	600	10000	6000000
AuditLog	管理員查詢	50	10000	500000
Seller	申請賣家	5	30	150
Seller	管理員驗證	5	30	150
Seller	讀取賣家資料	15	30	450
Store	瀏覽商店	1500	30	45000
Store	建立/更新商店資訊	20	30	600
ProductType	商品分類篩選	500	50	25000
ProductType	管理員維護	10	50	500
Product	商品搜尋	1000	500	500000
Product	商品瀏覽	3000	500	1500000
Product	瀏覽次數更新	3000	500	1500000
Product	上下架管理、更新	100	500	50000
ProductImage	讀取商品圖片	9000	1500	13500000
ProductImage	上傳/刪除圖片	50	1500	75000
Inventory	庫存查詢	3000	500	1500000
Inventory	庫存預留	60	500	30000
Inventory	庫存回補	5	500	2500
Inventory	商家手動修改庫存	30	500	15000
CartItem	加入商品	300	800	240000
CartItem	更新數量	150	800	120000
CartItem	移除商品	100	800	80000

CartItem	列出項目	50	800	40000
CartHistory	結帳時建立快照	25	500	12500
CartHistory	付款失敗時讀取	10	500	5000
Order	建立訂單	25	1000	25000
Order	更新狀態	30	1000	30000
Order	買家查詢訂單列表	100	1000	100000
Order	買家查詢訂單詳情	50	1000	50000
Order	賣家查詢商店訂單	60	1000	60000
Order	管理員查詢所有訂單	10	1000	10000
OrderItem	建立訂單項目	60	2500	150000
OrderItem	讀取訂單明細	250	2500	625000
Discount	使用折扣	50	100	5000
Discount	列出可用折扣	50	100	5000
Discount	建立折扣	5	100	500
Discount	編輯折扣	10	100	1000
OrderDiscount	套用折扣	15	300	4500
OrderDiscount	讀取訂單折扣	65	300	19500
SeasonalDiscount	讀取季節折扣	30	20	600
SeasonalDiscount	管理員新增/編輯	2	20	40
SpecialDiscount	讀取特殊折扣	80	50	4000
SpecialDiscount	賣家新增/編輯	10	50	500
ShippingDiscount	讀取運費折扣	25	10	250
ShippingDiscount	管理員新增/編輯	2	10	20
Review	買家讀取評論	300	1500	450000
Review	買家新增評論	10	1500	15000
Review	管理員刪除評論	1	1500	1500
ReviewImage	買家讀取圖片	300	1500	450000
ReviewImage	買家新增圖片	10	1500	15000
EmailVerification	使用者註冊	10	600	6000
EmailVerification	使用者忘記密碼	1	600	600

4.4 SQL Statements Used to Construct the Schema

```
CREATE TABLE User (
    user_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    login_id VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone_number VARCHAR(20),
    role ENUM('buyer', 'seller', 'admin') NOT NULL,
    deleted_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
    UPDATE CURRENT_TIMESTAMP
)
```

```
CREATE TABLE UserToken (
    token_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL,
    token_hash VARCHAR(255) NOT NULL UNIQUE,
    expires_at TIMESTAMP NOT NULL,
    is_revoked BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT FK_usertoken_user FOREIGN KEY (user_id)
    REFERENCES User(user_id) ON DELETE CASCADE
)
```

```
CREATE TABLE ShippingAddress (
    address_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL,
    recipient_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    city VARCHAR(50) NOT NULL,
    district VARCHAR(50),
    postal_code VARCHAR(10),
    address_line1 VARCHAR(255) NOT NULL,
    address_line2 VARCHAR(255),
    is_default BOOLEAN DEFAULT FALSE,
    CONSTRAINT FK_shippingaddress_user FOREIGN KEY (user_id)
    REFERENCES User(user_id) ON DELETE CASCADE
)
```

```
CREATE TABLE AuditLog (
    audit_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    event_id CHAR(36) NOT NULL UNIQUE,
    event_timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    table_name VARCHAR(100) NOT NULL,
    record_id BIGINT NOT NULL,
    action VARCHAR(20) NOT NULL,
    user_id BIGINT,
    request_id VARCHAR(128),
    ip_address VARCHAR(45), -- MariaDB/MySQL 沒有原生 inet 型別，通常用 VARCHAR(45) 儲存 IPv4/IPv6
    user_agent TEXT,
    service_name VARCHAR(50),
    old_data JSON,
    new_data JSON,
    changes JSON,
    checksum VARCHAR(64),
    CONSTRAINT FK_auditlog_user FOREIGN KEY (user_id)
        REFERENCES User(user_id) ON DELETE SET NULL
)
```

```
CREATE TABLE Seller (
    seller_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL UNIQUE,
    bank_account_reference VARCHAR(255),
    verified BOOLEAN DEFAULT FALSE,
    verified_at TIMESTAMP NULL,
    verified_by BIGINT NULL,
    CONSTRAINT FK_seller_user FOREIGN KEY (user_id)
        REFERENCES User(user_id) ON DELETE CASCADE,
    CONSTRAINT FK_seller_verified_by FOREIGN KEY (
        verified_by) REFERENCES User(user_id) ON DELETE SET NULL
)
```

```
CREATE TABLE Store (
    store_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    seller_id BIGINT NOT NULL,
    store_name VARCHAR(200) NOT NULL,
    store_description TEXT,
    store_address TEXT,
    store_email VARCHAR(100),
    store_phone VARCHAR(20),
    average_rating DECIMAL(2,1) DEFAULT 0.0,
    total_ratings INT DEFAULT 0,
    deleted_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
    UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT FK_store_seller FOREIGN KEY (seller_id)
    REFERENCES Seller(seller_id) ON DELETE RESTRICT
)
```

```
CREATE TABLE ProductType (
    product_type_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    type_code VARCHAR(50) NOT NULL UNIQUE,
    type_name VARCHAR(100) NOT NULL,
    parent_type_id BIGINT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    CONSTRAINT FK_producttype_parent FOREIGN KEY (
        parent_type_id) REFERENCES ProductType(product_type_id)
    ON DELETE CASCADE
)
```

```
CREATE TABLE Product (
    product_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    store_id BIGINT NOT NULL,
    product_type_id BIGINT NOT NULL,
    product_name VARCHAR(200) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    view_count BIGINT DEFAULT 0,
    average_rating DECIMAL(2,1) DEFAULT 0.0,
    total_reviews INT DEFAULT 0,
    deleted_at TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT FK_product_store FOREIGN KEY (store_id)
        REFERENCES Store(store_id) ON DELETE RESTRICT,
    CONSTRAINT FK_product_producttype FOREIGN KEY (
        product_type_id) REFERENCES ProductType(product_type_id) ON
        DELETE RESTRICT
)
```

```
CREATE TABLE ProductImage (
    image_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    product_id BIGINT NOT NULL,
    image_url VARCHAR(500) NOT NULL,
    display_order INT NOT NULL DEFAULT 1,
    CONSTRAINT FK_productimage_product FOREIGN KEY (product_id)
        REFERENCES Product(product_id) ON DELETE CASCADE
)
```

```
CREATE TABLE Inventory (
    inventory_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    product_id BIGINT NOT NULL UNIQUE,
    quantity INT NOT NULL DEFAULT 0,
    reserved INT NOT NULL DEFAULT 0,
    last_updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT FK_inventory_product FOREIGN KEY (product_id)
        REFERENCES Product(product_id) ON DELETE CASCADE
)
```

```
CREATE TABLE CartHistory (
    cart_history_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL,
    cart_snapshot JSON NOT NULL,
    item_count INT NOT NULL,
    order_ids JSON,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT FK_carthistory_user FOREIGN KEY (user_id)
        REFERENCES User(user_id) ON DELETE CASCADE
)
```

```
CREATE TABLE CartItem (
    cart_item_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    cart_id BIGINT NOT NULL,
    product_id BIGINT NOT NULL,
    quantity INT NOT NULL,
    added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT FK_cartitem_cart FOREIGN KEY (cart_id)
        REFERENCES Cart(cart_id) ON DELETE CASCADE,
    CONSTRAINT FK_cartitem_product FOREIGN KEY (product_id)
        REFERENCES Product(product_id) ON DELETE RESTRICT
)
```

```

CREATE TABLE Order (
    order_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    order_number VARCHAR(50) NOT NULL UNIQUE,
    user_id BIGINT NOT NULL,
    store_id BIGINT NOT NULL,
    order_status ENUM(
        'pending_payment',
        'payment_failed',
        'paid',
        'processing',
        'shipped',
        'delivered',
        'completed',
        'cancelled'
    ) NOT NULL DEFAULT 'pending_payment',
    subtotal DECIMAL(10,2) NOT NULL,
    shipping_fee DECIMAL(10,2) NOT NULL DEFAULT 60,
    total_discount DECIMAL(10,2) DEFAULT 0,
    total_amount DECIMAL(10,2) NOT NULL,
    payment_method VARCHAR(50),
    payment_reference VARCHAR(255),
    idempotency_key VARCHAR(128) UNIQUE,
    shipping_address_snapshot JSON NOT NULL,
    notes TEXT,

```

```

notes TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
paid_at TIMESTAMP NULL,
shipped_at TIMESTAMP NULL,
delivered_at TIMESTAMP NULL,
completed_at TIMESTAMP NULL,
cancelled_at TIMESTAMP NULL,
CONSTRAINT FK_order_user FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE RESTRICT,
CONSTRAINT FK_order_store FOREIGN KEY (store_id) REFERENCES Store(store_id) ON DELETE RESTRICT
)

```

```
CREATE TABLE OrderItem (
    order_item_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    order_id BIGINT NOT NULL,
    product_id BIGINT NULL,
    product_snapshot JSON NOT NULL,
    quantity INT NOT NULL,
    original_price DECIMAL(10,2) NOT NULL,
    item_discount DECIMAL(10,2) DEFAULT 0,
    unit_price DECIMAL(10,2) NOT NULL,
    subtotal DECIMAL(10,2) NOT NULL,
    CONSTRAINT FK_orderitem_order FOREIGN KEY (order_id)
        REFERENCES \`Order\`(order_id) ON DELETE CASCADE,
    CONSTRAINT FK_orderitem_product FOREIGN KEY (product_id)
        REFERENCES Product(product_id) ON DELETE SET NULL
)
```

```
CREATE TABLE Discount (
    discount_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    discount_code VARCHAR(50) NOT NULL UNIQUE,
    discount_type ENUM('seasonal', 'shipping', 'special') NOT NULL,
    name VARCHAR(200) NOT NULL,
    description TEXT,
    min_purchase_amount DECIMAL(10,2) NOT NULL DEFAULT 0,
    start_datetime TIMESTAMP NOT NULL,
    end_datetime TIMESTAMP NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    usage_limit INT,
    usage_count INT DEFAULT 0,
    created_by_type ENUM('system', 'seller') NOT NULL,
    created_by_id BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
```

```

CREATE TABLE OrderDiscount (
    order_discount_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    order_id BIGINT NOT NULL,
    discount_id BIGINT NOT NULL,
    discount_type ENUM('seasonal', 'shipping', 'special') NOT NULL,
    discount_amount DECIMAL(10,2) NOT NULL,
    applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT FK_orderdiscount_order FOREIGN KEY (order_id)
        REFERENCES \`Order\`(order_id) ON DELETE CASCADE,
    CONSTRAINT FK_orderdiscount_discount FOREIGN KEY (
        discount_id) REFERENCES Discount(discount_id) ON DELETE RESTRICT
)

```

```

CREATE TABLE SeasonalDiscount (
    seasonal_discount_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    discount_id BIGINT NOT NULL UNIQUE,
    discount_rate DECIMAL(5,4) NOT NULL,
    max_discount_amount DECIMAL(10,2),
    CONSTRAINT FK_seasonaldiscount_discount FOREIGN KEY (
        discount_id) REFERENCES Discount(discount_id) ON DELETE CASCADE
)

```

```

CREATE TABLE SpecialDiscount (
    special_discount_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    discount_id BIGINT NOT NULL UNIQUE,
    store_id BIGINT NOT NULL,
    product_type_id BIGINT,
    discount_rate DECIMAL(5,4),
    max_discount_amount DECIMAL(10,2),
    CONSTRAINT FK_specialdiscount_discount FOREIGN KEY (
        discount_id) REFERENCES Discount(discount_id) ON DELETE CASCADE,
    CONSTRAINT FK_specialdiscount_store FOREIGN KEY (store_id)
        REFERENCES Store(store_id) ON DELETE CASCADE,
    CONSTRAINT FK_specialdiscount_producttype FOREIGN KEY (
        product_type_id) REFERENCES ProductType(product_type_id) ON DELETE RESTRICT
)

```

```
CREATE TABLE ShippingDiscount (
    shipping_discount_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    discount_id BIGINT NOT NULL UNIQUE,
    discount_amount DECIMAL(10,2) NOT NULL,
    CONSTRAINT FK_shippingdiscount_discount FOREIGN KEY (
        discount_id) REFERENCES Discount(discount_id) ON DELETE
    CASCADE
)
```

4.5 SQL Statements Used to Insert the data

```
INSERT INTO User (login_id, password_hash, name, email, phone_number, role, deleted_at, created_at, updated_at) VALUES ('admin001', '$2b$10$xJwL5vGzQzKPL8mRvPzYXe0ZqKPLQ2vZkVjN3xFzR7YmKpJdK8mXe', '系統管理員', 'admin@rtmart.com.tw', '0912345678', 'admin', NULL, '2025-01-01 00:00:00', '2025-01-01 00:00:00'),
```

```
INSERT INTO UserToken (user_id, token_hash, expires_at, is_revoked, created_at) VALUES (6, '$2b$10$tokenHashExample001ForBuyer001ActiveSession', '2025-12-01 00:00:00', FALSE, '2025-11-15 10:00:00'),
```

```
INSERT INTO ShippingAddress (user_id, recipient_name, phone, city, district, postal_code, address_line1, address_line2, is_default) VALUES (6, '陳志明', '0967890123', '台北市', '大安區', '106', '忠孝東路四段100號', '5樓之1', TRUE),
```

```
INSERT INTO AuditLog (event_id, event_timestamp, table_name, record_id, action, user_id, request_id, ip_address, user_agent, service_name, old_data, new_data, changes, checksum) VALUES ('550e8400-e29b-41d4-a716-446655440001', '2025-02-10 16:20:00', 'User', 6, 'INSERT', NULL, 'req-001-register', '192.168.1.100', 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/120.0.0.0', 'auth-service', NULL, '{"login_id": "buyer001", "name": "陳志明", "email": "buyer001@email.com"}', '{"fields": ["login_id", "name", "email", "password_hash"]}', 'sha256:abc123def456'),
```

```
INSERT INTO Seller (user_id, bank_account_reference, verified, verified_at, verified_by) VALUES (3, 'BANK-REF-001-CTBC-****1234', TRUE, '2025-01-20 10:00:00', 1),
```

```
INSERT INTO Store (seller_id, store_name, store_description, store_address, store_email, store_phone, average_rating, total_ratings, deleted_at, created_at, updated_at) VALUES (1, '大明3C旗艦店', '專營各類3C電子產品，提供原廠保固與專業售後服務。iPhone、iPad、MacBook、Samsung、Sony等品牌應有盡有。', '台北市中正區忠孝西路一段50號', 'store001@rtmart.com.tw', '02-23456789', 4.7, 1250, NULL, '2025-01-20 12:00:00', '2025-11-01 08:00:00'),
```

```
INSERT INTO ProductType (type_code, type_name, parent_type_id, is_active) VALUES ('ELECTRONICS', '3C電子', NULL, TRUE), ('FASHION', '服飾配件', NULL, TRUE),
```

```
INSERT INTO Product (store_id, product_type_id, product_name, description, price, view_count, average_rating, total_reviews, deleted_at, created_at, updated_at) VALUES
(1, 6, 'Apple iPhone 15 Pro 256GB', '最新A17  
Pro晶片，鈦金屬設計，4800萬像素主相機，支援USB-C充電。顏色：原色鈦金屬。', 36900.00, 15680, 4.8, 523, NULL, '2025-03-01 10:00:00', '2025-11-10 12:00:00'),
```

```
INSERT INTO ProductImage (product_id, image_url, display_order) VALUES
(1, 'https://cdn.rtmart.com.tw/products/iphone15pro-1-main.jpg', 1),
(1, 'https://cdn.rtmart.com.tw/products/iphone15pro-2-back.jpg', 2),
```

```
INSERT INTO Inventory (product_id, quantity, reserved, last_updated) VALUES
(1, 50, 3, '2025-11-15 10:00:00'),
(2, 35, 2, '2025-11-15 10:00:00'),
(3, 25, 1, '2025-11-15 10:00:00')
```

```
INSERT INTO CartHistory (user_id, cart_snapshot, item_count, order_ids, created_at) VALUES
(6, '{"items": [{"product_id": 1, "product_name": "Apple iPhone 15 Pro 256GB", "quantity": 1, "price": 36900}], "total": 36900}', 1, '[1]', '2025-03-15 14:30:00'),
```

```
INSERT INTO CartItem (cart_id, product_id, quantity, added_at, updated_at) VALUES
(1, 5, 1, '2025-11-14 20:25:00', '2025-11-14 20:25:00'),
```

```
INSERT INTO Order (order_number, user_id, store_id, order_status, subtotal, shipping_fee, total_discount, total_amount, payment_method, payment_reference, idempotency_key, shipping_address_snapshot, notes, created_at, updated_at, paid_at, shipped_at, delivered_at, completed_at, cancelled_at) VALUES
('ORD20250315001', 6, 1, 'completed', 36900.00, 0.00, 0.00, 36900.00, 'credit_card', 'PAY-CC-20250315-001', 'idem-key-001', '{"recipient_name": "陳志明", "phone": "0967890123", "city": "台北市", "district": "大安區", "postal_code": "106", "address_line1": "忠孝東路四段100號", "address_line2": "5樓之1"}', '請在上班時間配送', '2025-03-15 14:30:00', '2025-03-22 10:00:00', '2025-03-15 14:35:00', '2025-03-17 09:00:00', '2025-03-20 14:30:00', '2025-03-22 10:00:00', NULL),
```

```

INSERT INTO OrderItem (order_id, product_id, product_snapshot,
quantity, original_price, item_discount, unit_price, subtotal)
VALUES
(1, 1, '{"product_name": "Apple iPhone 15 Pro 256GB",
"description": "最新A17 Pro晶片，鈦金屬設計", "store_name": "大明3C旗艦店"}', 1, 36900.00, 0.00, 36900.00, 36900.00),
(2, 1, '{"product_name": "Apple iPhone 15 Pro 256GB",
"description": "最新A17 Pro晶片，鈦金屬設計", "store_name": "大明3C旗艦店"}', 1, 36900.00, 0.00, 36900.00, 36900.00),
(3, 1, '{"product_name": "Apple iPhone 15 Pro 256GB",
"description": "最新A17 Pro晶片，鈦金屬設計", "store_name": "大明3C旗艦店"}', 1, 36900.00, 0.00, 36900.00, 36900.00);

INSERT INTO Discount (discount_code, discount_type, name,
description, min_purchase_amount, start_datetime, end_datetime,
is_active, usage_limit, usage_count, created_by_type, created_by_id,
created_at) VALUES
('WINTER2025', 'seasonal', '2025冬季大促銷',
'全站商品95折優惠，把握年末購物好時機！', 1000.00, '2025-11-01 00:00:00',
'2025-12-31 23:59:59', TRUE, NULL, 156, 'system', 1, '2025-10-15
10:00:00'),
('NEWYEAR2026', 'seasonal', '2026新年優惠！新年新氣象，全站商品9折！');

INSERT INTO OrderDiscount (order_id, discount_id, discount_type,
discount_amount, applied_at) VALUES
(2, 3, 'shipping', 60.00, '2025-04-20 11:00:00'),
(2, 6, 'special', 40.00, '2025-04-20 11:00:00'),
(5, 1, 'seasonal', 1895.00, '2025-11-15 10:00:00');

INSERT INTO SeasonalDiscount (discount_id, discount_rate,
max_discount_amount) VALUES
(1, 0.0500, 2000.00),
(2, 0.1000, 3000.00);

INSERT INTO SpecialDiscount (discount_id, store_id, product_type_id,
discount_rate, max_discount_amount) VALUES
(5, 1, 1, 0.2000, 3000.00),
(6, 2, NULL, 0.1500, NULL),
(7, 2, NULL, 0.1000, 500.00);

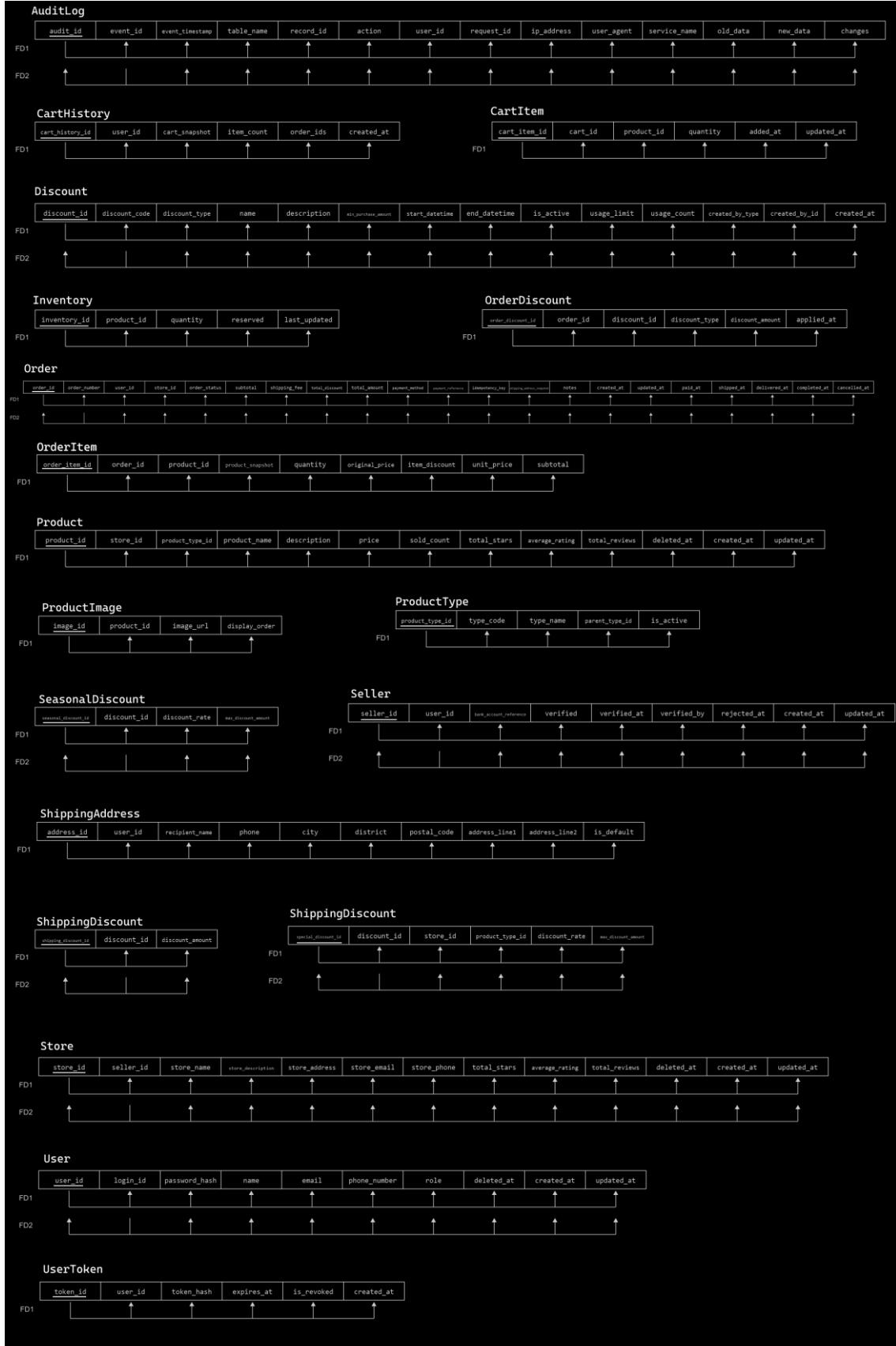
INSERT INTO ShippingDiscount (discount_id, discount_amount) VALUES
(2, 60.00),
(3, 60.00);

INSERT INTO SpecialDiscount (discount_id, store_id, product_type_id,
discount_rate, max_discount_amount) VALUES
(5, 1, 1, 0.2000, 3000.00),
(6, 2, NULL, 0.1500, NULL),
(7, 2, NULL, 0.1000, 500.00);

```

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

5.1 Functional Dependencies



Section 6 邏輯資料庫綱要 (Logic Database Schema)

6.1 系統安裝 (System Installation Description)

1. 安裝 docker desktop

<https://www.docker.com/products/docker-desktop/>

2. 將專案 repo 下載到本地

```
git clone https://github.com/chuni2005/RT-MART.git
```

3. 從 CLI 進入專案目錄後，透過 docker 執行 webapp

```
cd RT-MART
```

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

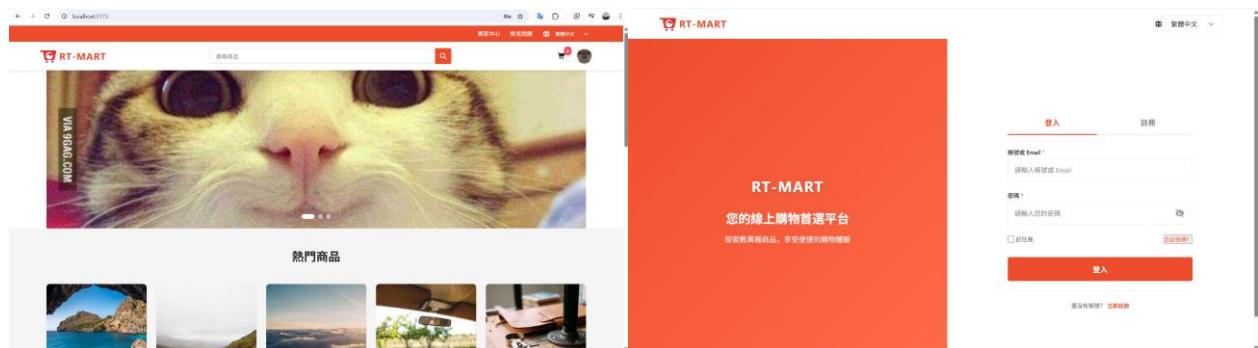
4. 打開瀏覽器，輸入 frontend 的網址

<http://localhost:5173>

6.2 系統使用 (The Use of the System)

6.2.1 買家使用介面

(a) 首頁、登入



(b) 商品瀏覽

The figure consists of three vertically stacked screenshots of the RT-MART website.

- Screenshot 1 (Top): Product Search Results**
This screenshot shows a search results page for "蓝牙". It includes a sidebar for filtering by brand, category, and price range. The main area displays three products: "降噪无线蓝牙耳机(耳罩式)" (\$3,990), "智能语音控制蓝牙音箱" (\$2,080), and "轻巧型户外防水蓝牙喇叭" (\$1,150). Each item has a small image, price, and a star rating.
- Screenshot 2 (Middle): Detailed Product Page**
This screenshot shows a detailed view of the "降噪无线蓝牙耳机(耳罩式)" product. It features a large image of the headphones, a brief description, and a summary of its features. Below the main image are smaller thumbnail images. The product has a 4.4-star rating with 3,301 reviews. A "Buy Now" button is prominently displayed.
- Screenshot 3 (Bottom): Shopping Cart and Order Summary**
This screenshot shows the user's shopping cart and the order summary page. The cart contains one item: the previously viewed headphones. The order summary provides a breakdown of the total cost (\$3,990), shipping (\$0), and tax (-\$60). The total amount due is \$3,990.

The screenshots illustrate the user interface for managing account details on the RT-MART website.

Screenshot 1: Payment Methods

- Left Panel:** Shows a summary of purchases: \$1990 (商品總額), \$3990 (消費總額), -\$40 (消費折扣). It also displays the shipping address: 江蘇省 (江苏省) 0510-465-445, 蘇州市太倉市 106, 久安新村 1號. Below is a payment method section with two options: 銀聯卡 (UnionPay Card) and 寶到付款 (Baodao Payment).
- Middle Panel:** Shows the total amount due: \$3990. A button labeled “確認訂單 (Confirm Order)” is visible.
- Right Panel:** Displays personal information: 姓名 (Name): 江蘇華, Email: jianghuah123@gmail.com, Phone: 0566-465-445. A “儲存變更 (Save Changes)” button is present.

Screenshot 2: Password Management

- Left Panel:** Shows the current password field with placeholder “請輸入舊密碼 (Enter old password)”. Below it is a new password field with placeholder “請輸入新密碼 (Enter new password)”, and a confirmation field with placeholder “請再次輸入新密碼 (Enter new password again)”. A “更新密碼 (Update Password)” button is at the bottom.
- Middle Panel:** Shows a “密碼設定 (Password Settings)” section with a note: “請勿將密碼與其他資料結合使用 (Do not combine the password with other data)”. A “刪除我的帳號 (Delete Account)” button is highlighted with a red border.

Screenshot 3: Address Management

- Left Panel:** Shows the shipping address: 江蘇省 (江苏省) 0510-465-445, 蘇州市太倉市 106, 久安新村 1號. A “新增地址 (Add New Address)” button is visible.
- Middle Panel:** Shows the order summary: 總金額 (Total Amount): \$4050, 支付狀態 (Payment Status): 待付款 (Pending Payment). Buttons for “查看詳情 (View Details)”, “刪除 (Delete)”, and “取消訂單 (Cancel Order)” are shown.

6.2.2 商家使用介面

The screenshots demonstrate the RT-MART merchant interface, which includes:

- Order History:** Shows order details, status (e.g., Order Created, Payment Received), and product information.
- Dashboard:** Provides an overview of sales performance with a chart showing sales trends over time.
- Store Settings:** Allows managing basic store information, contact details, and email settings.
- Product Management:** Manages existing products, adds new ones, and lists them with images and descriptions.
- Product Edit:** Edits product details, including images and descriptions.
- Order Management:** Manages a list of orders, with detailed views showing payment status and shipping information.
- Order Details:** Shows the breakdown of an order, including payment method (Credit Card), shipping address (台中市安平区), and product items.

6.2.3 管理員使用介面

The image displays four screenshots of the RT-MART administration panel, illustrating various management features:

- Order Management (订单管理):** This section shows detailed order information for Order ID ORG17670148967926400. It includes fields for Order Number, Order Date (2025/12/28 下午 09:28), Payment Method (信用卡), and Shipping Address (收件人: 江蕙蕙, Email: jianghuai129@gmail.com). The status is set to 全部状态 (All Status).
- Change Order Status (修改訂單狀態):** A modal window allows changing the status of an order from 未送達 (Not Delivered) to 已送達 (Delivered). The order ID is 36AAAAE1.
- System Discount Settings (系統折扣設定):** This page lists existing discounts:
 - 全場 15% 折扣活動 (SEAS_8635):** Discount rate 15%, Valid Until 2025/5/8, Total usage 211 / 1000 次.
 - 全場 20% 折扣活動 (SEAS_6375):** Discount rate 20%, Valid Until 2025/2/5, Total usage 22 / 100 次.
 - 運費折抵 60 元優惠 (SEAS_6375):** Discount rate 60 NT\$, Valid Until 2025/2/5, Total usage 0 / 100 次.
- User Accounts (會員管理):** This screen shows a list of users with their basic information and account status.

Section 7 Suggestions of the Database Turning

在這次實作 RT-MART 電商系統的過程中，我們不僅完成了基礎功能，也針對未來如果使用者變多、資料量變大時，思考了幾個可以改進的效能方向：

1. 提升搜尋速度：將「模糊查詢」改為「全文索引」

- 目前的狀況：現在我們搜尋商品名稱是用 LIKE %關鍵字%。這在資料庫課堂上有學過，這屬於「線性搜尋」，資料庫必須一筆一筆去比對，當商品有幾萬件時，搜尋就會變得很慢。
- 改進方向：我們可以參考老師教過的 **Index (索引)** 概念，在商品名稱和描述欄位建立 **Full-Text Index (全文索引)**。這樣資料庫就像有一本字典的索引頁，可以直接跳到關鍵字所在的位置，加快搜尋速度。

2. 減輕資料庫壓力：引入快取機制 (**Redis Caching**)

- 目前的狀況：現在使用者每次刷首頁、看購物車，後端都要去資料庫跑一次複雜的查詢（例如 Join 好幾個資料表來算折扣）。如果同時有一百個人在逛網頁，資料庫會非常忙碌。
- 改進方向：我們可以引入 **Redis** 這種「記憶體資料庫」當作快取。把一些常常被讀取、但不會秒秒都在變動的資料（例如熱門商品列表、個人的購物車內容）先放在記憶體裡。這樣下次有人要看，直接從記憶體拿，不用再去煩資料庫，反應時間會快非常多。

3. 處理爆量日誌：針對大表格進行「分區」處理 (**Table Partitioning**)

- 目前的狀況：我們的專案有實作 AuditLog (審核日誌)，這張表會記錄系統所有的動作。隨時間過去，這張表會長得非常大（幾十萬甚至幾百萬筆）。以後要做資料備份或刪除一年前的舊資料時，會卡住很久。
- 改進方向：我們可以使用 **Table Partitioning (分區)** 技術。簡單說就是把一張大表依照「時間」切成好幾塊。例如 2025 年的資料放一塊、2026 年放一塊。查詢時資料庫只要去對應的那一塊找就好，不用翻遍全部資料庫，管理起來也方便很多。

4. 分工合作：將「讀取」與「寫入」分開處理 (**Read/Write Splitting**)

- 目前的狀況：目前的架構是一台資料庫伺服器又要處理「買東西下單（寫入）」又要處理「逛商品看資訊（讀取）」。
- 改進方向：參考業界常見的「讀寫分離」概念。我們可以設一台主機專門負責下單、改資料，另一台（或多台）分身主機負責供大家查詢。這樣就算有很多人在看商品，也不會影響到關鍵的「付錢下單」流程，讓系統更穩定。

5. 精準撈取資料：優化 ORM 查詢效率

- 目前的狀況：我們實作時使用了 TypeORM 框架，有時候為了方便會直接把整個物件連同關聯資料（例如訂單裡的商品快照、圖片連結、賣家資訊）全部撈出來。但如果我們只是要在列表顯示「訂單編號」，撈這麼多資料會造成網路傳輸的浪費。
- 改進方向：我們應該改寫查詢語句，做到「要什麼才撈什麼 (**Select what you need**)」。只選取必要的欄位，並注意避免發生 N+1 查詢問題（也就是抓 10 筆訂單卻跑了 11 次資料庫查詢）。這樣可以大幅降低後端伺服器與資料庫之間的傳輸負擔。

Section 8 附加查詢和視圖(Additional Queries and Views)

8.1 資料查詢語法(Database Queries)

動態查詢商品及打折後的價格

```
SELECT
    p.product_id,
    p.product_name,
    p.price AS original_price,
    MAX(sd.discount_rate) AS max_discount_rate,
    p.price * (1 - COALESCE(MAX(sd.discount_rate), 0)) AS effective_price,
    s.store_name,
    pt.type_name,
    i.quantity AS stock
FROM Product p
LEFT JOIN SpecialDiscount sd
    ON sd.store_id = p.store_id
    AND (sd.product_type_id IS NULL OR sd.product_type_id = p.product_type_id)
LEFT JOIN Discount d
    ON d.discount_id = sd.discount_id
    AND d.is_active = true
    AND d.start_datetime <= NOW()
    AND d.end_datetime >= NOW()
LEFT JOIN Store s ON p.store_id = s.store_id
LEFT JOIN ProductType pt ON p.product_type_id = pt.product_type_id
LEFT JOIN Inventory i ON p.product_id = i.product_id
WHERE p.is_active = true
GROUP BY p.product_id
ORDER BY effective_price DESC;
```

商家每月報表生成

```
SELECT
    DATE_FORMAT(o.created_at, '%Y-%m') AS month,
    COUNT(*) AS order_count,
    SUM(o.total_amount) AS total_revenue,
    AVG(o.total_amount) AS avg_order_value
FROM `Order` o
WHERE o.order_status = 'completed'
    AND o.created_at >= DATE_SUB(NOW(), INTERVAL 12 MONTH)
GROUP BY month
ORDER BY month ASC;
```

商家產能 Dashboard

```
SELECT
    s.seller_id,
    u.name AS seller_name,
    st.store_name,
    st.average_rating AS store_rating,
    COUNT(DISTINCT o.order_id) AS total_orders,
    SUM(CASE WHEN o.order_status = 'completed' THEN o.total_amount ELSE 0 END) AS total_revenue,
    COUNT(DISTINCT p.product_id) AS total_products,
    AVG(p.average_rating) AS avg_product_rating,
    SUM(p.sold_count) AS total_units_sold
FROM Seller s
INNER JOIN User u ON s.user_id = u.user_id
INNER JOIN Store st ON s.seller_id = st.seller_id
LEFT JOIN `Order` o ON st.store_id = o.store_id
LEFT JOIN Product p ON st.store_id = p.store_id AND p.is_active = true
WHERE s.verified = true
GROUP BY s.seller_id
ORDER BY total_revenue DESC;
```

庫存數量不足警告

```
SELECT
    p.product_id,
    p.product_name,
    s.store_name,
    i.quantity AS current_stock,
    i.reserved AS reserved_stock,
    (i.quantity - i.reserved) AS available_stock,
    p.sold_count
FROM Product p
INNER JOIN Store s ON p.store_id = s.store_id
INNER JOIN Inventory i ON p.product_id = i.product_id
WHERE p.is_active = true
    AND (i.quantity - i.reserved) < 10
ORDER BY available_stock ASC, p.sold_count DESC;
```

偵測未付款訂單

```
SELECT
    o.order_id,
    o.order_number,
    u.name AS buyer_name,
    u.email AS buyer_email,
    s.store_name,
    o.total_amount,
    o.created_at,
    TIMESTAMPDIFF(HOUR, o.created_at, NOW()) AS hours_pending
FROM `Order` o
INNER JOIN User u ON o.user_id = u.user_id
INNER JOIN Store s ON o.store_id = s.store_id
WHERE o.order_status = 'pending_payment'
    AND o.created_at < DATE_SUB(NOW(), INTERVAL 24 HOUR)
ORDER BY o.created_at ASC;
```

遞迴搜尋商品父類別

```
WITH RECURSIVE product_type_tree AS (
    -- Base case: start with the specified type
    SELECT
        product_type_id,
        type_code,
        type_name,
        parent_type_id,
        0 AS depth
    FROM ProductType
    WHERE product_type_id = ? -- parameter

    UNION ALL

    -- Recursive case: find children
    SELECT
        pt.product_type_id,
        pt.type_code,
        pt.type_name,
        pt.parent_type_id,
        ptt.depth + 1
    FROM ProductType pt
    INNER JOIN product_type_tree ptt ON pt.parent_type_id = ptt.product_type_id
    WHERE pt.is_active = true
)
SELECT * FROM product_type_tree
ORDER BY depth, type_name;
```

Section 9 Conclusion and Future Work

9.1 Conclusions (結論)

在這次「RT-MART 電商平台」的開發過程中，我們學到的不僅僅是 SQL 語法，更是整套系統設計與開發的邏輯。以下是我們最深刻的幾點體會：

1. 設計與實務的動態調整：

我們發現當初在紙上設計的 **ER Model** 和 Schema 雖然邏輯正確，但隨著前端 React 功能的擴充（例如評價系統的加入與用戶頭像），原先的資料庫設計必須不斷地進行微調與增刪欄位。這讓我們學到：資料庫設計並不是一成不變的，必須具備足夠的靈活性來支撐前端的需求。

2. 串接外部服務的挑戰與成果：

為了讓系統更接近真實商用環境，我們花了不少時間研究並串接了 **Cloudinary (圖片管理)** 與 **MailService (郵件通知)**。雖然在環境變數配置與 API 串接上遇到了不少坑，但當我們看到商品圖片能成功上傳雲端、使用者能收到下單成功的確認信時，那種成就感讓我們覺得所有的努力都非常值得。

3. 從「不適應框架」到「理解架構」：

這也是我們第一次接觸 **NestJS** 與 **React** 這類大型框架。一開始確實非常不適應框架規定的開發格式（例如 Module、Service 的切分），但在經歷過一段磨合期後，我們逐漸理解了框架設計背後的哲學，這讓我們在設計複雜功能時能更有系統性地思考，而不是寫出雜亂無章的程式碼。

4. 在重構中成長：

所謂「計畫趕不上變化」，在開發中我們經歷了多次因為原始設計不符合實際需求而進行的「**重構 (Refactoring)**」。雖然過程很痛苦，但也正是在這些反覆修改的過程中，我們才真正理解如何更好地設計一個系統的架構，並學會如何寫出更易於維護的程式碼。

9.2 Future Work (未來展望)

雖然目前系統已具備核心交易功能，但我們對 RT-MART 的規劃不只於此，未來我們希望從以下幾個面向持續進化：

1. 社交與互動功能：

計畫加入即時聊天系統 (**Chat System**)，讓買家與賣家能直接溝通；並開發願望清單 (**Wishlist**)，增加使用者對平台的黏著度。

2. 即時通知系統：

目前僅有郵件通知，未來預計整合 **WebSocket** 或 **SSE** 技術，實現站內的即時訊息推送（如訂單狀態變更通知、系統公告）。

3. 效能調優與深度重構：

如前述調校建議，我們將引入 **Redis 快取** 與 **Full-Text Index** 搜尋優化。同時，我們也會針對目前專案中較不完善的舊程式碼進行持續的重構，優化 API 的反應速度與程式碼品質。

4. 自動化測試與安全性：

目前專案在單元測試與壓力測試上還有進步空間。未來希望能加入更多的自動化測試案例，

並加強資料庫的存取權限控管與資料加密，讓系統更加穩固安全。

5. 整合多元支付金流 API (Payment Gateway Integration) :

目前的系統僅模擬了訂單的付款狀態。未來我們希望能正式串接台灣常見的第三方支付服務（如 **Line Pay**、**綠界 ECPay** 或 **藍新 NewebPay**），實作真正的線上付款功能。這不僅能提升使用者體驗，也能讓我們學習如何處理更嚴謹的資料庫交易事務 (Transaction)、金流安全驗證，以及透過 **Webhook** 機制即時回傳並更新訂單狀態。

Glossary

DBMS

Database Management System，是一種針對物件資料庫，為管理資料庫而設計的大型電腦軟體管理系統。

NestJS

是一個 server-side、以 Node.js 為核心的 web framework，以 MIT license 開源的免費軟體

MariaDB

是一個社群開發的開源 DBMS 軟體。為原先 MySQL 開發團隊，因 MySQL 被 Oracle 收購後產生疑慮，而 fork 出來的另一個版本。同時支援 MySQL 的所有功能。

ReactJS

是一個開源、以 JavaScript 撰寫的前端 Library，目前 Meta 是主要維護者

Cloudinary

是一個 SaaS 軟體，主要業務是給予各種網頁、軟體開發者提供雲端儲存的服務

References

NestJS Documentation: <https://docs.nestjs.com/>

MariaDB Documentation: <https://mariadb.com/docs/>

ReactJS Documentation: <https://react.dev/reference/react>

Cloudinary Documentation: <https://cloudinary.com/documentation>

GoogleAPI Documentation: <https://googleapis.dev/nodejs/translate/latest/v2.Translate.html>

Appendix

demo slides:

https://www.canva.com/design/DAG3Pl5Q25Y/G9R2N5FjoGAYz2vaS67cfw/edit?utm_content=DAG3Pl5Q25Y&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

source code: <https://github.com/chuni2005/RT-MART>