

線上商城系統 (Web-based Shop Store System)

系統需求規格書

Software Requirements Specification (SRS)

Version: Fianl version

姓名	E-mail
李文至	t108590001@ntut.org.tw
陸永強	luyongqiang1827@gmail.com
黃明萱	t108590003@ntut.org.tw
林聖祐	t108590008@ntut.org.tw
張景辰	t108590010@ntut.org.tw
王裕詮	t108590031@ntut.org.tw
林峻霆	timlin891207@gmail.com
宋宇倫	t108360143@ntut.org.tw

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

01/21/2021

目錄 (Table of Contents)

Section 1 系統(System)	3
1.1 目的 (Purpose).....	3
1.2 系統名稱 (Identification).....	3
1.3 概觀 (Overview)	3
1.4 符號描述 (Notation Description) (if any).....	4
Section 2 系統(System)	5
2.1 系統描述 (System Description).....	5
2.2 操作概念 (Operational Concepts or User Stories).....	6
2.3 功能性需求 (Functional Requirements)	6
2.4 非功能性需求 (Non-Functional Requirements)	6
2.4.1 效能需求 (Performance Requirements)	6
2.5 介面需求 (Interface Requirements)	6
2.5.1 使用者介面需求 (User Interfaces Requirements)	6
2.5.2 外部介面需求 (External Interface Requirements) (if any).....	6
2.5.3 內部介面需求 (Internal Interface Requirements) (if any)	7
2.6 其他需求 (Other Requirements).....	7
2.6.1 環境需求 (Environmental Requirement)	7
2.6.2 安裝需求 (Installation Requirement)	7
2.6.3 測試需求 (Test Requirements) (if any)	7
Section 3 資料庫概念設計 (Conceptual Design of the Database)	8
3.1 Entity-Relationship (ER) Diagram.....	8
Section 4 資料庫概念設計 (Conceptual Design of the Database)	9
4.1 Database Schema	9
4.2 Description of ER Diagram.....	9
4.3 SQL statements for database construction	12
4.4 SQL statements for data population	13
4.5 The implementation of tables in target DBMS.....	16
Section 5 功能性依賴(Functional Dependencies and Database Normalization).....	17
5.1 功能性依賴 (Functional Dependencies)	17
Section 6 數據庫系統的使用(The Use of the Database System).....	18
6.1 系統安裝 (System Installation Description)	18
6.2 系統使用 (The Use of the System)	20
6.2.1 使用者頁面.....	20
6.2.2 管理員頁面.....	27
Section 7 資料庫優化建議(Suggestions of Database Tuning)	31
Section 8 附加查詢和視圖(Additional Queries and Views)	32
8.1 附加查詢 (Additional Queries)	32

8.2 查詢展示 (Views).....	43
Section 9 結論與未來規劃(Conclusions and Future Work).....	44
9.1 結論 (Conclusions)	44
9.2 未來規劃 (Future Work)	46
Glossary	47
References.....	49
Appendix.....	50

Section 1 系統(System)

1.1 目的 (Purpose)

藉由本學期修習的資料庫系統課程，為了更加理解資料庫的運作以及表格設計實作，並且可以有效運用曾經學過的網際網路技術與應用的觀念，以及各項程式語言的基礎，我們分配了工作給每位成員並且完成使用資料庫的系統，因此我們經由指導後目標完成一個完整線上商城的系統，能讓使用者可以在網路上瀏覽、搜索、購買商品。讓大家擁有賓至如歸的網路購物體驗。我們的主要目標為：

- 讓顧客可以搜尋到想要看商品並且購買
- 管理員可管理會員、賣場
- 計算銷售狀況與統計資料
- 可提供所有人查詢商品資料

1.2 系統名稱 (Identification)

此專案有分為主系統與子系統

主系統為：

- 線上商城系統(Web-based Shop Store System,WSSS)

子系統分別為：

- 使用者管理子系統(User Management Subsystem,UMS)
- 購物車管理子系統(Shopping Car Management Subsystem,SHMS)
- 上下架管理子系統(Down and Up Management Subsystem,DUMS)
- 商品管理子系統(Products Management Subsystem,PMS)
- 商品搜尋管理子系統(Search Management Subsystem,SMS)
- 商品暢銷管理子系統(Sell Best Management Subsystem,SBMS)
- 資料庫子系統(Database Subsystem, DBS)
- 結帳管理子系統(Bill Management Subsystem,BMS)

1.3 概觀 (Overview)

在資料庫系統課程的學習中，我們了解資料庫其中所包含的內容，像是關聯性的資料庫，分散式資料庫的管理等等，資料庫在目前網際網路應用是一個不可或缺的一項功能與技術，很多網頁都是由資料庫所建成的，因此，我們在此次的專案做一個和資料庫相關的系統，我們採用 MySQL 這個平台，其目的是 MySQL 方便安裝與架設，並且也可以支援許多程式語言，藉此設計出一個可以讓大眾方便使用的購物系統提供大家使用。

1.4 符號描述 (Notation Description) (if any)

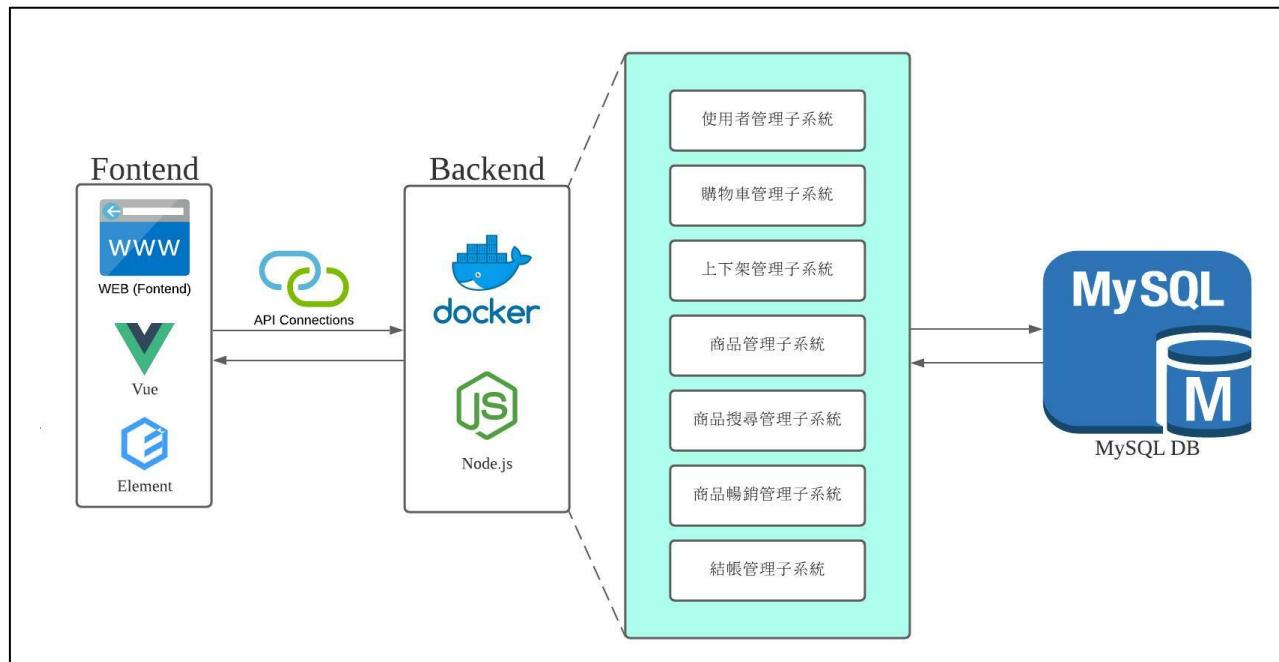
WSSS 1.0.0	The WSSS system will be labeled with the number 1.0.0.
UMS 1.1.n	The UMS components will be labeled with the number 1.1.n.
SHMS 1.2.n	The SHMS components will be labeled with the number 1.2.n.
DUMS 1.3.n	The DUMS components will be labeled with the number 1.3.n.
PMS 1.4.n	The PMS components will be labeled with the number 1.4.n.
SMS 1.5.n	The SMS components will be labeled with the number 1.5.n.
SBMS 1.6.n	The SBMS components will be labeled with the number 1.6.n.
DBS 1.7.n	The DBS components will be labeled with the number 1.7.n.
BMS 1.8.n	The BMS components will be labeled with the number 1.8.n.

Section 2 系統(System)

2.1 系統描述 (System Description)

- 線上商城系統(Web-based Shop Store System,WSSS)：這個系統涵蓋了以下子系統。
- 使用者管理子系統(User Management Subsystem,UMS)：將使用者帳號做各種不同的身分區別，分別為管理者和會員，例如：管理者可以上下架商品、會員才能購買商品、等。
- 購物車管理子系統(Shopping Car Management Subsystem,SHMS)：每個帳號都有分別對應的購物車，購物車的每個商品都可以有刪除，新增的功能
- 上下架管理子系統(Down and Up Management Subsystem,DUMS)：帳號為管理者的人才可以使用此系統，此系統可以用來管理所有商品上下架狀況。
- 商品管理子系統(Products Management Subsystem,PMS)：帳號為管理者的人才可以使用此系統，此系統可以定義價錢、基本介紹。
- 商品搜尋管理子系統(Search Management Subsystem,SMS)：可以讓使用者以各種不同方式搜尋商品，像是類別、名稱，相似字詞，以及價格範圍。
- 商品暢銷管理子系統(Sell Best Management Subsystem,SBMS)：以購買多寡，次數來做排行，並顯示給消費者以供參考。
- 訂單管理子系統(Orders Management System,OMS)：可以讓管理員查詢所有會員的訂單紀錄，以及會員可以查詢自己的訂單紀錄，或取消訂單。
- 資料庫系統(Databases System,DBS)：存放使用者、商品以及訂單的所有資訊。

2.1.1 系統架構圖 (System Architecture Diagram)



2.2 操作概念 (Operational Concepts or User Stories)

訪客操作觀念：以訪客的身分可以瀏覽及查詢商城內的商品，但是將要購買的商品放入購物車以及付款必須先註冊成為會員才能操作。

會員操作觀念：使用者註冊成為會員，與訪客功能相同，並新增結帳、個人資料、信用卡綁定、查詢交易紀錄及查看訂單狀況或取消訂單之功能。

管理員操作概念：使用者以系統管理員身分登錄，可管理所有會員的帳號及權限、查看所有訂單，以及上下架商品之功能。

2.3 功能性需求 (Functional Requirements)

需求編號	需求描述
FMS 001	可對資料做新增/修改/刪除/查詢
FMS 002	結帳系統
FMS 003	缺貨通知系統

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

2.4.1 效能需求 (Performance Requirements)

需求編號	需求描述
NFR 001	網頁/資料庫故障頁面通知
NFR 002	網頁的顯示速度

2.5 介面需求 (Interface Requirements)

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

需求編號	需求描述
MIR 001	簡潔明瞭的介面方便使用者操作
MIR 002	將產品分別顯示出來
MIR 003	可選擇付款方式
MIR 004	可對產品評價
MIR 005	缺貨時會通知

2.5.2 外部介面需求 (External Interface Requirements) (if any)

需求編號	需求描述
EIR 001	透過 HTTP 通訊協定主機通訊

2.5.3 內部介面需求 (Internal Interface Requirements) (if any)

需求編號	優先順序	需求描述
IIR 001	1	SHMS 和 DBS 間必須能傳送與接收使用者購物車資訊
IIR 002	1	UMS 和 SHMS 間必須能傳送與接收使用者所選取之商品資訊
IIR 004	1	SMS 和 UMS 間必須能傳送與接收使用者感興趣及追蹤的商品資訊
IIR 005	1	SMS 和 OMS 間必須能傳送與接收使用者追蹤清單及感興趣商品的相關資訊
IIR 006	1	SMS 和 SMS 間必須能傳送與接收使用者感興趣及追蹤的商店資訊
IIR 008	1	SMS 和 UMS 間必須能傳送與接收該店家登入的相關資訊
IIR 009	1	UMS 和 DBS 間必須能傳送與接收各種使用者登入資訊
IIR 010	1	SMS 和 UMS 間必須能傳送與接收商場管理員登入的相關資訊
IIR 011	1	SMS 和 CMS 間必須能傳送與接收管理商品的相關資訊
IIR 012	1	PMS 和 DBS 間必須能傳送與接收商品的相關資訊

2.6 其他需求 (Other Requirements)

2.6.1 環境需求 (Environmental Requirement)

需求編號	需求描述
ER 001	開發環境為 Docker4.2 以及 MySQL 8

2.6.2 安裝需求 (Installation Requirement)

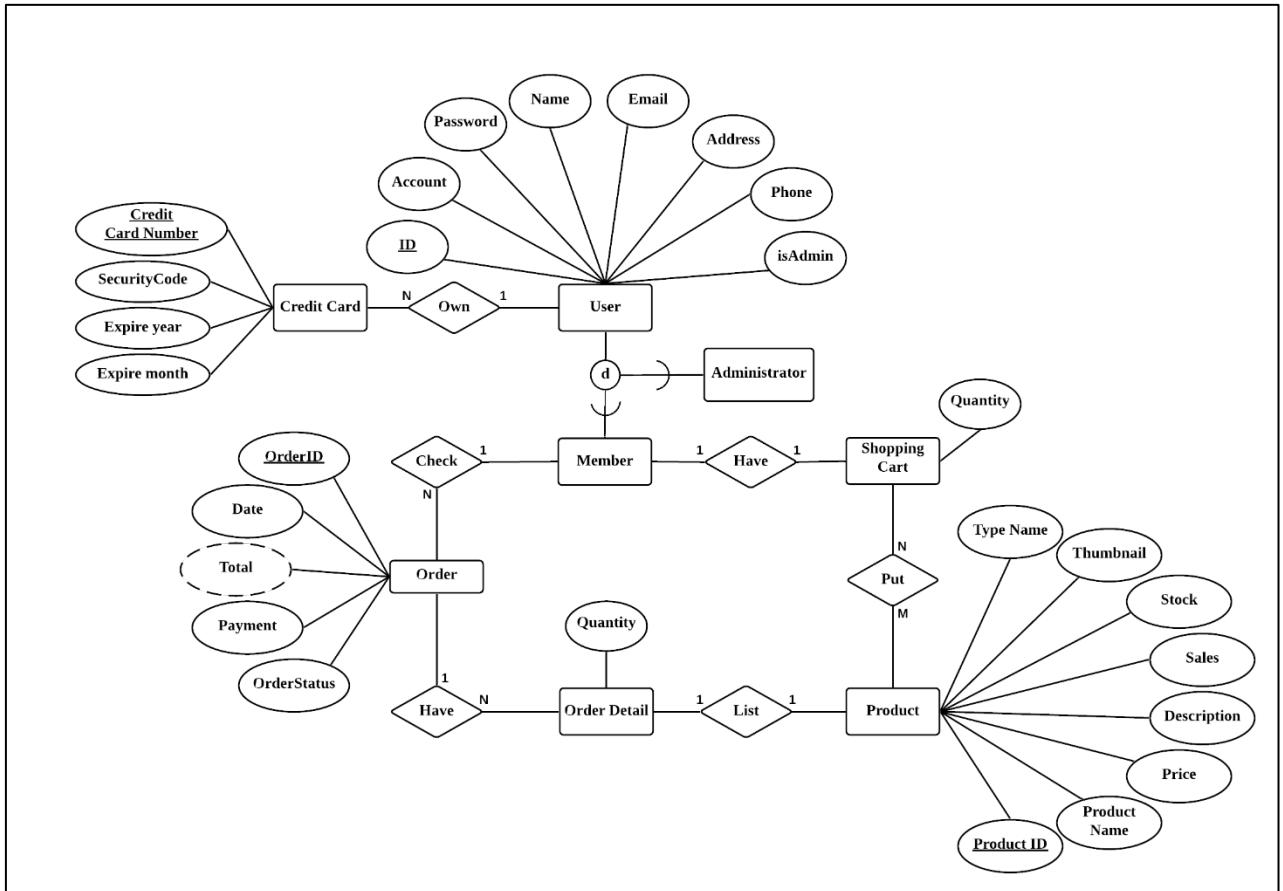
需求編號	需求描述
IR 001	使用 Docker Compose 自動建置服務，降低人工建置發生錯誤的機率

2.6.3 測試需求 (Test Requirements) (if any)

需求編號	需求描述
TR 001	提供使用者 GUI 介面的測試
TR 002	每個功能都有對應的測試內容

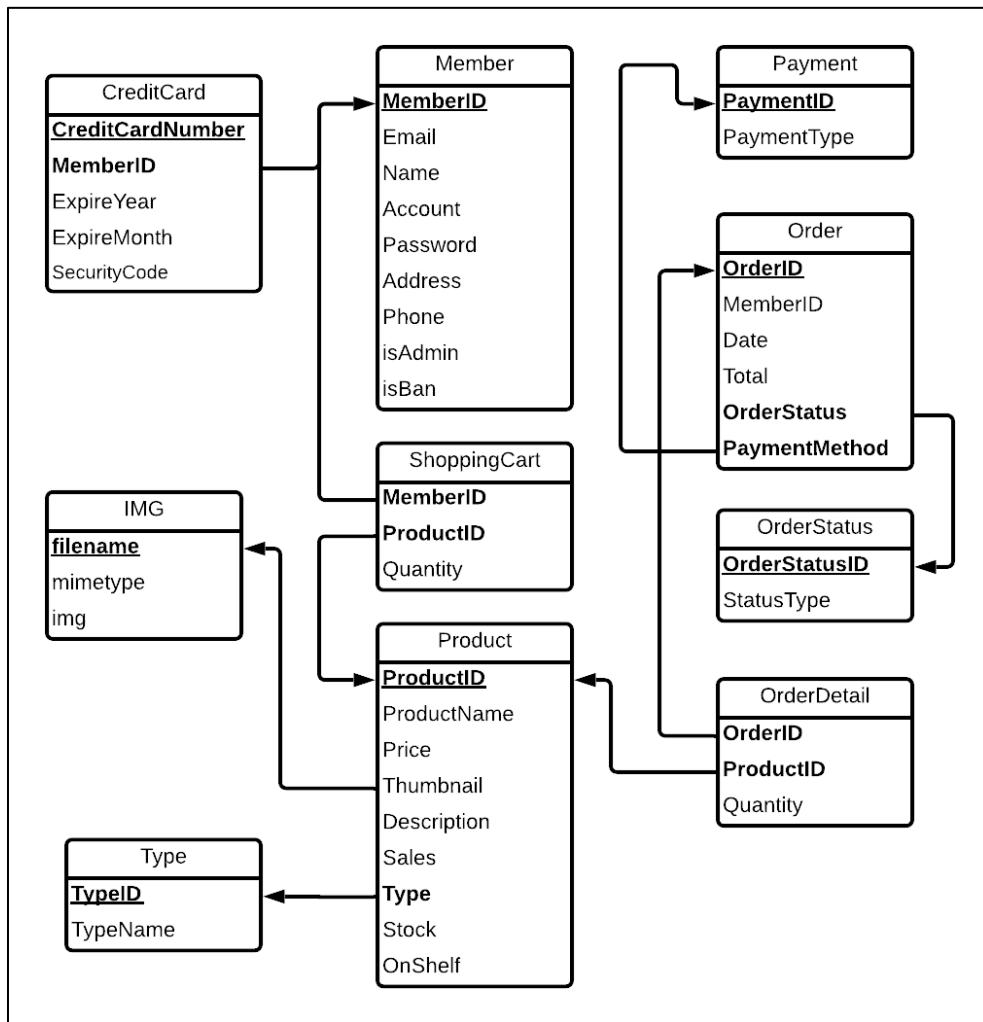
Section 3 資料庫概念設計 (Conceptual Design of the Database)

3.1 Entity-Relationship (ER) Diagram



Section 4 資料庫概念設計 (Conceptual Design of the Database)

4.1 Database Schema



4.2 Description of ER Diagram

Member				
Description : 存放會員資料				
Attribute	Domain Type	Key	Nullable	Description
<u>MemberID</u>	int	primary	not null	會員編號(id>0)
Email	varchar		not null	會員電子信箱
Name	varchar		not null	會員姓名
Account	varchar		not null	會員帳號
Password	varchar		not null	會員密碼
Address	varchar			會員地址
Phone	varchar			會員手機號碼

IsAdmin	int		not null	權限等級 0 一般會員 1 管理員
IsBan	int		not null	會員是否被停權 0 被停權 1 正常使用

Payment

Description : 存放付款方式

Attribute	Domain Type	Key	Nullable	Description
PaymentID	int	primary	not null	付款分類
PaymentType	varchar		not null	付款方式

Product

Description : 存放商品資料

Attribute	Domain Type	Key	Nullable	Description
ProductID	int	primary	not null	商品 ID
ProductName	varchar		not null	商品名稱
Price	int		not null	商品價格
Thumbnail	varchar			商品縮圖網址
Description	mediumtext			商品介紹
Sales	int		not null	商品銷售量
Type	int	foreign	not null	商品類別
Stock	int		not null	商品庫存
OnShelf	varchar		not null	商品是否上架

ShoppingCart

Description : 存放購物車資料

Attribute	Domain Type	Key	Nullable	Description
MemberID	int	foreign	not null	會員 ID
ProductID	int	foreign	not null	商品 ID
Quantity	int		not null	商品數量

OrderDetail

Description : 存放交易清單資料

Attribute	Domain Type	Key	Nullable	Description
OrderID	int	foreign	not null	交易 ID
ProductID	int	foreign	not null	商品 ID
Quantity	int		not null	商品數量

Order				
Description : 存放交易紀錄資料				
Attribute	Domain Type	Key	Nullable	Description
OrderID	int	primary	not null	交易 ID
MemberID	int	foreign	not null	會員 ID
Date	datetime		not null	交易時間
Total	int		not null	交易總額
OrderStatus	int	foreign	not null	訂單狀態
PaymentMethod	int	foreign	not null	付款方式

Type				
Description : 存放商品類別資料				
Attribute	Domain Type	Key	Nullable	Description
TypeID	int	primary	not null	商品類別 ID
TypeName	varchar		not null	商品類別名稱

OrderStatus				
Description : 存放訂單狀態資料				
Attribute	Domain Type	Key	Nullable	Description
OrderStatusID	int	primary	not null	訂單狀態 ID
StatusType	varchar		not null	訂單狀態類別

CreaditCard				
Description : 存放信用卡資料				
Attribute	Domain Type	Key	Nullable	Description
CreaditCardNumber	varchar	primary	not null	信用卡號
MemberID	int	foreign	not null	會員 ID
ExpireYear	int		not null	有效年份
ExpireMonth	int		not null	有效月份
SecurityCode	varchar		not null	信用卡安全碼

Image				
Description : 存放商品圖片				
Attribute	Domain Type	Key	Nullable	Description
filename	varchar	primary	not null	圖片名稱
mimetype	varchar		not null	圖片類型
img	longblob		not null	商品圖片

4.3 SQL statements for database construction

```
CREATE TABLE `Member` (
  `MemberID` int NOT NULL AUTO_INCREMENT,
  `Email` varchar(255) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Account` varchar(255) NOT NULL,
  `Password` varchar(255) NOT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `Phone` varchar(255) DEFAULT NULL,
  `IsAdmin` int NOT NULL,
  `isBan` int NOT NULL,
  PRIMARY KEY (`MemberID`)
);
```

```
CREATE TABLE `Payment` (
  `PaymentID` int NOT NULL,
  `PaymentType` varchar(255)NOT NULL,
  PRIMARY KEY (`PaymentID`)
);
```

```
CREATE TABLE `Product` (
  `ProductID` int NOT NULL AUTO_INCREMENT,
  `ProductName` varchar(255) NOT NULL,
  `Price` int NOT NULL,
  `Thumbnail` varchar(255) DEFAULT NULL,
  `Description` mediumtext NULL,
  `Sales` int NOT NULL,
  `Type` int NOT NULL,
  `Stock` int NOT NULL,
  `OnShelf` varchar(255)NOT NULL,
  PRIMARY KEY (`ProductID`),
  CONSTRAINT `product_ibfk_1` FOREIGN KEY (`Type`) REFERENCES `Type` (`TypeID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

```
CREATE TABLE `ShoppingCart` (
  `MemberID` int NOT NULL,
  `ProductID` int NOT NULL,
  `Quantity` int NOT NULL,
  CONSTRAINT `shoppingcart_ibfk_1` FOREIGN KEY (`MemberID`) REFERENCES `Member` (`MemberID`) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT `shoppingcart_ibfk_2` FOREIGN KEY (`ProductID`) REFERENCES `Product` (`ProductID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

```
CREATE TABLE `OrderDetail` (
  `OrderID` int NOT NULL,
  `ProductID` int NOT NULL,
  `Quantity` int NOT NULL,
  CONSTRAINT `orderdetail_ibfk_2` FOREIGN KEY (`ProductID`) REFERENCES `Product` (`ProductID`) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT `orderdetail_ibfk_3` FOREIGN KEY (`OrderID`) REFERENCES `Order` (`OrderID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

```

CREATE TABLE `Order` (
    `OrderID` int NOT NULL AUTO_INCREMENT,
    `MemberID` int NOT NULL,
    `Date` datetime NOT NULL ON UPDATE CURRENT_TIMESTAMP,
    `Total` int NOT NULL,
    `OrderStatus` int NOT NULL,
    `PaymentMethod` int NOT NULL,
    PRIMARY KEY (`OrderID`),
    CONSTRAINT `order_ibfk_1` FOREIGN KEY (`MemberID`) REFERENCES `Member` (`MemberID`) ON DELETE RESTRICT ON UPDATE RESTRICT,
    CONSTRAINT `order_ibfk_2` FOREIGN KEY (`OrderStatus`) REFERENCES `OrderStatus` (`OrderStatusID`) ON DELETE RESTRICT ON UPDATE RESTRICT,
    CONSTRAINT `order_ibfk_3` FOREIGN KEY (`PaymentMethod`) REFERENCES `Payment` (`PaymentID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);

```

```

CREATE TABLE `Type` (
    `TypeID` int NOT NULL AUTO_INCREMENT,
    `TypeName` varchar(255) NOT NULL,
    PRIMARY KEY (`TypeID`)
);

```

```

CREATE TABLE `OrderStatus` (
    `OrderStatusID` int NOT NULL,
    `StatusType` varchar(255) NOT NULL,
    PRIMARY KEY (`OrderStatusID`)
);

```

```

CREATE TABLE `CreditCard` (
    `CreditCardNumber` varchar(30) NOT NULL,
    `MemberID` int NOT NULL,
    `ExpireYear` int NOT NULL,
    `ExpireMonth` int NOT NULL,
    `SecurityCode` varchar(30) NOT NULL,
    PRIMARY KEY (`CreditCardNumber`),
    CONSTRAINT `creditcard_ibfk_1` FOREIGN KEY (`MemberID`) REFERENCES `Member` (`MemberID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);

```

```

DROP TABLE IF EXISTS `Image`;
CREATE TABLE `Image` (
    `filename` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
    `mimetype` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL,
    `img` longblob NULL,
    PRIMARY KEY (`filename`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = DYNAMIC;

```

4.4 SQL statements for data population

```

INSERT INTO `Member` VALUES (1, 'admin@test.com', 'administrator', 'admin', '12345', NULL, NULL, 1, 0);
INSERT INTO `Member` VALUES (2, '123@gmail.com', '123', 'tim1207', 'tim1207', '123', NULL, 0, 1);
INSERT INTO `Member` VALUES (3, 'test123@gmail.com', 'test123', 'test123', '12345', 'abc', NULL, 0, 0);
INSERT INTO `Member` VALUES (4, 'test456@gmail.com', 'test', 'test458', '123', '', 0, 0);

```

```

INSERT INTO `Payment` VALUES (1, '信用卡');
INSERT INTO `Payment` VALUES (2, '貨到付款');

```

```

INSERT INTO `Product` VALUES (1, '甜蜜香蕉', 10, '/img/2a5bec4b1ede9fbe95b591392a569f06', '有機香蕉採用台蕉五號及新北蕉品種，搭配旗山特有的環境，生產的香蕉風味獨特，口感Q彈，酸甜適中，果肉細緻，香濃無比。')
INSERT INTO `Product` VALUES (2, '富士山蘋果', 10, '/img/e642d6c24fc2ce7c895bb5bd6592b77', '由「國光」與「Delicious」二種品種交配而成，也是日本代表性的蘋果品種。大小均勻，果肉鮮嫩多汁，酸甜適中，果皮薄，果肉厚，果肉細緻，香濃無比。')
INSERT INTO `Product` VALUES (3, 'Airpods', 5000, '/img/490f79945af6a9dd2de04ddd375d32d7', 'AirPods 由 Apple H1 耳機晶片驅動，能與你的各個裝置建立更快速、更穩定的無線連接。')
INSERT INTO `Product` VALUES (4, '巧克力', 100, '/img/bb1b2bddc5e1816fec2cc2752e7829500', '巧克力是以可可豆和可可脂為主要原料製成的一種甜食。它不但口感細膩甜美，而且還含有豐富的營養成分。')
INSERT INTO `Product` VALUES (5, '哇沙米脆豌豆151g', 57, '/img/6c9a110e61153f9151827f16af04dac', '巧克力是以可可豆和可可脂為主要原料製成的一種甜食。它不但口感細膩甜美，而且還含有豐富的營養成分。')
INSERT INTO `Product` VALUES (6, 'IPHONE 13藍256G MLQA3TA/A【全國電子】', 29400, '/img/18f55df5401afacce9efcb63612e91a', 'iPhone 13 帶來突破性的相機技術，強勁的 A15 處理器，更長的續航時間，以及前所未有的耐用性。')
INSERT INTO `Product` VALUES (7, '【孩之寶 hasbro】粉紅豬小妹 佩佩的遊樂場遊戲組 F24025L00', 1199, '/img/63e8630e89a1fbde4a6134f2df03899f', '佩佩的派對時光！孩子們可以用這款遊戲組來扮演佩佩，享受她喜歡的遊戲。')
INSERT INTO `Product` VALUES (8, '【獨立包裝】口罩KF94韓版氧化銅離子滅活口罩 四層含熔噴布 魚嘴型摺疊口罩 KF94口罩 立體口罩 10只/包', 1, '/img/64911cc79f52ee6249f49df9', 'KF94口罩，韓國製造，採用四層結構，內層為熔噴布，外層為聚丙烯，中間兩層為無纺布。')
INSERT INTO `Product` VALUES (9, '台灣優紙 成人3D立體醫療口罩 耳塞式(未滅菌) 立體口罩 25片盒裝 口罩 醫療口罩 預用口罩 優紙成人口罩', 150, '/img/974d4a76db2ceb1869t', '台灣優紙，台灣製造，採用3D立體設計，佩戴舒適，防護效果優異。')
INSERT INTO `Product` VALUES (10, '魔芋爽 4元 單獨包裝 3包', 3, '/img/ff38b81990418e7f16af04dac', '魔芋爽，單獨包裝，方便攜帶，低卡路里，低脂肪，低熱量，是理想的減重食品。')
INSERT INTO `Product` VALUES (11, 'adidas ULTRABOOST 21 TOKYO 跑鞋 男 S23863', 4159, '/img/974da476db2ceb1869b71cd3d9758e19', 'adidas Ultraboost 21 跑鞋提供絕佳舒適感，強勁的回彈力，讓你跑得更遠。')
INSERT INTO `Product` VALUES (12, '衛龍魔芋爽 風吃海帶 純嘴燒 麵筋【手機批發網】《現貨 多種口味 快速出貨》熱賣爆款 零食 零嘴', 3, '/img/55098479612b68b8802e15db12ac28t', '衛龍魔芋爽，風吃海帶，純嘴燒，麵筋，多種口味，方便快捷，是理想的零食選擇。')
INSERT INTO `Product` VALUES (13, 'IG高品質台灣粉 ig台灣粉 純身保固IG粉絲 Instgram粉絲 華人粉 全球粉 真人粉 IG粉 IG粉絲', 1, '/img/55098479612b68b8802e15db12ac2899', 'IG高品質台灣粉，ig台灣粉，純身保固IG粉絲，Instgram粉絲，華人粉，全球粉，真人粉，IG粉，IG粉絲，IG粉，IG粉絲，IG粉，IG粉絲。')
INSERT INTO `Product` VALUES (14, '要變 長夾 手拿包 手包 男款錢包 錢包男式長款錢包 真皮錢包 時尚款', 1, '/img/269f4f8e01e64ed568fbaf7e22db8764', '要變，長夾，手拿包，手包，男款錢包，錢包男式長款錢包，真皮錢包，時尚款。')
INSERT INTO `Product` VALUES (15, '吊帶交叉假兩件式上衣【3010】秋冬新款轉系INS氣質露肩上衣 繩針上衣 鉤針衫 編織腰上衣 女生上衣 PAPI', 198, '/img/248c263dcraf1434b06b6f0b5c6', '吊帶交叉假兩件式上衣，秋冬新款轉系INS氣質露肩上衣，繩針上衣，鉤針衫，編織腰上衣，女生上衣，PAPI。')
INSERT INTO `Product` VALUES (16, '【可信用卡】陸版抖音 20000抖幣 直播 Tik Tok 抖音充值 儲值 代儲 教學 咨詢 全網最低價 中科抖音', 9240, '/img/58be69df6a6c1ce21685t', '【可信用卡】陸版抖音，20000抖幣，直播 Tik Tok，抖音充值，儲值，代儲，教學，諮詢，全網最低價，中科抖音。')
INSERT INTO `Product` VALUES (17, '馬桶小花清香廁所除臭香薰凝膠潔凈靈 馬桶清潔劑 衛生間空氣清新劑 馬桶小花 馬桶香薰凝膠 馬桶潔凈靈', 19, '/img/08aa2dd2f9fc9372bb9a81ec', '馬桶小花清香廁所除臭香薰凝膠潔凈靈，馬桶清潔劑，衛生間空氣清新劑，馬桶小花，馬桶香薰凝膠，馬桶潔凈靈。')
INSERT INTO `Product` VALUES (18, 'ScottyBear™那隻熊 Z04008 新款貓子女中筒 ins潮 棉白色夏季可愛日系黑色字母長襪', 39, '/img/35ee7d3d04dd4af83a1ade86b79f32ab', 'ScottyBear™那隻熊 Z04008 新款貓子女中筒 ins潮 棉白色夏季可愛日系黑色字母長襪。')
INSERT INTO `Product` VALUES (19, '【OOTD批發】女生秋冬外套 韓版復古chic燈芯絨外套長袖秋季 簡約純色高領內搭上衣 2020寬鬆襯衫 女生衣著', 168, '/img/e1bd26d73356a0eb7d774', '【OOTD批發】女生秋冬外套，韓版復古chic燈芯絨外套長袖秋季，簡約純色高領內搭上衣，2020寬鬆襯衫，女生衣著。')
INSERT INTO `Product` VALUES (20, '台灣24H出貨【送-絕美香氛蠟燭】融蠟燭 蠟燭暖蠟 融蠟燭 香氛蠟燭 暖蠟 融蠟燭 香氛蠟燭 暖蠟 燭 暖蠟 神聖禮物 交換禮物', 679, '/img/8bf1fc3ab296', '台灣24H出貨【送-絕美香氛蠟燭】融蠟燭，蠟燭暖蠟，融蠟燭，暖蠟，蠟燭，暖蠟，香氛蠟燭，暖蠟，蠟燭，暖蠟，聖誕禮物，交換禮物。')
INSERT INTO `Product` VALUES (21, '小林製藥 小白兔暖暖包 10片/包 (手提袋 24小時持續恆溫) 專品藥局', 149, '/img/d3f026f4c74975ab119653f435cea59', '小林製藥，小白兔暖暖包，10片/包，(手提袋 24小時持續恆溫) 專品藥局。')
INSERT INTO `Product` VALUES (22, '日本mdc左旋肉鹼小綠袋纖體丸 48回增量版 減重產品 減脂瘦身 窮竈 產後暴瘦 經期暴食 飽腹 抑制食欲', 420, '/img/f511d7fb3b3199770901b8501', '日本mdc左旋肉鹼小綠袋纖體丸，48回增量版，減重產品，減脂瘦身，窮竈，產後暴瘦，經期暴食，飽腹，抑制食欲。')
INSERT INTO `Product` VALUES (23, 'DayDay少女 | 雨兩穿~韓國學院系加絨加厚羊羔毛絨棉服 冬季厚實保暖連帽外套 女生外套 飛行外套 加厚外套 學生外套', 598, '/img/50a0a9867eb', 'DayDay少女 | 雨兩穿~韓國學院系加絨加厚羊羔毛絨棉服，冬季厚實保暖連帽外套，女生外套，飛行外套，加厚外套，學生外套。')
INSERT INTO `Product` VALUES (24, '【男款 現貨即出】發熱衣 保暖衣 長袖 輕薄發熱衣 透氣保暖衣 輕摩厚發熱衣 順滑衣 衛生衣', 99, '/img/79d9d1fc435', '【男款 現貨即出】發熱衣 保暖衣 長袖 輕薄發熱衣 透氣保暖衣 輕摩厚發熱衣 順滑衣 衛生衣。')

```

```

INSERT INTO `Product` VALUES (306, '【文具王子】現貨！手牌SDI I-PUSH輕鬆按修正內帶.替換內帶.補充帶 (10入裝)', 195, '/img/6346156480e426859b5ec6c1a16c7fb8', '規格:5mm')
INSERT INTO `Product` VALUES (307, '【文具王子】現貨！手牌SDI I-PUSH輕鬆按修正內帶.替換內帶.補充帶 (10入裝)', 195, '/img/6346156480e426859b5ec6c1a16c7fb8', '規格:5mm')
INSERT INTO `Product` VALUES (308, '【優悅小舖】國曆Kokuyo一米新純筆袋簡約純色帆布可立筆袋可手拉化妝品收納包', 359, '/img/6346156480e426859b5ec6c1a16c7fb8', '帆布封頭')
INSERT INTO `Product` VALUES (309, '【銀河文具坊】成功 21326A 強力磁鐵 強力磁石 強力吸鐵 酒杯型 工字型 裸裝', 8, '/img/6346156480e426859b5ec6c1a16c7fb8', '材質：壓克力')
INSERT INTO `Product` VALUES (310, '柒柒的小舖 角落生物裝尸裝殼學生獎品 角落小夥伴三角板18cm直尺量角器學生文具禮物', 15, '/img/6346156480e426859b5ec6c1a16c7fb8', '產品')
INSERT INTO `Product` VALUES (311, '【好的家居】台灣現貨貨炒出,秒開置物架推車/收納層架,免安裝層架,置物架,波浪架,置物櫃,收納架,折疊置物架', 650, '/img/5d6424a8c010')
INSERT INTO `Product` VALUES (312, '現貨免運抽屜簡易衣櫃加粗19管衣櫃布藝鋼架大號布衣櫃 150寬大人加大衣櫃經濟型 鋼管加粗加固學生宿舍出租屋簡易衣櫃', 1200, '/img/5d6424a8c010')
INSERT INTO `Product` VALUES (313, '【免運】【年底大促】年前可收貨多功能可折疊簡約懶人沙發床小戶型客廳雙人坐臥兩用科技布可儲物', 12000, '/img/5d6424a8c010b25c0ba72d7974', '現貨')
INSERT INTO `Product` VALUES (314, '【本賣場銷售冠軍】現代沙發 | 免運費 | 台灣現貨 | 快速出貨 | 可拆洗 | 單人沙發 | 雙人沙發 | 小家庭沙發 | 6000', '/img/5d6424a8c010b25c0ba72d7974c0b6')
INSERT INTO `Product` VALUES (315, '日式防塵組合衣櫃 衣櫃diy魔片 簡易收納衣櫃 衣櫃 組合衣櫃 收納衣櫃 DIY衣櫃 居家收納組合櫃 多層組合衣櫃衣櫥', 2000, '/img/5d6424a8c010')
INSERT INTO `Product` VALUES (316, '魔方五一多功能收納椅(四色)折疊收納椅 /多功能/收納凳/魔方椅/台灣現貨/沙發凳/組合椅/儲物箱/組合凳/居家/魔方凳', 2380, '/img/5d6424a8c010')
INSERT INTO `Product` VALUES (317, '台灣現貨 實木貓墻壁掛式 貓窩 貓爬架 墻壁式 跳台 跳板 木質 墻上 貓家具白色 Sm3v', 2500, '/img/5d6424a8c010b25c0ba72d7974f67d21', '材質：實木')
INSERT INTO `Product` VALUES (318, '金屬底座家具裝飾摆件輕奢水晶球底座', 150, '/img/5d6424a8c010b25c0ba72d7974f67d21', '品牌：夢馨緣水晶 材質：金屬材質 颜色：金色')
INSERT INTO `Product` VALUES (319, '25mm品質護角三面直角金屬包角木盒邊角禮盒裝飾鐵質防撞角', 10, '/img/5d6424a8c010b25c0ba72d7974f67d21', '賣場現貨下標2天之內寄出')
INSERT INTO `Product` VALUES (320, '(插心) 家具雙輪 (辦公椅腳輪) 【禁昌信】辦公椅 家具輪 桌腳輪 桌腳雙輪 腳輪 鋼軸輪 辦公椅輪', 30, '/img/5d6424a8c010b25c0b')
INSERT INTO `Product` VALUES (321, '板橋區家具 T309-5A 卡米白橡7尺高低櫃，大台北都會區免運費', 13300, '/img/5d6424a8c010b25c0ba72d7974f67d21', '此賣場為【T309-5A】下標')
INSERT INTO `Product` VALUES (322, '【新款】M8螺桿可調節鋁合金家具腳輪腳輪沙發腳底座 家具腳輪腳輪洗衣機腳', 200, '/img/5d6424a8c010b25c0ba72d7974f67d21', '親愛的水水/大大')
INSERT INTO `Product` VALUES (323, '~誠築生活家具~雕花玻璃早餐杯 茶杯 水杯 牛奶杯 杯具', 150, '/img/5d6424a8c010b25c0ba72d7974f67d21', '商品名稱:雕花玻璃早餐杯 (如遇缺貨請見說明欄) ')
INSERT INTO `Product` VALUES (324, '【米朵Miduo】三門三抽塑鋼碗盤櫃 防水塑鋼家具 塑鋼櫈櫃', 6750, '/img/5d6424a8c010b25c0ba72d7974f67d21', '免組裝家具，不需要再自行DIY')
INSERT INTO `Product` VALUES (325, '【逸雅傢俱】木芯貼皮板/夾板', 3000, '/img/5d6424a8c010b25c0ba72d7974f67d21', '木芯板 厚度：1.8mm(六分) 外層一夾板貼皮。內層一實木板')
INSERT INTO `Product` VALUES (326, '【米朵Miduo】四門四抽塑鋼碗盤櫃 防水塑鋼家具 櫃櫃', 8750, '/img/5d6424a8c010b25c0ba72d7974f67d21', '免組裝家具，不需要再自行DIY')
INSERT INTO `Product` VALUES (327, 'Muji 無印良品 壁掛家具 轉角架 胡桃木 全新品', 860, '/img/5d6424a8c010b25c0ba72d7974f67d21', '全新品！ Muji 無印良品 壁掛家具 轉角架')
INSERT INTO `Product` VALUES (328, '日本安心君 (家具防震板) 耐震7級/60~120公分', 550, '/img/5d6424a8c010b25c0ba72d7974f67d21', '置於傢俱底部，免施工，免螺絲，可裁切，可DIY')
INSERT INTO `Product` VALUES (329, 'DIY手工 沙盤模型 模型材料 面剖戶型 迷你傢俱 歐式梳妝檯1比25', 48, '/img/5d6424a8c010b25c0ba72d7974f67d21', '#建築模型材料 #沙盤模型')
INSERT INTO `Product` VALUES (330, '【AKALI】家易淨100ml 長效除臭 家具去味 裝潢去甲醛 銀離子 淨化空氣 奈米光觸媒 去除異味 分解異味 除菌', 239, '/img/5d6424a8c010b25c0b')

```

```

INSERT INTO `ShoppingCart` VALUES (1, 8, 5);
INSERT INTO `ShoppingCart` VALUES (1, 7, 10);

```

```

INSERT INTO `OrderDetail` VALUES (1, 1, 1);
INSERT INTO `OrderDetail` VALUES (1, 2, 1);
INSERT INTO `OrderDetail` VALUES (1, 3, 1);
INSERT INTO `OrderDetail` VALUES (4, 1, 3);
INSERT INTO `OrderDetail` VALUES (4, 2, 2);
INSERT INTO `OrderDetail` VALUES (5, 1, 3);
INSERT INTO `OrderDetail` VALUES (5, 2, 2);
INSERT INTO `OrderDetail` VALUES (6, 1, 3);
INSERT INTO `OrderDetail` VALUES (6, 2, 2);
INSERT INTO `OrderDetail` VALUES (7, 4, 10);
INSERT INTO `OrderDetail` VALUES (7, 5, 10);
INSERT INTO `OrderDetail` VALUES (8, 1, 3);
INSERT INTO `OrderDetail` VALUES (8, 2, 2);
INSERT INTO `OrderDetail` VALUES (9, 7, 10);
INSERT INTO `OrderDetail` VALUES (9, 8, 10);

```

```

INSERT INTO `Order` VALUES (1, 1, '2021-12-08 16:48:33', 5020, 2, 1);
INSERT INTO `Order` VALUES (2, 2, '2021-12-04 11:10:10', 50, 1, 1);
INSERT INTO `Order` VALUES (3, 2, '2021-12-04 11:10:10', 50, 1, 1);
INSERT INTO `Order` VALUES (4, 2, '2021-12-04 11:10:10', 50, 1, 1);
INSERT INTO `Order` VALUES (5, 2, '2021-12-04 11:10:10', 50, 1, 1);
INSERT INTO `Order` VALUES (6, 2, '2021-12-04 11:10:10', 50, 1, 1);
INSERT INTO `Order` VALUES (7, 1, '2021-12-04 11:10:10', 200, 1, 1);
INSERT INTO `Order` VALUES (8, 1, '2021-12-04 11:10:10', 50, 3, 1);
INSERT INTO `Order` VALUES (9, 1, '2021-12-04 11:10:10', 200, 3, 1);

```

```

INSERT INTO `Type` VALUES (1, '水果');
INSERT INTO `Type` VALUES (2, '食物');
INSERT INTO `Type` VALUES (3, '3c');
INSERT INTO `Type` VALUES (4, '衣物');
INSERT INTO `Type` VALUES (5, '日用品');
INSERT INTO `Type` VALUES (6, '飲料');
INSERT INTO `Type` VALUES (7, '其他');
INSERT INTO `Type` VALUES (8, '文具');
INSERT INTO `Type` VALUES (9, '家具');

```

```

INSERT INTO `OrderStatus` VALUES (1, '訂單完成');
INSERT INTO `OrderStatus` VALUES (2, '訂單取消');
INSERT INTO `OrderStatus` VALUES (3, '訂單成立');

```

```

INSERT INTO `CreditCard` VALUES ('1234567896543217', 2, 2024, 12, '123');
INSERT INTO `CreditCard` VALUES ('1234567899876543', 1, 2025, 8, '456');

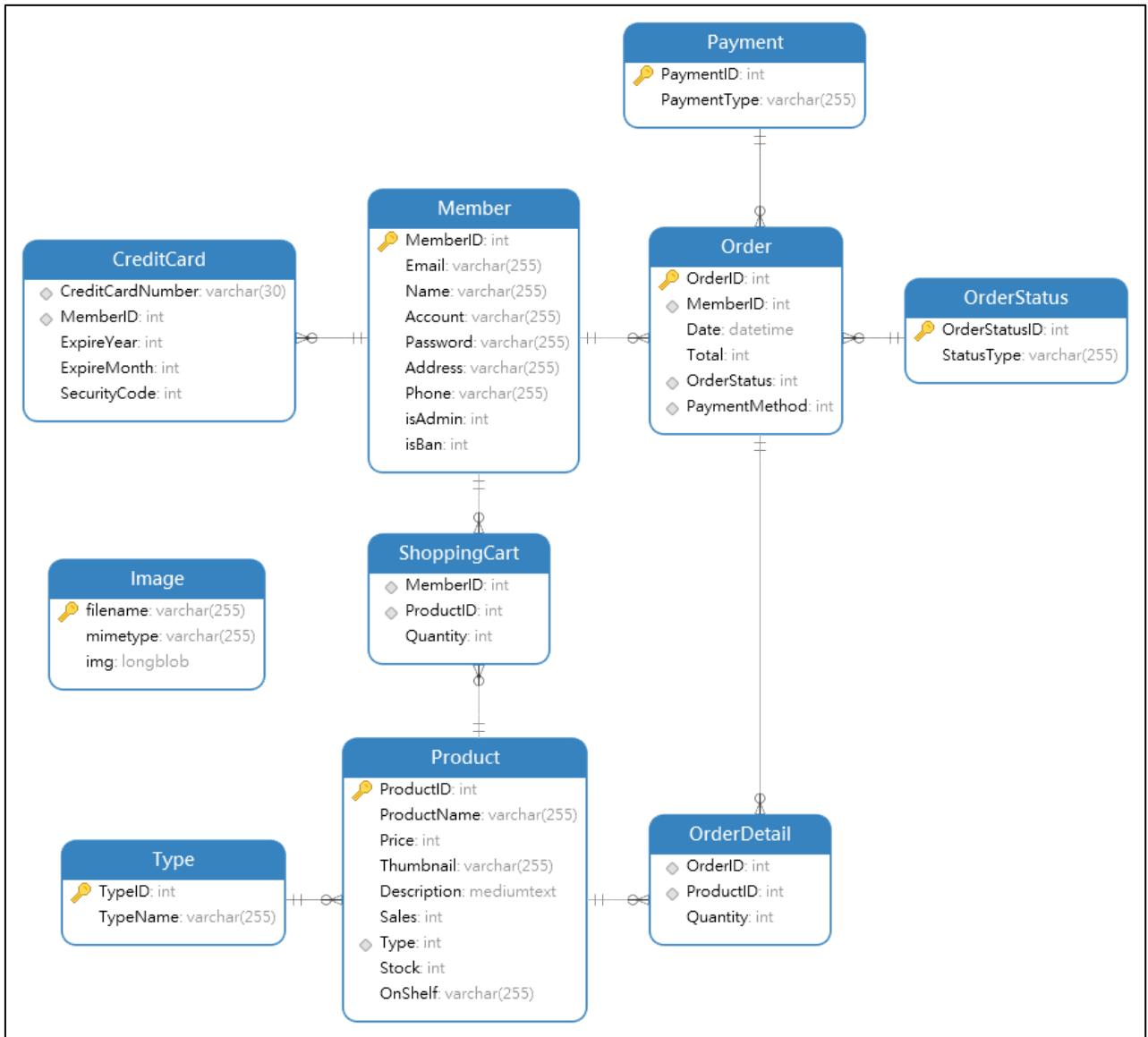
```

```

INSERT INTO `Image` VALUES ('0041608c11590fba63fdff255266e4e4', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708151512181515141212112121919121
INSERT INTO `Image` VALUES ('01c81fb285f2c8b161cc89f048ae70a', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0608151513171515131817191A1C1C1A1
INSERT INTO `Image` VALUES ('05a6c0df2441bf451073061627cc5df2', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0608151513171515131717191A1C1A1A1
INSERT INTO `Image` VALUES ('06859e285dd517cebb467672e40971da', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000906071313121513131516151718191A181
INSERT INTO `Image` VALUES ('08aaad2d2f9fc9372bb9a81ec9f3630e', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000906071212121012101010151500F0F151
INSERT INTO `Image` VALUES ('0b8bf4d0fc2967315a8735a2784a038', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708151615181615151819181A1818181A1
INSERT INTO `Image` VALUES ('0d8895464891fbee762bcc5aa106f5b', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A07081515141714141817181B211A1B1
INSERT INTO `Image` VALUES ('0d8895464891fbee762bcc5aa106f5b', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A07081615151812151519181D1812181
INSERT INTO `Image` VALUES ('0f0f72aad23c00cf1bea40e4286e8d7d7', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A070816151512151618181A191A1
INSERT INTO `Image` VALUES ('1126f7b0157d6c05bfff9a238a595ff7f', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A070815161518151518191818181A1A1
INSERT INTO `Image` VALUES ('153fa9b3e0122bd94b4374ff1fc5e35', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708161517161618181A19211C1C1
INSERT INTO `Image` VALUES ('15a6b7cfc550e909bffa91561dd196ac', 'image/jpeg', 0xFFD8FFE000104A464946000102000064000FFEC001144756367900010004000000290000FFEE000E4
INSERT INTO `Image` VALUES ('18ac5bbbeb69f9c3209573802be547696', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A060714141316131413161619191A1A1
INSERT INTO `Image` VALUES ('18e746c2ab41feabfea2237e0fza0034', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB008400090607131312151313161516191F19181
INSERT INTO `Image` VALUES ('18f55df5401afacce9efcb6d3612e91a', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708111211120F1110101011111111010111
INSERT INTO `Image` VALUES ('1c311b1908c828709e81093360ffd9df', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A060813151317131515181717191A1A19181
INSERT INTO `Image` VALUES ('207ee16elbc2d63879a175789a38044e', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A060714141316131413171716181818181
INSERT INTO `Image` VALUES ('2204890950adb6007d8fa215ab3c2a5c', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708121212111121212121111111
INSERT INTO `Image` VALUES ('227fc27d3ccf7af78d51ba85656857b', 'image/jpeg', 0xFFD8FFE000104A46494600010101000060000FFDB0043000A07079097060A09080908080A0C0F19100F0
INSERT INTO `Image` VALUES ('22e2a4774ec1cc0c3ffba43f29408a28', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0608131513171315151817171A1B191A1
INSERT INTO `Image` VALUES ('232518ecbe16b9911ea7c6c6d2f415b7', 'image/jpeg', 0xFFD8FFE000104A464946000102000064000FFEC001144756367900010004000000290000FFEE000E4
INSERT INTO `Image` VALUES ('248c263caf1434b06b6fb05c624447e', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000A0708151514181515141819181A1B1D1D1A1
INSERT INTO `Image` VALUES ('24c9d0fc9b81569f2fe9cebb2963b4b9', 'image/jpeg', 0xFFD8FFE000104A4649460001010000100010000FFDB0084000906071312121513121516151718181B181

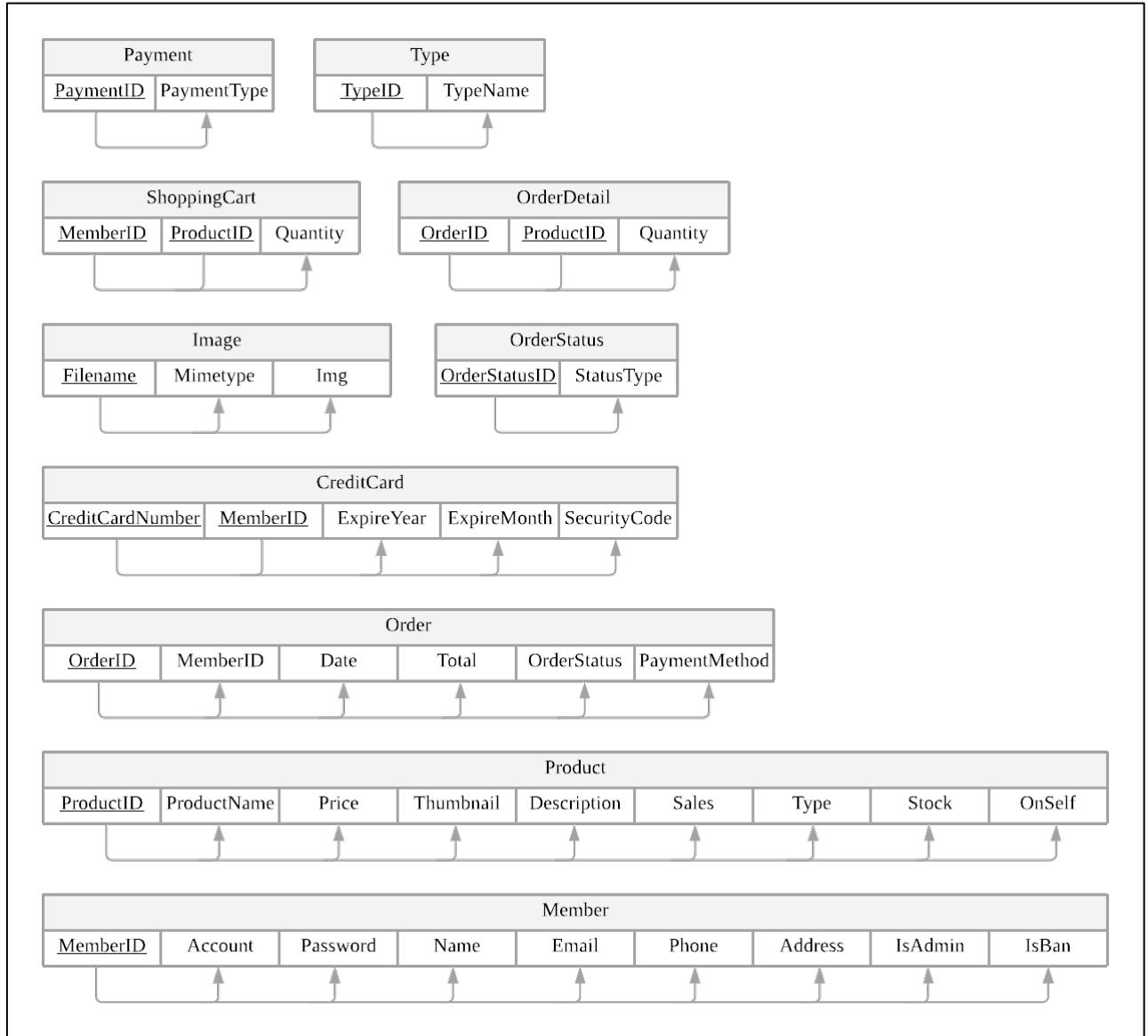
```

4.5 The implementation of tables in target DBMS



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

5.1 功能性依賴 (Functional Dependencies)

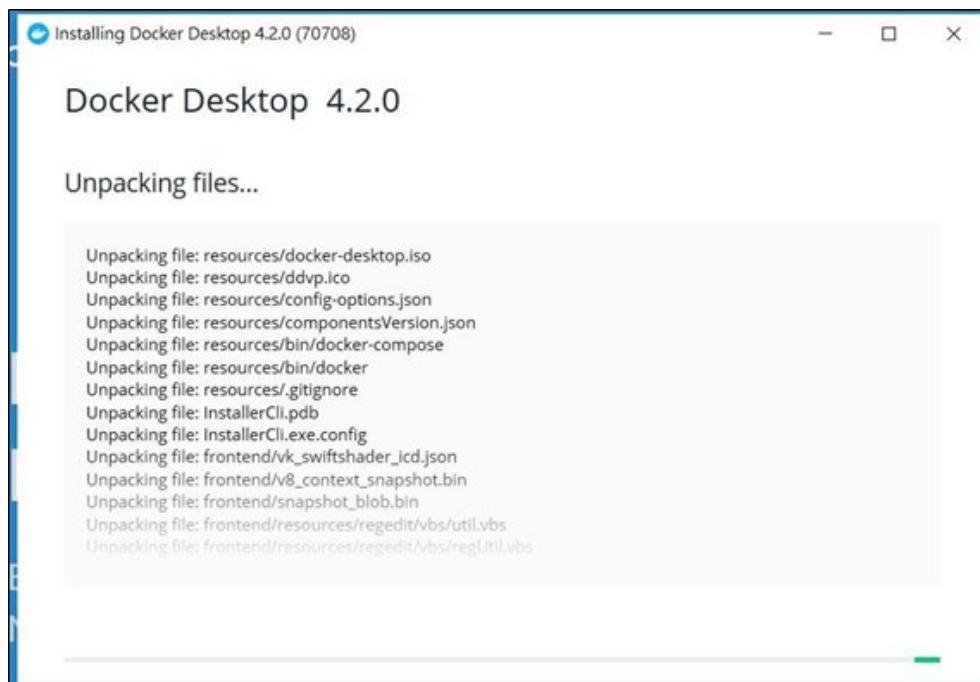


Section 6 數據庫系統的使用(The Use of the Database System)

6.1 系統安裝 (System Installation Description)

安裝 Docker 4.2.0

[Docker for Windows release notes | Docker Documentation](#)



下載 Linux 核心更新套件

[較舊版本的 WSL 手動安裝步驟 | Microsoft Docs](#)



將 wsl 2 設定為預設版本，打開 PowerShell 輸入 wsl --set-default-version 2



```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。
請嘗試新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Users\user> wsl --set-default-version 2
有關 WSL 2 的主要差異詳細資訊，請瀏覽 https://aka.ms/wsl2
操作順利完成。
PS C:\Users\user>
```

安裝 Navicat

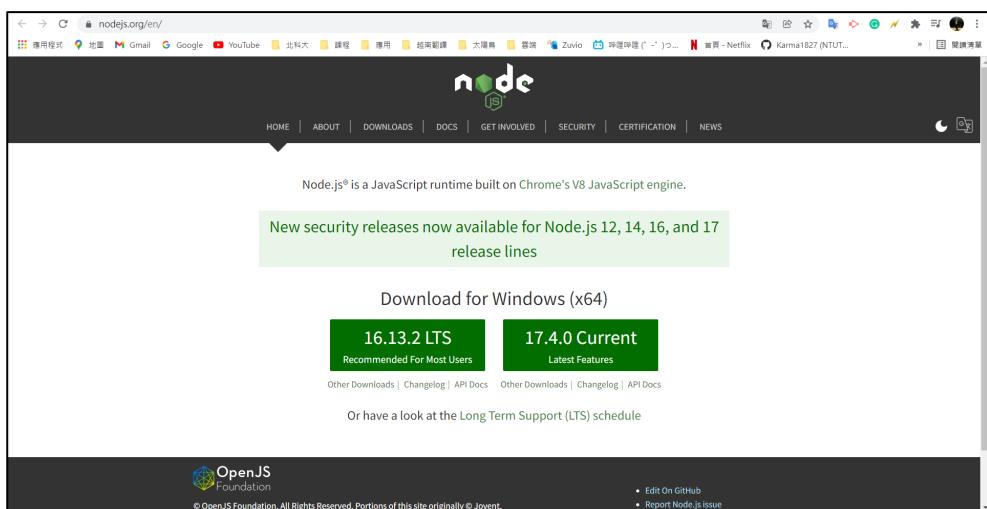
下載 Navicat，<https://www.navicat.com/cht/download/navicat-premium>



安裝和註冊 Navicat 教程 <https://kknews.cc/code/x4nymer.html>

安裝 Nodejs

下載 nodejs，<https://nodejs.org/en/>



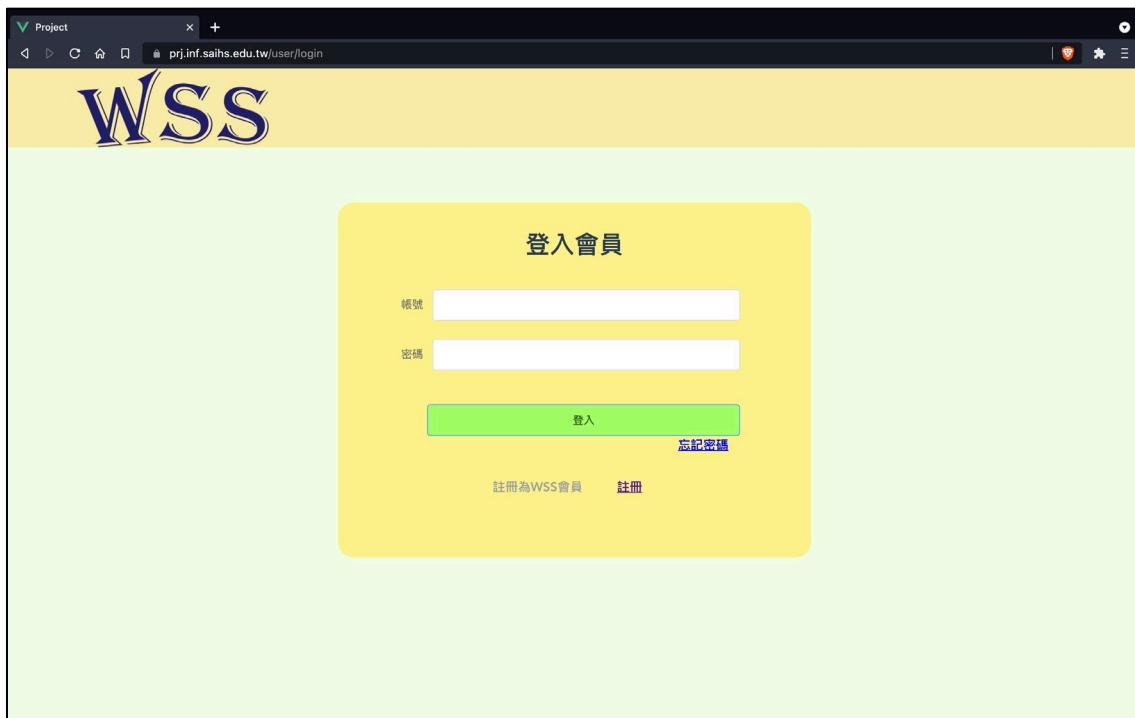
安裝 nodejs 教程 <https://seanacnet.com/node-js/node-js-express-insert/>

FrontEnd & BackEnd：請參考放在雲端 team09 的 code 資料夾裡的 README

6.2 系統使用 (The Use of the System)

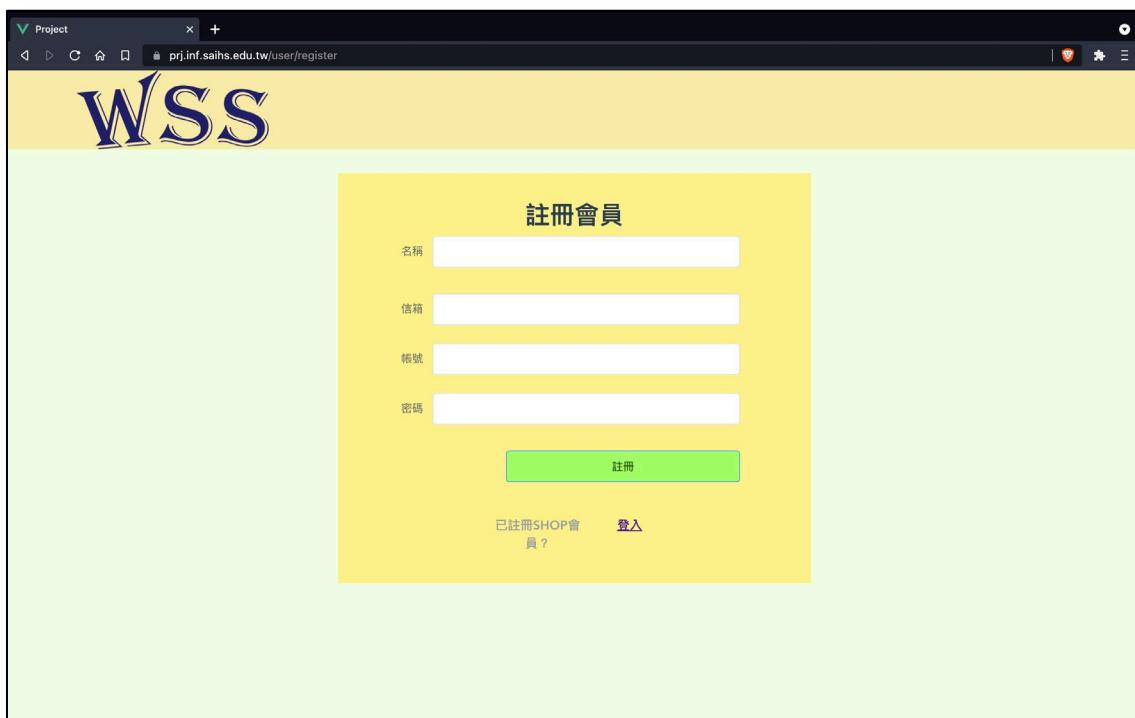
6.2.1 使用者頁面

a) 登入畫面



The screenshot shows a web browser window with the URL `prj.inf.saihs.edu.tw/user/login`. The page has a yellow header with the letters "WSS". Below the header is a yellow rectangular form titled "登入會員". It contains two input fields: "帳號" and "密碼", both with placeholder text. Below these fields is a green "登入" button. To the right of the button is a blue link "忘記密碼". At the bottom of the form are two links: "註冊為WSS會員" and "註冊".

b) 忘記密碼

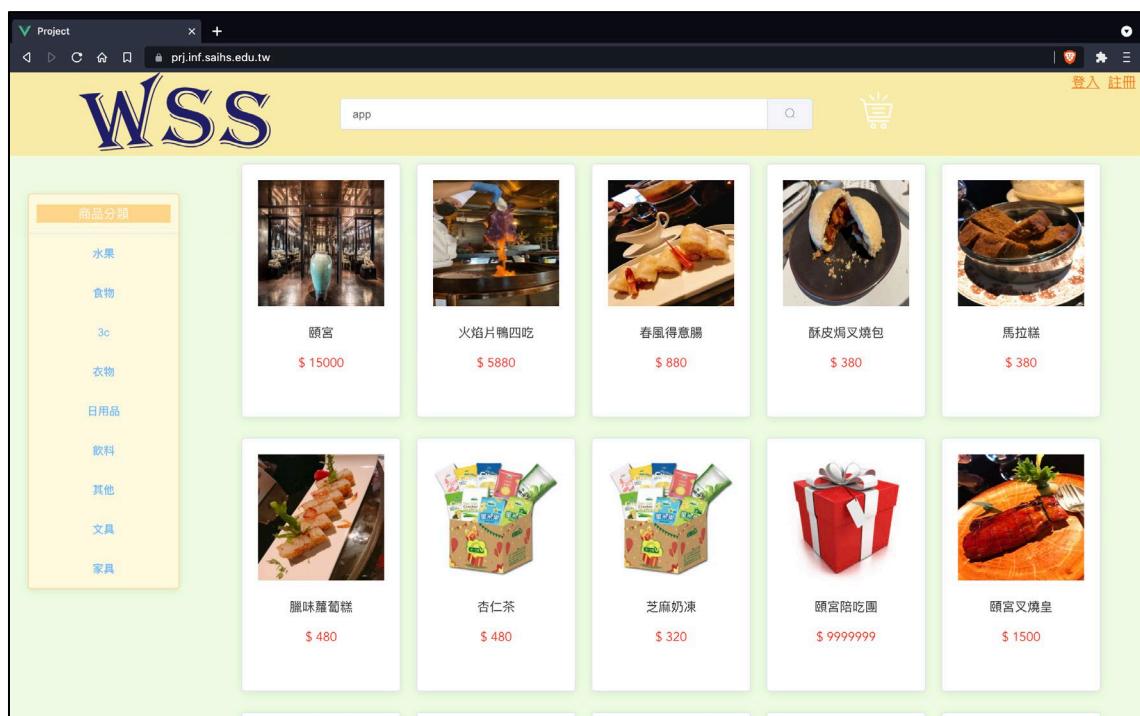


The screenshot shows a web browser window with the URL `prj.inf.saihs.edu.tw/user/register`. The page has a yellow header with the letters "WSS". Below the header is a yellow rectangular form titled "註冊會員". It contains four input fields: "名稱", "信箱", "帳號", and "密碼", all with placeholder text. Below these fields is a green "註冊" button. At the bottom of the form are two links: "已註冊SHOP會員？" and "登入".

c)註冊畫面

The screenshot shows a registration form titled "註冊會員" (Register Member). It includes four input fields: "名稱" (Name), "信箱" (Email), "帳號" (Account), and "密碼" (Password). Below the fields is a green "註冊" (Register) button. At the bottom of the form, there are links for "已註冊SHOP會員？" (Already registered as a SHOP member?) and "登入" (Log in).

d)未登入時的首頁



e)使用者登入後首頁

The screenshot shows a user interface for a food delivery or ordering system. At the top, there's a navigation bar with a search bar containing "Please Search Product" and a shopping cart icon. On the right, it says "Hi andy 登出 會員中心". A sidebar on the left lists categories: 水果 (Fruit), 食物 (Food), 3c, 衣物 (Clothing), 日用品 (Household), 飲料 (Drinks), 其他 (Others), 文具 (Stationery), and 家具 (Furniture). The main area displays a grid of 10 product cards:

商品	價格
頤宮	\$ 15000
火焰片鴨四吃	\$ 5880
春風得意腸	\$ 880
酥皮焗叉燒包	\$ 380
馬拉糕	\$ 380
臘味蘿蔔糕	\$ 480
杏仁茶	\$ 480
芝麻奶凍	\$ 320
頤宮陪吃圓	\$ 9999999
頤宮叉燒皇	\$ 1500

f)管理員登入時的首頁

This screenshot shows the same user interface as above, but from the perspective of an administrator. It features the same layout with a sidebar and a grid of 10 product cards. The products listed are identical to those in the user version, with their respective prices.

g) 使用搜尋欄位查詢商品

The screenshot shows a search results page for the term "app". The results are displayed in a grid format. The first item is an iPhone 13 Pro Max with a price of \$25900. The second item is a "快幣" (Fast Coin) with a price of \$1. The third item is a bottle of perfume with a price of \$159. The fourth item is an Apple Watch with a price of \$249. The fifth item is another iPhone 13 Pro Max with a price of \$32900. Below these are three more items: a phone case for \$77, a Taiwan-made original company product for \$495, and a pineapple for \$100.

h) 搜尋"水果"分類中價格由低到高，且在 10 元~150 元之間的商品

The screenshot shows a search results page for the category "水果" (Fruit), sorted by price from low to high between \$10 and \$150. The results are displayed in a grid format. The items shown are: Sweet Banana (\$10), Fuji Apple (\$15), Kiwi (\$100), grape (\$100), pineapple (\$100), lemon (\$100), Chayote (\$100), and Chirimoya (\$150).

i)商品頁面

請點擊此處

春風得意腸

\$ 880

數量 0 還剩下9個

加入購物車

腸粉是廣州街頭再尋常不過的一款小吃，在廣州，每到早晨腸粉店都供不應求，也成「搶粉」。腸粉傳統以碎肉、魚肉、鮮蝦為餡料，「白如雪，薄如紙，油光閃亮，香滑可口」。就連賜名達人乾隆品嘗過後也要欣然賜名。

j)購物車頁面

商品圖片	商品名稱	單價	購買數量	小計	操作
	酥皮焗叉燒包	\$ 380	- 1 +	\$ 380	
	富士山蘋果	\$ 15	- 5 +	\$ 75	

共 6 件商品，已選擇 5 件

合計: 75元

買單

k)結帳頁面

The screenshot shows a payment page for WSS. At the top right, it says "Hi andy 登出 會員中心". The main area displays an order for 5 Fuji apples at \$15 each, totaling \$75. Below the order summary, there's a section for shipping information with fields for name, phone number, and address, all filled with "andy", "0912345678", and "Taipei". There are tabs for payment methods: "貨到付款" (selected), "信用卡", and "修改". Below the payment method, there are two radio buttons: "貨到付款" (selected) and "貨到付款". The total cost is listed as \$80 for shipping and \$155 元 (total). A green "結帳" button is at the bottom.

1)會員中心 - 我的檔案

The screenshot shows the member center profile page for "andy". On the left, there are four teal buttons: "我的檔案" (selected), "信用卡", "交易紀錄", and "修改密碼". The main content area shows a table with personal information:

我的個人資料	
姓名	andy
電話	0912345678
信箱	t108590001@ntut.org.tw
住址	Taipei

A blue "修改" button is located at the top right of the profile table.

m)會員中心 - 信用卡

The screenshot shows the 'Credit Card' section of the WSS Member Center. On the left, there is a sidebar with four teal-colored buttons: '我的檔案', '信用卡' (selected), '交易紀錄', and '修改密碼'. The main content area has a yellow header bar with the WSS logo and a search bar. Below the header, the title '我的信用卡' is displayed, followed by a table showing a single credit card entry: Card Number 1234678901234890, Year 2023, Month 05. A red 'Delete' button is located at the bottom right of the table.

n)會員中心-交易紀錄

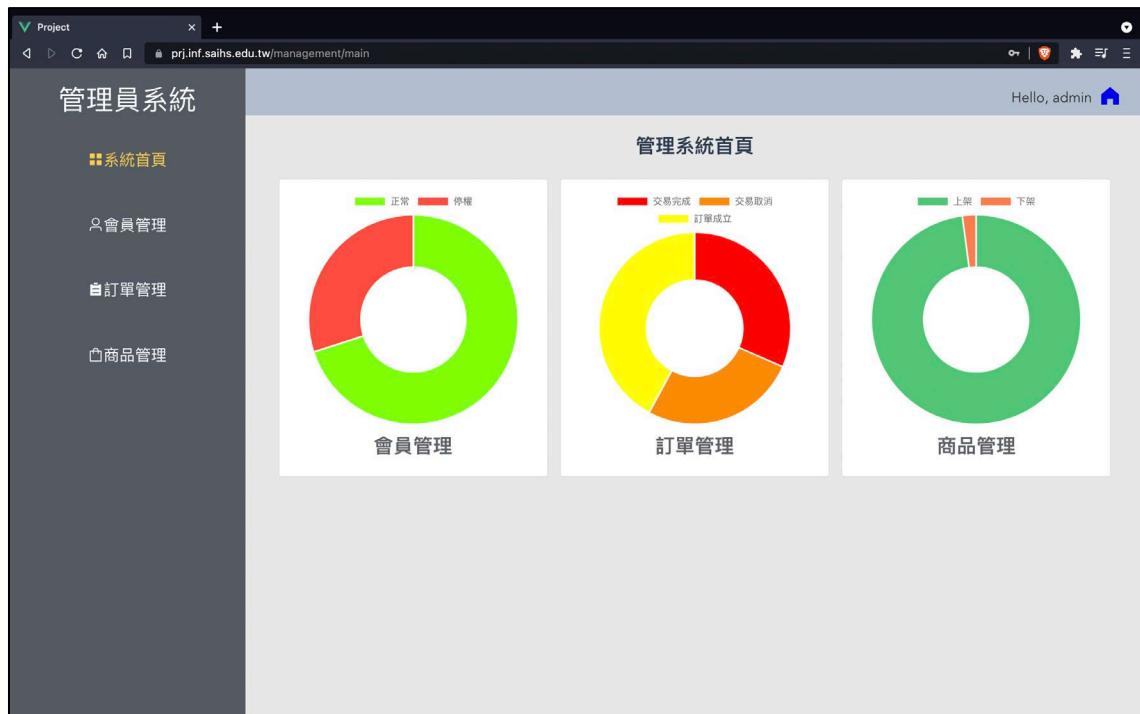
The screenshot shows the 'Order History' section of the WSS Member Center. The sidebar on the left includes '我的檔案', '信用卡', '交易紀錄' (selected), and '修改密碼'. The main content area features a yellow header bar with the WSS logo and a search bar. The title '我的訂單紀錄' is shown above a table of order details. The table has columns: 訂單編號 (Order ID), 下單日期 (Order Date), 總金額 (Total Amount), and 訂單狀態 (Order Status). One row is visible: Order ID 22, Order Date 2022-01-19T10:03:33.000Z, Total Amount 135000, and Order Status 訂單取消 (Order Canceled). Below the table, there is a breakdown of the total amount: 項目 (Item), 單價 (Unit Price), 數量 (Quantity), and 小計 (Subtotal), with a total of \$ 135000.

o)會員中心 - 修改密碼

The screenshot shows the 'Member Center' page of the WSS system. At the top right, it says 'Hi andy 登出 會員中心'. On the left, there's a sidebar with four teal buttons: '我的檔案', '信用卡', '交易紀錄', and '修改密碼'. The main area has a yellow header bar with a search bar 'Please Search Product' and a shopping cart icon. Below the header is a white box titled '修改密碼' containing three input fields: '原來密碼', '新密碼', and '再次輸入', followed by a blue '修改' button.

6.2.2 管理員頁面

a)管理員系統-首頁



b) 管理員系統 - 會員管理

The screenshot shows the 'Member Management' page of the Admin System. The left sidebar has a dark theme with white text. The '會員管理' (Member Management) option is selected and highlighted in yellow. The main content area has a light gray background. At the top right, it says 'Hello, admin' with a blue house icon. Below that is a search bar labeled 'Search'. The main table has columns: 會員ID (Member ID), 會員名稱 (Member Name), 帳號 (Account), 信箱 (Email), 是否為管理員 (Is Admin), 狀態 (Status), and 操作 (Actions). There are 7 rows of data, each with a unique ID, name, account, email, admin status (green for yes, red for no), status (green for normal, red for suspended), and an edit button. At the bottom, there's a pagination bar showing 'Total 8' and a page number '1'.

會員ID	會員名稱	帳號	信箱	是否為管理員	狀態	操作
1	admin	admin	123	是	正常	編輯
2	123	tim1207	123@gmail.com	是	停權	編輯
5	test123	test123	test123@gmail.com	否	停權	編輯
6	karma	karma	karma@gmail.com	否	停權	編輯
7	test456	test456	twm0989256926@gmail.com	否	正常	編輯

c) 管理員系統 - 訂單管理

The screenshot shows the 'Order Management' page of the Admin System. The left sidebar has a dark theme with white text. The '訂單管理' (Order Management) option is selected and highlighted in yellow. The main content area has a light gray background. At the top right, it says 'Hello, admin' with a blue house icon. Below that is a search bar labeled 'Search'. The main table has columns: 訂單編號 (Order ID), 會員ID (Member ID), 日期 (Date), 總金額 (Total Amount), 付款方式 (Payment Method), 狀態 (Status), and 操作 (Actions). There are 5 rows of data, each with a unique ID, member ID, date, amount, payment method, status (green for completed, red for canceled), and an edit button. At the bottom, there's a pagination bar showing 'Total 17' and a page number '1'.

訂單編號	會員ID	日期	總金額	付款方式	狀態	操作
1	1	2022-01-11T23:38:49.000Z	\$ 5020	信用卡	交易完成	編輯
2	1	2022-01-12T00:05:48.000Z	\$ 50	信用卡	交易取消	編輯
3	1	2022-01-11T20:56:47.000Z	\$ 50	信用卡	交易完成	編輯
4	1	2022-01-11T20:56:49.000Z	\$ 50	信用卡	交易完成	編輯
5	1	2022-01-11T23:19:09.000Z	\$ 50	信用卡	交易取消	編輯

d) 會員系統 - 商品管理

The screenshot shows the 'Product Management' page of the Admin System. The left sidebar has a dark theme with yellow highlights for '商品管理'. The main area has a light blue header '商品管理'. Below it, there are two tabs: '選擇商品' and '新增/編輯商品', with '新增/編輯商品' being active. A search bar with placeholder 'Search' is at the top right. The main content area displays a table of products:

商品ID	商品名稱	商品圖片	商品價格	商品庫存	在架上	操作
1	甜蜜香蕉		10	96	是	編輯 下架
2	富士山蘋果		15	4	是	編輯 下架
3	Airpods		5000	13	是	編輯 下架

Total 332 < 1 2 3 4 5 6 7 > Go to 1

e) 會員管理 - 新增商品

The screenshot shows the 'New Product' form in the Admin System. The left sidebar has a dark theme with yellow highlights for '商品管理'. The main area has a light blue header '商品管理'. Below it, there are two tabs: '選擇商品' and '新增/編輯商品', with '新增/編輯商品' being active. The form fields are:

- 商品名稱:
- 商品價格: NT\$
- 剩餘數量:
- 商品類型:

Below the form is a rich text editor toolbar with icons for Normal, Bold, Italic, Underline, etc. At the bottom are '取消' (Cancel) and '儲存' (Save) buttons.

商品詳情頁面模擬

f) 會員管理 - 編輯商品

The screenshot displays the 'Management System' interface. On the left is a dark sidebar with navigation links: 系統首頁, 會員管理, 訂單管理, and **商品管理**. The main area is titled '商品管理' (Product Management) and shows a sub-section '編輯商品' (Edit Product). It features a preview image of three bananas, input fields for '商品名稱' (Product Name: 甜蜜香蕉), '商品價格' (Product Price: NT\$ 10), and '剩餘數量' (Remaining Quantity: 96). Below the form is a rich text editor toolbar and a descriptive text block about organic bananas. At the bottom are '取消' (Cancel) and '儲存' (Save) buttons.

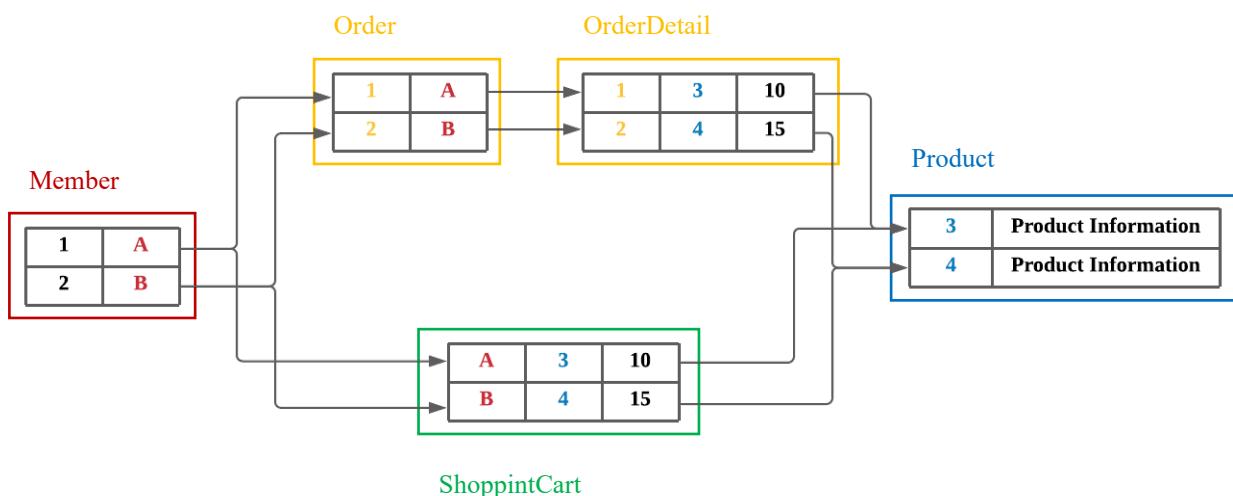
The second screenshot shows a product detail page for '甜蜜香蕉'. It includes a large image of the bananas, the product name '甜蜜香蕉' at the top, the price '\$ 10' in red, a quantity selector set to '0' with a maximum of '96個' (96 pieces), and a green '加入購物車' (Add to Cart) button. A small descriptive text block is also present.

Section 7 資料庫優化建議(Suggestions of Database Turning)

在這次的專題實作當中，對於系統效能、細節以及資料庫優化；我們組員討論過後，發現還有一些可以改善的地方，以下列出三點：

1. 將系統讀寫分離：可以利用讀寫分離技術增加機器的處理能力及效率。
2. 新增 index：老師在課程後半段有所提到，因為搜尋時需要查找較多資料，建立 index 可以有效提高在查詢時的效率，例如本系統在查詢 order，product，shoppingCart 等大量資料的資料庫時，會因資料量過大導致整體效能降低，利用新增 index 的方式能夠克服此困難。

EX: Member 可以透過 shoppingCart 和 order 去查詢 product 資訊



3. 使用 Queue 隊列排程：老師上課也有提到 Queue 排程任務來避免請求阻塞，可有效解決查詢資料時過於龐大造成請求時間過長。

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

8.1 附加查詢 (Additional Queries)

上傳圖片

```
const uploadImage = (file, host) => {
    return new Promise((resolve, reject) => {
        const filename = customAlphabet("0123456789abcdef", 32)()
        query("INSERT INTO `Image` values(?, ?, ?)", [filename, file.mimetype, file.buffer]).then((result) => {
            resolve({
                code: 200,
                message: "上傳成功",
                imageUrl: `http://${host}/img/${filename}`
            });
        }).catch((error) => {
            reject(error);
        })
    })
};
```

取得所有交易清單

```
const getOrderList = (page) => {
    return new Promise((resolve, reject) => {
        if(page === undefined || page === "")
            page = 1
        const dataPerPage = 50
        const minLimit = (Number(page) - 1) * dataPerPage ;
        query('SELECT COUNT(*) as _count FROM `Order` ').then((result)=>{
            const total = Number(result[0]._count);
            const pages = Math.ceil(total / dataPerPage);
            query('SELECT * FROM `Order` LIMIT ?,?', [minLimit, dataPerPage]).then((result) => {
                resolve({
                    result,
                    total,
                    pages,
                });
            }).catch((error) => {reject(error);});
        }).catch((error) => {reject(error);});
    });
};
```

修改訂單狀態

```
const modifyOrder = (orderId, status) => {
    return new Promise((resolve, reject) => {
        if(status === "finish"){
            query('UPDATE `Order` SET OrderStatus = 1 WHERE OrderID = ?', [orderId]).then(() => {
                resolve({
                    code: 200,
                    message: '訂單交易成功',
                });
            }).catch((error) => {reject(error);})
        }else if(status === "cancel"){
            query('UPDATE `Order` SET OrderStatus = 2 WHERE OrderID = ? AND OrderStatus != 2', [orderId]).then(
            (result) => {
                if(result.affectedRows > 0 ){
                    query('UPDATE Product,OrderDetail SET Product.Stock = Product.Stock + OrderDetail.Quantity
                    WHERE Product.ProductID = OrderDetail.ProductID AND OrderID = ?', [orderId])
                    .then(() =>{
                        resolve({
                            code: 200,
                            message: '訂單取消成功',
                        });
                    })
                }else{
                    reject(error.APIError("訂單取消失敗", new Error()));
                }
            }).catch((error) => {reject(error);})
        }else{
            reject(error.APIError("訂單操作失敗", new Error()));
        }
    });
};
```

查看所有訂單狀態

```
const getAllOrderStatus = () => {
    return new Promise((resolve,reject) => {
        query("SELECT StatusType,COUNT(StatusType) AS 'total' FROM `Order` LEFT JOIN OrderStatus ON `Order`.OrderStatus = OrderStatus.OrderStatusID GROUP BY StatusType ")
        .then((result) => {
            resolve(result);
        }).catch((error) => {
            reject(error);
        });
    });
}
```

上架商品

```
const addProduct = (product) => {
    return new Promise((resolve,reject) => {
        query('INSERT INTO `Product`(`ProductName`, `Price`, `Thumbnail`, `Description`, `Sales`, `Type`, `Stock`, [product.name, product.price, product.thumbnail, product.description, 0, product.type, product.stock, "Yes"])
        .then(()=>{
            resolve({
                code: 200,
                message: "商品上架成功",
            })
        }).catch((error) => {reject(error);})
    });
};
```

下架商品和重新上架商品

```
const operateProduct = (productId, onShelf) => {
    return new Promise((resolve,reject) => {
        if(onShelf === undefined)
            reject(error.APIError("商品操作失敗", new Error()));
        const shelfStatus = onShelf ? "Yes" : "No";

        query('UPDATE `Product` SET OnShelf = ? WHERE ProductID = ?', [shelfStatus,productId]).then(() => {
            resolve({
                code: 200,
                message: onShelf ? "商品重新上架成功" : "商品下架成功",
            })
        }).catch((error) => {reject(error);})
    });
};
```

修改商品資料

```
const modifyProductData = (id, values) => {
    return new Promise((resolve,reject) => {
        query('UPDATE `Product` SET Price = ?, Thumbnail = ?, Description = ?, Stock = ? WHERE ProductID = ?', [values.price, values.thumbnail, values.description, values.stock, id]).then(() => {
            resolve({
                code: 200,
                message: "商品資訊更新成功",
            })
        }).catch((error) => {reject(error);})
    });
};
```

取得所有商品狀態

```
const getAllProductStatus = () =>{
    return new Promise((resolve,reject) => {
        query("SELECT OnShelf,COUNT(OnShelf) AS 'total' FROM Product GROUP BY OnShelf").then((result) => {
            resolve(result);
        }).catch((error) => {
            reject(error);
        });
    });
};
```

取得所有商品資料

```
const getAllProduct = (page) => {
    return new Promise((resolve,reject) => {
        if(page === undefined || page === "") {
            page = 1
        }
        const dataPerPage = 50;
        const minLimit = (Number(page) - 1) * dataPerPage;
        query("SELECT COUNT(*) AS COUNT FROM Product").then((result) => {
            const total = Number(result[0].COUNT);
            const pages = Math.ceil(total / dataPerPage);
            query("SELECT ProductID,ProductName,Price,Thumbnail,Stock,OnShelf FROM Product limit ?,?", [minLimit, da
                resolve({
                    result,
                    total,
                    pages
                })
            }).catch((error) => {
                reject(error);
            });
        }).catch((error) => {
            reject(error);
        });
    });
};
```

列出所有使用者清單

```
const listUser = (page) => {
    return new Promise((resolve,reject) => {
        if(page === undefined || page === "") {
            page = 1
        }
        const dataPerPage = 50;
        const minLimit=(Number(page) - 1) * dataPerPage
        query('SELECT COUNT(*) as _count FROM Member ').then((result)=>{
            const total = Number(result[0]._count);
            const pages = Math.ceil(total / dataPerPage);
            query('SELECT MemberID,Account,Name,Email,isAdmin,isBan FROM Member  LIMIT ?,?', [minLimit,dataPerPage]
                resolve({
                    result,
                    total,
                    pages,
                });
            }).catch((error) => {reject(error);});
        }).catch((error) => {reject(error);});
    });
};
```

修改使用者的權限和狀態

```
const modifyUserStatus = (userId, operate) => {
    return new Promise((resolve,reject) => {
        const expectColumns = ["isAdmin", "isBan"];
        const params = []
        const operateColumns = []
        expectColumns.forEach(column => {
            if(operate[column] !== undefined){
                operateColumns.push(` ${column} = ?`);
                params.push(operate[column] ? 1 : 0);
            }
        });
        params.push(userId);
        const sql = `UPDATE Member SET ${operateColumns.join(",")} WHERE MemberID = ?`;
        query(sql, params).then(() => {
            resolve({
                code: 200,
                message: "操作成功"
            });
        }).catch((error) => {reject(error);});
    });
};
```

查看所有使用者的停權狀況

```
const getAllUserStatus = () => {
  return new Promise((resolve, reject) => {
    query("SELECT isBan,COUNT(*) AS 'total' FROM Member GROUP BY isBan").then((result) => {
      resolve({
        unban: result[0].total,
        ban: result[1].total,
      })
    }).catch((error) => {
      reject(error);
    })
  });
}
```

查看購物車

```
const getCart = (user) => [
  return new Promise((resolve, reject) => {
    query("SELECT ProductID, ProductName, Thumbnail, Price, Stock, Quantity FROM ShoppingCart NATURAL JOIN Product WHERE ShoppingCart.MemberID = ? ", [user.id]).then((result) => {
      resolve(result);
    }).catch((error) => {
      reject(error);
    })
  });
]
```

放商品進購物車

```
const putProduct = (user, values) => {
  return new Promise((resolve, reject) => {
    query('SELECT Stock FROM Product WHERE ProductID = ?', [values.productID]).then((result) => {
      let stock;
      if(result.length > 0 ){
        stock = Number(result[0].Stock);
      }else{
        reject(error.APIError("這是一個不存在的商品", new Error()));
      }

      if (stock >= values.quantity) {
        query("SELECT * FROM ShoppingCart WHERE MemberID = ? AND ProductID = ?", [user.id, values.productID]).then((result) => {
          if (result.length === 0) {
            query('INSERT INTO `ShoppingCart` (`MemberID`, `ProductID`, `Quantity`) VALUES (?, ?, ?)', [user.id, values.productID, values.quantity]).then(() => {
              resolve({
                code: 200,
                message: "放置購物車成功",
              });
            }).catch((error) => {
              reject(error);
            })
          } else {
            query('UPDATE `ShoppingCart` SET Quantity = Quantity + ? WHERE MemberID = ? AND ProductID = ?', [values.quantity, user.id, values.productID]).then(() => {
              resolve({
                code: 200,
                message: "商品數量增加成功",
              });
            }).catch((error) => {
              reject(error);
            })
          }
        }).catch((error) => {
          reject(error);
        })
      } else {
        reject(error.APIError("商品庫存不足", new Error()));
      }
    }).catch((error) => {
      reject(error);
    });
  });
}
```

從購物車移除商品

```
const removeProduct = (user, values) => {
  return new Promise((resolve, reject) => {
    query('DELETE FROM ShoppingCart WHERE MemberID = ? and ProductID =?', [user.id, values.productId]).then(
      (result) => {
        resolve({
          code: 200,
          message: "移除成功",
        });
      }).catch((error) => {
        reject(error);
      });
  });
}
```

修改購物車的商品數量

```
const modifyProductQuantity = (user, values) => {
  return new Promise((resolve, reject) => {
    query('SELECT Stock FROM Product WHERE ProductID = ?', [values.productId]).then((result) => {
      let stock;
      if(result.length > 0){
        stock = Number(result[0].Stock);
      }else{
        reject(error.APIError("這是一個不存在的商品", new Error()));
      }
      query("SELECT * FROM ShoppingCart WHERE MemberID = ? AND ProductID = ?", [user.id, values.productId]).then((result) => {
        if (result.length === 0) {
          reject(error.APIError("購物車內並無此商品", new Error()));
        } else {
          if (stock >= values.quantity) {
            query('UPDATE `ShoppingCart` SET Quantity = ? WHERE MemberID = ? AND ProductID = ? ', [values.quantity, user.id, values.productId]).then((result) => {
              resolve({
                code: 200,
                message: "商品數量更改成功",
              });
            }).catch((error) => {
              reject(error);
            });
          } else {
            reject(error.APIError("商品庫存不足", new Error()));
          }
        }
      }).catch((error) => {
        reject(error);
      });
    });
  });
}
```

取得 Type ID 對應的 Type Name

```
const getCategories = (user) => {
  return new Promise((resolve,reject) => {
    query("SELECT TypeID AS id, TypeName as name FROM Type").then((result) => {
      resolve(result);
    }).catch((error) => {
      reject(error);
    })
  });
};
```

讀取圖片

```
const loadImg = (filename) => {
  return new Promise((resolve,reject) => {
    query("SELECT img,mimetype FROM `Image` WHERE filename =?", [filename]).then((result) => {
      if(result.length > 0)
        resolve(result[0]);
      reject(error.APIError("找不到圖片", new Error()));
    }).catch((error) => {
      reject(error);
    })
  });
};
```

查看訂單根據訂單編號

```
const getOrdersByID = (user, page) => {
  return new Promise((resolve, reject) => {
    if(page === undefined || page === "") {
      page = 1
    }
    const dataPerPage = 20
    const minLimit = (Number(page) - 1) * dataPerPage
    query('SELECT COUNT(*) as _count FROM `Order` WHERE MemberID = ?',[user.id]).then((result)=>{
      const total = Number(result[0]._count);
      const pages = Math.ceil(total / dataPerPage);
      query('SELECT OrderID,Date,Total,StatusType FROM `Order` LEFT JOIN `OrderStatus` ON `Order`. `OrderStatus` = `OrderStatus`. `OrderStatusID` WHERE MemberID = ? LIMIT ?,?', [user.id, minLimit, dataPerPage]).then((result) => {
        resolve([
          result,
          total,
          pages
        ]);
      }).catch((error) => {reject(error);});
    }).catch((error) => {reject(error);});
  });
};
```

建立訂單

```
const createOrder = (user, values) => {
  return new Promise(async (resolve, reject) => {
    let totalPrice = 0;
    for(let i = 0 ; i < values.products.length; i++){
      await query('SELECT Price FROM `Product` WHERE ProductID = ? AND OnShelf = "Yes"', values.products[i].productId).then((result) => {
        if(result.length > 0){
          totalPrice += result[0].Price * values.products[i].quantity
        }else{
          reject(error.APIError("建立訂單失敗", new Error()));
        }
      }).catch((error) => {reject(error);});
    }

    const now = date.format(new Date(), "YYYY/MM/DD hh:mm:ss")
    query(`INSERT INTO `Order` (`MemberID`, `Date`, `Total`, `OrderStatus`, `PaymentMethod`) VALUES (?, ?, ?, ?, ?)`, [user.id, now, totalPrice, 3, values.paymentMethod]).then((result) => {
      const orderId = result.insertId;
      let sql = `INSERT INTO `OrderDetail` (`OrderID`, `ProductID`, `Quantity`) values`;
      const parameterBracket = [];
      const parameters = [];
      values.products.forEach(element) => {
        parameterBracket.push("(?, ?, ?)");
        parameters.push(orderId, element.productId, element.quantity);
        query(`UPDATE `Product` SET Sales = Sales + ?, Stock = Stock - ? WHERE ProductID = ?`, [element.quantity, element.quantity, element.productId]);
      }
      query(sql+ parameterBracket.join(","),
        parameters).then((result) => {
          resolve({
            code: 200,
            message: "購買成功",
          });
        });
    }).catch((error) => {reject(error);});
  });
};
```

刪除訂單

```
const deleteOrder = (user, orderId) =>{
  return new Promise((resolve, reject) => {
    query('UPDATE `Order` SET OrderStatus = 2 WHERE OrderID = ? AND MemberID = ?', [orderId, user.id]).then(() => {
      query('UPDATE Product,OrderDetail \
      SET Product.Stock = Product.Stock + OrderDetail.Quantity, \
      Product.sales = Product.sales - OrderDetail.Quantity\
      WHERE Product.ProductID = OrderDetail.ProductID AND OrderID = ?',[orderId])
      .then(() =>{
        resolve({
          code: 200,
          message: '取消成功',
        });
      }).catch((error) => {reject(error);})
    }).catch((error) => {reject(error);})
  });
};
```

取得訂單的詳細資料

```
const getOrderDetail = (user, orderId) => {
  return new Promise((resolve, reject) => {
    query('SELECT Product.ProductName, o.Quantity, Product.Price FROM (SELECT ProductID,Quantity FROM `Order` LEFT JOIN OrderDetail ON `Order`.OrderID = OrderDetail.OrderID WHERE MemberID = ? AND `Order`.OrderID = ?) AS o LEFT JOIN Product ON o.ProductID = Product.ProductID', [user.id, orderId]).then((result) => {
      resolve(result);
    }).catch((error) => {reject(error);});
  });
}
```

根據分類列出商品

```
const getProductsByCategory = (categoryId, page, filter, sort, maxPrice, minPrice) => {
  return new Promise((resolve, reject) => {
    const filterOptions = [
      filter,
      sort,
      minPrice,
      maxPrice,
      page
    ];

    const dictionaryCondition = {
      expressions: ["Product.Type = ?","OnShelf = ?"],
      parameters:[categoryId,"YES"]
    };

    const getProductsSql = `SELECT Thumbnail,ProductName,Price,ProductID FROM Product`;
    productDatabase.filterProducts(getProductsSql, filterOptions, dictionaryCondition).then((result) => {
      resolve(result);
    }).catch((error) => {reject(error)});
  });
};
```

根據商品明在不同分類中搜尋

```
const searchCategoryProductByName = (productName, categoryId, page, filter, sort, maxPrice, minPrice) => {
  return new Promise((resolve, reject) => {
    const filterOptions = [
      filter,
      sort,
      minPrice,
      maxPrice,
      page
    ];
    const dictionaryCondition = {
      expressions: ["ProductName Like ?","OnShelf = ?","`Type` = ?"],
      parameters:[`%${productName}%`, "YES", categoryId]
    };

    const getProductsSql = 'SELECT Thumbnail,ProductName,Price,ProductID FROM Product';
    productDatabase.filterProducts(getProductsSql, filterOptions, dictionaryCondition).then((result) => {
      resolve(result);
    }).catch((error) => {reject(error)});
  });
};
```

取得商品的詳細資料

```
const getProductDetail = (productId) => {
  return new Promise((resolve, reject) => {
    query('SELECT Thumbnail ,ProductName,Price,Stock,Description FROM `Product` WHERE ProductID = ?', [productId])
      .then((result) => {
        if(result.length > 0){
          resolve(result[0]);
        }else{
          reject(error.APIError("查無此商品", new Error()));
        }
      }).catch((error) => {reject(error);});
  });
};
```

根據銷量列出商品

```
✓ const getProductsBySales = (page, sort, maxPrice, minPrice) => {           Identifier
✓   return new Promise((resolve,reject) => {
✓     const filterOptions={
✓       filter: "Sales",
✓       sort,
✓       minPrice,
✓       maxPrice,
✓       page
✓     }
✓
✓     const dictionaryCondition = {
✓       expressions: ["Onshelf = ?"],
✓       parameters:[ "YES"]
✓     }
✓
✓     const getProductsSql = 'SELECT Thumbnail,ProductName,Price,ProductID FROM Product';
✓     productDatabase.filterProducts(getProductsSql, filterOptions, dictionaryCondition).then((result) => {
✓       resolve(result);
✓     }).catch((error) => {reject(error)});
✓   });
✓});
```

根據分類列出類似此商品的數量

```
const countProductByCategory = (productName) => {
  return new Promise((resolve,reject) => {
    console.log(productName)
    query(`SELECT TypeName,count(*) as quantity FROM Product LEFT JOIN Type on Type.TypeID = TypeID WHERE ProductName LIKE '%${productName}%}`).then((result) => {
      resolve(result)
    }).catch((error) => {reject(error);})
  });
};
```

使用者登入

```
const Login = (values) => {
  return new Promise((resolve,reject) => {
    query("SELECT * FROM Member WHERE Account = ?", values.account).then((result) => {
      if(result.length > 0) {
        const queryPassword = result[0].Password;
        const userPassword = values.password;
        if (queryPassword === userPassword) {
          // 產生 JWT
          const payload = {
            user_id: result[0].MemberID,
            user_name: result[0].Name,
            user_mail: result[0].Email
          };
          const isAdmin = result[0].isAdmin ? true : false;
          // 取得 API Token
          const token = jwt.sign({ payload, exp: Math.floor(Date.now() / 1000) + (86400 * 365) },
            config.secretKey);
          resolve({
            code: 200,
            message: '登入成功',
            token,
            isAdmin
          });
        } else {
          reject(error.APIError("輸入的密碼有誤", new Error()));
        }
      } else{
        reject(error.APIError("輸入的帳號不存在", new Error()));
      }
    }).catch((error) => {reject(error);})
  });
};
```

根據名字列出商品

```
const searchAllProductByName = (productName, page, filter, sort, maxPrice, minPrice) => {
    return new Promise((resolve, reject) => [
        const filterOptions = {
            filter,
            sort,
            minPrice,
            maxPrice,
            page
        }
        const dictionaryCondition = { Variable Declaration
            expressions: ["ProductName Like ?","OnShelf = ?"],
            parameters:[`%${productName}%`,"YES"]
        }

        const getProductsSql = 'SELECT Thumbnail,ProductName,Price,ProductID FROM Product';
        productDatabase.filterProducts(getProductsSql, filterOptions, dictionaryCondition).then((result) => {
            resolve(result);
        }).catch((error) => {reject(error)}));
    ]);
};
```

重設密碼

```
const resetPassword = (value) => {
    return new Promise((resolve, reject) => {
        const newPassword = nanoid(16) Variable Declaration
        query("UPDATE Member SET Password = ? WHERE Email = ? AND Account = ?", [newPassword, value.email, value.account]).then((result) => {
            if(result.affectedRows === 0 ){
                reject(error.APIError("找不到此信箱或帳號", new Error()))
            }
            const mailOptions = {
                from: config.mailUser,
                to: value.email,
                subject: '密碼重置請求',
                text: `您好!\\n\\n這是您的新密碼:${newPassword}`
            };
            config.mailService.sendMail(mailOptions, (error, info) => {
                if(error){
                    reject(error);
                }
                resolve({
                    code: 200,
                    message: `已將新的密碼送至 ${value.email}`
                })
            });
        }).catch((error) => {
            reject(error)
        })
    });
};
```

使用者註冊

```
const Register = (values) => {
    return new Promise((resolve, reject) => {
        console.log(values);
        query(`SELECT * FROM Member WHERE Account = ? OR Email = ?`, [values.account, values.email]).then((result) => {
            if (Object.keys(result).length === 0) {
                query(`INSERT INTO `Member`(`Email`, `Account`, `Password`, `Name`) VALUES (?, ?, ?, ?)`, [values.email, values.account, values.password, values.name]).then((result) => {
                    const payload = {
                        user_id: result.insertId,
                        user_name: values.name,
                        user_mail: values.email
                    };
                    const token = jwt.sign({ payload, exp: Math.floor(Date.now() / 1000) + (86400 * 365) }, config.secretKey);

                    resolve({
                        code: 200,
                        message: '註冊成功',
                        token
                    });
                }).catch((error) => {reject(error);})
            } else {
                reject(error.APIError("註冊失敗", new Error()));
            }
        }).catch((error) => {reject(error);})
    });
};
```

增加信用卡

```
const addCreditCard = (user, credit) => {
  return new Promise((resolve, reject) => {
    query("SELECT * FROM `CreditCard` WHERE CreditCardNumber = ? AND SecurityCode = ?", [credit.cardNumber, credit.securityCode]).then((result) => {
      if (result.length === 0) {
        query('INSERT INTO `CreditCard`(`CreditCardNumber`, `MemberID`, `ExpireYear`, `ExpireMonth`, `SecurityCode`) VALUES (?, ?, ?, ?, ?)', [credit.cardNumber, user.id, credit.year, credit.month, credit.securityCode]).then(() => {
          resolve({
            code: 200,
            message: '新增成功',
          });
        }).catch((error) => {reject(error)});
      } else {
        reject(error.APIError("新增失敗", new Error()));
      }
    }).catch((error) => {reject(error);})
  });
}
```

查詢信用卡

```
const findCreditCard =(user,page)=>{
  return new Promise((resolve,reject) => [
    if(page === undefined || page === "") {
      page = 1
    }
    const dataPerPage = 50
    const minLimit = (Number(page) - 1) * dataPerPage
    query('SELECT CreditCardNumber,ExpireYear,ExpireMonth,SecurityCode FROM `CreditCard` WHERE `MemberID` = ? LIMIT ?, ?, [user.id, minLimit, dataPerPage]).then((result) => {
      resolve(result);
    }).catch((error) => {reject(error);});
  ])
}
```

刪除信用卡

```
const deleteCreditCard =(user,value)=>{
  return new Promise((resolve,reject) => {
    console.log (value.cardNumber);
    query('DELETE FROM `CreditCard` WHERE `MemberID` = ? AND `CreditCardNumber` = ? ', [user.id, value.cardNumber]).then((result) => [
      resolve({
        code: 200,
        message: '刪除成功',
      });
    ]).catch((error) => {reject(error);});
  })
}
```

取得個人資料

```
const getInformation = (user) => {
  return new Promise((resolve, reject) => {
    query("SELECT `Name`,Phone,Email,Address,isAdmin FROM Member WHERE MemberID = ? ", user.id).then((result) => [
      const response = result[0];
      response.isAdmin = response.isAdmin ? true : false;
      resolve(result[0]);
    ]).catch((error) => {
      reject(error);
    })
  });
}
```

修改個資

```
const modifyInformation = (user,value) =>{
  return new Promise((resolve,reject) => {
    query('UPDATE `Member` SET Email = ? , Name = ? ,Address = ? ,Phone = ? WHERE `MemberID` = ? ', [value.email, value.name, value.address, value.phone, user.id,]).then((result) => {
      resolve([
        code: 200,
        message: '修改成功',
      ]);
    }).catch((error) => {reject(error);});
  })
}
```

修改密碼

```
const modifyPassword = (user,value) =>{
    return new Promise((resolve,reject) => {
        query("SELECT * FROM `Member` WHERE MemberID = ? AND Password = ?", [user.id, value.oldPassword]).then(
            (result) => {
                if (result.length === 0) {
                    reject(error.APIError("舊密碼錯誤", new Error()));| Expression Statement
                } else {
                    query('UPDATE `Member` SET Password = ? WHERE MemberID = ?',
                        [value.newPassword,user.id]).then(() => {
                            resolve({
                                code: 200,
                                message: '修改成功',
                            });
                        }).catch((error) => {reject(error)});
                }
            }).catch((error) => {reject(error);})
    });
}
```

根據商品 ID 查看庫存

```
function checkStockByProductID(values) {
    return new Promise(async (resolve, reject) => {
        for(let i = 0; i < values.products.length; i++)[]|
            await query('SELECT Stock FROM Product WHERE ProductID = ? AND OnShelf = "Yes"', [values.products[i].productId]).then((result) => {
                if(result.length === 0)
                    reject(error.APIError("查無此商品的庫存", new Error()));

                if (values.products[i].quantity > result[0].Stock) {
                    resolve(false);
                }
            }).catch((error) => {
                reject(error);
            });
    }
    resolve(true);
})
}
```

根據使用者 ID 查看權限

```
function checkAdminByUserID(userID){
    return new Promise((resolve, reject) => {
        query('SELECT isAdmin FROM Member WHERE MemberID = ?', userID).then((result) => {
            if(result.length > 0){
                if(result[0].isAdmin === 1){
                    resolve(true)
                }
                else
                    resolve(false)
            }
        }).catch((error) => {reject(error);})
    })
}
```

根據使用者 ID 查看是否停權

```
function checkBanByUserID(userID){ Identifier
    return new Promise((resolve, reject) => {
        query('SELECT isBan FROM Member WHERE MemberID = ?', userID).then((result) => {
            if(result.length > 0){
                if(result[0].isBan === 1){
                    resolve(true)
                }
                else
                    resolve(false)
            }
        }).catch((error) => {reject(error);})
    })
}
```

8.2 查詢展示 (Views)

將商品的 Type 擬成所對應的類別名字 並整合列出

```

1 • CREATE VIEW project3 AS SELECT
2     t.ProductName,t.TypeName,t.Price,t.Description FROM
3     (SELECT * FROM Product NATURAL LEFT JOIN Type WHERE Type.TypeID=Product.Type) as t;
4
5 • select * from project3;

```

ProductName	TypeName	Price	Description
甜蜜香蕉	水果	10	有機香蕉採用台蕉五號及新北蕉品種，搭配...
富士山蘋果	水果	10	由「國光」與「Delicious」二種品種交配而...
IG高品質台灣粉 ig台灣粉 終身保固IG粉絲 In...	水果	1	good
大湖草莓	水果	100	苗栗大湖草莓栽種面積達五百多公頃，面積...
紐西蘭奇異果	水果	100	紐西蘭奇異果的原生種其實來自中國，當地...
grape	水果	100	果實風味甜美，供鮮食、釀酒或製果乾，釀...
pineapple	水果	100	鳳梨是多年生草本果樹，種植後三年內可以...
lemon	水果	100	檸檬富含維他命C及枸櫞酸，用途極廣，可...
超俗辣芭樂	水果	100	台灣的芭樂，有點甜又不會太甜，甘、濃的...
太陽鳥鳳梨	水果	100	鳳梨原產於熱帶南美洲，鳳梨是熱帶地區極...
釋迦	水果	150	果葉排列整齊，果實鱗目大突起，果實硬熟...
巧克力	食物	100	巧克力是以可可漿和可可脂為主要原料製成...
哇沙米脆豌豆 151g	食物	57	巧克力是以可可漿和可可脂為主要原料製成...
衛龍辣條 魔芋爽 4元 親嘴燒 3元 小麵筋 魔...	食物	3	爆款商品～解饑抗餓.....好吃不怕胖（每小...
衛龍魔芋爽 風吃海帶 親嘴燒 麵筋【手機批...	食物	3	魔芋爽獨特的麻辣口味，爽滑Q彈的口感，...
日本mdc左旋肉鹼小綠袋纖體丸 48回增量版 ...	食物	420	宿便的危害：色斑、痤瘡、皺紋、口苦、口...
海底撈【白者火鍋（麻辣嫩牛/番茄牛肉）】	食物	199	操作方便只要有水，立即享用美味的海底撈...

將所有會員的交易紀錄細項以 view 列出呈現

```

1 • CREATE VIEW project2 AS SELECT
2     c.Name,c.OrderID,c.Date,c.Quantity,c.ProductName FROM
3     (SELECT * FROM
4     (SELECT * FROM
5         (SELECT * FROM Member NATURAL LEFT JOIN `Order` WHERE Member.MemberID= `Order`.MemberID)  as t
6     NATURAL LEFT JOIN OrderDetail WHERE t.OrderID= OrderDetail.OrderID)  as t
7     NATURAL LEFT JOIN Product WHERE t.ProductID= Product.ProductID) as c;
8 • select * from project2;

```

	Name	OrderID	Date	Quantity	ProductName
▶	admin	1	2021-12-08 16:48:33	1	甜蜜香蕉
	admin	1	2021-12-08 16:48:33	1	富士山蘋果
	admin	1	2021-12-08 16:48:33	1	Airpods
	123	4	2021-12-04 11:10:10	3	甜蜜香蕉
	123	4	2021-12-04 11:10:10	2	富士山蘋果
	123	5	2021-12-04 11:10:10	3	甜蜜香蕉
	123	5	2021-12-04 11:10:10	2	富士山蘋果
	123	6	2021-12-04 11:10:10	3	甜蜜香蕉
	123	6	2021-12-04 11:10:10	2	富士山蘋果
	admin	7	2021-12-14 04:08:18	10	巧克力
	admin	7	2021-12-14 04:08:18	10	哇沙米脆豌豆 151g
	admin	11	2021-12-04 11:10:10	3	甜蜜香蕉
	admin	11	2021-12-04 11:10:10	2	富士山蘋果

Section 9 結論與未來規劃(Conclusions and Future Work)

9.1 結論 (Conclusions)

李文至

我在上這門課之前，就已經對資料庫有簡單的瞭解了，但這學期上完之後我學到了許多之前都不瞭解的東西，不論是課堂上教的還是做 project 時學到的，也加增了許多資料庫的知識，像是我原本對於資料庫的只有簡單的一對一、一對多、多對多，在應用上只有使用一對多跟 Primary key 被參考以後不可以被刪除而已，但經過這學期上課以後，我才知道原來只要在創建表時加入一些指令之後，就可以讓他自動跟著 Primary key 作變更(編輯、刪除)等、在做 Project 的時候我是負責寫前端的人之一，我們使用的 vue 框架，我沒有寫過這個框架的語法，在這次寫 Project 的時候，我學會了使用這個框架來寫網頁前端，也學會如何使用 node.js 串接後端 api，並將資料顯示在網頁上面，我覺得這次做 project 的感覺很棒，因為大家都在互相討論，如果有人有不會的地方，問其他人大家都會互相幫忙，當自己會的地方，別人不會的都會不吝嗇地去教對方，如果兩個人都不會的話，就會一起上網找資料，一起學會，這種互相討論、協助共同做出 project 是大家的心血，共同完成一個 project 的感覺真的很棒。

陸永強

暑假在做 Sunbird 計畫時，有聽學長姐常常提到資料庫和我當 driver 也有時在學長姐的指導下操作資料庫與其相關程式碼，但那時還不太了解資料庫。經過這學期的上課，讓我更了解資料庫系統。我覺得老師課程安排很不錯，光是聽概念肯定沒辦法理解，於是老師讓我們分組做專題，從專題中學到更多東西。大家一開始討論專題主題、系統架構圖、設計 ER model、然後 schema 是以 ER 圖去做設計。但期中報告時，老師給了我們很多回饋去做更改，例如：要先有 schema 再去設計 ER model、如何設計系統架構圖、在設計 ER model 時要以使用者的方面去考慮，不要參雜太多實作的部分。

我在這次專題負責的部分是後端，從建資料庫到寫 API。雖然我程式方面沒有到很強，但經過暑假有在 Sunbird 練習，在這次的專寫程式方面很明顯有提升，也可以跟組員們一起討論程式碼（之前我都是聆聽比較多）。寫完每隻 API 過後，我們都會使用 PostMan 去測試看我們的邏輯對不對。經過一整個學期的合作與努力，我們完成了線上商城系統，雖然介面有點陽春~但是我們一起做出來的專題，我很喜歡也很享受跟大家一起討論開發的過程。謝謝各位組員，辛苦了~

黃明萱

這學期在資料庫系統課堂上學到了很多，原來寫一個資料庫前還要設計 Entity-relationship model 然後 schema 是以 ER 圖去寫相互的屬性，還又最基礎的 MySQL 建立資料庫的語法與用法和規則，還有很多該注意的細節，如果沒有就不行等等…的。

專題小組裡也學到不少，我們組是先分前端與後端小組，每組都會有小組長去盯，而我是前端的組員，網頁部份我也是因為這次課堂要寫出一個網站，不然我現在可能還

不會寫，用了很多 Vue.js 的架構，也學到一些基礎簡單的網頁設計的寫法，雖然也是很多都是網路上參考或是與組員討論，在進行修改，但寫出來就覺得很不錯，也學到 API 的接法，才知道原來一個網頁介面要好看真的不容易，要一直去修改排版，又要實用。

林聖祐

在這次的資料庫專題中，我在小組內所扮演的角色是系統架構師，整個專案的架構都是由我進行規劃，在專案開發的過程中我十分要求 Code 的品質，並且前後端的分層也有稍微規劃，讓組員們依照我所制定的規則去 Coding，因此最後的 Code 我認為理解起來不會太困難，如果有其他人要找到專案內前端的某個元件或是後端的某支 API 應該都相當容易。在這次的專案中我還大量使用了 Git，讓大家在不同的分支上開發，最後再由我 merge 到主要分支。

這次是我第一次在一個小組內擔任系統架構師的角色，對於系統規劃、前後端分層也沒有甚麼明確的概念，這些部份我都是依照著自身經驗、網路資源及詢問朋友來一步一步調整，因此在調整的時候難免會與先前規劃互相衝突。由於我對自己實作的標準比較高，所以這次專題成品在我心目中只有勉強的及格分數，我原先預想這個電商平台能夠串接第三方登入、金流、驗證碼等，但礙於組員內大家都沒有這種實務開發的經驗，導致開發進度緩慢，最後只能砍到剩下最基本的功能，雖說開發進度很緩慢，但我也沒有打算苛責他們，多數組員都非常積極的學習，幾乎每天都有組員傳 Line 來求助我開發上遇到的困難。

張景辰

這次的資料庫，說實話對我來說，我有一點挫折，有些東西上課聽過之後，還是有點不瞭解其中的原理，導致我花了不少的時間，在嘗試著理解之前的東西。而在專題的實作上，我也必須對我的夥伴們說聲抱歉，我知道我並沒有對整個團隊做出太大的貢獻，因為我自身能力的不足，以至於他們必須幫我收拾殘局。但是我在這門課當中，也有學習到一個好的網站，背後是透過非常多的資料庫以及 API 去做整合，才能讓這個系統能夠在眾多的網站當中脫穎而出。我也期許自己能夠在之後的路途上，持續的精進自己，讓自己有朝一日，也能夠打造出一個所謂「好」的系統。

王裕詮

在資料庫系統這門課學到了很多，像是寫資料庫前還需要設計 Entity-relationship model 與 schema，他們互相的關係，什麼是需要的甚麼是不需要的，在 MySQL 也學到了很多新的語法及用法，還有正規化，可以避免在對資料庫操作的時候，會產生的問題，實際寫出一個完整的網站有前端與後端。

這次小組專題，我們分組就是以前端後端去分組，我是前端的，之前有修網頁設計課多少會一些，之前都只是單純用 html 去寫，這次多了 Vue.js 的架構，讓整個網頁更好看，寫了很多網頁上的功能，有時還是會卡住，但多上網查一下試一下就解決了，一直試成功寫得非常的愉快，雖然小組員的人都說我設計的很醜，真是太難過了，也學到

API 怎麼接上，真是收穫滿滿。

宋宇倫

我是一名外系的學生，自己本來在電子係是分組在計算機工程也比較擅長於一些控制韌體的程式，但是在我的領域有些專業跟資工系有重疊，因此決定在本學期嘗試選這門課，我想要藉由這門課去熟悉整個軟體的架構，再從自己本科韌體端的系統去擴展到軟體端。

在課堂專題中，因為對軟體的不熟悉以及我自己個人的事物在時間分配上沒有做到很好，我認為我自己沒有跟夥伴們做到很好的連結，不過我的專題夥伴還是做出了非常好的作品，如果給自己評分的話肯定是不能及格的，不過我在這個過程中也因為組員的努力讓我學到非常多，我開始熟悉什麼叫做軟體工程什麼叫網頁・前端後端資料庫，也非常努力地去瞭解組員們的 code 希望自己能趕快趕上他們，但是大家真的都很快或許我花的時間還不夠，總之我還是非常高興自己修了這麼課，現在也已經在應用這些知識開始拓展自己的專題，或許結果來看自己的表現真的不是很好，不過我還是非常感謝組員及老師的付出及教導，自己真的學習到了不少東西。

林峻霆

這學期的資料庫讓我學到很多，前端、後端、資料庫的應用等等。一開始什麼也不懂，只能上網找資料從頭學起。我主要和組員一起討論如何畫 ER model 和 Functional dependences，負責的是後端的部分，讓我了解到 api 的撰寫，以及資料庫和 SQL 的應用，現在也駕輕就熟了。當後端都完成了所有 api 的部分，我有去幫忙前端接 api，從中與前端的組員討論也學到一點前端的東西。很感謝這學期有其他組員們和我一起完成這個專題，不然一個人去做不但無法討論，甚至停滯不前。說真的我很喜歡這種可以一起做專題的感覺。

9.2 未來規劃 (Future Work)

在未來展望的部分，我們的首要目標會是先將介面優化，把介面設計的更簡潔更人性化，讓使用者有更佳的使用體驗。另外我們想在之後能新增一些折價優惠卷以及優惠活動來吸引消費者的購買，並且在消費者的訂單完成之後，可以對該商品留下一個簡短的評價，以便於其他消費者在日後購買該產品時能夠有一個參考的依據。同時在管理員的後台端，我們會整理出一個產品的銷售報表，讓管理員能夠清楚地看到產品的銷售情況，包括所有產品的銷售總金額，以及個別產品的銷售趨勢圖，這樣可以讓管理員針對個別產品做出相對應的處理。最後為了讓我們的網頁更加的貼近生活，我們會將系統設計成讓新加入的使用者可以透過第三方平台(例如 Google、Facebook、Line 等)去做快速註冊會員的動作。我們希望這個系統並不會只侷限於這堂課，而是在日後可以實際的應用在我們的日常生活當中。

Glossary

DBMS

資料庫管理系統(Database Management System)是一種操縱和管理資料庫的大型軟體，用於建立、使用和維護資料庫，簡稱 DBMS。它對資料庫進行統一的管理和控制，以保證資料庫的安全性和完整性。用戶通過 DBMS 訪問資料庫中的數據，管理員也通過 DBMS 進行資料庫的維護工作。它可使多個應用程式和用戶用不同的方法在同時或不同時刻去建立，修改和詢問。

Docker

Docker 是一種軟體平台，可讓您快速地建立、測試和部署應用程式。Docker 將軟體封裝到名為容器的標準化單位，其中包含程式庫、系統工具、程式碼和執行時間等執行軟體所需的所有項目。使用 Docker，您可以將應用程式快速地部署到各種環境並加以擴展，而且知道程式碼可以執行。

Element plus

功能 Element 因為能夠透過 Matrix 將其他通訊橋接至應用程式上而聞名，如 IRC、Slack、Telegram 與其他通訊軟體。同時也整合了透過 WebRTC 傳輸的點對點音訊與視訊聊天及群組聊天。由於可以自行建立應用程式與其背後的聊天伺服器，所以 Element 常被倡導隱私權的人推薦。

MySQL

MySQL 資料庫系統是關聯式資料庫管理系統(relational database management system)RDBMS。MySQL 的基本運作，MySQL 是如何處理來自使用者或應用程式的請求、如何處理資料修改、不同的 isolation，以及他們與資料庫 ACID (Atomicity, Consistency, Isolation, Durability) 特性之間的關係。

Navicat Premium

Navicat Premium 是一套資料庫開發工具，讓你從單一應用程式中同時連接 MySQL、MariaDB、MongoDB、SQL Server、Oracle、PostgreSQL 和 SQLite 資料庫。它與 Amazon RDS、Amazon Aurora、Amazon Redshift、Microsoft Azure、Oracle Cloud、Google Cloud 和 MongoDB Atlas 等雲端資料庫相容。你可以快速輕鬆地建立、管理和維護資料庫。

Node.js

Node.js 可透過 JavaScript 和一系列模組來編寫伺服器端應用和網路相關的應用。核心模組包括檔案系統 I/O、網路 (HTTP、TCP、UDP、DNS、TLS/SSL 等)、二進位資料流、加密演算法、資料流等等。Node 模組的 API 形式簡單，降低了編程的複雜度。使用框架可以加速開發。

Postman

Postman 是一個可以模擬 HTTP Request 的工具，其中包含常見的 HTTP 的請求方式，例如： GET 、POST 、PUT 、DELETE ，而它的主要功能就是能夠快速的測試你的 API 是否能夠正常的請求資料，並得到正確的請求結果。除了快速測試的功能以外， Postman 還擁有非常容易使用的介面，以及 Collection 的功能，這篇主要先介紹 Request 的功能，並實際操作一次 GET 與 Post 這兩個 Method 。

Vue

音同 View ，是一套以視圖層為基礎發展的 JavaScript 漸進式框架。與其他前端框架 / 函式庫不同的是， Vue.js 的標透的 API 供開發者實作資料定操作的件時也因為 Vue.js 的核心把焦點關注在狀態與畫面的同步層級上，遂能夠輕易地與其他 JavaScript 式庫前端開發工具整合使用，成為一套完整的前端開發方案。

References

伺服器端的介紹

https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/First_steps/Introduction#E4%BC%BA%E6%9C%8D%E7%AB%AF%E8%88%87%E7%94%A8%E6%88%B6%E7%AB%AF%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88%E6%98%AF%E7%9B%B8%E5%90%8C%E7%9A%84%E5%97%8E%EF%BC%9F

Wireframe

<https://tw.alphacamp.co/blog/wireframe>

Node.js backEnd

<https://ithelp.ithome.com.tw/m/articles/10194773>

Figma

<https://www.figma.com/files/recent?fuid=1062255556062200338>

Appendix