

客製化交易系統

系統需求規格書

Software Requirements Specification (SRS)

Version: 1.0

姓名	學號	E-mail
呂育璋	109590004	t109590004@ntut.org.tw
莊喆安	109590008	t109590008@ntut.org.tw
歐銘耘	109590037	benson99921@yahoo.com.tw
范遠皓	109590041	t109590041@ntut.org.tw
柯瑞霖	109590043	t109590043@ntut.org.tw
劉濬龔	109590048	f779849464977@gmail.com

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

01/03/2023

目錄 (Table of Contents)

Section 1 簡介 (Introduction)	1
1.1 目的 (Purpose)	1
1.2 系統名稱	2
1.3 概觀 (Overview)	2
1.4 符號描述 (Notation Description)	2
Section 2 系統 (System)	5
2.1 系統描述 (System Description)	5
2.1.1 系統架構圖 (System Context Diagram)	5
2.2 操作概念 (Operational Concepts)	5
2.3 設計限制 (Design, Data, and Implementation Constraints)	5
2.4 技術限制 (Technological Limitations)	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)	7
2.5.3 內部介面需求 (Internal Interface Requirements)	7
2.6 功能性需求 (Functional Requirements)	7
2.7 非功能性需求 (Non-Functional Requirements)	8
2.7.1 效能需求 (Performance Requirements)	8
2.7.2 測試需求 (Test Requirements)	8
2.8 其他需求 (Other Requirements)	9
2.8.1 環境需求 (Environmental Requirement)	9
Section 3 資料庫概念設計	10
3.1 Entity Relationship ER Model	10
Section 4 邏輯資料庫綱要	11
4.1 Schema of the Database	11
4.2 SQL Statements Used to Construct the Schema	16
4.3 The implementation of tables in target DBMS	20
Section 5 功能性依賴	21
Section 6 資料庫概念設計	22
Section 7 Additional Queries and Views	24
Section 8 Conclusions and Future Work	35
Section 9 References	35

Section 1 簡介 (Introduction)

1.1 目的 (Purpose)

在本學期的資料庫課程中，我們學到了設計資料庫該有的基本工具。為了更加熟練現有的技術，並增強實作上的深度。題目是：「客製化商品訂購系統」，旨在提供一個 B2C 的線上訂購系統，客戶可以在平台上搜尋特定商品類型，並講述所需要有的功能或是外型，並且可以隨時追蹤到商品的進度，方便客戶可以跟廠商進行修改。

根據身分將擁有以下各功能：

● 經理：

- 查看商品報表
- 管理員工的權限
- 管理平台上的商品(包含優惠)
- 確認處理訂單

● 員工：

- 提供最新的訂單進度
- 能上報自己部門的進度

● 會員：

- 瀏覽、搜尋已上架的商品類型
- 創建訂單與確認購買
- 提供的產品需求

● 訪客：

- 瀏覽、搜尋已上架的商品類型
- 可以創建自己的帳號

1.2 系統名稱

主系統名稱為：

客製化商品訂購系統(Customized Product Order System CPO)

各子系統分別為：

權限管理子系統 (Permissions to subsystems PTS)

員工管理子系統 (Employee Management Subsystem EMS)

進度管理子系統 (Order Progress Management Subsystem OPMS)

商品瀏覽與查詢子系統 (Product Browsing Query Subsystem PBQS)

會員訂單子系統 (Member Order Subsystem MOS)

訂單客製化細節子系統 (Customized Subsystems CS)

會員資訊管理子系統 (Customer Information Management Subsystem CIMS)

商品進度檢視子系統 (Interactive subsystem for product and customer departments ISCD)

優惠操作子系統 (Preferential Operating Subsystem POS)

商品管理子系統 (Product Management Subsystem PMS)

資料維護子系統 (Data maintenance subsystem DMS)

1.3 概觀 (Overview)

我們這組希望可以讓客戶能夠對於自己的需求進行需求的修改，資料包含許多不同的來源其中有客戶名，需要的商品，類型，細節，細節等等數據，將這些資料分別放置於各處並互相互動形成資料庫，所以我們這組打算使用方便入門MySQL入手進行開發。建立資料庫系統後，我們還要去對資料進行分類，看那些資料要進行共用或是需要獨立存在，而使用者可以讀到那些資料庫或是資料由其身分進行決定。

1.4 符號描述 (Notation Description)

CPO 1.0.0	The CPO system will be labeled with the number 1.0.0
PTS 1.1.n	The PTS components will be labeled with the number 1.1.n.
EMS 1.2.n	The EMS components will be labeled with the number 1.2.n
OPMS 1.3.n	The OPMS components will be labeled with the number 1.3.n.

PBQS 1.4.n	The PBQS components will be labeled with the number 1.4.n
MOS 1.5.n	The MOS components will be labeled with the number 1.5.n
CS 1.6.n	The CS components will be labeled with the number 1.6.n
CIMS 1.7.n	The CIMS components will be labeled with the number 1.7.n
ISCD 1.9.n	The ISCD components will be labeled with the number 1.9.n.
POS 1.10.n	The POS components will be labeled with the number 1.10.n.
PMS 1.11.n	The PMS components will be labeled with the number 1.11.n.
DMS 1.12.n	The DMS components will be labeled with the number 1.12.n.

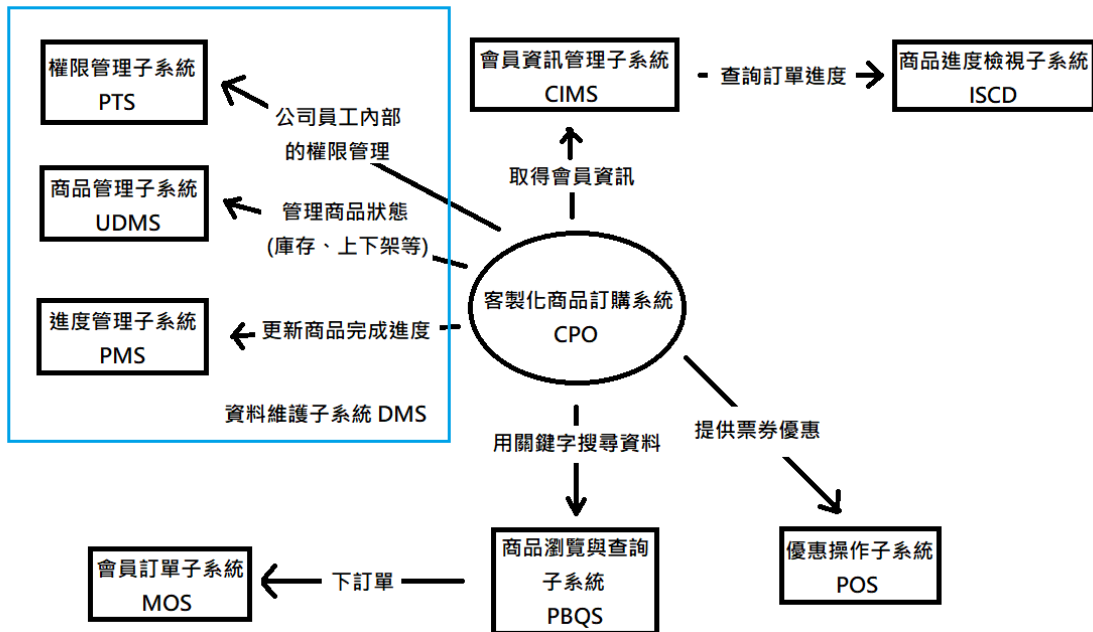
CPO-F-nnn	CPO 功能性需求(Functional Requirements)
CPO-N-nnn	CPO 非功能性需求(Non-Functional Requirements)
PTS-F-nnn	PTS 功能性需求(Functional Requirements)
PTS-N-nnn	PTS 非功能性需求(Non-Functional Requirements)
EMS-F-nnn	EMS 功能性需求(Functional Requirements)
EMS-N-nnn	EMS 非功能性需求(Non-Functional Requirements)
OPMS-F-nnn	OPMS 功能性需求(Functional Requirements)
OPMS-N-nnn	OPMS 非功能性需求(Non-Functional Requirements)
PBQS-F-nnn	PBQS 功能性需求(Functional Requirements)
PBQS-N-nnn	PBQS 非功能性需求(Non-Functional Requirements)
MOS-F-nnn	MOS 功能性需求(Non-Functional Requirements)
MOS-N-nnn	MOS 非功能性需求(Non-Functional Requirements)
CS-F-nnn	CS 功能性需求(Functional Requirements)

CS-N-nnn	CS 非功能性需求(Non-Functional Requirements)
CIMS-F-nnn	CIMS 功能性需求(Functional Requirements)
CIMS-N-nnn	CIMS 非功能性需求(Non-Functional Requirements)
ISCD-F-nnn	ISCD 功能性需求(Functional Requirements)
ISCD-N-nnn	ISCD 非功能性需求(Non-Functional Requirements)
POS-F-nnn	POS 功能性需求(Functional Requirements)
POS-N-nnn	POS 非功能性需求(Non-Functional Requirements)
PMS-F-nnn	PMS 功能性需求(Functional Requirements)
PMS-N-nnn	PMS 非功能性需求(Non-Functional Requirements)
DMS-F-nnn	DMS 功能性需求(Functional Requirements)
DMS-N-nnn	DMS 非功能性需求(Non-Functional Requirements)

Section 2 系統 (System)

2.1 系統描述 (System Description)

2.1.1 系統架構圖 (System Context Diagram)



2.2 操作概念 (Operational Concepts)

Scenario 1: 經理操作概念 (Manager Operational Concepts) 可以編輯員工權限。能讀取進度頁面，用來檢視員工進度。可以檢視訂單紀錄頁面。在聊天室頁面與會員進行溝通。可在商品新增優惠折扣。可以在商品編輯頁面新增、編輯、刪除商品。

Scenario 2: 員工操作概念 (Employee Operational Concepts) 在進度頁面編輯工作進度。根據訂單紀錄頁面上的客製化細項進行商品製作。

Scenario 3: 會員操作概念 (Member Operational Concepts) 從登入頁面登入後，進入主頁瀏覽、查詢商品。若要下單，可直接點擊主頁中商品後進入商品頁面，選擇數量和規格。如果對預設選項不滿意，可點選客製化按鈕進行細項修改。在聊天室頁面與經理進行溝通。

Scenario 4: 訪客操作概念 (Visitor Operational Concepts) 只能瀏覽頁面、查詢商品，或是進入登入註冊頁面進行註冊登入。

所有身分組都能查看、編輯自己的個人資料。

2.3設計限制 (Design, Data, and Implementation Constraints)

需求編號	需求描述
DDIC 001	使用 MySQL 作為 DBMS
DDIC 002	使用 Vue.js作為前端框架
DDIC 003	使用 flask 作為後端框架

2.4技術限制 (Technological Limitations)

需求編號	需求描述
TL 001	金流系統 (LINE pay、綠界)
TL 002	會員認證 (Google)

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
UI 001	會員登入及註冊介面
UI 002	商品瀏覽介面
UI 003	購物車介面
UI 004	訂單查詢介面
UI 005	商家管理介面
UI 006	搜尋介面
UI 007	權限給予介面
UI 008	客戶資訊管理介面
UI 009	平台商品管理介面

2.5.2外部介面需求 (External Interface Requirements)

需求編號	需求描述
ETI 001	使用者透過HTTP來瀏覽網頁。
ETI 002	網頁使用Vue框架與主機連接。

2.5.3內部介面需求 (Internal Interface Requirements)

需求編號	需求描述
ITI 001	PTS、EMS、CIMS 可讓經理編輯權限和管理會員
ITI 002	PMS、POS 可讓經理新增、修改、下架商品以及優惠操作
ITI 003	OPMS 可讓員工編輯管理訂單狀態
ITI 004	PBQS 可讓會員與訪客透過網頁瀏覽商品資訊
ITI 005	CIMS 可讓會員或訪客進行登入、註冊會員
ITI 006	MOS、CS 可讓會員編輯訂單與添加客製化敘述
ITI 007	ISCD 可讓會員檢視訂單進度

2.6功能性需求 (Functional Requirements)

需求編號	需求描述
FR 001	訪客可註冊與註銷會員
FR 002	訪客與會員可搜尋商品
FR 003	訪客與會員可將商品加入與移出購物車
FR 004	訪客將商品結帳前須先加入會員
FR 005	會員購物車內商品能夠結帳
FR 006	會員可以追蹤訂單狀態

FR 007	會員可以查看歷史訂單
FR 008	會員收到商品後能夠對商店進行評分
FR 009	經理擁有關閉商店的權限
FR 010	經理能夠查看總體營業狀況
FR 011	經理能夠建立與管理其他使用者權限
FR 012	經理與員工能夠釋出優惠卷
FR 013	員工可以上下架與編輯商品資訊

2.7非功能性需求 (Non-Functional Requirements)

2.7.1效能需求 (Performance Requirements)

需求編號	需求描述
NFP 001	訪客與會員瀏覽商品頁面 讀取時間應不大於 5 秒
NFP 002	訪客與會員查詢商品頁面 讀取時間應不大於 3 秒
NFP 003	資料同步處理時間應不大於 2 秒

2.7.2測試需求 (Test Requirements)

需求編號	需求描述
NFT 001	系統維護時，訪客與會員必須顯示維護中頁面
NFT 002	每個子系統需經測試且並無任何問題
NFT 003	提供多系統運行服務平台測試
NFT 004	提供使用者 GUI 介面測試
NFT 005	同時在線人數至少在20位以上 且系統能負擔之壓力測試

2.8其他需求 (Other Requirements)

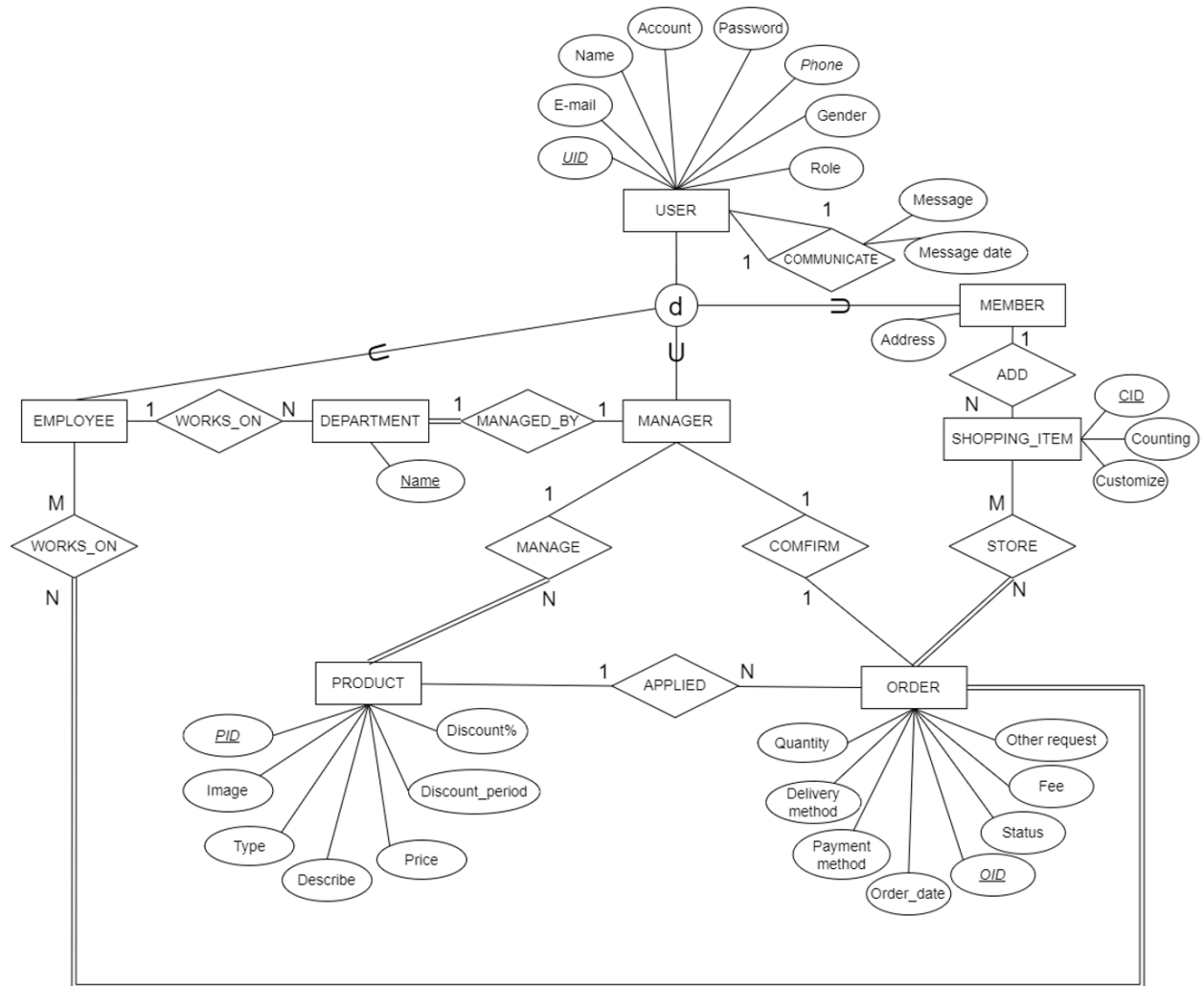
2.8.1 環境需求 (Environmental Requirement)

需求編號	需求描述
ER 001	需要上網環境

Section 3 資料庫概念設計

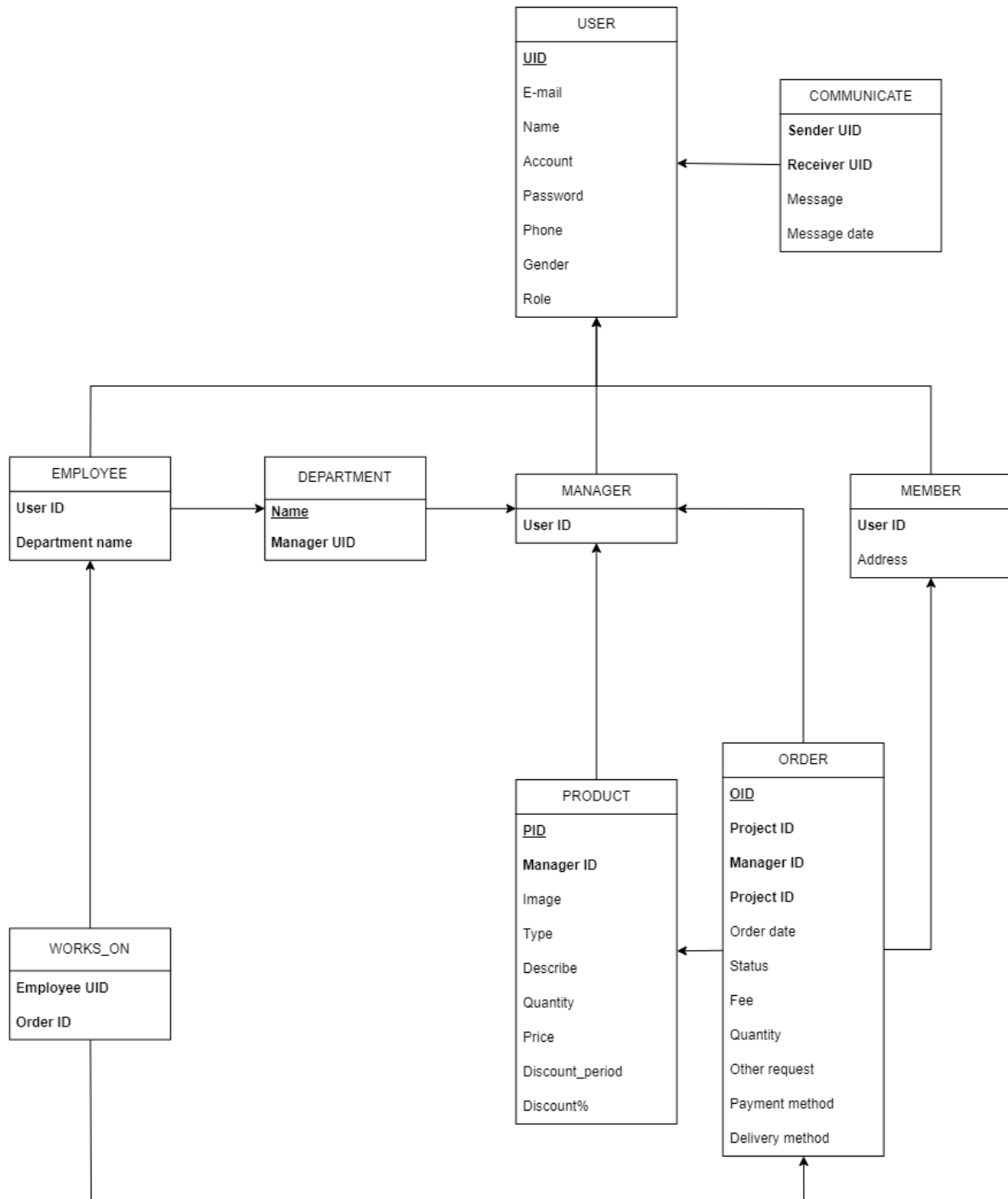
(Conceptual Design of the Database)

3.1 Entity Relationship ER Model



Section4 邏輯資料庫綱要 (Logic Database Schema)

4.1 Schema of the Database



USER				
Description:存放使用者資料				
Attribute	Type	Key	Nullable	Description
UID	INT	Primary	Not Null	使用者的ID (ID >= 1)
Name	VARCHAR(32)		Not Null	使用者名稱
E-mail	VARCHAR(64)		Not Null	使用者電子郵件
Account	VARCHAR(16)		Not Null	使用者帳號
Password	VARCHAR(16)		Not Null	使用者密碼
Role	VARCHAR(16)		Not Null	使用者等級
Phone	VARCHAR(16)		Not Null	使用者電話
Gender	VARCHAR(16)		Not Null	使用者性別

MANAGER				
Description:管理者身分組的資料				
Attribute	Type	Key	Nullable	Description
Manager_ID	INT	P/F	Not Null	管理者ID

EMPLOYEE				
Description:員工身分組的資料				
Attribute	Type	Key	Nullable	Description
Employee_ID	INT	P/F	Not Null	員工ID
Department	VARCHAR(16)	Foreign	Not Null	員工的部門

MEMBER				
Description:會員身分組的資料				
Attribute	Type	Key	Nullable	Description
Member_ID	INT	P/F	Not Null	會員ID
Address	VARCHAR(256)		Not Null	會員地址

DEPARTMENT				
Description:部門的資料				
Attribute	Type	Key	Nullable	Description
Name	VARCHAR(16)	Primary	Not Null	部門名稱
Manager_ID	INT	Foreign	Not Null	部門的管理者

PRODUCT				
Description:存放商品資料				
Attribute	Type	Key	Nullable	Description
PID	INT	Primary	Not Null	商品的ID(ID >=0)
Image	BLOB			商品圖片
Manager_ID	INT	Foreign	Not Null	管理者ID
Type	VARCHAR(16)		Not Null	商品類型
Describe	VARCHAR(256)		Not Null	商品描述
Quantity	INT		Not Null	商品總量
Price	INT		Not Null	商品價格
Discount	DECIMAL(3,2)		Not Null	商品折扣

Discount_period	DATETIME			商品折扣期限
-----------------	----------	--	--	--------

ORDER				
Description: 訂單的資料				
Attribute	Type	Key	Nullable	Description
OID	INT	P/F	Not Null	訂單ID
PID	INT	Foreign	Not Null	商品ID
Manager_ID	INT	Foreign	Not Null	管理者ID
Member_ID	INT	Foreign	Not Null	會員ID
Order_date	DATETIME		Not Null	訂單成立日期
Quantity	INT		Not Null	訂購數量
Status	VARCHAR(16)		Not Null	訂單狀態
Payment_method	VARCHAR(16)		Not Null	付款方式
Delivery_method	VARCHAR(16)		Not Null	運送方式
Fee	INT		Not Null	訂單價錢
Other_request	VARCHAR(256)			其他需求

WORKS_ON				
Description: 員工所在的部門資料				
Attribute	Type	Key	Nullable	Description
Employee_ID	INT	P/F	Not Null	員工ID
OID	INT	Foreign	Not Null	訂單編號

COMMUNICATE				
Description:員工與經理對話資料				
Attribute	Type	Key	Nullable	Description
Sender	INT	Primary	Not Null	寄送者ID
Receiver	INT		Not Null	接收者ID
Message	varchar(100)		Not Null	訊息
Date	datetime		Not Null	訊息發送時間

4.2 SQL Statements Used to Construct the Schema

USER
<pre>CREATE TABLE `user` (`UID` INT AUTO_INCREMENT, `Name` VARCHAR(32) NOT NULL, `E-mail` VARCHAR(64) NOT NULL, `Account` VARCHAR(16) NOT NULL, `Password` VARCHAR(16) NOT NULL, `Role` VARCHAR(16) NOT NULL, `Phone` VARCHAR(16) NOT NULL, `Gender` VARCHAR(16) NOT NULL, PRIMARY KEY(`UID`));</pre>

MEMBER
<pre>CREATE TABLE `member` (`Member_ID` INT NOT NULL, `Address` VARCHAR(256) NOT NULL, PRIMARY KEY(`Member_ID`), FOREIGN KEY (`Member_ID`) REFERENCES `user` (`UID`) ON DELETE CASCADE);</pre>

DEPARTMENT
<pre>CREATE TABLE `department` (`Name` VARCHAR(16) NOT NULL, `Manager_ID` INT NOT NULL, PRIMARY KEY(`Name`), FOREIGN KEY (`Manager_ID`) REFERENCES `manager` (`Manager_ID`) ON DELETE CASCADE);</pre>

EMPLOYEE

```
CREATE TABLE `employee` (  
  `Employee_ID` INT NOT NULL,  
  `Department` VARCHAR(16) NOT NULL,  
  PRIMARY KEY(`Employee_ID`),  
  FOREIGN KEY (`Employee_ID`) REFERENCES `user`(`UID`) ON DELETE CASCADE,  
  FOREIGN KEY (`Department`) REFERENCES `department`(`Name`) ON DELETE CASCADE  
);
```

MANAGER

```
CREATE TABLE `manager` (  
  `Manager_ID` INT NOT NULL,  
  PRIMARY KEY(`Manager_ID`),  
  FOREIGN KEY (`Manager_ID`) REFERENCES `user`(`UID`) ON DELETE CASCADE  
);  
drop tables `manager`;
```

PROJECT

```
CREATE TABLE `product` (  
  `PID` INT AUTO_INCREMENT,  
  `Image` BLOB,  
  `Manager_ID` INT NOT NULL,  
  `Type` VARCHAR(16) NOT NULL,  
  `Describe` VARCHAR(256) NOT NULL,  
  `Quantity` INT NOT NULL,  
  `Price` INT NOT NULL,  
  `Discount` DECIMAL(3,2) NOT NULL,  
  `Discount_period` DATETIME,  
  
  PRIMARY KEY(`PID`),  
  FOREIGN KEY (`Manager_ID`) REFERENCES `manager`(`Manager_ID`) ON DELETE CASCADE  
);
```

ORDER

```
CREATE TABLE `order` (  
  `OID` INT AUTO_INCREMENT,  
  `PID` INT NOT NULL,  
  `Manager_ID` INT NOT NULL,  
  `Member_ID` INT NOT NULL,  
  `Order_date` DATETIME NOT NULL,  
  `Quantity` INT NOT NULL,  
  `Status` VARCHAR(16) NOT NULL,  
  `Payment_method` VARCHAR(16) NOT NULL,  
  `Delivery_method` VARCHAR(16) NOT NULL,  
  `Fee` INT NOT NULL,  
  `Other_request` VARCHAR(256),  
  PRIMARY KEY(`OID`),  
  FOREIGN KEY (`PID`) REFERENCES `product`(`PID`) ON DELETE CASCADE,  
  FOREIGN KEY (`Manager_ID`) REFERENCES `manager`(`Manager_ID`) ON DELETE CASCADE,  
  FOREIGN KEY (`Member_ID`) REFERENCES `member`(`Member_ID`) ON DELETE CASCADE  
);
```

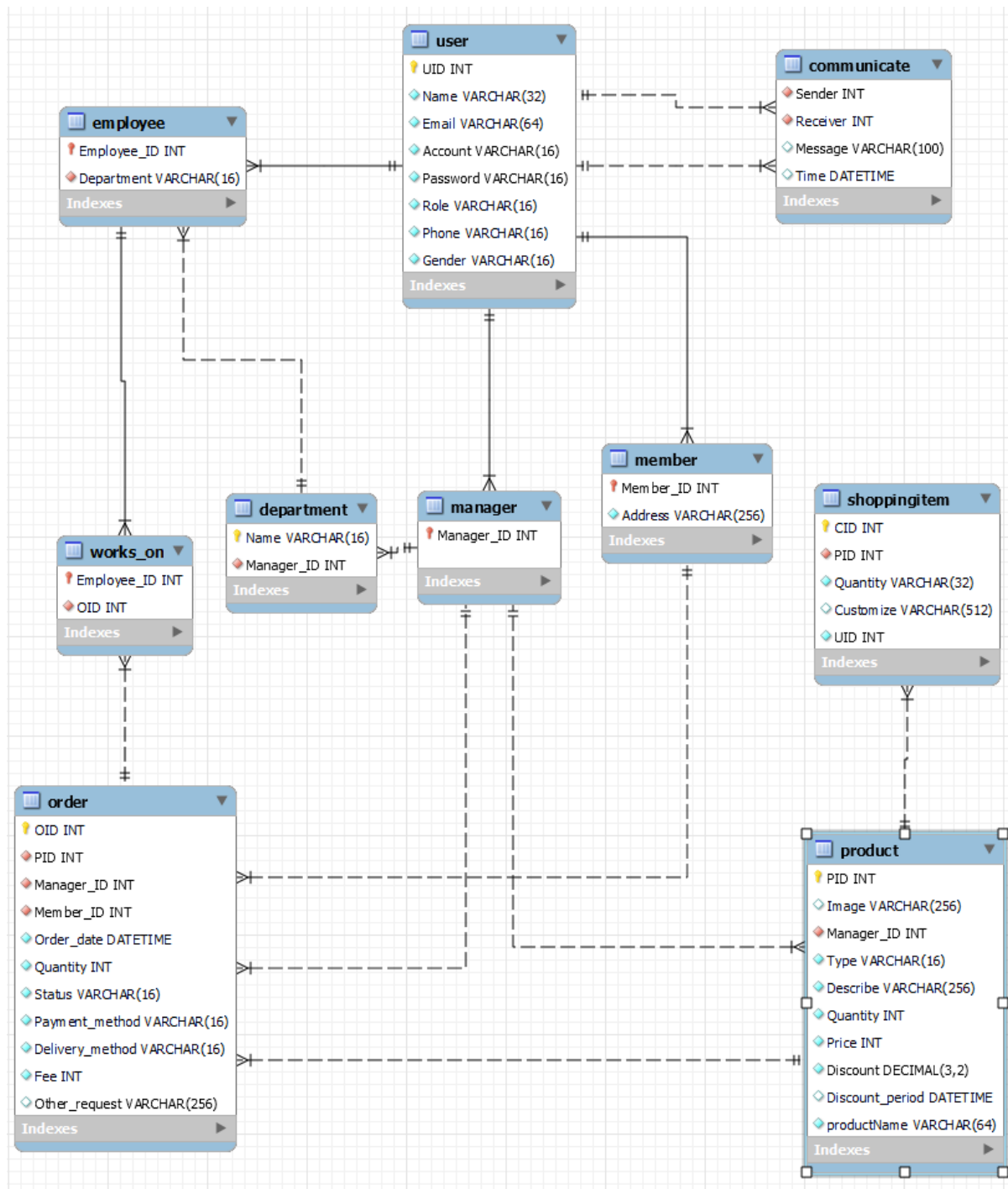
COMMUNICATE

```
CREATE TABLE `communicate` (  
  `Sender` INT NOT NULL,  
  `Receiver` INT NOT NULL,  
  `Message` varchar(100),  
  `Time` DATETIME,  
  FOREIGN KEY (`Sender`) REFERENCES `user`(`UID`) ON DELETE CASCADE,  
  FOREIGN KEY (`Receiver`) REFERENCES `user`(`UID`) ON DELETE CASCADE  
);
```

WORKS_ON

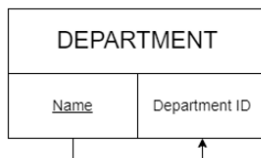
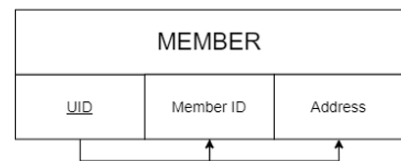
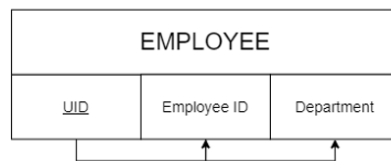
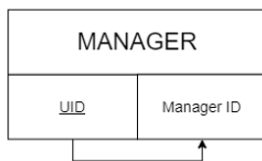
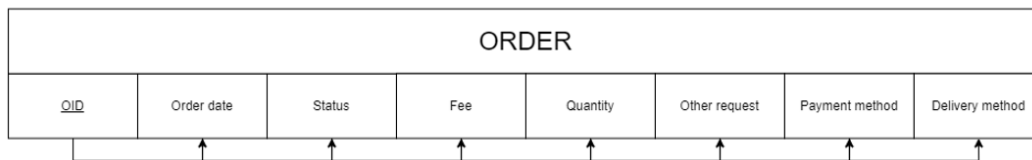
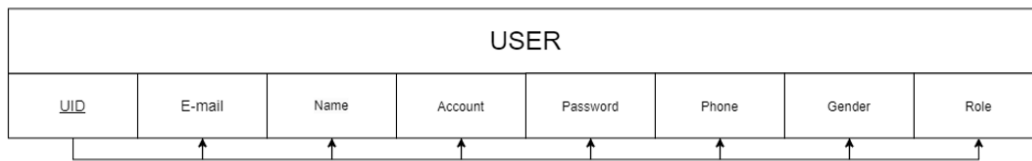
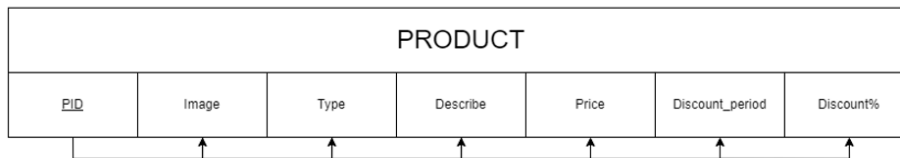
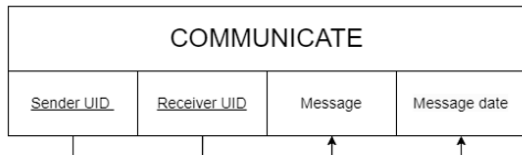
```
CREATE TABLE `works_on` (  
  `Employee_ID` INT NOT NULL,  
  `OID` INT NOT NULL,  
  PRIMARY KEY(`Employee_ID`),  
  FOREIGN KEY (`Employee_ID`) REFERENCES `employee` (`Employee_ID`) ON DELETE CASCADE,  
  FOREIGN KEY (`OID`) REFERENCES `order` (`OID`) ON DELETE CASCADE  
);
```

4.3 The implementation of tables in target DBMS



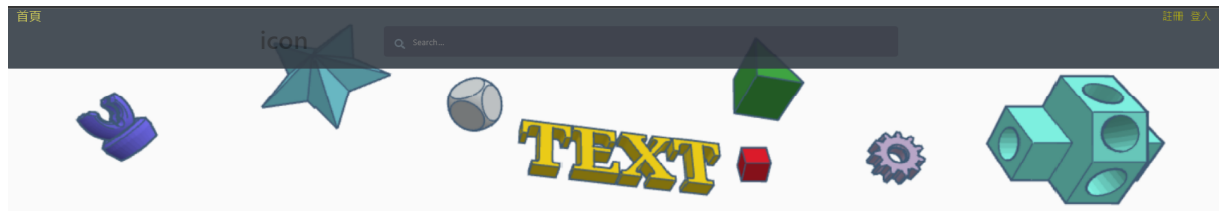
Section 5 功能性依賴

(Functional Dependencies and Database Normalization)



Section 6 資料庫概念設計 (Additional Queries and Views)

6.1 首頁



6.2 註冊頁面

註冊帳號

姓名*

帳號*

密碼*

確認密碼*

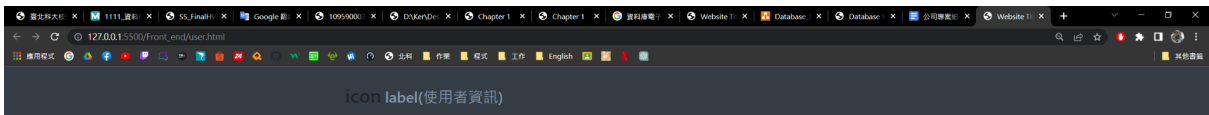
電子郵件*

電話*

地址*

性別 ☐ 男 ☐ 女 ☐ 不願透露

6.3 使用者頁面



User Information

密碼:

Email:

電話:

地址

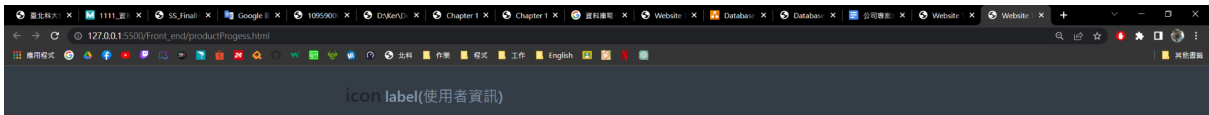
訂單紀錄

專案管理


登出



6.4 訂單狀態



User Information



隱形

材料工數

繪圖

製作

測試

備註



Section 7 Additional Queries and Views

1.會員使用

a.註冊

```
# _____ 註冊 _____

# 註冊
@app.route('/signUp', methods=['GET', 'POST'])
def signUp():
    conn = mysql.connect()
    cursor = conn.cursor()
    try:
        data = request.get_json()
        cursor.execute('SELECT Account, Password FROM user WHERE (Account = %s)', data['Account'])
        user = cursor.fetchall()
        if (user == ()):
            sql = ("INSERT INTO user (Name, Email, Account, Password, Role, Phone, Gender) VALUES (%s, %s, %s, %s, %s, %s, %s)")
            val = (data['Name'], data['Email'], data['Account'], data['Password'], 'member', data['Phone'], data['Gender'])
            cursor.execute(sql, val)
            conn.commit()
            cursor.execute("SELECT UID FROM user WHERE Account = %s", data['Account'])
            userID = cursor.fetchall()
            sql = ("INSERT INTO member (Member_ID, Address) VALUES (%s, %s)")
            val = (userID, data['Address'])
            cursor.execute(sql, val)
            conn.commit()
            return jsonify({
                'status': 'success',
                'values': '註冊成功'
            })
        else:
            return jsonify({
                'status': 'success',
                'values': '此帳號已被註冊'
            })
    finally:
        cursor.close()
        conn.close()
```

b.登入

```

# 登入
@app.route('/signIn', methods=['GET', 'POST'])
def signIn():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        data = request.get_json()
        sql = ("SELECT Account, Password FROM user WHERE (Account = %s)")
        val = (data['Account'])
        cursor.execute(sql, val)
        user = cursor.fetchall()
        if (user == ()):
            print('This account is not registerd.')
            return jsonify({
                'status': 'success',
                'values': '帳號未被註冊'
            })
        elif (data['Password'] != user[0]['Password']):
            print('Wrong password.')
            return jsonify({
                'status': 'success',
                'values': '錯誤的密碼'
            })
        elif (data['Password'] == user[0]['Password']):
            print('Success.')
            cursor.execute("SELECT UID FROM user WHERE (Account = %s)", val)
            global UID
            UID = cursor.fetchall()[0]['UID']
            print('SignIn UID = ' + str(UID))
            return jsonify({
                'status': 'success',
                'values': '登入成功',
                'UID': UID
            })
        else:
            return jsonify({
                'status': 'success',
                'values': '-1'
            })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

c. 登出

```

# 登出
@app.route('/signOut', methods=['POST'])
def signOut():
    global UID
    UID = 0
    print('SignOut UID = ' + str(UID))
    return jsonify({
        'status': 'success',
        'values': '登出成功',
        'UID': UID
    })

```

2. 針對使用者進行判斷

a. 查看會員

```
# 查看會員
@app.route('/userList', methods=['GET'])
def userList():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        cursor.execute("SELECT * FROM user")
        userslist = cursor.fetchall()
        return jsonify({
            'status': 'success',
            'values': userslist
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

b. 取得使用者資料

```

#取得使用者資料
@app.route('/user', methods=['GET'])
def getUser():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        global UID
        cursor.execute("SELECT * FROM user WHERE UID = %s", UID)
        user = cursor.fetchall()
        cursor.execute("SELECT Address FROM member WHERE Member_ID = %s", UID)
        Address = cursor.fetchall()
        return jsonify({
            'status': 'success',
            'values': user,
            'Address': Address
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

c.取得使用者身分

```

#取得身分
@app.route('/getRole', methods=['GET'])
def getRole():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        global UID
        cursor.execute("SELECT Role FROM user WHERE UID = %s", UID)
        role = cursor.fetchall()
        return jsonify({
            'status': 'success',
            'values': role
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

d.修改使用者資料

```
#修改使用者資料
@app.route('/modifyUser', methods=['GET', 'POST'])
def modifyUser():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        data = request.get_json()
        global UID
        val = (data['Email'], data['Phone'], UID)
        cursor.execute("UPDATE user SET Email= %s, Phone= %s WHERE UID = %s", val)
        conn.commit()
        val = (data['Address'], UID)
        cursor.execute("UPDATE member SET Address= %s WHERE Member_ID = %s", val)
        conn.commit()
        Address = cursor.fetchall()
        return jsonify({
            'status': 'success',
            'values': '修改成功'
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

3.網頁產品

a.首頁產品

```
# 主頁產品
@app.route('/getAllProduct', methods=['GET'])
def getAllProduct():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        cursor.execute("SELECT * FROM product")
        productList = cursor.fetchall()
        return jsonify({
            'status' : 'success',
            'values' : productList
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

b.查看產品細項

```
# 進入產品
@app.route('/getProduct', methods=['GET'])
def getProduct():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        global PID
        cursor.execute("SELECT * FROM product WHERE PID = %s", PID)
        product = cursor.fetchall()
        return jsonify({
            'status' : 'success',
            'values' : product
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

4.使用購物車

a.加入購物車

```
#加入購物車
@app.route('/addToShoppingItem', methods=['POST'])
def addToShoppingItem():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        data = request.get_json()
        global PID
        global UID
        val = (PID, data['Quantity'], data['Customize'], UID)
        cursor.execute("INSERT INTO shoppingItem (PID, Quantity, Customize, UID) VALUES (%s, %s, %s, %s)", val)
        conn.commit()
        shoppingItem = cursor.fetchall()
        return jsonify({
            'status' : 'success',
            'values' : '已加入購物車'
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

b.進入購物車

```

#進入購物車
@app.route('/getShoppingItem', methods=['GET'])
def getShoppingItem():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        global UID
        cursor.execute("SELECT * FROM shoppingItem AS S, product AS P WHERE S.UID = %s AND P.PID = S.PID", UID)
        shoppingItem = cursor.fetchall()
        return jsonify({
            'status' : 'success',
            'values' : shoppingItem
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

c.刪除購物車中的商品

```

#刪除購物車
@app.route('/deleteShoppingItem', methods=['POST'])
def deleteShoppingItem():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        data = request.get_json()
        global UID
        val = (UID, data['CID'])
        cursor.execute("DELETE FROM shoppingItem AS S WHERE S.UID = %s AND S.CID = %s", val)
        conn.commit()
        return jsonify({
            'status' : 'success',
            'values' : '成功刪除'
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

d.購物車結帳訂單成立


```

#購物單新增訂單成立
@app.route('/shoppingItemAddOrder', methods=['POST'])
def shoppingItemAddOrder():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        data = request.get_json()
        global UID
        val = (data['PID'], 9, UID, data['Order_date'], data['Quantity'], '剛剛成立', '货到付款', '货到付款', data['Fee'], data['Other_request'])
        cursor.execute("INSERT INTO `order` (PID, Manager_ID, Member_ID, Order_date, Quantity, Status, Payment_method, Delivery_method, Fee, Other_request) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)", val)
        conn.commit()
        cursor.execute("DELETE FROM `shoppingItem`")
        conn.commit()
        return jsonify({
            'status': 'success',
            'values': '訂單成立'
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

5. 訂單使用

a. 根據使用者身分取得訂單

```

@app.route('/getOrderByUser', methods=['GET'])
def getOrderByUser():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    cursor.execute("SELECT OID, Member_ID AS UID, product.Image AS Image, productName AS `Name`, Other_request AS Other, Price, `order`.Quantity AS Quantity FROM `order` INNER JOIN product USING(PID) WHERE Member_ID = %s;", UID)
    order = cursor.fetchall()
    return jsonify({
        'status': 'success',
        'values': order
    })

```

b. 取得所有訂單

```

@app.route('/getAllOrder', methods=['GET'])
def getAllOrder():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    cursor.execute("SELECT OID, Member_ID AS UID, product.Image AS Image, productName AS `Name`, Other_request AS Other, Price, `order`.Quantity AS Quantity FROM `order` INNER JOIN product USING(PID);")
    order = cursor.fetchall()
    return jsonify({
        'status': 'success',
        'values': order
    })

```

c. 取得特定訂單的進度

```

@app.route('/getOrderStatusByOID', methods=['GET'])
def getOrderStatusByOID():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    global OID
    cursor.execute("SELECT OID, Member_ID AS UID, `Status` FROM `order` WHERE OID = %s;", OID)
    order = cursor.fetchall()
    return jsonify({
        'status': 'success',
        'values': order
    })

```

d. 更新目前訂單進度

```

@app.route('/modifyOrderStatus', methods=['POST'])
def modifyOrderStatus():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        data = request.get_json()
        global OID
        val = (data['Status'], OID)
        cursor.execute("UPDATE `order` SET `Status` = %s WHERE OID = %s", val)
        conn.commit()
        Address = cursor.fetchall()
        return jsonify({
            'status': 'success',
            'values': '修改成功'
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()

```

6.換頁時傳遞資訊

a.更新目前使用者身分

```

# PID POST
@app.route('/PID/<int:pid>', methods=['POST'])
def Update(pid):
    global PID
    PID = pid
    return jsonify({
        'status' : 'success',
        'values' : pid
    })

```

7.商品設定

a.編輯商品

```
# 編輯商品
@app.route('/editProduct', methods=['POST'])
def editProduct():
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)

    try:
        data = request.get_json()
        global PID
        val = (data['name'], data['image'], data['price'],
              data['discount'], data['describe'], PID)
        cursor.execute(
            "UPDATE product SET productName= %s, Image= %s, Price= %s, Discount= %s, `Describe`= %s WHERE PID = %s", val)

        conn.commit()
        return jsonify({
            'status': 'success',
            'values': '編輯成功'
        })
    finally:
        cursor.close()
        conn.close()
```

b.取得商品

```
#取得商品
@app.route('/getTargetProduct/<int:pid>', methods=['get'])
def getTargetProduct(pid):
    conn = mysql.connect()
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    try:
        cursor.execute("SELECT * FROM product WHERE PID = %s", pid)
        product = cursor.fetchall()
        return jsonify({
            'status' : 'success',
            'values' : product
        })
    # except Exception as e:
    #     print(e)
    finally:
        cursor.close()
        conn.close()
```

SQL

```

7 • SELECT Gender, COUNT(*)
8   FROM `user`
9   GROUP BY Gender;

```

Result Grid | Filter Rows: | Export:

	Gender	COUNT(*)
▶	male	2
	female	2
	none	1

```

11 • SELECT *
12 FROM `user`
13 ORDER BY Gender DESC;

```

Result Grid | Filter Rows: | Edit: | Export/Import:

	UID	Name	Email	Account	Password	Role	Phone	Gender
▶	9	man01	man01@gmail.com	man01	man01	manager	090201	none
	7	mem1	mem1@gmail.com	mem1	mem1	member	090001	male
	10	mem2	mem2@gmail.com	mem2	mem2	member	09445213	male
	8	emp1	emp1@gmail.com	emp1	emp1	employee	090101	female
	11	mem3	mem3@gmail.com	mem3	mem3	member	091234678	female
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```

15 • SELECT *
16 FROM `order`
17 WHERE `Status` = '剛剛成立'
18 ORDER BY Order_date DESC;
19

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	OID	PID	Manager_ID	Member_ID	Order_date	Quantity	Status	Payment_method	Delivery_method	Fee	Other_request
▶	4	9	9	10	2023-01-10 21:41:17	1	剛剛成立	貨到付款	貨到付款	90	mem2 black
	1	8	9	7	2023-01-10 21:40:18	1	剛剛成立	貨到付款	貨到付款	70	mem1 black thin
	3	9	9	7	2023-01-10 21:40:18	2	剛剛成立	貨到付款	貨到付款	180	mem1 green fat
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Section 8 Conclusions and Future Work

在這次的專案中，我們充分的學習了html、vue.js以及MySQL這三種程式語言的使用與開發。並且很順利的將這三者結合並建構出一個完整的系統。透過實作，我們也更了解EER Diagram和Schema的涵義以及其實際的應用。對我們來說，這是一次寶貴的資料庫專案開發經驗。未來如果要再鑽研資料庫方面的系統學習，可能會考慮試試看其他的前後端工具。並且在EER Diagram上會再審視圖表的邏輯清晰度 最後再擴展我們的編程能力，朝可靠的全端工程師前進。

Section 9 References

Python & JavaScript Web Development [Flask & VueJS Full Stack]

by Parwiz Forogh

<https://www.youtube.com/watch?v=nCMB5p74WJQ>

JavaScript基本功修練

<https://ithelp.ithome.com.tw/articles/10253259>