**水果線上訂購系統**

**(Food Selling Online System)**

**系統需求規格書**

**Software Requirements Specification (SRS)**

**Version: 1.0**

|  |  |  |
| --- | --- | --- |
| **姓名** | **學號** | **E-mail** |
| 鄭耀飛 | 104440026 | t104440026@ntut.edu.tw |
| 廖彥澤 | 106590018 | t106590018@ntut.edu.tw |
| 王柏偉 | 106590021 | t106590021@ntut.edu.tw |
| 蔡政哲 | 106590022 | t106590022@ntut.edu.tw |
| 高楷杰 | 106590026 | t106590026@ntut.edu.tw |
| 溫致綱 | 106590040 | t106590040@ntut.edu.tw |
| 陳風平 | 106590048 | t106590048@ntut.edu.tw |

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

**10/08/2019**

# ***目錄(Table of Contents)***

[**目錄(Table of Contents)**](#_v219w2245i1c)1

[**Section 1 簡介 (Introduction)**](#_xfjcn7pk2sp)3

[1.1 目的 (Purpose)](#_hgvjm332vobi) 3

[1.2 系統名稱 (Identification)](#_hgvjm332vobi) 3

[1.3 概觀(Overview)](#_j6zd5q1kml1i) 3

[1.4 符號描述(Notation Description)](#_bxeojf6zt964) 3

**Section 2 系統 (System)**  4

[2.1 系統描述 (SystemJ Description)](#_bxmsogmfi8i2) 4

[2.1.1 系統架構圖 (System Context Diagram)](#_fpsrbhxxu5ka) 4

[2.2 設計限制 (Design, Data, and Implementation Constrains)](#_ogishbb45wf0) 4

[2.3 技術限制 (Technological Limitations)](#_x8o241hmt2y0) 5

[2.4 介面需求 (Interface Requirements)](#_slw3h9lc639s) 5

[2.4.1 使用者介面需求 (User Interfaces Requirements)](#_e237a05amd15) 5

[2.4.2 外部介面需求 (External Interface Requirements)](#_qkym3anksl31) 6

[2.4.3 內部介面需求 (Internal Interface Requirements)](#_5x4xle5oqtp2) 6

[2.5 功能性需求 (Functional Requirements)](#_oi90jay3umki) 7

[2.6 非功能性需求 (Non-Functional Requirements)](#_kv7dy3gkgjb8) 7

[2.7 操作概念 (Operational Concepts)](#_dch6dxriq7f4) 8

[2.7.1 網頁導覽概念 (Anonymous Operational Concepts)](#_6ji2kcuk27q6) 8

[2.7.2 顧客使用者操作概念 (Customer Operational Concepts)](#_3kc2vhuiy8av) 8

[2.7.3 管理者操作概念(Manager Operational Concepts)](#_9hjz0e9fgiy4) 8

[2.7.4 系統管理者操作概念(Administrator Operational Concepts)](#_rkn6wnn4tvsn) 8

**Section 3 資料庫概念設計(Conceptual Design of the Database)** 8

3.1 Entity-Relationship (ER) Model 8

3.2 企業規則 (Business Rules) 9

**Section 4 邏輯資料庫綱要(Logical Database Schema)** 12

4.1 Schema of the Database 12

4.2 Domain Definitions 14

4.3 Relation Domains 16

4.4 Expectation of Possible Database Operations 19

4.5 SQL Statements for Database Construction 22

4.6 Implementation of Tables in Target DBMS 25

**Section 5 Functional Dependencies and Database Normalization** 27

5.1 Functional Dependencies 27

**Section 6 Additional Queries and Views**  29

6.1 System Installation Description 29

6.2 The Use of the System 34

**Section 7 Suggestions of Database Turning** 40

**Section 8 Additional Queries and Views** 41

**Section 9 Conclusions and Future Work** 47

9.1 Conclusions 47

9.2 Future work 47

**Glossary** 49

**References** 50

# Section 1 簡介 (Introduction)

## **1.1 目的 (Purpose)**

想要更深入了解資料庫設計方式、技術運用、需求分析和效能評估等相關知識，此次實作水果線上訂購系統 (UBS) 是個很好的學習機會。

台灣為水果農業大國，具有需多高經濟價值的水果可以外銷國外，但大多都藉由傳統通路居多，在網路店商發達的年代，販賣各式不同的商品，卻缺乏知名的水果的販賣平台，藉由此次機會設計統合台灣各地農民水果的線上訂購網站，有助於產業的轉型與發展，也幫助農民增加銷售管道和收入。

## **1.2 系統名稱 (Identification)**

水果線上訂購系統(Food Selling Online System，簡稱FSOS)。

## **1.3 概觀(Overview)**

資料庫系統是資訊工程當中相當重要的一個環節，在這數據的時代，懂得建立、管理好的資料庫，不僅能讓應用程式運行時更加流暢，在大數據的分析下，也能帶來更多的獲利，而本專案利用 MySQL 作為 DBMS，他具有以下優點：便宜、高度最佳化、好學且操作簡單， 基於以上幾個優點，我們在專案的設計上採用 MySQL 做為資料庫的系統，而透過這次的專案，讓我們能夠更加了解資料庫的設計與運用。

## **1.4 符號描述(Notation Description)**

|  |  |
| --- | --- |
| **MS 1.0n** | 會員管理系統使用編號 MS 1.0n |
| **RS 1.1n** | 審查系統使用編號 RS 1.1n |
| **HS 1.2n** | 榮譽商家商品系統使用編號 HS 1.2n |
| **SS 1.3n** | 當季農產品系統使用編號 SS 1.3n |
| **BS 1.4n** | 農產品上下架系統使用編號 BS 1.4n |
| **SFS 1.5n** | 查詢篩選系統使用編號 SFS 1.5n |
| **ES 1.6n** | 農產品評語系統使用編號 ES 1.6n |
| **TS 1.7n** | 追縱清單系統使用編號 TS 1.7n |
| **SCS 1.8n** | 購物車系統使用編號 SCS 1.8n |
| **CS 1.9n** | 結帳系統使用編號 CS 1.9n |

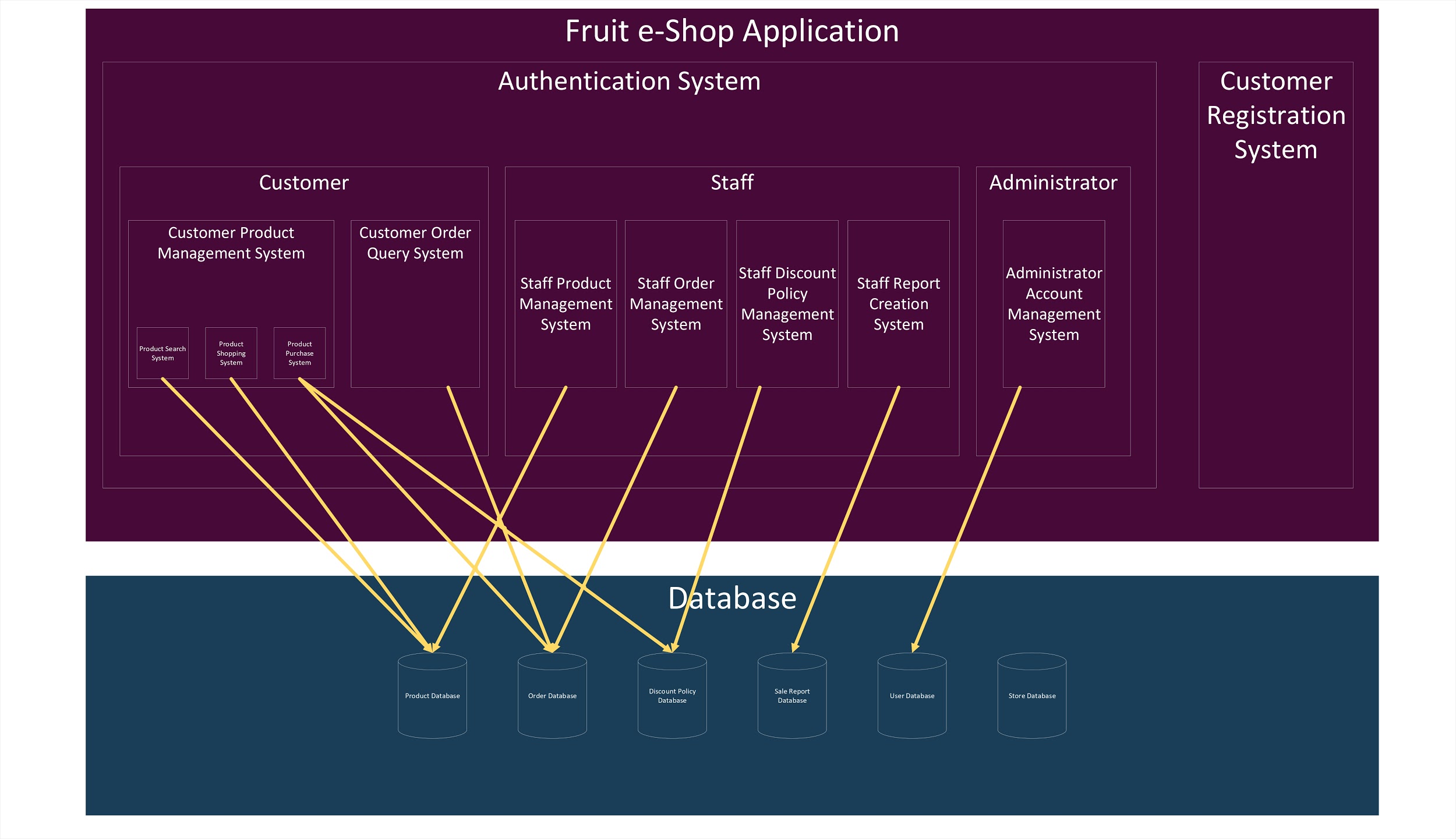
# Section 2 系統 (System)

## **2.1 系統描述 (System Description)**

本系統為農民蔬果系統，此系統之設計為方便管理者管理與農民線上註冊以及消費者能夠線上購買，詳細功能包含：

* **會員註冊與認證系統**：此部分可讓使用者註冊，成為會員後，依照不同的身分可以 執行相對應的功能，如：管理員可以上下架所有農產品以及管理所有會員資料，同時也可以認證商品使其成為減碳商品以及認證農民使其成為榮譽農民、經認證的農民可以 指定欲販賣的蔬果以及欲販賣之數量、會員可以下訂單等等。
* **農產品上下架系統**：本系同提供一個整合的平台讓工作人員可以很方便的對販賣之農產品進行 管理，相關功能包括：新增商品（新增）、將商品下架（刪除）。
* **農產品瀏覽與查詢**：一般使用者在登入前即可使用的功能，由於沒有經過身分確認， 所以不能夠訂商品也不能對資料庫內之資料進行修改，只能讀取資料庫內容，瀏覽商品資料等等。
* **農產品查詢篩選系統**：為查詢功能得額外功能，能對商品篩選。功能包括：價格從低至高、價格從高至低、當季商品、減碳商品、榮譽商家。
* **追蹤清單及購物車**：會員經過登入後，可以將有興趣的農產品加入追蹤清單，免去每次登入後需要重新再找一次想訂的商品，而想同時買多項以上時，可先將與購買之商品加入購物車，稍後再一次購買。
* **線上訂購**：一般會員可以在登入後進行訂購的動作，將過身分確認等相關手續後， 即可下單訂商品，而此時店員將收到訂單，並由系統發出訂單信件。
* **農產品評語**：會員可對農產品作評語，這樣一來可提供想買此農產品的會員有所參考。 訂單歷史紀錄查詢：會員可查詢所有訂單紀錄，包含以購買和尚未購買。
* **缺貨農產品通知**：會員可對缺貨的農產品做追蹤。
* **審查系統**：審查員(管理員)可對農產品登記為減碳商品，也可以登記農民(會員)使其成為榮譽商家。
* **當季農產品瀏覽**：依照不同季節推薦該季盛產之農產品。

## **2.1.1 系統架構圖 (System Context Diagram)**



## **2.2 設計限制 (Design, Data, and Implementation Constrains)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| IC 001 | 使用MYSQL作為DBMS |
| IC 002 | ajax設計使用jQuery、javascript |
| IC 003 | 網頁設計使用Golang |

## **2.3 技術限制 (Technological Limitations)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| TL 001 | 物流系統 (使用宅急便) |
| TL 002 | 金流系統 (使用線上刷卡、ATM、便利商店等付款方式) |
| TL 003 | 手機認證 (加入會員使用手機簡訊認證) |
| TL 004 | 農產品產地認證 (當地農會或工會團體) |

## **2.4介面需求 (Interface Requirements)**

### **2.4.1使用者介面需求 (User Interfaces Requirements)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| UIR 001 | 網站的首頁，提供使用者登入、登出，並且可供搜索、導引至其他頁面的功能，同時會列出熱銷商品。 |
| UIR 002 | 登入、註冊頁面，提供使用者輸入帳號密碼，或是註冊的功能。 |
| UIR 003 | 客戶資訊頁面，能夠看到會員的資訊，包括預設地址、手機…...等資料，以及悄悄話的功能。 |
| UIR 004 | 購物車頁面，提供客戶能夠看到自己加入購物車的商品和歷史紀錄，並進行刪除或是結帳的動作。 |
| UIR 005 | 追蹤訂單頁面，使用者可以在此頁面中了解購買商品的配送狀況。 |
| UIR 006 | 賣家賣場頁面，員工可以在其中管理自己的商品，例如上架、下架、折扣、更新……等操作。 |
| UIR 007 | 報表分析頁面，賣家可以在其中檢視自己賣場內商品的各種報表分析。 |
| UIR 008 | 系統管理員頁面，系統管理員可於此處管理帳號資訊。 |

### **2.4.2外部介面需求 (External Interface Requirements)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| EIR 001 | 使用者用瀏覽器透過HTTP 通訊協定與主機通訊。 |

### 2.4.3內部介面需求 (Internal Interface Requirements)

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| IIR 001 | 系統管理員帳號管理子系統必須要能夠在使用者資料庫內新增、查詢、修改、刪除使用者資料。 |
| IIR 002 | 產品搜尋子系統必須要能正確的查詢產品資料庫的資料。 |
| IIR 003 | 產品瀏覽子系統必須要能正確的查詢產品資料庫的資料。 |
| IIR 004 | 產品購買子系統必須要能在訂單資料庫內新增產品。 |
| IIR 005 | 產品購買子系統必須要能夠藉由折扣資料庫打折。 |
| IIR 006 | 訂單查詢子系統必須要能夠在訂單資料庫內查詢訂單資料。 |
| IIR 007 | 商品管理子系統必須要能夠在產品資料庫新增、修改產品。 |
| IIR 008 | 訂單管理子系統必須要能夠在訂單資料庫讀取、刪除訂單。 |
| IIR 009 | 折扣管理子系統必須要能夠在折扣資料庫新增、查詢、修改、刪除折扣。 |
| IIR 010 | 創建報表子系統必須要能夠藉由報表資料庫的資料產生報表。 |

## 

## **2.5 功能性需求 (Functional Requirements)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| FR 001 | 利用表格呈現水果 |
| FR 002 | 可新增/修改/查詢水果 |
| FR 003 | 驗證/忘記密碼/訂單確認mail |
| FR 004 | 加入購物車/我的最愛 |
| FR 005 | 結帳系統 |
| FR 006 | 驗證是不是水果 |
| FR 007 | 貨品追蹤 |
| FR 008 | 評價系統 |
| FR 009 | 購買/販賣歷史紀錄 |
| FR 010 | 搜尋水果並且動態推薦 |
| FR 011 | 登入時顯示客製化內容 |

## **2.6 非功能性需求 (Non-Functional Requirements)**

|  |  |
| --- | --- |
| **需求編號** | **需求描述** |
| NFR 001 | 資料表設計應正規化節省空間 |
| NFR 002 | 操作即時回應 (使用 async await) |
| NFR 003 | 前端有 error handling |
| NFR 004 | 有跑 CI/CD |
| NFR 005 | 後端有Unit test |
| NFR 006 | 上線前會有壓力測試 |
| NFR 007 | Respond time 不應超過 5秒 |

## **2.7 操作概念 (Operational Concepts)**

### **2.7.1網頁導覽概念 (Anonymous Operational Concepts)**

任何使用者經由**水果訂購系統(FSOS)**可以觀看各種水果以及商家的資料，其  
 中的平台提供**農產品查詢篩選系統**過濾掉不需要的資訊

### **2.7.2顧客使用者操作概念 (Customer Operational Concepts)**

使用者經過**會員註冊與認證系統**登入為顧客的身分之後，可以使用**購物車，**將有  
 意購買的產品加入購物車，若欲購買產品數量不足還有**缺貨農產品通知**來接收第  
 一線的消息，決定購買時利用 **線上訂購**進行交易。

### **2.7.3管理者操作概念(Manager Operational Concepts)**

使用者經過**會員註冊與認證系統**登入為管理者的身分之後，可以利用**農產品  
 上下架系統**進行產品管理。

### **2.7.4系統管理者操作概念(Administrator Operational Concepts)**

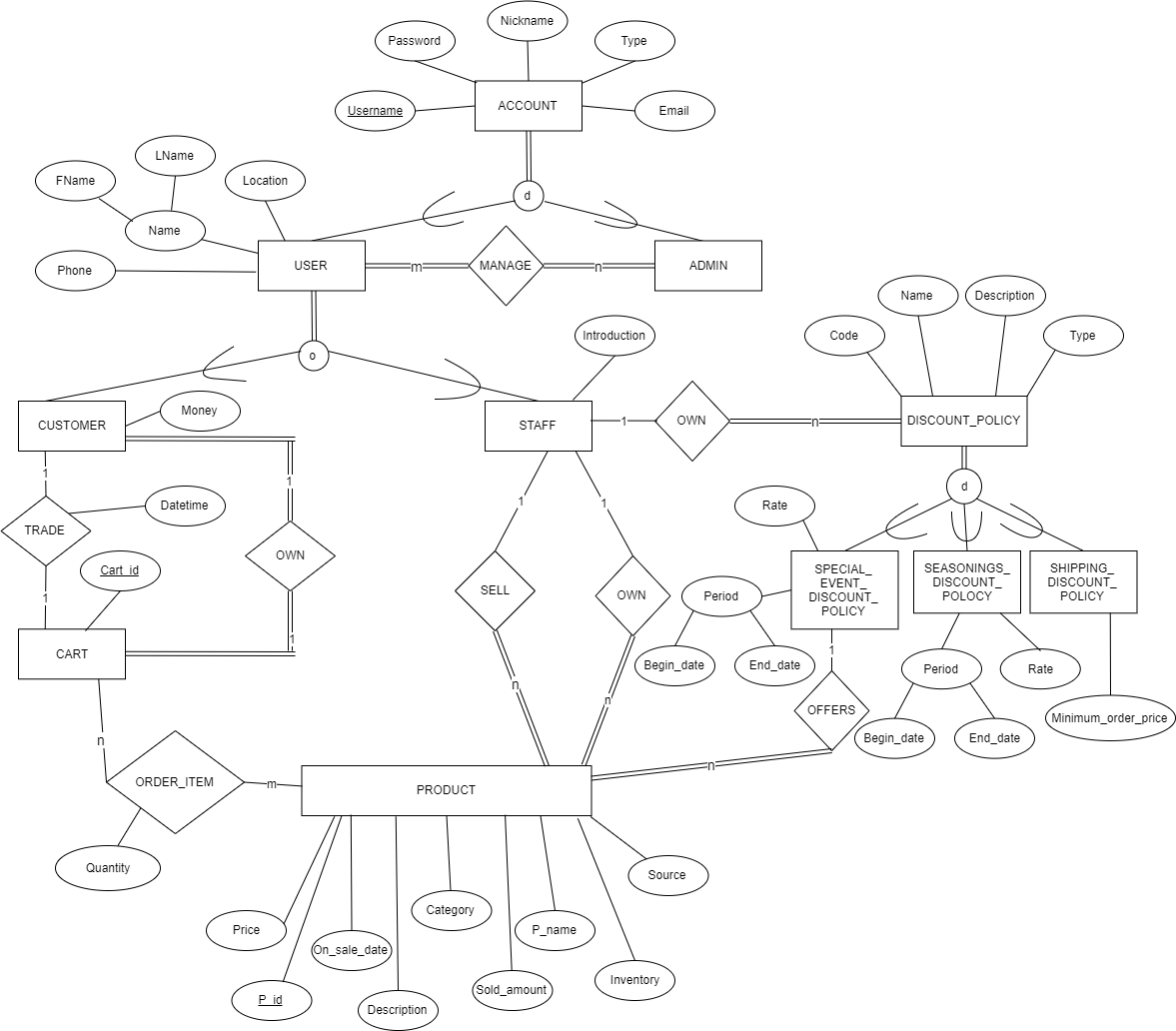
使用者經過**會員註冊與認證系統**登入為系統管理者的身分之後，可以進行帳  
 號資訊的管理，還能對賣家進行**審查系統**。

# Section 3資料庫概念設計

(Conceptual Design of the Database)

## **3.1 Entity-Relationship (ER) Mode**l

# 



# **3.2 Business Rules**

1. Admin為系統直接指定
2. Admin 用於管理使用者，能夠新增或刪除使用者
3. 使用者可同時為顧客和賣家
4. 商品購買時得支付60元運費
5. 賣家可以新增以下折扣
6. 折扣有Special event discount、Seasonings discount、Shipping discount
7. Special event discount為賣家自定義的折扣

* 針對商品
* 使用折扣代碼 Special event 時該賣家指定的商品會依照代碼的優惠進行折扣

1. Seasoning discount為賣家依據季節而進行的折扣

* 針對帳單
* 使用折扣代碼Seasoning時該賣家刊登的所有商品會依照代碼的優惠進行折扣

1. Shipping discount為達到指定價錢可以免運費

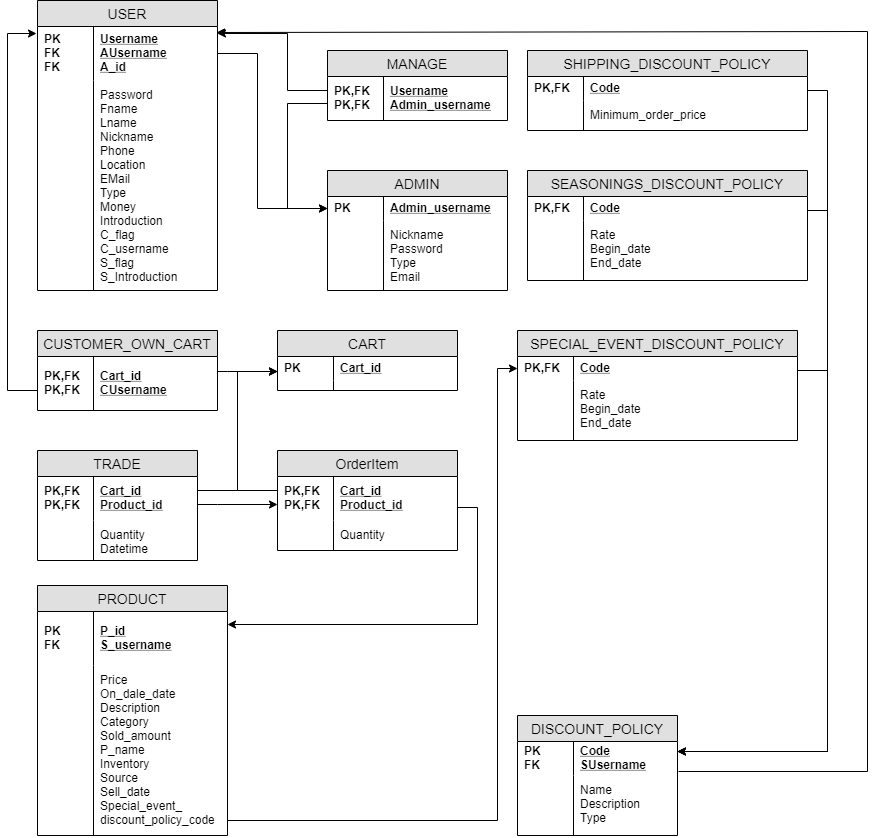
* 針對帳單
* 使用折扣代碼Shipping時該賣家的商品達到所規定的金額後即可免60元的運費

1. 使用者購買商品時，會把商品放入購物車，購物車內能夠修改商品數量或是刪除已選的商品
2. 使用者結帳時會購買購物車內的所有物品
3. 使用者能夠產生報表，內容為訂單的歷史紀錄

# Section 4 邏輯資料庫綱要

(Logical Database Schema)

## **4.1 Schema of the Database**



## **4.2 Domain Definitions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Logical definition** | **Data type** | **Format** | **Format clarification** |
| Nonnegative\_ integers | The set of nonnegative integers. | Integer number |  | The integer number must be nonnegative. |
| Names | The set of names. | Character string |  | The character string must be 1 to 255 characters. |
| Texts | The set of texts. | Character string |  | The character string must be 1 to 1023 characters. |
| Taiwan\_  mobile\_phone\_numbers | The set of ten-digit mobile phone numbers in Taiwan. | Character string | dddd-ddd-ddd | Each d is a number (0-9). The first digit must be 0. |
| Emails | The set of emails. | Character string |  | The character string must be a valid email. |
| Dates | The set of dates. | Character string | yyyy:mm:dd | Each y, m, and d is a number (0-9). Specifically, yyyy, mm, and dd must be a valid year, month, and day of month, respectively. |
| Time | The set of time. | Character string | HH:MM:SS | Each H, M, and S is a number (0-9). Specifically, HH, MM and SS must be a valid hour, minute, and second, respectively. |
| Datetime | The set of Date and Time | Character string | yyyy:mm:dd:HH:MM:SS | Same as above |
| Account\_usernames | The set of account usernames in a website. | Character string |  | The character string must be 6 to 12 characters. It must contains only English letters (a-z/ A-Z) and/ or numbers (0-9). |
| Account\_passwords | The set of account passwords in a website. | Character string |  | The character string must be 8 to 16 characters. It must contains only English letters (a-z/ A-Z) and/ or numbers (0-9). It must contains at least one uppercase English letter, at least one lowercase English letter, and at least one number. |
| Product\_id\_ numbers | The set of nine-digit product identification numbers in a website. | Character string | ddd-ddd-ddd | Each d is a number (0-9). |
| Cart\_id\_ numbers | The set of twelve-digit cart identification numbers in a website. | Character string | dddd-dddd-dddd | Each d is a number (0-9). |
| Discounts | The set of discounts on products provided by a company to its customers. | Character string | ddddddddd | Each d is a number (0-9).or English letters (a-z / A-Z) |

# **4.3 Relation Domains**

ACCOUNT

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Username | Account\_usernames |
| Password | Account\_passwords |
| Nickname | Names |
| Email | Emails |

USER

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Fname | Names |
| Lname | Names |
| Location | Texts |
| Phone | Taiwan\_mobile\_phone\_numbers |

CUSTOMER

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Money | Nonnegative\_integers |

TRADE

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Datetime | Datetime |

CART

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Cart\_id | Cart\_id\_numbers |

PRODUCT

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Price | Nonnegative\_integers |
| P\_id | Product\_id\_numbers |
| OnSaleDate | Dates |
| Description | Texts |
| Category | Texts |
| Pname | Names |
| Inventory | Nonnegative\_integers |
| Source | Texts |

DISCOUNT\_POLICY

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Code | Discounts |

SPECIAL\_EVENT\_DISOUNT\_POLICY

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Code | Discounts |

SHIPPING\_DISCOUNT\_POLICY

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Code | Discounts |

SEASONING\_DISCOUNT\_POLICY

|  |  |
| --- | --- |
| **Attributes** | **Domains** |
| Code | Discounts |

# **4.4 Expectation of Possible Database Operations**

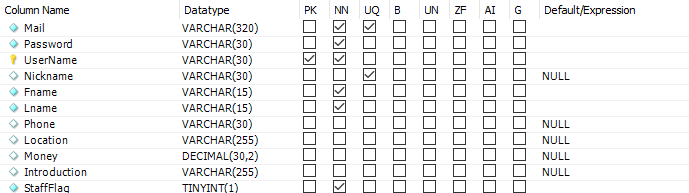
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tables** | **可能操作** | **預估使用頻率**  **(per day)** | **表格資料量**  **(tuples)** | **系統負擔 (worst case per day)** |
| USERS | 使用者登入時 驗證身分 | 100 | 500 | 50,000 Query |
| USERS | 註冊帳號 | 50 | 500 | 50 Insert |
| USERS | 更改使用者帳號密碼 | 50 | 500 | 25,000 Query /  50 Update |
| USERS | 申請升級管理員 | 10 | 500 | 50 Update |
| PRODUCT | 加入商品 | 200 | 5,000 | 200 Insert |
| PRODUCT | 修改商品資訊 | 200 | 5000 | 1,000,000 Query /  200 Update |
| PRODUCT | 刪除商品 | 200 | 5,000 | 1,000,000 Query /  200 Delete |
| PRODUCT | 搜尋商品 | 2,000 | 5,000 | 10,000,000 Query |
| ORDER\_ITEM | 新增商品至 購物車 | 150 | 1,000 | 150 Insert |
| ORDER\_ITEM | 刪除購物車內 商品 | 60 | 1,000 | 60,000 Query /  60 Delete |
| ORDER\_ITEM | 修改購物車內 商品數量 | 60 | 1,000 | 60,000 Query /  60 Update |
| ORDER\_ITEM / PRODUCT | 查看當前購物車商品 | 150 | 1,000 | 150,000 Query |
| TRADE / PRODUCT / CUSTOMER\_OWN\_CART | 取得賣家的訂單 | 75 | 10000 | 750,000 Query |
| USERS / CART | 為每一位用戶 新增購物車 | 150 | 500 | 150 Insert |
| ORDER\_ITEM / PRODUCT / | 從購物車內購買商品 | 75 | 10,000 | 150 Insert /  150,000 Query /  150 Delete |
| ORDER\_ITEM | 查詢訂單 歷史紀錄 | 20 | 10,000 | 200,000 Query |
| TRADE / PRODUCT | 查詢賣出的紀錄 | 75 | 10,000 | 750,000 Query |
| ADMIN | 查詢全部使用者帳號資料 | 10 | 500 | 5,000 Query |
| ADMIN | 刪除使用者 | 5 | 500 | 5 Delete |
| DISCOUNT\_POLICIES | 管理員新增折扣 | 100 | 150,000 | 10,000 Insert |
| DISCOUNT\_POLICIES | 管理員查詢自己的折扣 | 500 | 150,000 | 50,000 Query |
| DISCOUNT\_POLICIES | 使用者使用輸入折扣碼 | 10,000 | 150,000 | 100,000 Query |
| DISCOUNT\_POLICIES | 管理員刪除自己的折扣 | 100 | 150,000 | 10,000 Delete |

# **4.5 SQL Statements for Database Construction**

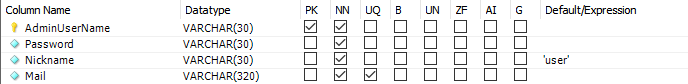
* **CREATE** **TABLE** IF NOT EXISTS users (
* Mail **VARCHAR**(320) NOT NULL,
* **Password** **VARCHAR**(30) NOT NULL,
* UserName **VARCHAR**(30) NOT NULL,
* Nickname **VARCHAR**(30) ,
* Fname **VARCHAR**(15) NOT NULL,
* Lname **VARCHAR**(15) NOT NULL,
* Phone **VARCHAR**(30) ,
* Location **VARCHAR**(255) ,
* Money **DECIMAL**(30, 2) ,
* Introduction **VARCHAR**(255) ,
* StaffFlag BOOLEAN NOT NULL,
* **PRIMARY** **KEY**(UserName),
* **UNIQUE**(Mail),
* **UNIQUE**(Nickname)
* );
* **CREATE** **TABLE** IF NOT EXISTS admin (
* AdminUserName **VARCHAR**(30) NOT NULL,
* **Password** **VARCHAR**(30) NOT NULL,
* Nickname **VARCHAR**(30) NOT NULL **DEFAULT** 'user',
* Mail **VARCHAR**(320) NOT NULL,
* **PRIMARY** **KEY** (AdminUsername),
* **UNIQUE** (Mail)
* );
* **CREATE** **TABLE** IF NOT EXISTS product (
* ProductId **INTEGER** NOT NULL,
* StaffUserName **VARCHAR**(30) NOT NULL,
* Description **VARCHAR**(255) **DEFAULT** 'No description',
* Pname **VARCHAR**(255) NOT NULL,
* Category **VARCHAR**(255) **DEFAULT** 'None',
* Source **VARCHAR**(255) **DEFAULT** 'None',
* Price **INTEGER** NOT NULL,
* Inventory **INTEGER** NOT NULL,
* SoldQuantity **INTEGER** NOT NULL,
* OnSaleDate **DATE** NOT NULL,
* DiscountPolicyCode **CHAR(9)**,
* **PRIMARY** **KEY** (ProductId),
* **FOREIGN** **KEY** (StaffUserName) **REFERENCES** users (UserName) **ON DELETE CASCADE,**
* **FOREIGN** **KEY** (DiscountPolicyCode) **REFERENCES** special\_discount\_policy
* (DiscountPolicyCode) **ON DELETE SET NULL**,
* **CONSTRAINT** p\_id\_non\_negative **CHECK** (ProductId >= 0),
* **CONSTRAINT** price\_non\_negative **CHECK** (Price >= 0),
* **CONSTRAINT** inventory\_non\_negative **CHECK** (Inventory >= 0),
* **CONSTRAINT** sold\_quantity\_non\_negative **CHECK** (SoldQuantity >= 0)
* );
* **CREATE** **TABLE** IF NOT EXISTS cart (
* CartId **INTEGER** NOT NULL **AUTO\_INCREMENT**,
* **PRIMARY** **KEY**(CartId),
* **CONSTRAINT** cart\_id\_non\_negative **CHECK** (CartId >= 0)
* );
* **CREATE** **TABLE** IF NOT EXISTS order\_item (
* CartId **INTEGER** NOT NULL,
* ProductId **INTEGER** NOT NULL,
* Quantity **INTEGER** NOT NULL,
* **PRIMARY** **KEY**(CartId, ProductId),
* **FOREIGN** **KEY**(CartId) **REFERENCES** cart(CartId) **ON DELETE CASCADE**,
* **FOREIGN** **KEY**(ProductId) **REFERENCES** product(ProductId)
* **ON DELETE CASCADE** **ON UPDATE CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS customer\_own\_cart (
* CustomerUserName **VARCHAR**(30) NOT NULL,
* CartId **INTEGER** NOT NULL,
* **PRIMARY** **KEY**(CustomerUserName, CartId),
* **FOREIGN** **KEY**(CustomerUserName) **REFERENCES** users(UserName) **ON DELETE CASCADE**,
* **FOREIGN** **KEY**(CartId) **REFERENCES** cart(CartId) **ON DELETE CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS trade (
* ProductId **INTEGER** NOT NULL,
* CartId **INTEGER** NOT NULL,
* Quantity **INTEGER,**
* **PRIMARY** **KEY**(ProductId, CartId),
* **FOREIGN** **KEY**(ProductId) **REFERENCES** product(ProductId)
* **ON DELETE CASCADE ON UPDATE CASCADE**,
* **FOREIGN** **KEY**(CartId) **REFERENCES** cart(CartId) **ON DELETE CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS manage (
* AdminUserName **VARCHAR**(30) NOT NULL,
* UserName **VARCHAR**(30) NOT NULL,
* **PRIMARY** **KEY**(AdminUserName, UserName),
* **FOREIGN** **KEY**(AdminUserName) **REFERENCES** admin(AdminUserName) **ON DELETE CASCADE**,
* **FOREIGN** **KEY**(UserName) **REFERENCES** users(UserName) **ON DELETE CASCADE**
* );
* **CREATE TABLE** IF NOT EXISTS discount\_policies (
* Code                **CHAR**(9)         NOT NULL,
* **Name**                **VARCHAR**(255)    NOT NULL,
* Description     **VARCHAR**(255)    NOT NULL,
* Type                **VARCHAR**(255)    NOT NULL,
* StaffUserName   **VARCHAR**(30)     NOT NULL,
* **PRIMARY** **KEY**(Code),
* **FOREIGN** **KEY**(Type) **REFERENCES** discount\_policy\_types(**Name**)
* **ON** **UPDATE** **CASCADE**,
* **FOREIGN** **KEY**(StaffUserName) **REFERENCES** users(UserName)
* **ON** **DELETE** **CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS seasonings\_discount\_policies (
* Code            **CHAR**(9)         NOT NULL,
* Rate            **DECIMAL**(3, 2)   NOT NULL,
* BeginDate       **DATE**            NOT NULL,
* EndDate **DATE**            NOT NULL,
* **PRIMARY** **KEY**(Code),
* **FOREIGN** **KEY**(Code) **REFERENCES** discount\_policies(Code)
* **ON** **DELETE** **CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS shipping\_discount\_policies (
* Code                      **CHAR**(9)         NOT NULL,
* MinimumOrderPrice       **DECIMAL**(30, 2)  NOT NULL,
* **PRIMARY** **KEY**(Code),
* **FOREIGN** **KEY**(Code) **REFERENCES** discount\_policies(Code)
* **ON** **DELETE** **CASCADE**
* );
* **CREATE** **TABLE** IF NOT EXISTS special\_event\_discount\_policies (
* Code            **CHAR**(9)         NOT NULL,
* Rate            **DECIMAL**(3, 2)   NOT NULL,
* BeginDate       **DATE**            NOT NULL,
* EndDate **DATE**            NOT NULL,
* **PRIMARY** **KEY**(Code),
* **FOREIGN** **KEY**(Code) **REFERENCES** discount\_policies(Code)
* **ON** **DELETE** **CASCADE**
* );

# **4.6 Implementation of Tables in Target DBMS**

USERS



ADMIN



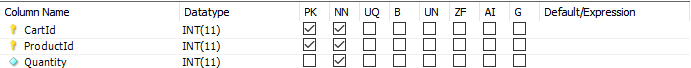
CART



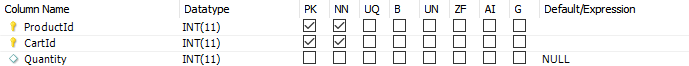
CUSTOMER\_OWN\_CART



ORDER\_ITEM



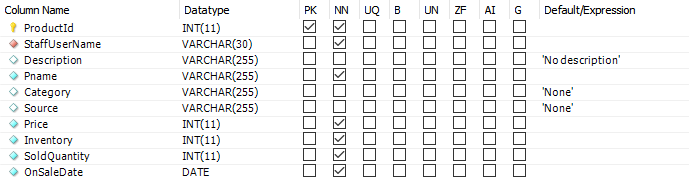
TRADE



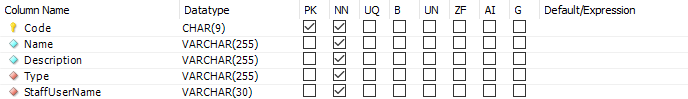
MANAGE



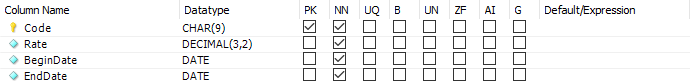
PRODUCT



DISCOUNT\_POLICIES



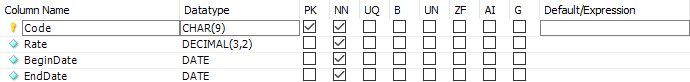
SEASONINGS\_DISCOUNT\_POLICIES



SHIPPING\_DISCOUNT\_POLICIES

https://lh6.googleusercontent.com/5-S_Tf3VE6I3xQje54DItCYovMGuR_F7yf3Ms4eqVfaZnresEzunY7qlyKBGr_IIuvHW9lQoqQtVrN2IUYiny6lBo-iqvTQ7F6PbxhllOA-qStl1nbI9A_oplEErYeVFOIRg2to3

SPECIAL\_EVENT\_DISCOUNT\_POLICIES



# Section 5 Functional Dependencies and

# Database Normalization

**5.1 Functional Dependencies**

|  |  |  |  |
| --- | --- | --- | --- |
| ADMIN | | | |
| UserName | Password | Nickname | Mail |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USER | | | | | |
| UserName | Password | Nickname | Mail | Fname | Lname |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phone | Location | Money | Introduction | Staff\_flag |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRODUCT | | | | |
| ProductId | StaffUserName | Description | Pname | Category |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source | Price | Inventory | SoldQuantity | OnSaleDate |

|  |  |  |
| --- | --- | --- |
| ORDER\_ITEM | | |
| CartId | ProductId | Quantity |

|  |  |  |  |
| --- | --- | --- | --- |
| TRADE | | | |
| CartId | ProductId | Quantity | DateTime |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DISCOUNT\_POLICY | | | | |
| Code | StaffUserName | Name | Description | Type |

|  |  |
| --- | --- |
| SHIPPING\_DISCOUNT\_POLICY | |
| Code | MinimumOrderPrice |

|  |  |  |  |
| --- | --- | --- | --- |
| SEASONINGS\_DISCOUNT\_POLICY | | | |
| Code | Rate | BeginDate | EndDate |

|  |  |  |  |
| --- | --- | --- | --- |
| SPECIAL\_EVENT\_DISCOUNT\_POLICY | | | |
| Code | Rate | BeginDate | EndDate |

# Section 6 The Use of the Database System

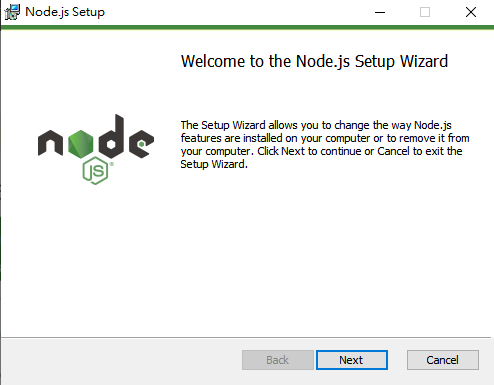
**6.1 System Installation Description**

### **6.1.1 安裝 Node.js**

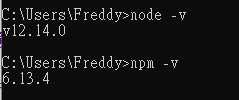
在 Node.js 官網下載安裝檔



Next直到安裝完成

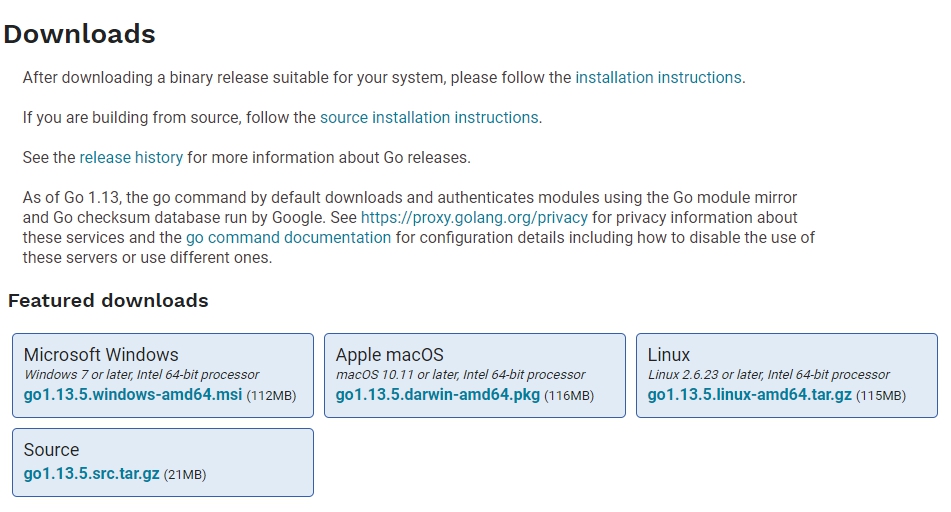
****

開啟CMD，輸入以下指令有回應就代表安裝成功

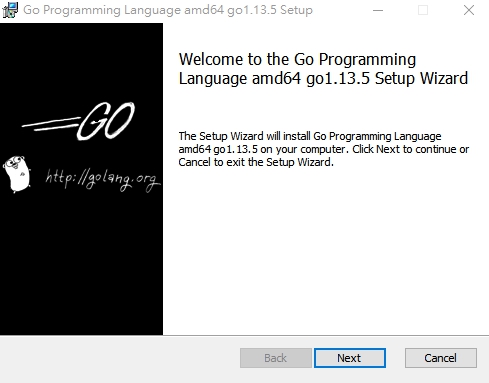
****

### **6.1.2 安裝 Golang**

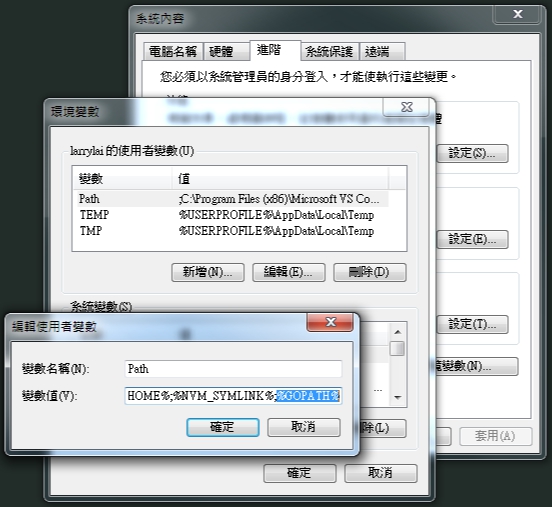
在 Golang官網下載安裝檔



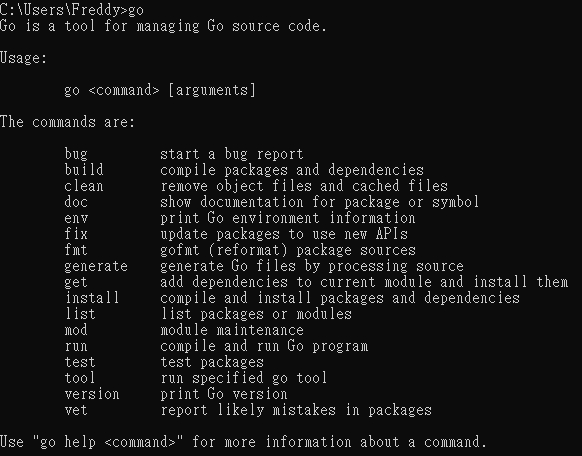
Next直到安裝完成



設定環境變數

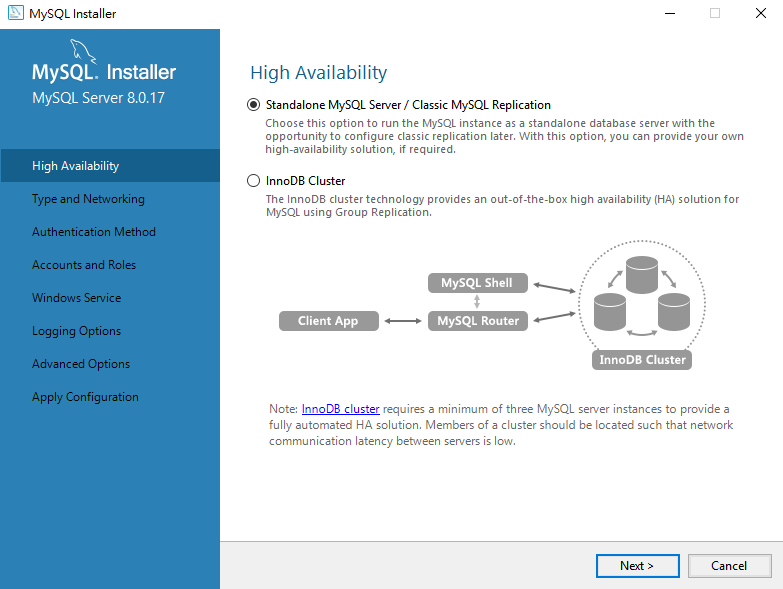


開啟CMD，輸入以下指令有回應就代表安裝成功

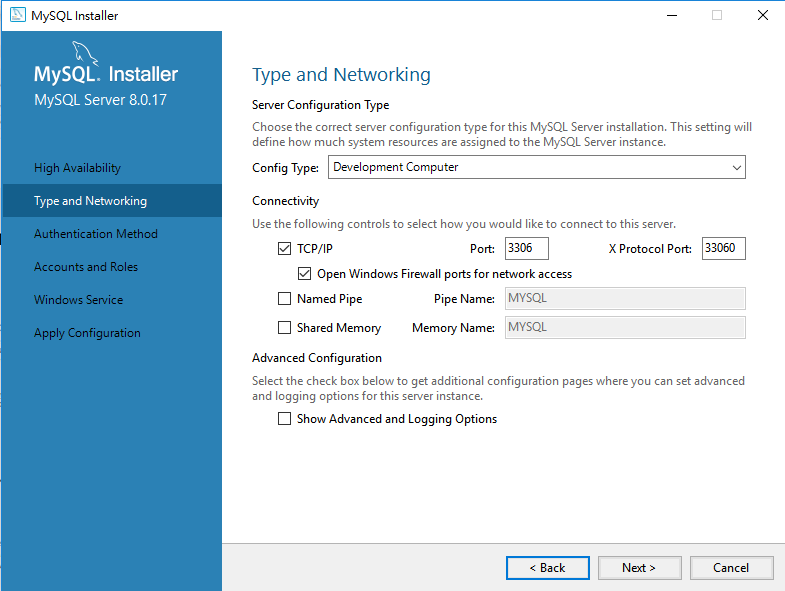


**6.1.3 安裝 MySQL**

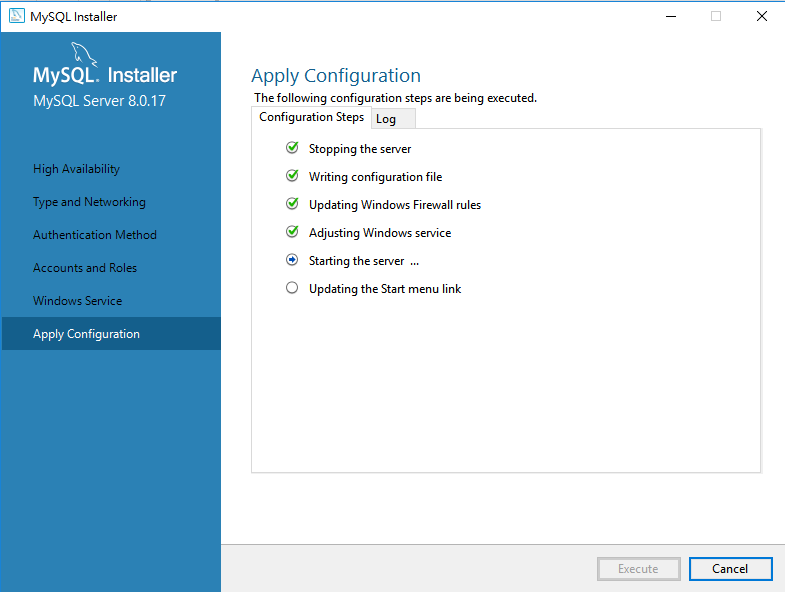
安裝 Standalone MySQL Server / Classic MySQL Replication 版本



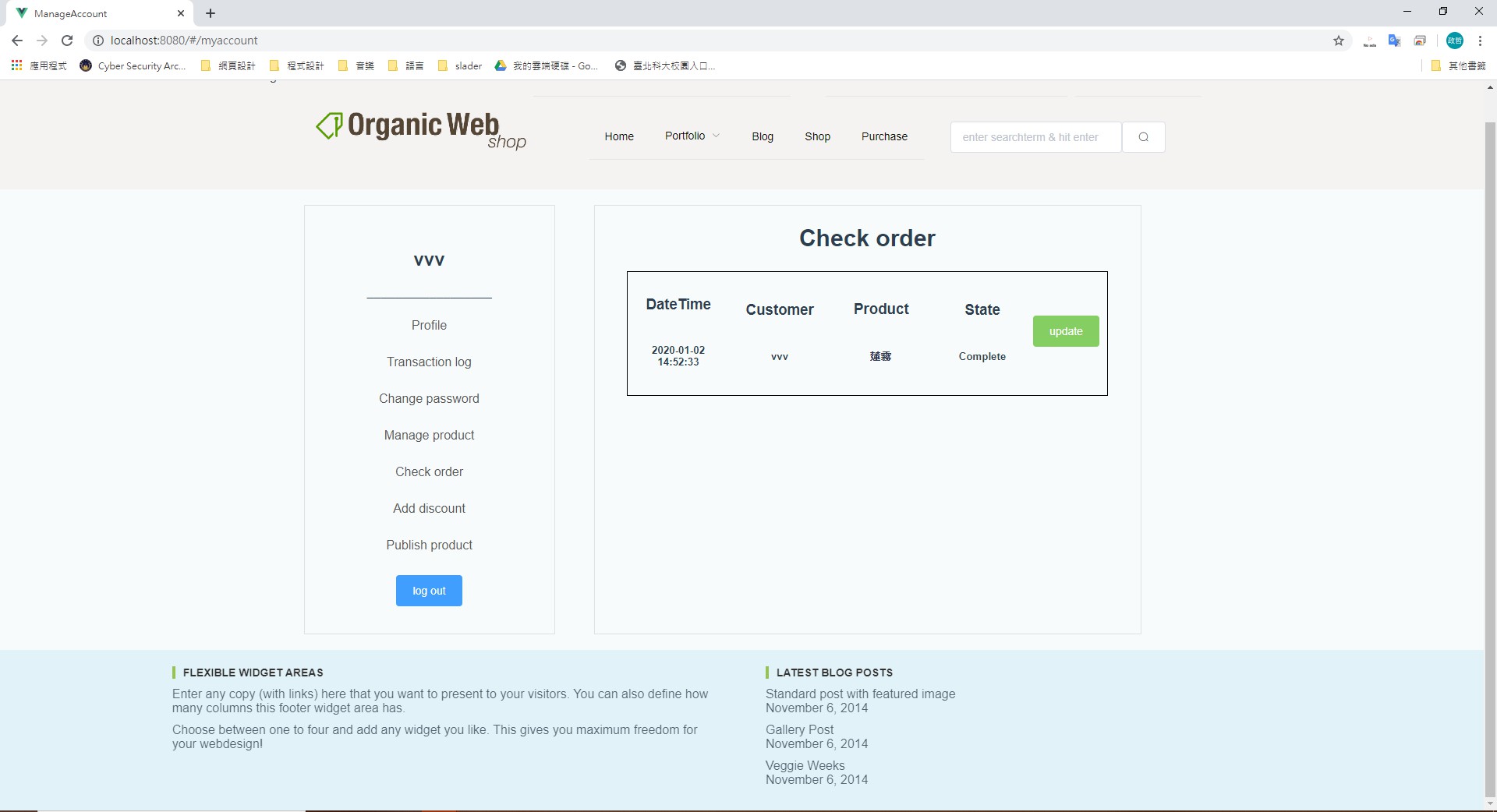
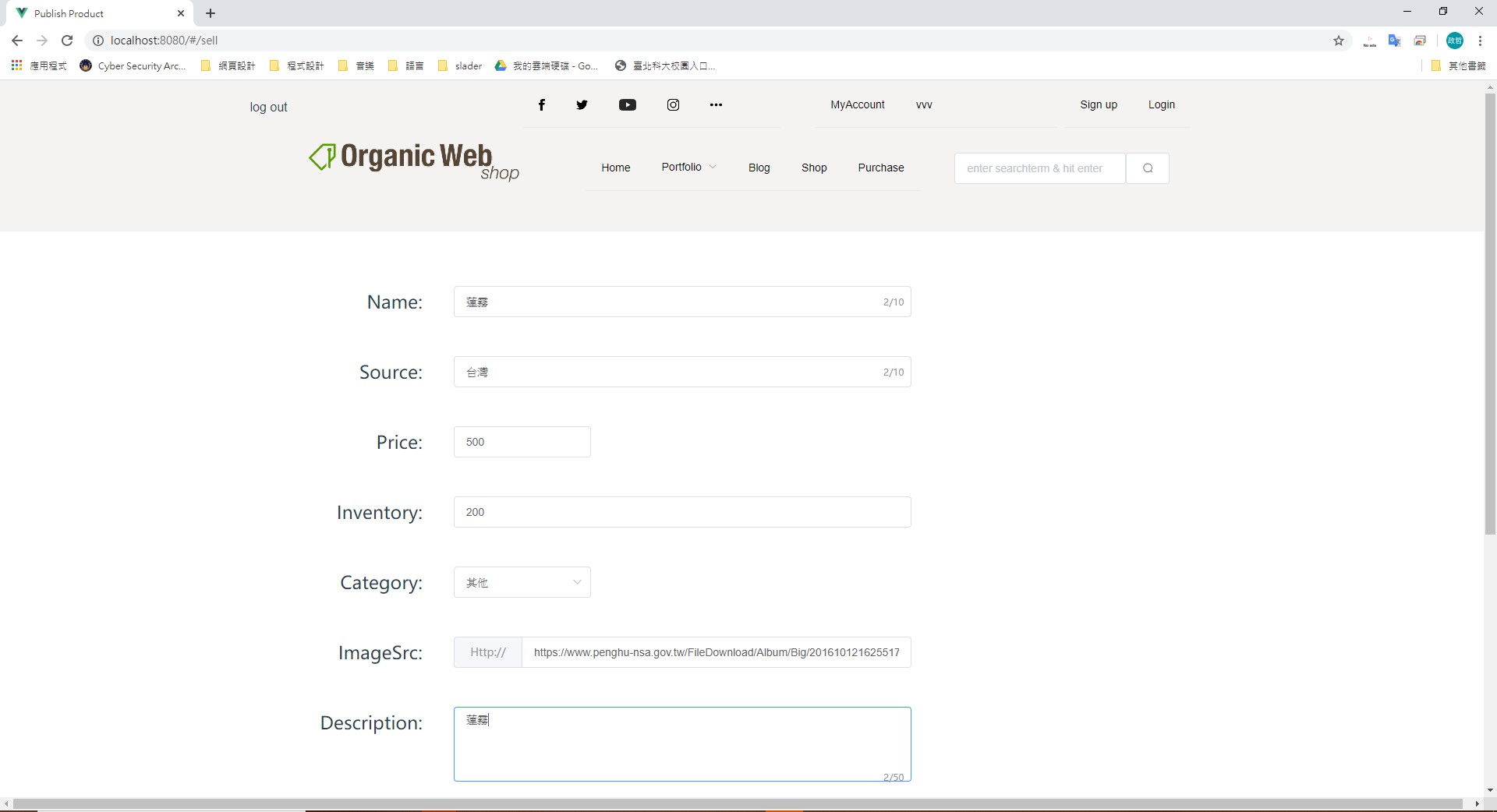
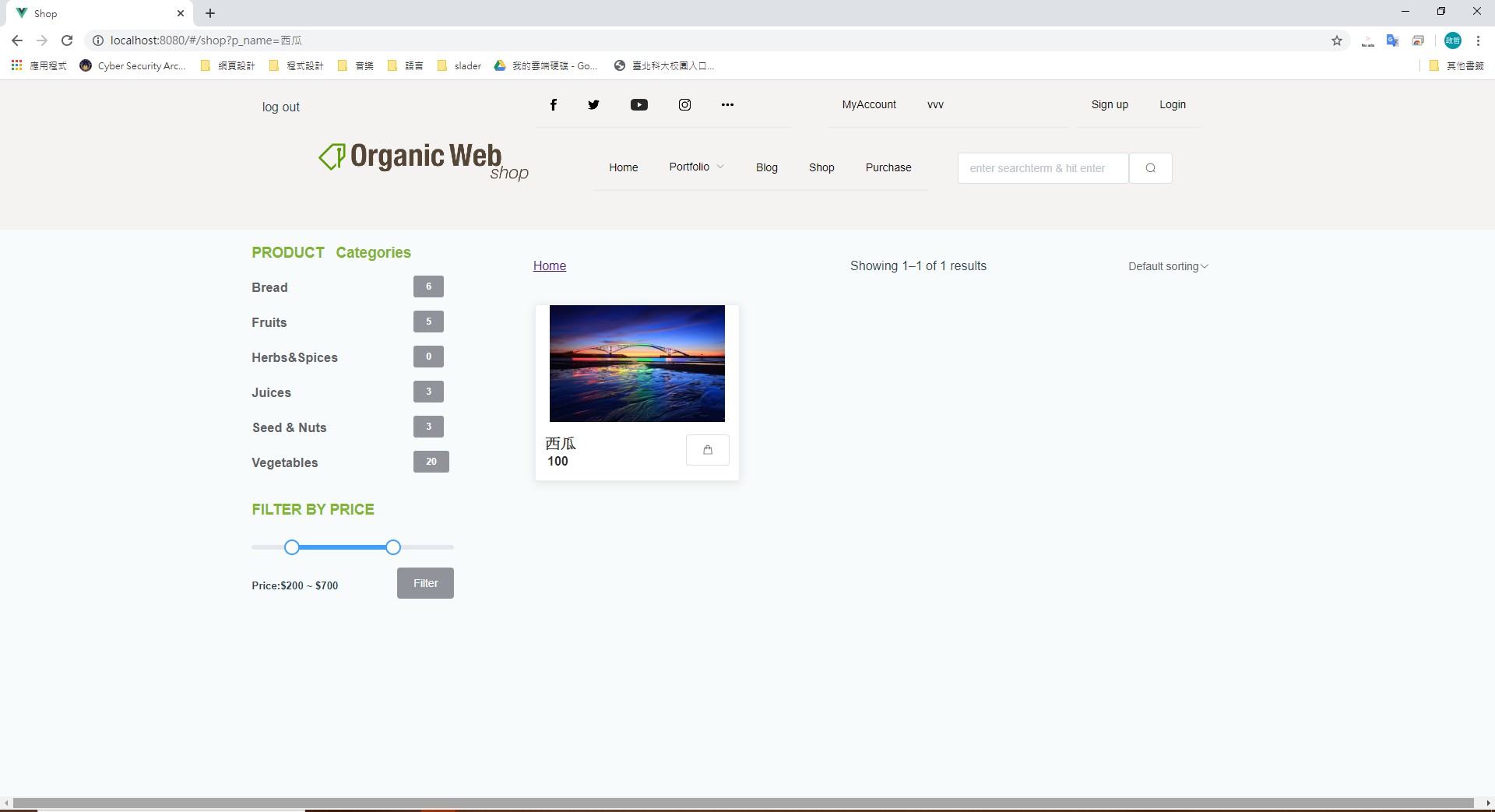
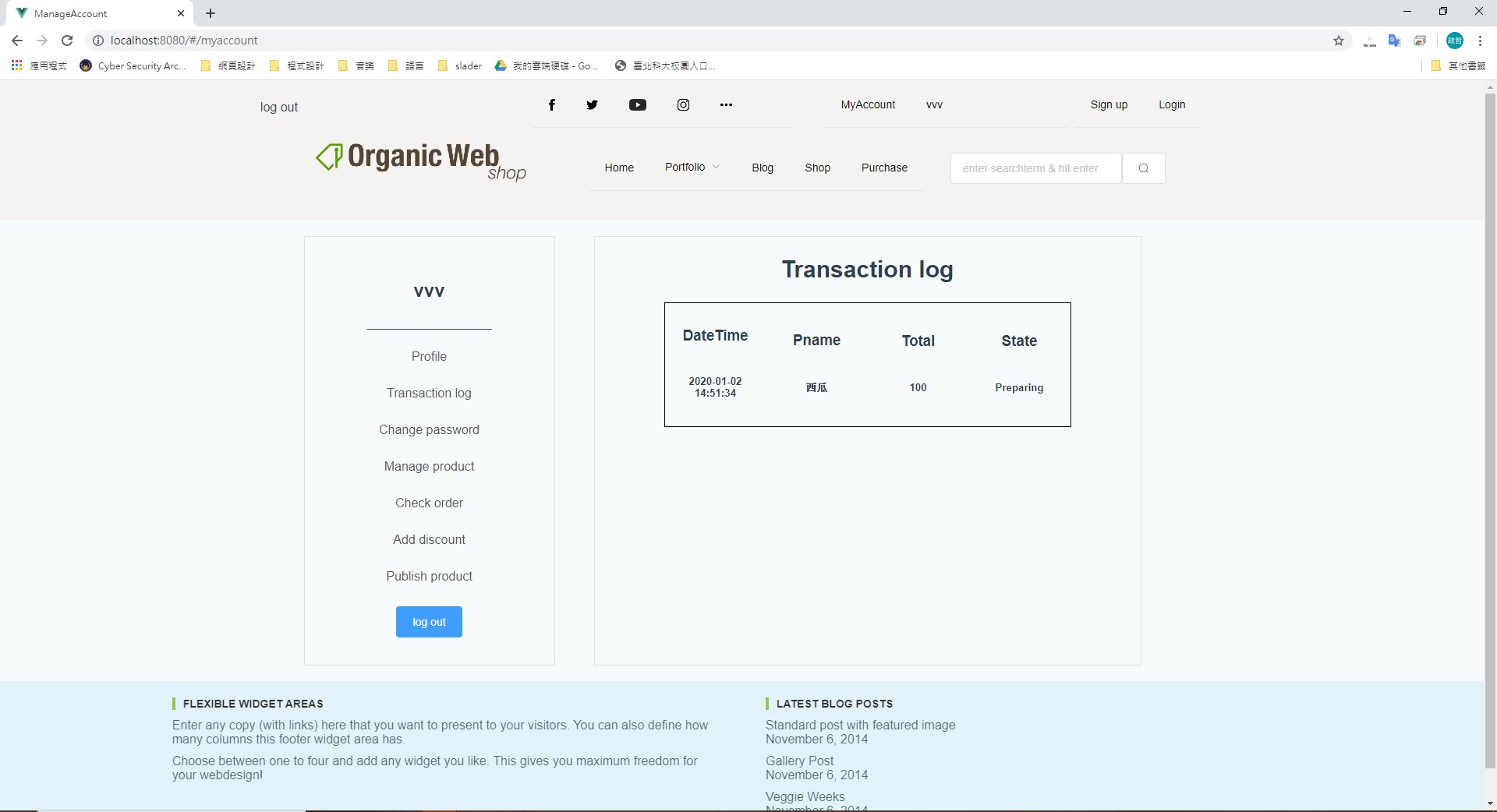
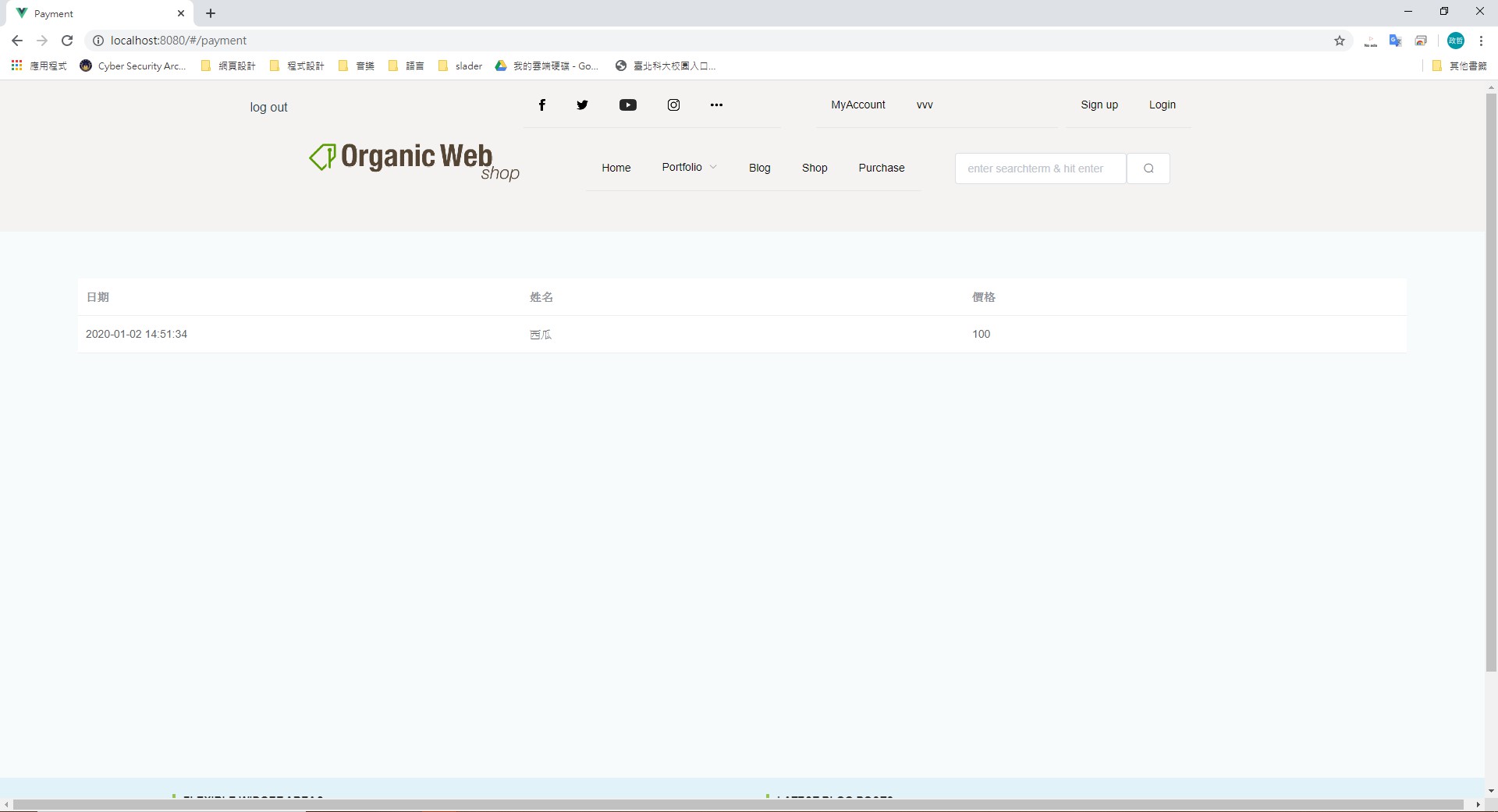
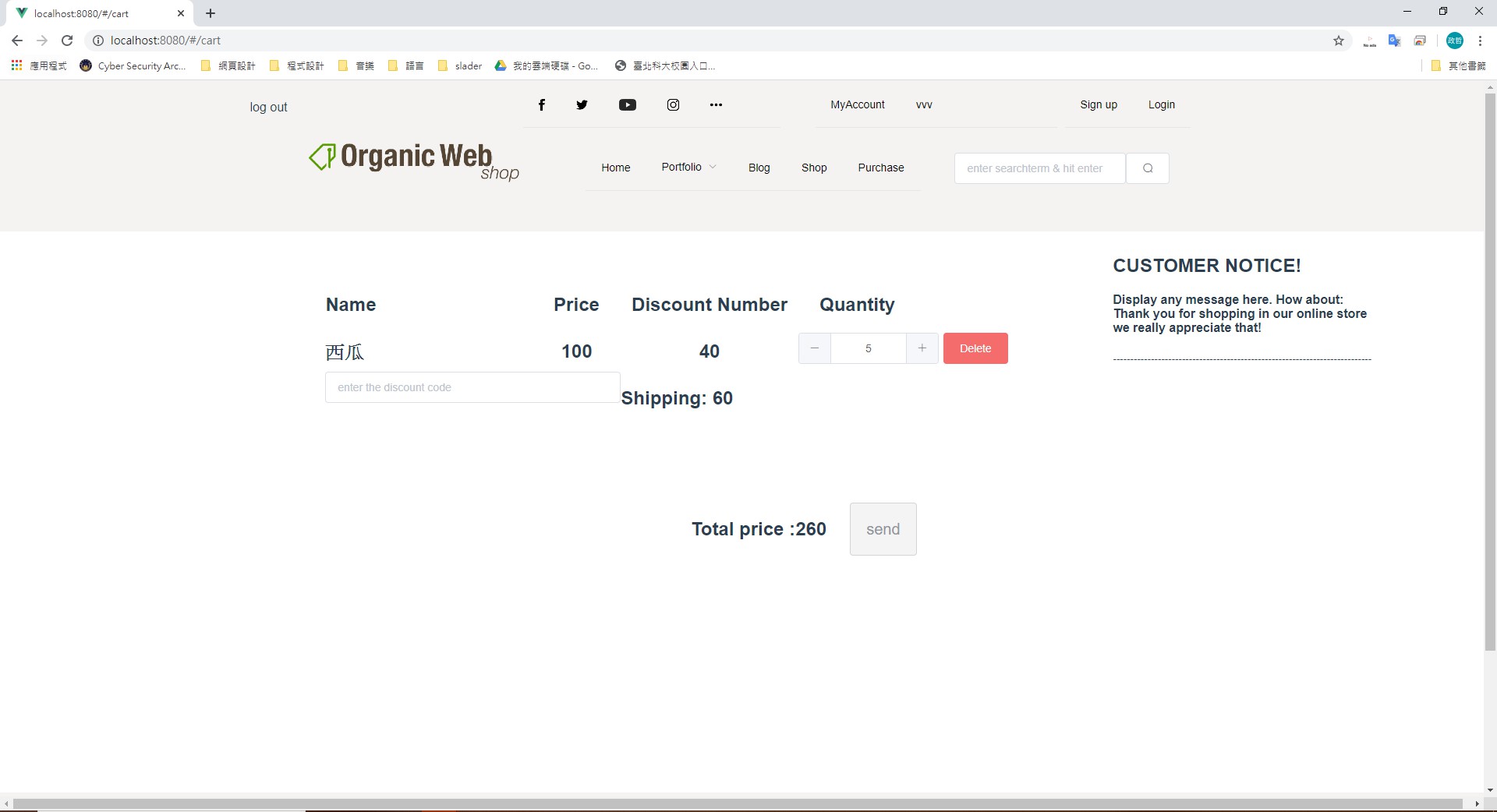
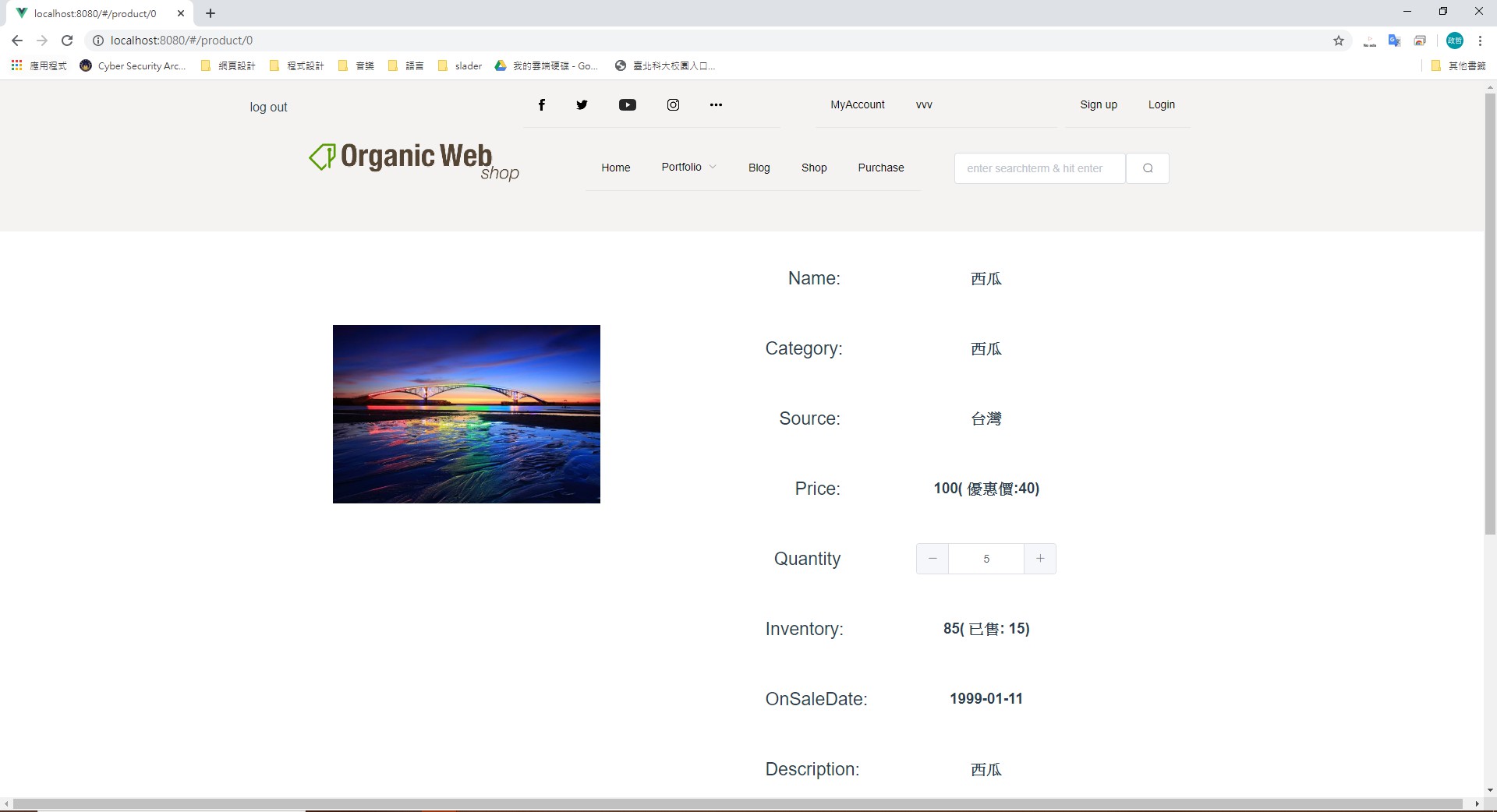
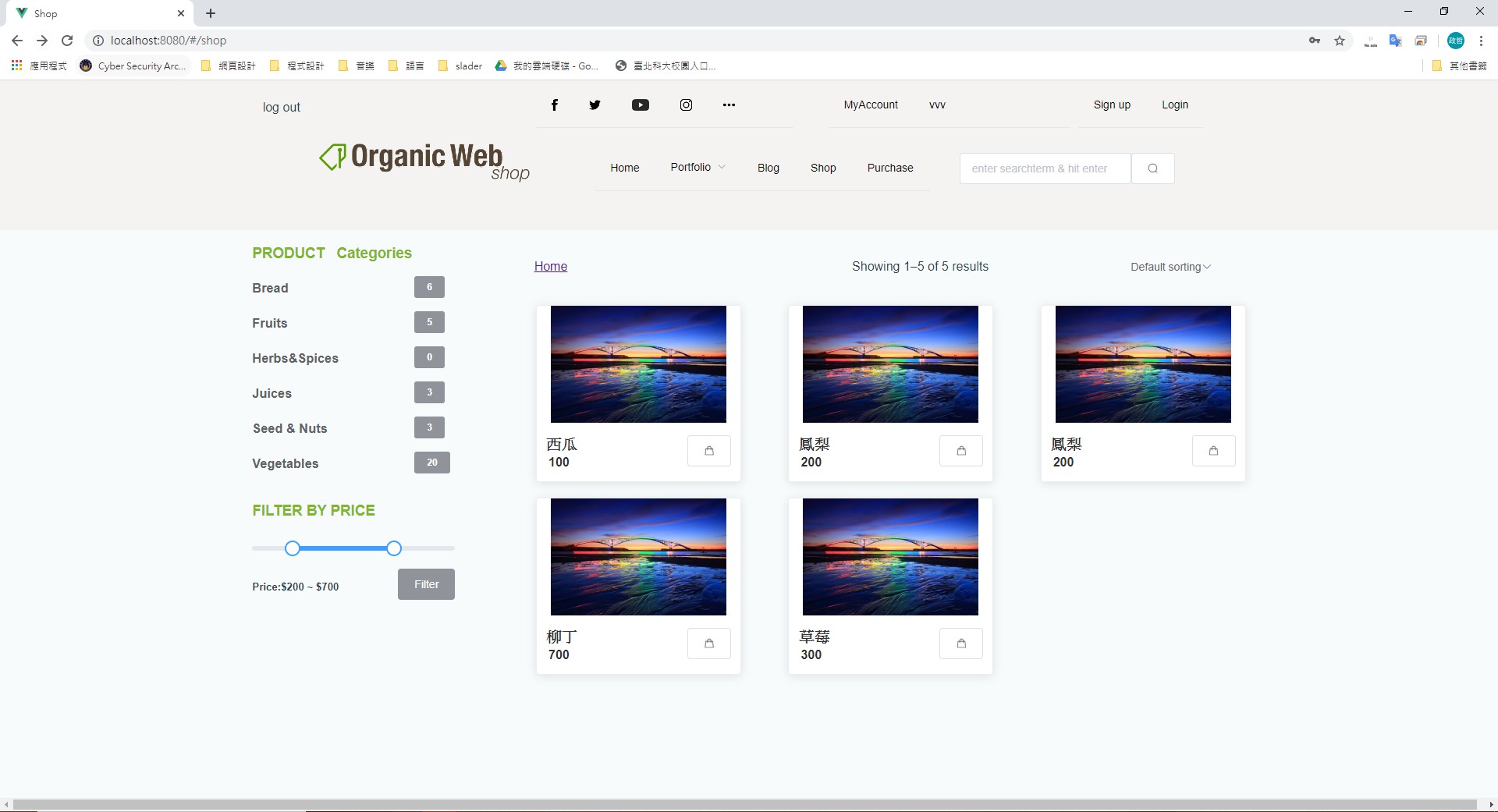
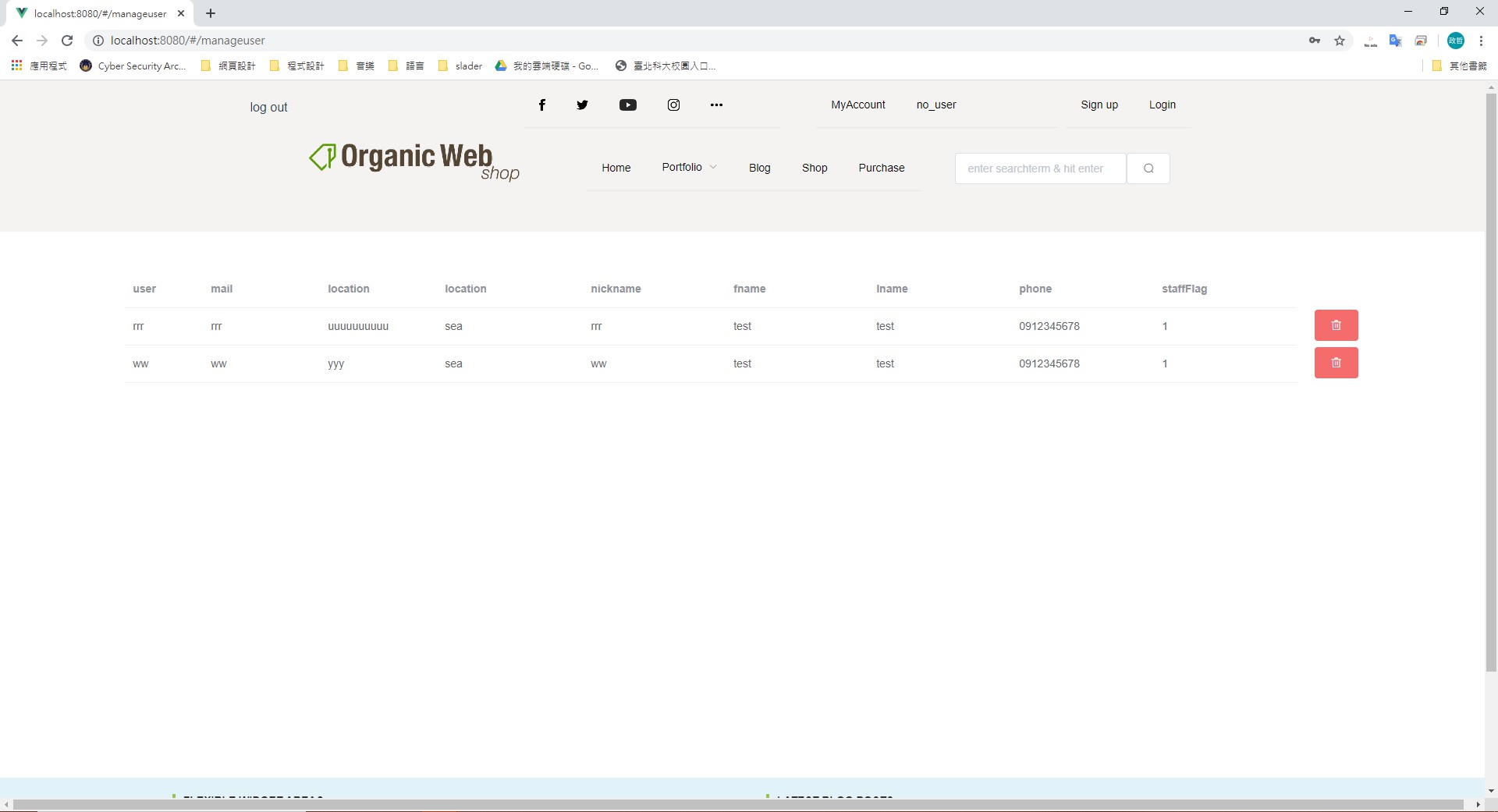
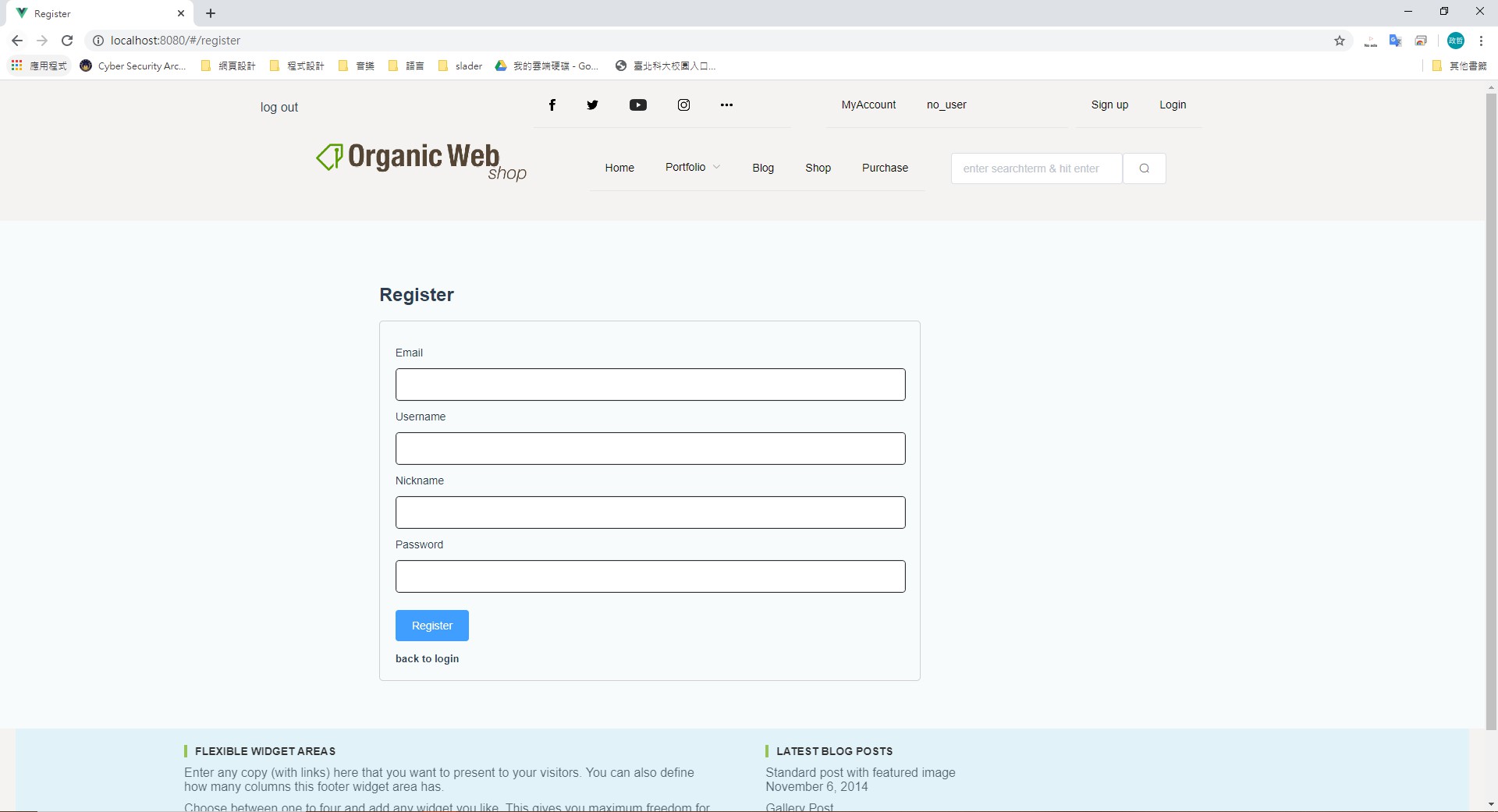
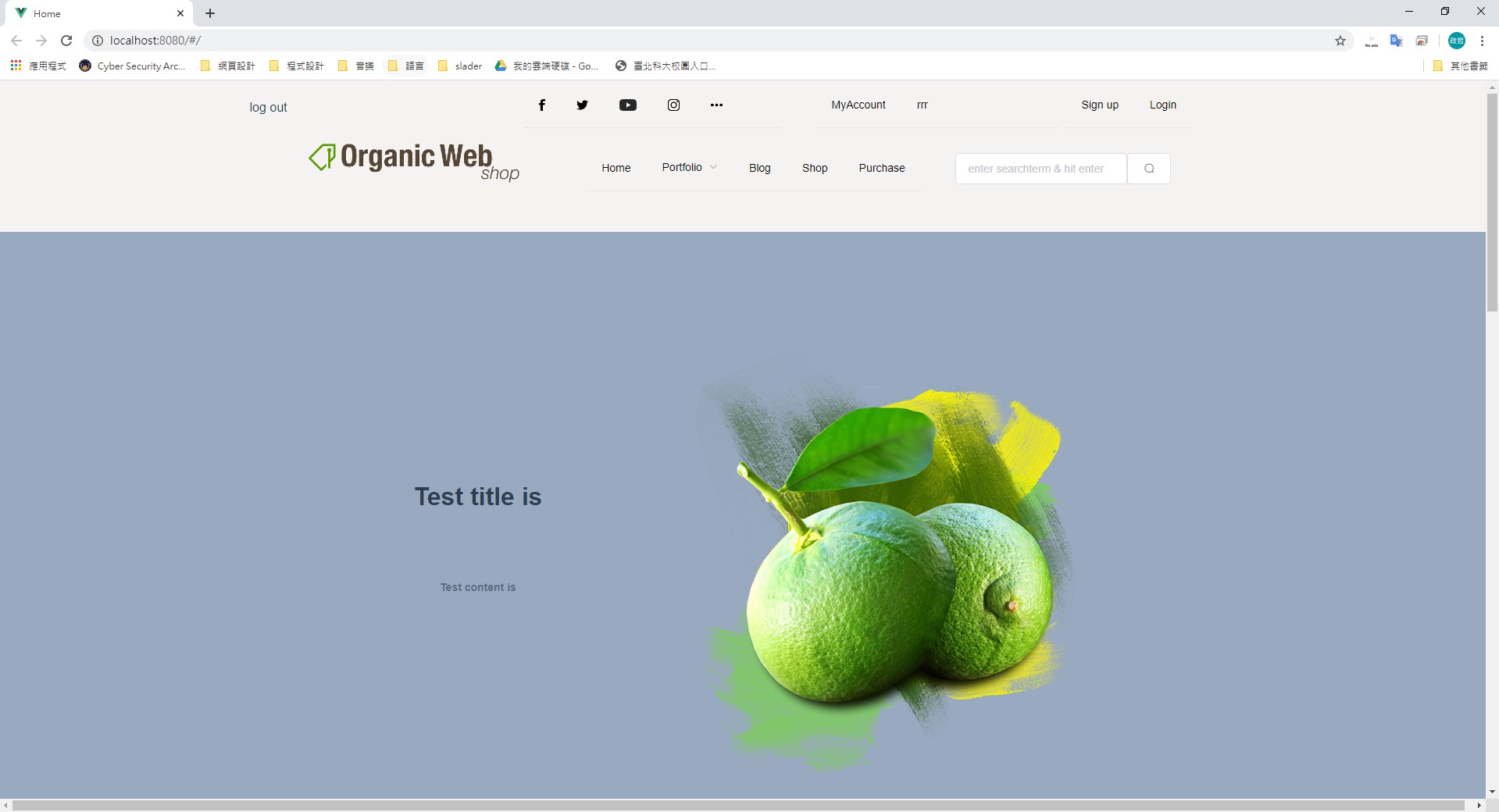
設定參數



安裝



**6.2 The Use of the System**



# Section 7 Suggestions of Database Turning

在這次資料庫的專案中，我們是實作一個 C2C 的購物網站，與商品相關的操作相當的多，有許多Query都必須關聯、Join到商品表格，由於商品表格資料眾多，因此在進行Query時耗時相當的久，因此我們認為可以運用這學期所學，將商品表格變為order file，雖然這麼做會降低Insert、Delete的效率，但我們認為我們的平台Select的次數將會遠大於Insert和Delete，因此將其變為order file是較佳的選擇，並且由於大多數都是與商品的Primary key有關的Query，因此為其建立Primary index最能夠提升效率，而折扣碼也是在商品表格中經常需要用其作Join的欄位，但由於其次數並沒有比Primary key還多，因此不考慮將其作為Clustered index，而是選擇它作為Secondary index，其他沒有需要關聯的屬性我們便不為其建立Secondary index，因為若是這樣做反而會拖低系統效能。

在購物車的部分，由於先前的ER Diagram設計時欠缺考慮，因此產生了多餘的表格 – cart 和 customer\_own\_cart，由於每個User都會有購物車，因此只需要用Username就可以達成購物車ID的功能，設計起來較為簡單，也不會有冗於表格，這部分需要改進。

在我們設計這個資料庫系統時，關於交易的部分，由於所學尚淺，也難以測試過其會發生的狀況，因此我們並未將其納入設計的考量內，例如雙方同時購買，但購買總數超過商品庫存這種情況，應該要拋出異常無法進行交易，這部份是我們需要改進的部分。

# Section 8 Additional Queries and Views

GetOrder

SELECT Pname, Price, Datetime

FROM (SELECT ProductId, Datetime

FROM trade

WHERE CartId = ?) as t join product as p

ON t.Product.Id = p.ProductId

ORDER BY t.Datetime

AddOrderIntoTrade

INSERT INTO trade

SELECT CartId, ProductId, Quantity, Now()

FROM order\_item

WHERE CartId = ?;

ModifyInformaitonsAfterTransaction

UPDATE product, order\_item

SET product.SoldQuantity = product.SoldQuantity + order\_item.Quantity,

WHERE product.ProductId = order\_item.ProductId;

DeleteOrderItemsInCart

DELETE FROM order\_item

WHERE  CartId = ?;

AddCartId

INSERT INTO cart VALUES(); // AUTO\_INCREMENT

AddRowInCustomerOwnCart

INSERT INTO customer\_own\_cart

VALUES (?, (SELECT \* FROM cart ORDEY BY CartId DESC LIMIT 0, 1));

GetCartIdWithUsername

SELECT CartId

FROM customer\_own\_cart

WHERE CustomerUserName

GetStaffOrder

SELECT CustomerUserName, Pname, Quantity, DateTime

FROM trade INNER JOIN product ON trade.ProductId = product.ProductId

INNER JOIN customer\_own\_cart ON trade.CartId =

                                customer\_own\_cart.CartId

WHERE product.StaffUserName = ?

ORDEY BY trade.Datetime;

AddOrderItemToCart

INSERT INTO order\_item

VALUES (?, ?, ?);

DeleteOrderItemInCart

DELETE FROM order\_item

WHERE ProductId = ?

AND CartId = ?;

GetOrderItemsInCart

SELECT order\_item.ProductId, Pname, Price, Quantity, Code

FROM product, order\_item

WHERE CartId = ?

AND order\_item.ProductId = product.ProductId;

ModifyOrderItemQuantity

UPDATE order\_item

SET Quantity = ?

WHERE ProductId = ?

AND CartId = ?;

GetNewProductId

SELECT MAX(ProductId)

FROM product;

AddProduct

INSERT INTO product

VALUES (GetNewProductId()+1, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);

DeleteProduct

DELETE FROM product

WHERE ProductId = ?;

ModifyProduct

UPDATE product

SET StaffUserName = ?,

    Description = ?,

    Pname = ?,

    Category = ?,

    Source = ?,

    Price = ?,

    Inventory = ?,

    SoldQuantity = ?,

    ImageSource = ?,

WHERE ProductId = ?;

QueryProduct

SELECT \*

FROM product

WHERE Pname like %?

OR StaffUserName = %?

OR ProductId = ?;

InsertDiscountPolicy

INSERT INTO discount\_policies

VALUES (?, ?, ?, ?, ?);

InsertUser

INSERT INTO users

VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);

GetAllUsers

SELECT \*

FROM users;

GetUserByUserName

SELECT \*

FROM users

WHERE UserName = ?;

GetUserByMail

SELECT \*

FROM users

WHERE Mail = ?;

UpdateUserPassword

UPDATE users

SET Password = ?

WHERE UserName = ?;

UpdateUserStaffFlag

UPDATE users

SET StaffFlag = ?

WHERE UserName = ?;

DeleteUser

DELETE FROM users

WHERE UserName = ?;

InsertDiscountPolicy

INSERT INTO discount\_policies VALUES(?, ?, ?, ?, ?)

GetDiscountPolicyByCode

SELECT

discount\_policies.Code,

discount\_policies.Name,

discount\_policies.Description,

discount\_policies.Type,

discount\_policies.StaffUserName,

shipping\_discount\_policies.MinimumOrderPrice AS ShippingMinimumOrderPrice,

seasonings\_discount\_policies.Rate AS SeasoningsRate,

seasonings\_discount\_policies.BeginDate AS SeasoningsBeginDate,

seasonings\_discount\_policies.EndDate AS SeasoningsEndDate,

special\_event\_discount\_policies.Rate AS SpecialEventRate,

special\_event\_discount\_policies.BeginDate AS SpecialEventBeginDate,

special\_event\_discount\_policies.EndDate AS SpecialEventEndDate

FROM discount\_policies

     LEFT OUTER JOIN shipping\_discount\_policies

ON discount\_policies.Code = shipping\_discount\_policies.Code

LEFT OUTER JOIN seasonings\_discount\_policies

ON discount\_policies.Code = seasonings\_discount\_policies.Code

LEFT OUTER JOIN special\_event\_discount\_policies

ON discount\_policies.Code = special\_event\_discount\_policies.Code

WHERE discount\_policies.Code = ?

GetStaffDiscountPolicies

SELECT

discount\_policies.Code,

discount\_policies.Name,

discount\_policies.Description,

discount\_policies.Type,

discount\_policies.StaffUserName,

shipping\_discount\_policies.MinimumOrderPrice AS ShippingMinimumOrderPrice,

seasonings\_discount\_policies.Rate AS SeasoningsRate,

seasonings\_discount\_policies.BeginDate AS SeasoningsBeginDate,

seasonings\_discount\_policies.EndDate AS SeasoningsEndDate,

special\_event\_discount\_policies.Rate AS SpecialEventRate,

special\_event\_discount\_policies.BeginDate AS SpecialEventBeginDate,

special\_event\_discount\_policies.EndDate AS SpecialEventEndDate

FROM discount\_policies

LEFT OUTER JOIN shipping\_discount\_policies

ON discount\_policies.Code = shipping\_discount\_policies.Code

LEFT OUTER JOIN seasonings\_discount\_policies

ON discount\_policies.Code = seasonings\_discount\_policies.Code

LEFT OUTER JOIN special\_event\_discount\_policies

ON discount\_policies.Code = special\_event\_discount\_policies.Code

WHERE discount\_policies.StaffUserName = ?

DeleteDiscountPolicy

DELETE FROM discount\_policies

WHERE Code = ?

InsertSeasoningsDiscountPolicy

INSERT INTO seasonings\_discount\_policies

VALUES(?, ?, ?, ?)

InsertShippingDiscountPolicy

INSERT INTO shipping\_discount\_policies

VALUES(?, ?)

InsertSpecialEventDiscountPolicy

INSERT INTO special\_event\_discount\_policies

VALUES(?, ?, ?, ?)

# Section 9 Conclusions and Future Work

**9.1 Conclusions**

在這次資料庫專案中，學到了不少東西，從前後端分離的架構，再到 SQL 的操作。

我們的前端使用的是現在最熱門的框架 Vue.js，而後端使用的也是現在最熱門的 golang，而框架則是選用 gin，在專案初期的時候，大多數人都沒有這兩種框架以及語言的相關經驗，甚至連 js 都不太會寫，所以我們初期在做的事情都是在打地基。

先開一個練習的 repo ，讓大家熟悉 vue.js 以及 gin 這兩個框架，再藉由這樣的練習去熟悉語言本身。另外因為大家的 git 其實也不太熟悉，所以在這過程中也順帶熟悉了 gitflow 的開發流程，以及要如何寫好 git commit message 以及如何開 pr，還好我們在初期有做好這些基本功，讓我們後續的開發有著不錯的效率。

因為大家的開發經驗都不太相同，以及熟悉的協作方式都不太一樣，並且大家能湊在一起討論的時間也沒有想像中的多，所以我們有很多時間都是平行開發，導致有一些溝通上的問題，這些是我們之後必須補齊的。並且因為課程的要求，所以我們還學會了先寫開發需求文件，然後開始設計系統再開發。

而這次也學到如何設計出ER Diagram 以及 model，再將這些 diagram 轉成 scheme ，再轉換成 SQL code，而要如何寫好 SQL query 也是一門學問，另外在寫這些東西的同時，也會一起討論如何避免 SQL injection ，要如何有效解決跟有哪些策略。

最後，開發後端的時候，大多數時間測試的時候都是使用 curl 指令，而不是使用 js 來呼叫，這樣會導致前端在串接的時候，會遇到同源政策的問題，而這問題也會連帶地增加大家的經驗。

**9.2 Future work**

未來展望則是會希望，可以將所有來不及寫的頁面，全部都完成，並且後端所有因為時間緊急的 api 也會開完。除此之虞，也會進行重構，將之前因為趕死線所欠下的技術債一次還清，並且更改架構，將所有服務拆成微服務並且 container 化。並且把前後端測試都加上，還有 ci/cd 也會一並補上。而影片也會重新拍攝，希望能夠將影片拍成類似宣傳影片的樣子，藉此放進履歷裡面。

# Glossary

**MySQL**  
 一個開放原始碼的關聯式資料庫管理系統，原開發者為瑞典的MySQL AB公司，該公司於2008年被昇陽微系統（Sun Microsystems）收購。2009 年，甲骨文公司（Oracle）收購昇陽微系統公司，MySQL成為Oracle 旗下產品。

**HTTP**  
HTTP是一個客戶端（用戶）和伺服器端（網站）之間請求和應答的標準，通常使用TCP協定。透過使用網頁瀏覽器、網路爬蟲或者其它的工具，客戶端發起一個HTTP請求到伺服器上指定埠（預設埠為80）。我們稱這個客戶端為用戶代理程式（user agent）。應答的伺服器上儲存著一些資源，比如HTML檔案和圖像。我們稱這個應答伺服器為源伺服器（origin server）。通常，由HTTP客戶端發起一個請求，建立一個到伺服器指定埠（預設是80埠）的TCP連接。HTTP伺服器則在那個埠監聽客戶端的請求。一旦收到請求，伺服器會向客戶端返回一個狀態，比如"HTTP/1.1 200 OK"，以及返回的內容，如請求的檔案、錯誤訊息、或者其它訊息。

**DBMS**  
是一種針對物件資料庫，為管理資料庫而設計的大型電腦軟體管理系統。具有代表性的資料管理系統有：Oracle、Microsoft SQL Server、Access、MySQL及PostgreSQL等。通常資料庫管理師會使用資料庫管理系統來建立資料庫系統。現代DBMS使用不同的資料庫模型追蹤實體、屬性和關係。在個人電腦、大型電腦和主機上應用最廣泛的資料庫管理系統是關係型DBMS（relational DBMS）。在關係型資料模型中，用二維表格表示資料庫中的資料。這些表格稱為關係。

# References

* element-ui   
  <https://element.eleme.io/#/zh-CN/component/button>
* vuex  
  <https://vuex.vuejs.org/zh/guide/>
* vue  
  <https://cn.vuejs.org/v2/guide/>
* vue router

<https://github.com/vuejs/vue-router>

* async await  
  <https://www.oxxostudio.tw/articles/201908/js-async-await.html>
* golang

<https://golang.org/doc/install>

* gin

<https://github.com/gin-gonic/gin>

* git flow

<https://gitbook.tw/chapters/gitflow/why-need-git-flow.html>

* how to write a good git commit

<https://chris.beams.io/posts/git-commit/>

* mysql doc

<https://dev.mysql.com/doc/>

* jwt

<https://jwt.io/>