CSIE4105 Database Systems

Stella Boutique Online Shop

Final Report
Version: 1.0

姓名	學號	**E-mail**
鄭立杰	107590013	t107590013@ntut.org.tw
王偉斌	107590050	t107590050@ntut.org.tw
陳小蘭	107590058	t107590058@ntut.org.tw
巫啟裕	107590061	t107590061@ntut.org.tw
李芷綺	107590452	t107590452@ntut.org.tw
詹家緯	107590059	t107590059@ntut.org.tw

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

01/06/2021

目錄 (Table of Contents)

Section 1 簡介 (Introduction)

1.1 目的 (Purpose)

透過本學期所修習之資料庫系統課程, 為了驗證資料庫系統之運作以及表格設計的理解程度, 並且結合之前所修習之網頁設計課程與其他程式設計課程之基礎, 嘗試分工完成資料庫系統與使用者終端介面之實作. 購物平台是我們日常生活中最常接觸之資料庫應用軟體, 經過討論, 決定這次的Project名稱為:「Stella Boutique」, 模擬線上服飾銷售網站。

透過此系統能讓使用者在網路上訂購貨品, 節省實體店面之花費以及庫存之問題. 並且透過網頁設計提升商品質感, 增加客人停留與購買之意願, 進而增加訂單量以及銷貨利潤。

1.2 系統名稱 (Identification)

本購物平台的主系統:

線上購物平台系統(Online Shopping Platform System, OSPS)

功能性子系統:

會員管理子系統(Member Management Subsystem, MMS) (Google API)
商品管理子系統(Product Management Subsystem, PMS)
購物車管理子系統(Cart Management Subsystem, CMS)
折扣子系統(Discount Subsystem, DS)
訂單與結帳子系統(Making Order and Payment Subsystem, MOPS)
銷售管理子系統(Financial Management Subsystem, FMS)
顧客訂單子系統(Customer Order Subsystem, COS)
商品瀏覽子系統(Product View Subsystem, PVS)

1.3 概觀 (Overview)

資料庫在近年來已成為應用軟體不可或缺的一部分, 各種網路平台都有它的蹤跡。這次的Project使用MySQL作為DBMS, 使用他所提供之規格實作資料庫之應用。

網站除了功能和效能之外, 使用者體驗(UX)也相當重要, 所以這次我們打算使用React來製作前端, 使用Spring Boot來實作後端之功能。透過這兩者之結合完成一個完整的系統。

1.4 符號描述 (Notation Description)

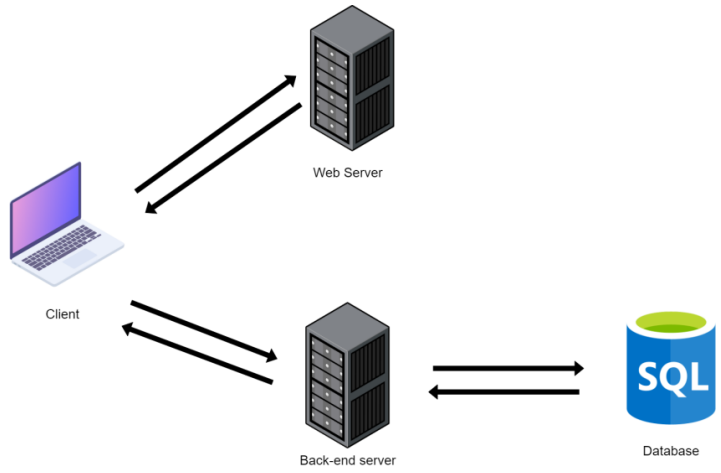
MMS 1.0.n	MMS 以1.0.n 代號表示
PMS 1.1.n	PMS 以1.1.n 代號表示
CMS 1.2.n	CMS 以1.2.n 代號表示
DS 1.3.n	DS 以1.3.n 代號表示
MOPS 1.4.n	MOPS 以1.4.n 代號表示
FMS 1.5.n	FMS 以1.5.n 代號表示
COS 1.6.n	COS 以1.6.n 代號表示
PVS 1.7.n	PVS 以1.7.n 代號表示

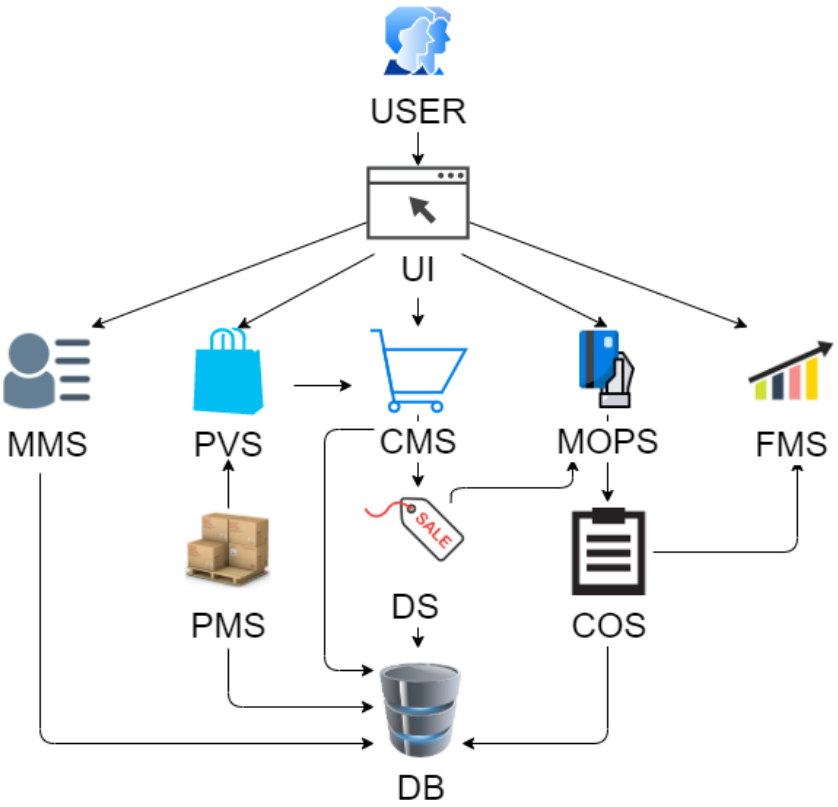
Section 2 系統(System)

2.1 系統描述 (System Description)

系統主要分爲8個部分，分別爲會員管理子系統(Member Management Subsystem, MMS)，商品管理子系統(Product Management Subsystem, PMS)，購物車管理子系統(Shopping Cart Management Subsystem, SCS)，折扣子系統(Discount Subsystem, DS)，訂單與結帳子系統(Making Order and Payment Subsystem, MOPS)，銷售管理子系統(Financial Management Subsystem, FMS)，顧客訂單子系統(Customer Order Subsystem, COS)，商品瀏覽子系統(Product View Subsystem, PVS)

2.1.1 系統架構圖 (System Context Diagram)





2.2 操作概念 **(Operational Concepts)**

Guest: 訪客能夠使用關鍵字搜尋(PVS)想要的商品資料，或者以分類標籤(PVS)根據類別篩選感興趣的商品種類。即使沒有登入, 也可以看到平台上現在有的折扣券(DS), 但無法進行購買的動作。

Buyer: 買家可以透過登入畫面(MMS)註冊帳號並登入, 利用關鍵字搜尋(PVS)想要的商品資 料。網頁側邊會有分類標籤(PVS)根據類別篩選感興趣的商品種類。將想買的東西加入購物車(CMS)以後, 如果持有折扣券(DS), 便可以使用折扣後的價格購買商品。結帳(MOPS)完後， 可以在購買紀錄查看訂單(COS)。

Admin/Seller: 賣家可以在商品管理後臺上架或修改(PMS)想要賣的商品, 並將商品利用分類標籤(PMS)將商品歸類。除了平台上現有的折扣券外, 賣家也 可以自行訂定專屬於賣場商品的優惠(DS)。在訂單列表(COS)中可以管理近期的交易紀錄, 視個人需求也可以利用銷售報表(FMS)來查看賣場的營運狀況。另外也可以使用銷售管理系統(FMS)查看整個網路平台的營收與交易量，當交易出狀況時能使用銷售管理系統(FMS)管理所有的訂單。

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

需求編號	優先順序	需求描述
C-01	1	本系統使用的DBMS為（MySQL）
C-02	1	本系統網頁使用Java連接後端

2.4 技術限制 **(Technological Limitations)**

需求編號	優先順序	需求描述
L-01	1	物流系統（宅配或超商代收）
L-02	1	金流系統（綫上刷卡、貨到付款、LinePay）
L-03	1	會員認證系統

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	優先順序	需求描述
UI-01	1	會員登入介面
UI-02	1	商品清單瀏覽介面
UI-03	1	商品資料顯示介面
UI-04	1	購物車清單介面
UI-05	1	購買下單介面
UI-06	1	會員訂單介面
UI-07	1	賣家管理介面

2.5.2 外部介面需求 (External Interface Requirements)

需求編號	優先順序	需求描述
EI-01	1	使用者使用瀏覽器透過HTTPS安全連線
EI-02	1	透過Nginx 建立Web Server

2.5.3 內部介面需求 (Internal Interface Requirements)

需求編號	優先順序	需求描述
II-01-01	1	會員管理子系統(MMS)能與商品管理子系統(PMS)交換資料
II-01-02	1	會員管理子系統(MMS)能與商品瀏覽子系統(PVS)交換資料
II-01-03	1	會員管理子系統(MMS)能與購物車管理子系統(CMS)交換資料
II-01-04	1	會員管理子系統(MMS)能與折扣子系統(DS)交換資料
II-01-05	1	會員管理子系統(MMS)能與訂單與結帳子系統(MOPS)交換資料
II-01-06	1	會員管理子系統(MMS)能與資料庫(DB)交換資料
II-02-01	1	商品管理子系統(PMS)能與商品瀏覽子系統(PVS)交換資料
II-02-02	1	商品管理子系統(PMS)能與資料庫(DB)交換資料
II-02-03	1	商品瀏覽子系統(PVS)能與購物車管理子系統(CMS)交換資料
II-03-01	1	購物車管理子系統(CMS)能與折扣子系統(DS)交換資料
II-03-02	1	購物車管理子系統(CMS)能與資料庫(DB)交換資料
II-04-01	1	折扣子系統(DS)能與訂單與結帳子系統(MOPS)交換資料
II-04-02	1	折扣子系統(DS)能與資料庫(DB)交換資料
II-05-01	1	訂單與結帳子系統(MOPS)能與顧客訂單子系統(COS)交換資料
II-06-01	1	顧客訂單子系統(COS)能與銷售管理子系統(FMS)交換資料
II-06-02	1	顧客訂單子系統(COS)能與資料庫(DB)交換資料

2.6 功能性需求 **(Functional Requirements)**

需求編號	優先順序	需求描述
F-01	1	使用者使用瀏覽器透過HTTPS安全連線
F-02	1	透過Nginx 建立Web Server

2.7 非功能性需求 **(Performance Requirements)**

2.7.1 效能需求 (Performance Requirements)

需求編號	優先順序	需求描述
N-P-01	1	操作功能順暢

2.7.2 測試需求 (Test Reiquirements)

需求編號	優先順序	需求描述
N-T-01	1	每個子系統功能都應被測試過
N-T-02	1	提供使用者 GUI 介面的測試
N-T-03	1	提供各式平台使用系統的測試

2.8 其他需求 **(Other Requirements)**

2.8.1 環境需求 (Environmental Requirement)

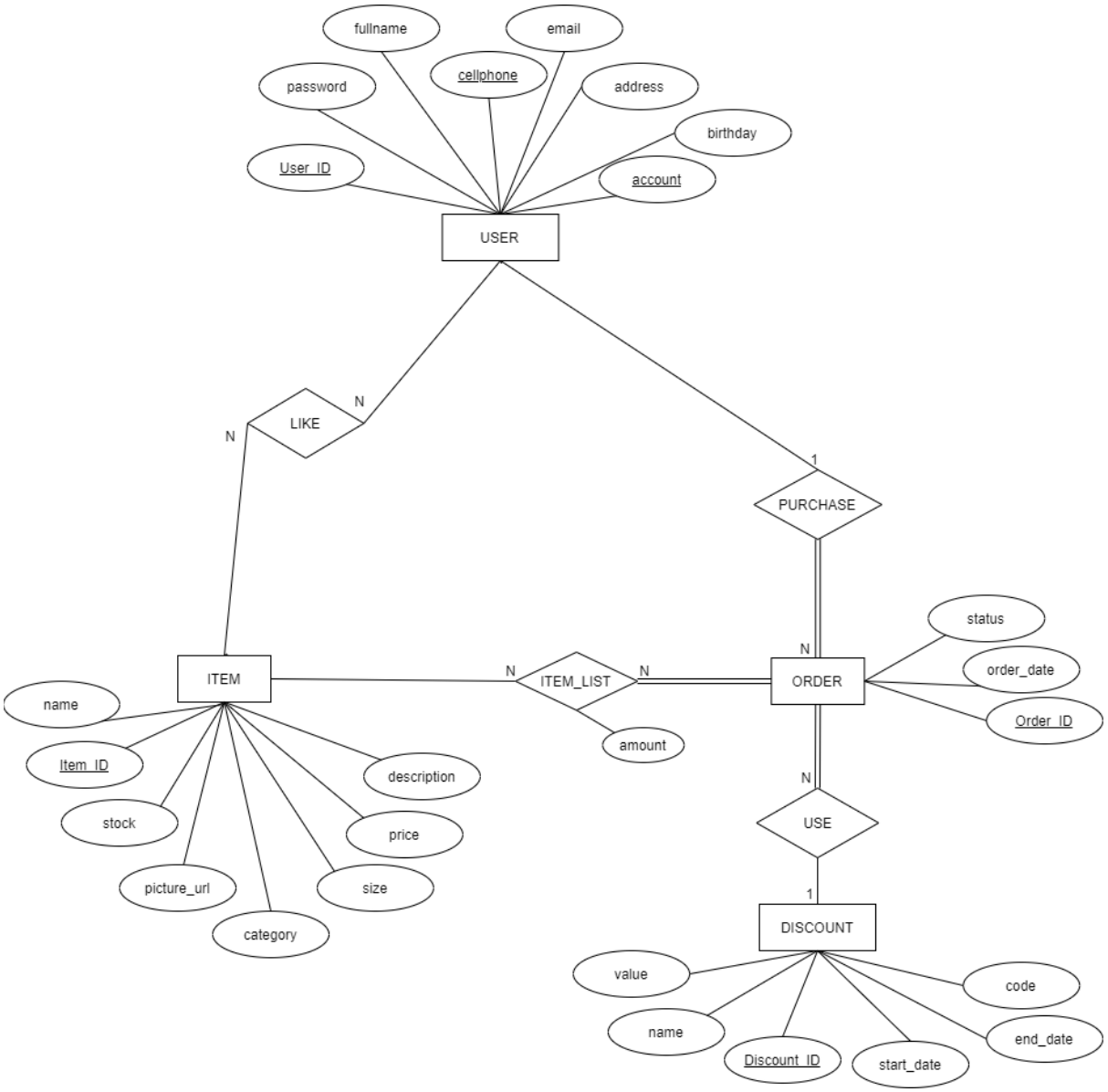
需求編號	優先順序	需求描述
N-E-01	1	伺服器端資料庫系統使用 MYSQL
N-E-02	1	伺服器端設備為 CPU :Intel i5、記憶體: 4GB RAM、硬碟空間 20GB 以上
N-E-03	1	客戶端硬體設備為 Intel P4 3.0GHz 以上及硬碟空間 1GB 以上

2.8.2 安裝需求 (Installation Requirement)

需求編號	優先順序	需求描述
N-I-01	1	伺服器端網頁伺服器採用Spring-Boot與 React (+Redux)
N-I-02	1	資料庫需建立資料表

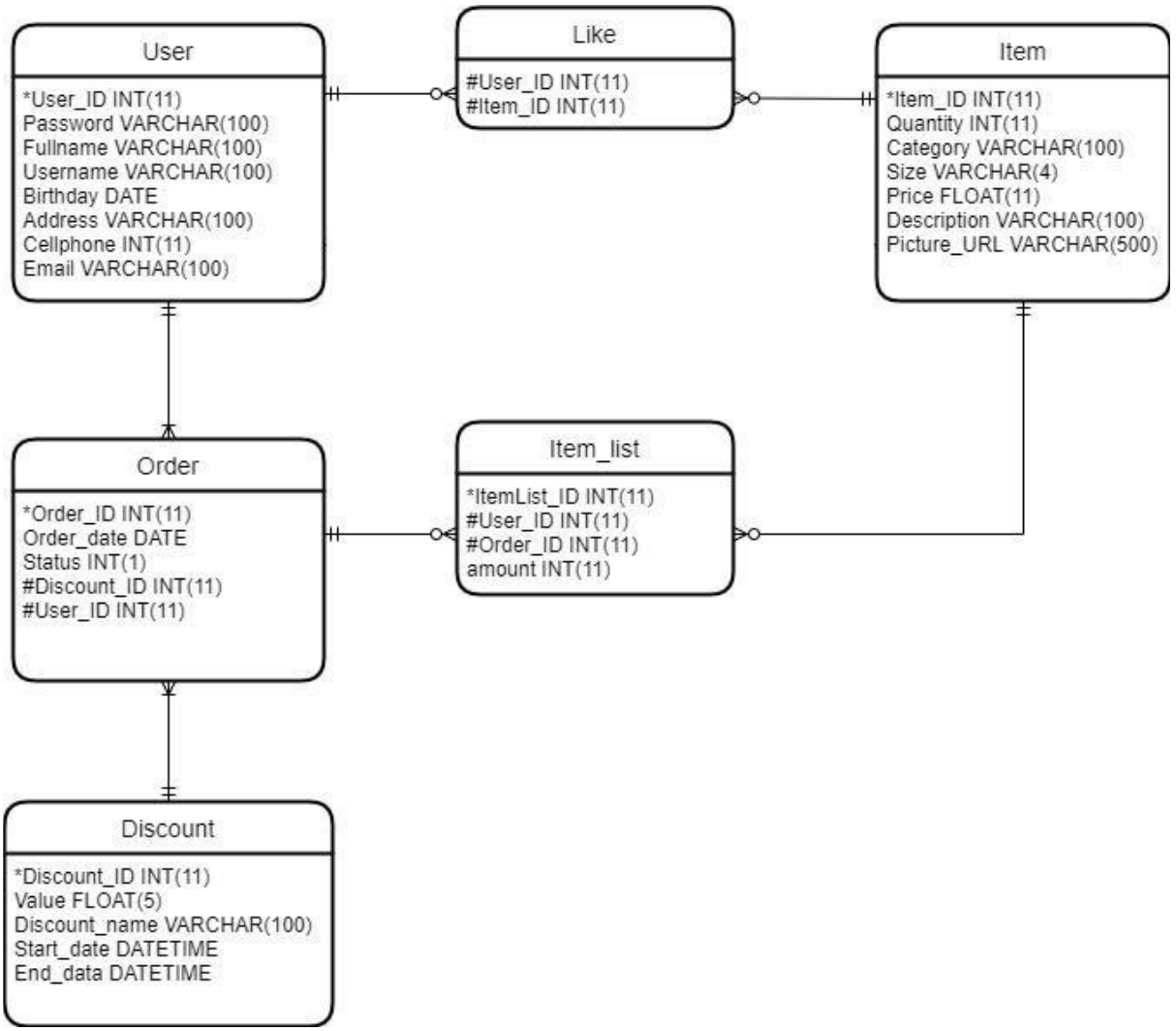
Section 3 資料庫概念設計(Conceptual Design of the Database)

3.1 Entity-Relationship (ER) Model



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

4.1 Schema of the Database



(*Primary key)
(#Foreign Key)

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

Table	可能操作	預估使 用頻率 (per day)	表格資料量 (tuples)	系統負擔 (worst case)
User	新增會員資料	10	Don’t care	10次Insert / day
User	更新會員資料	50	200	10000 次 Query / day 50 次 Update / day
User	查詢會員資料	50	200	10000 次 Query / day 50 次 Update / day
Item	新增商品資料	5	Don’t care	5次Insert / day
Item	更新商品資料	2	Don’t care	2次Update / day
Item	查詢商品資料	100	5000	500000次Query / day
Like	新增點贊資訊	25	Don’t care	25次Insert / day
Like	查詢點贊資訊	25	Don’t care	25次Query / day
Like	刪除點贊資訊	5	Don’t care	5次Delete / day
Order	新增訂單資料	50	5000	50次Insert / day
Order	更新追蹤訂單資訊	50	5000	5000次Query / day
Order	查詢訂單資料	100	5000	25000次Query / day
Order	刪除追蹤訂單資訊	50	5000	50次Delete / day
Discount	新增折扣資訊	10	100	10次Insert / day
Discount	查詢折扣資訊	20	200	20次Query / day
Discount	更新折扣資訊	5	50	50次Update / day
Discount	刪除折扣資訊	10	100	10次Delete / day
Item_list	新增庫存清單資訊	5	Don’t care	5次Insert / day
Item_list	查詢庫存清單資訊	2	Don’t care	2次Update / day
Item_list	更新庫存清單資訊	100	5000	50000次Query / day

4.3 SQL Statements Used to Construct the Schema

```
-- Database: `stella_boutique`
CREATE DATABASE IF NOT EXISTS stella_boutique;
USE stella_boutique;
```

```
-- Table structure for table `user`
CREATE TABLE IF NOT EXISTS `user` (
  `id` int(6) NOT NULL,
  `password` varchar(100) NOT NULL,
  `fullname` varchar(100) NOT NULL,
  `username` varchar(100) NOT NULL,
  `birthday` varchar(10) NOT NULL,
  `address` varchar(100) NOT NULL,
  `phoneNumber` varchar(10) NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY(`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `username_UNIQUE` (`username`)
);
```

```
-- Table structure for table `item`
CREATE TABLE IF NOT EXISTS `item` (
  `id` int(6) NOT NULL,
  `name` varchar(100) NOT NULL,
  `quantity` int(11) NOT NULL,
  `category` varchar(100) NOT NULL,
  `size` varchar(5) NOT NULL,
  `price` float NOT NULL,
  `description` varchar(100) DEFAULT NULL,
  `pictureURL` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
);
```

```
-- Table structure for table `discount`
CREATE TABLE IF NOT EXISTS `discount` (
  `id` int(6) NOT NULL,
  `value` float NOT NULL,
  `code` varchar(5) NOT NULL,
  `name` varchar(100) NOT NULL,
  `startDate` datetime NOT NULL,
  `endDate` datetime NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `code_UNIQUE` (`code`)
);
```

```
-- Table structure for table `like`
CREATE TABLE IF NOT EXISTS `like` (
  `userID` int(6) NOT NULL,
  `itemID` int(6) NOT NULL,
  PRIMARY KEY (`userID`,`itemID`),
  UNIQUE KEY `userID_UNIQUE` (`userID`),
  UNIQUE KEY `itemID_UNIQUE` (`itemID`),
  CONSTRAINT `userID` FOREIGN KEY (`userID`) REFERENCES `user` (`id`),
  CONSTRAINT `itemID` FOREIGN KEY (`itemID`) REFERENCES `item` (`id`)
);
```

```
-- Table structure for table `order`
CREATE TABLE IF NOT EXISTS `order` (
  `id` int(6) NOT NULL,
  `orderDate` datetime NOT NULL,
  `status` int(11) NOT NULL,
  `discountID` int(6) DEFAULT NULL,
  `orderUserID` int(6) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  CONSTRAINT `discountID` FOREIGN KEY (`discountID`) REFERENCES `discount` (`id`),
  CONSTRAINT `orderUserID` FOREIGN KEY (`orderUserID`) REFERENCES `user` (`id`)
);
```

```
-- Table structure for table `itemlist`
CREATE TABLE IF NOT EXISTS `itemlist` (
  `orderItemID` int(6) NOT NULL,
  `orderID` int(6) NOT NULL,
  `amount` int(11) NOT NULL,
  PRIMARY KEY (`orderItemID`,`orderID`),
  UNIQUE KEY `orderItemID_UNIQUE` (`orderItemID`),
  UNIQUE KEY `orderID_UNIQUE` (`orderID`),
  CONSTRAINT `orderItemID` FOREIGN KEY (`orderItemID`) REFERENCES `item` (`id`),
  CONSTRAINT `orderID` FOREIGN KEY (`orderID`) REFERENCES `order` (`id`)
);
```

4.4 The implementation of tables in target DBMS

User

Field	Type	Null	Key	Default
id	int	NO	PRI	NULL
password	varchar(100)	NO		NULL
fullname	varchar(100)	NO		NULL
username	varchar(100)	NO	UNI	NULL
birthday	varchar(10)	NO		NULL
address	varchar(100)	NO		NULL
phoneNumber	varchar(10)	NO		NULL
email	varchar(100)	NO		NULL

Item

Field	Type	Null	Key	Default
id	int	NO	PRI	NULL
name	varchar(100)	NO		NULL
quantity	int	NO		NULL
category	varchar(100)	NO		NULL
size	varchar(5)	NO		NULL
price	float	NO		NULL
description	varchar(100)	YES		NULL
pictureURL	varchar(500)	YES		NULL

Discount

Field	Type	Null	Key	Default
id	int	NO	PRI	NULL
value	float	NO		NULL
code	varchar(5)	NO	UNI	NULL
name	varchar(100)	NO		NULL
startDate	datetime	NO		NULL
endDate	datetime	NO		NULL

Like

Field	Type	Null	Key	Default	Extra
userID	int	NO	PRI	NULL	
itemID	int	NO	PRI	NULL	

Order

	Field	Type	Null	Key	Default
▶	id	int	NO	PRI	NULL
	orderDate	datetime	NO		NULL
	status	int	NO		NULL
	discountID	int	YES	MUL	NULL
	orderUserID	int	NO	MUL	NULL

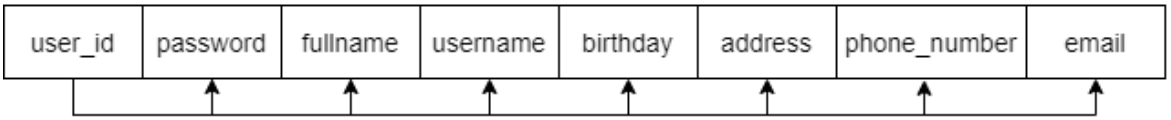
ItemList

	Field	Type	Null	Key	Default
▶	id	int	NO	PRI	NULL
	orderDate	datetime	NO		NULL
	status	int	NO		NULL
	discountID	int	YES	MUL	NULL
	orderUserID	int	NO	MUL	NULL

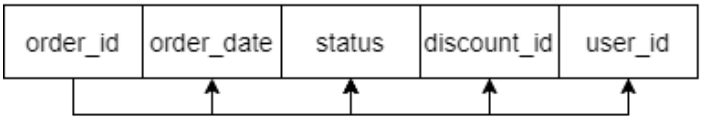
Section 5 Functional Dependencies and Database Normalization

5.1 Functional Dependencies

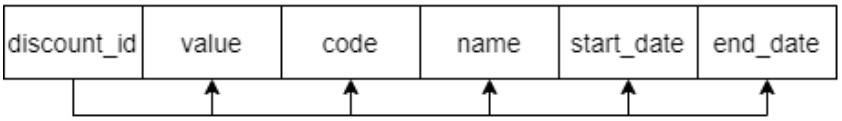
User



Order

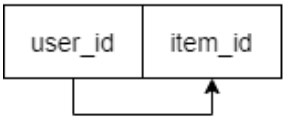


Discount

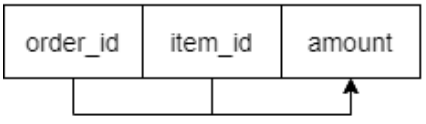


Item

Like



ItemList

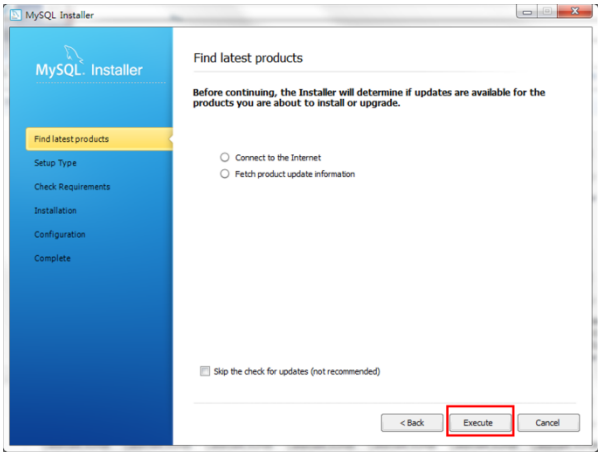


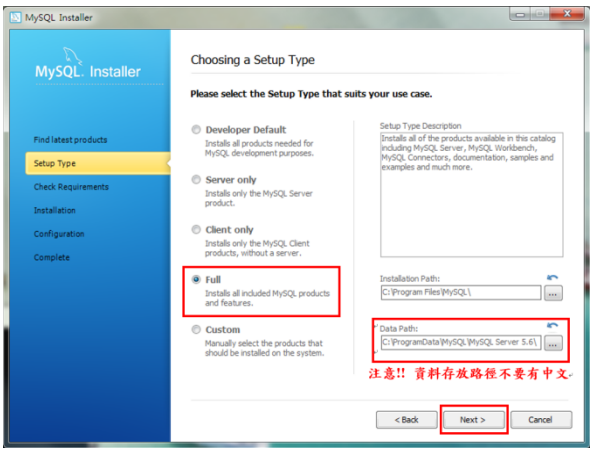
Section 6 The Use of the Database System

6.1 System Installation Description

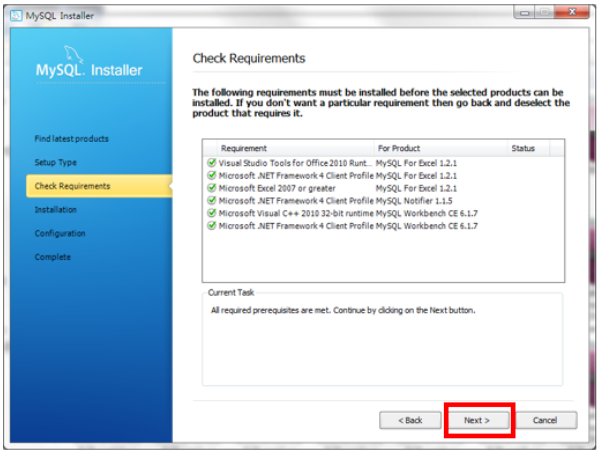
6.1.1 MySQL

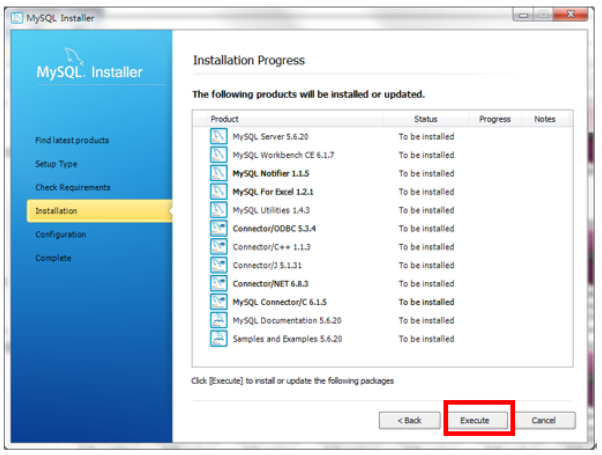
點選「Execute」檢查更新。選擇「Full」，可修改安裝路徑。接著點擊「Next」



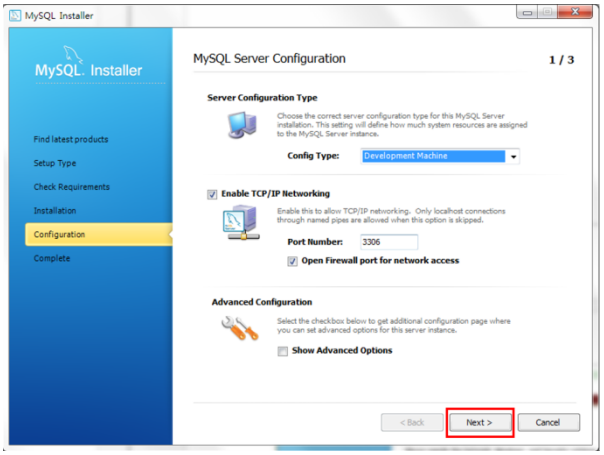


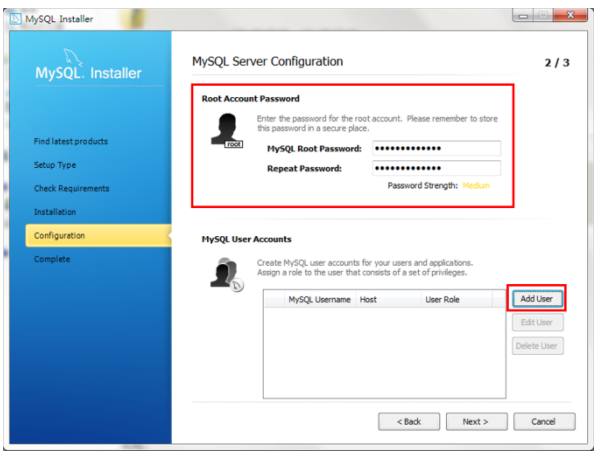
點擊「Next」安裝必要元件，點擊「Execute」開始安裝 MySQL





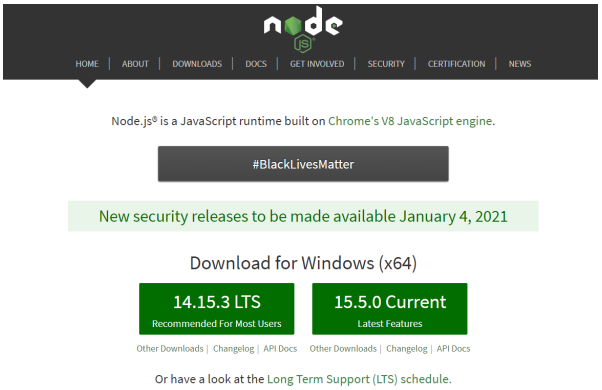
接下來要設定 Mysql Server，可以設定 Config type 和 Port number
設定 root 帳號以及密碼並新增一位使用者





6.1.2 npm以及react

先到Node.js官網下載安裝npm。裝好之後，重新打開 Terminal 輸入: npm- -v 確認版本



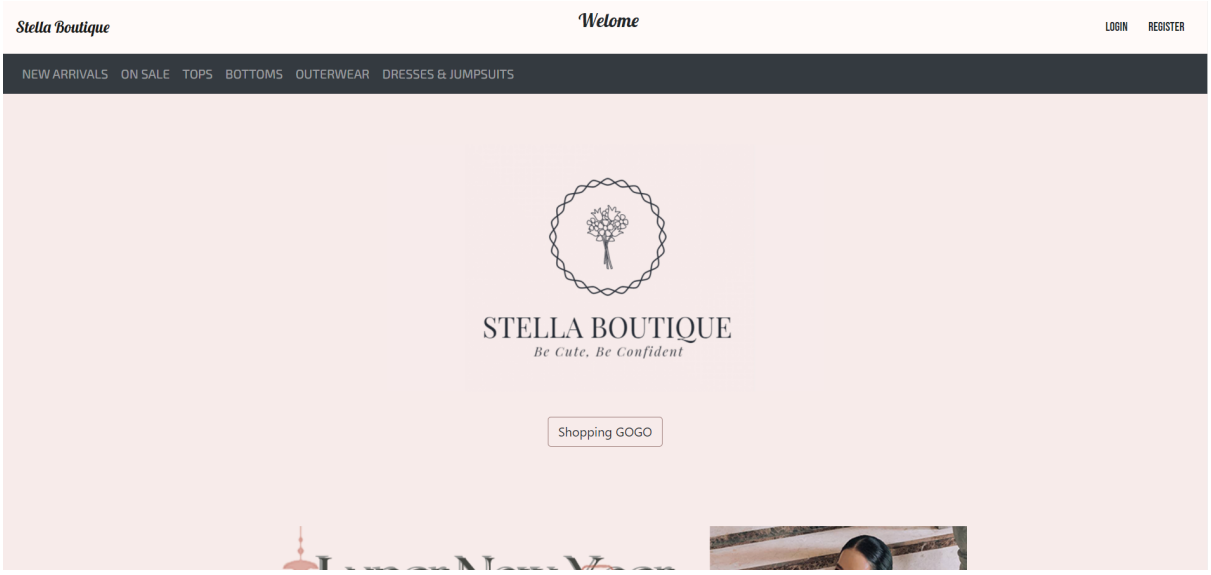
接下來透過terminal 輸入以下指令 安裝react並且建立新的專案

```
# 安裝 create-react-app
npm install -g create-react-app
```

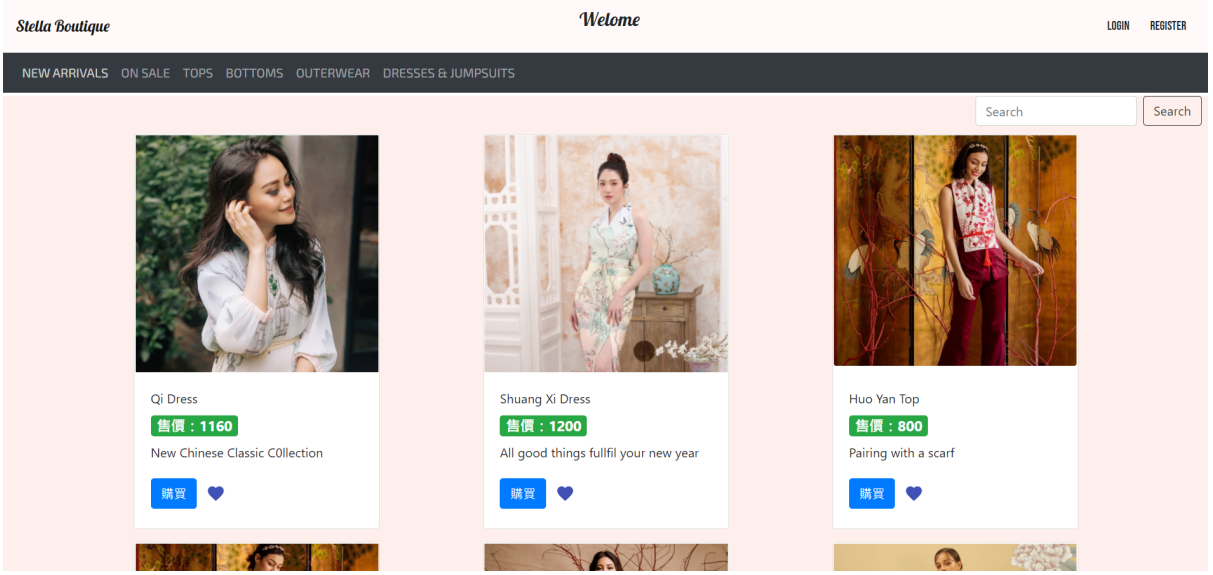
最後輸入 `npm run build` 可以在localhost:3000

6.2 The Use of the System

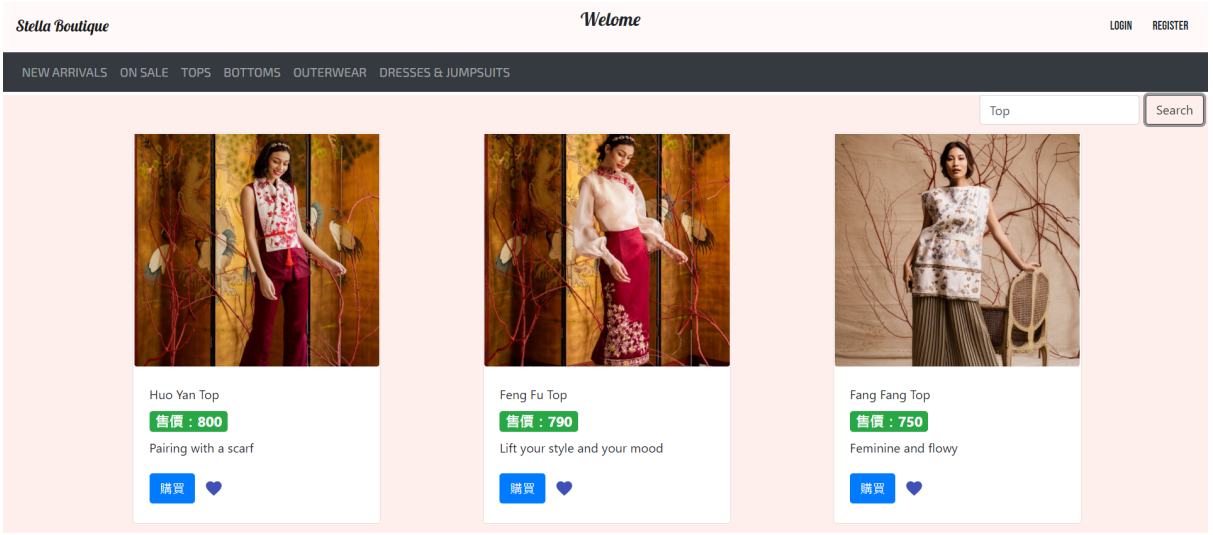
訪客首頁畫面



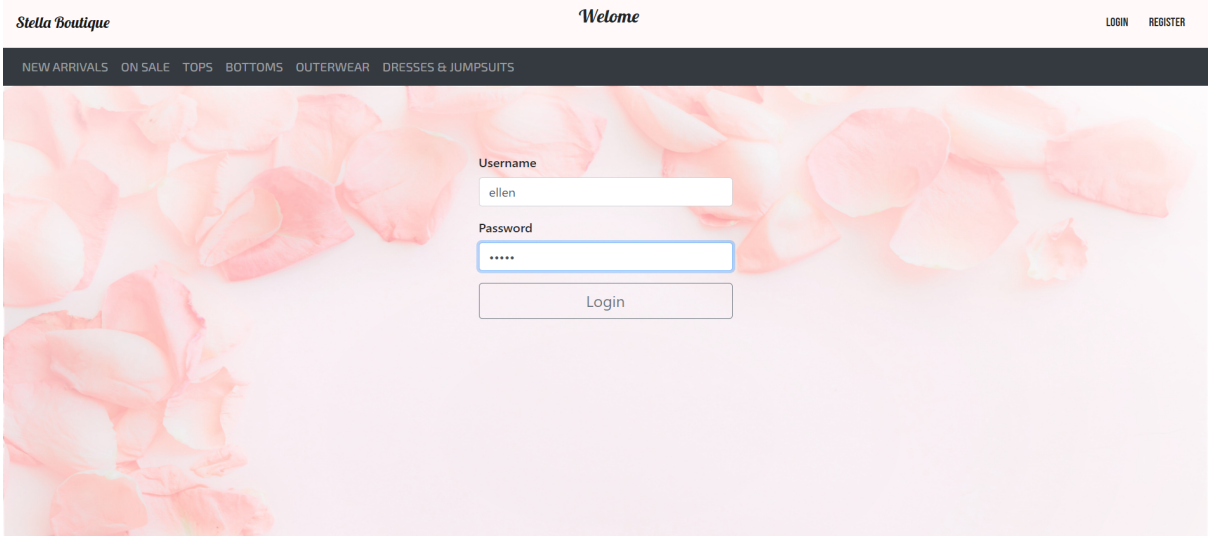
顧客瀏覽商品頁面



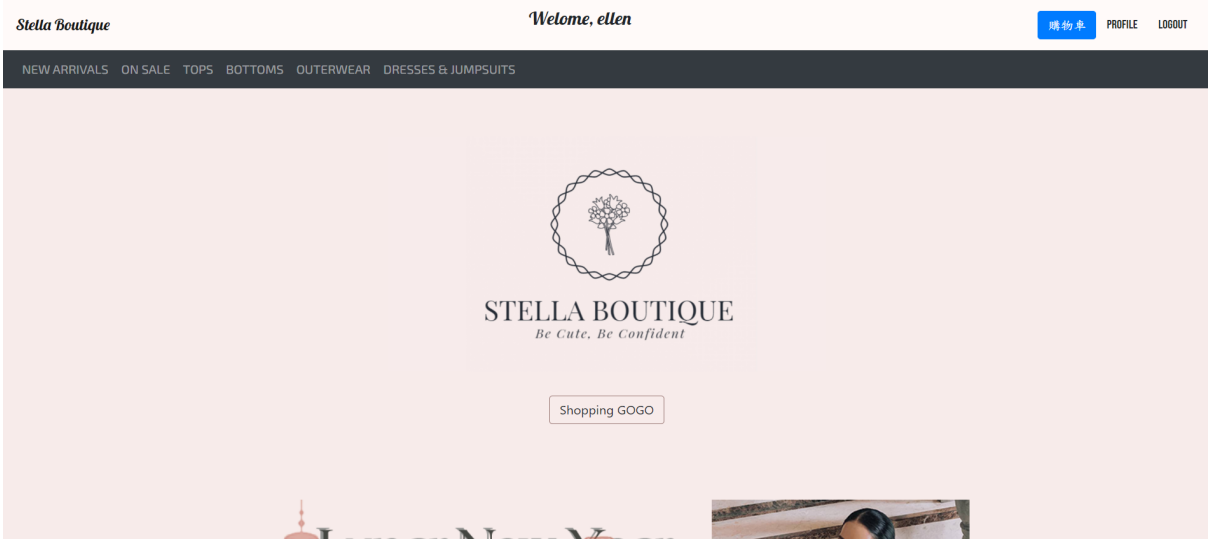
顧客搜尋商品



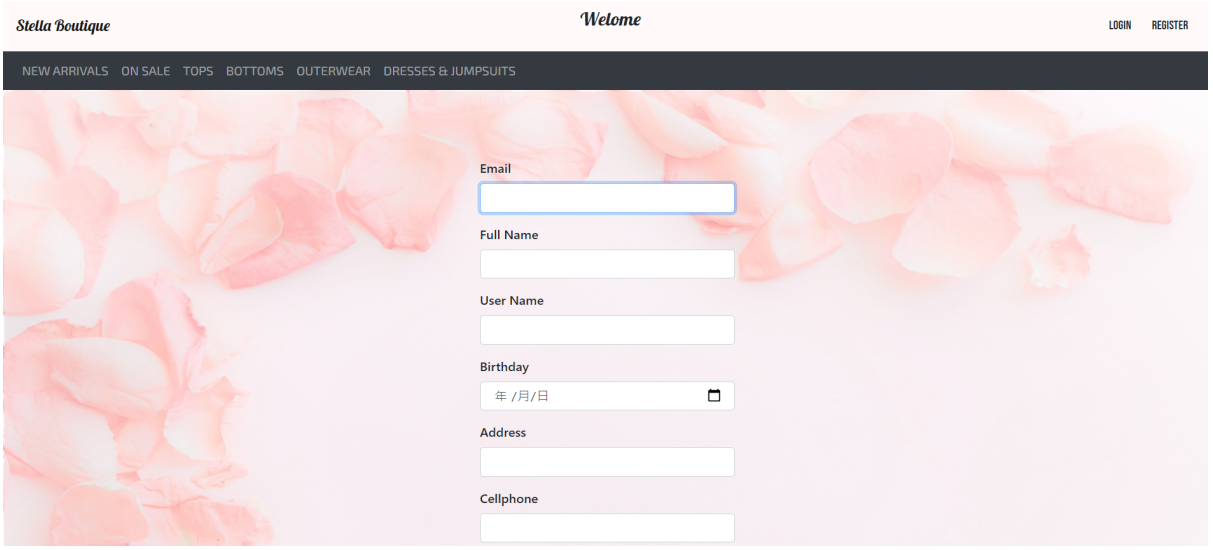
顧客登入畫面



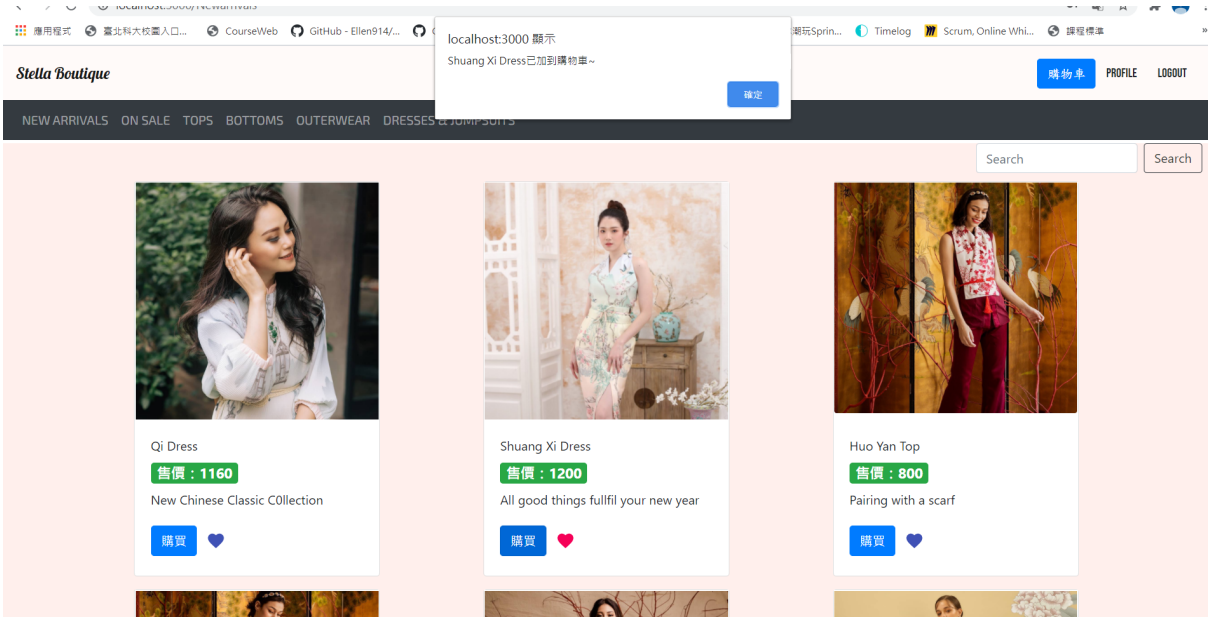
登入後跳轉首頁



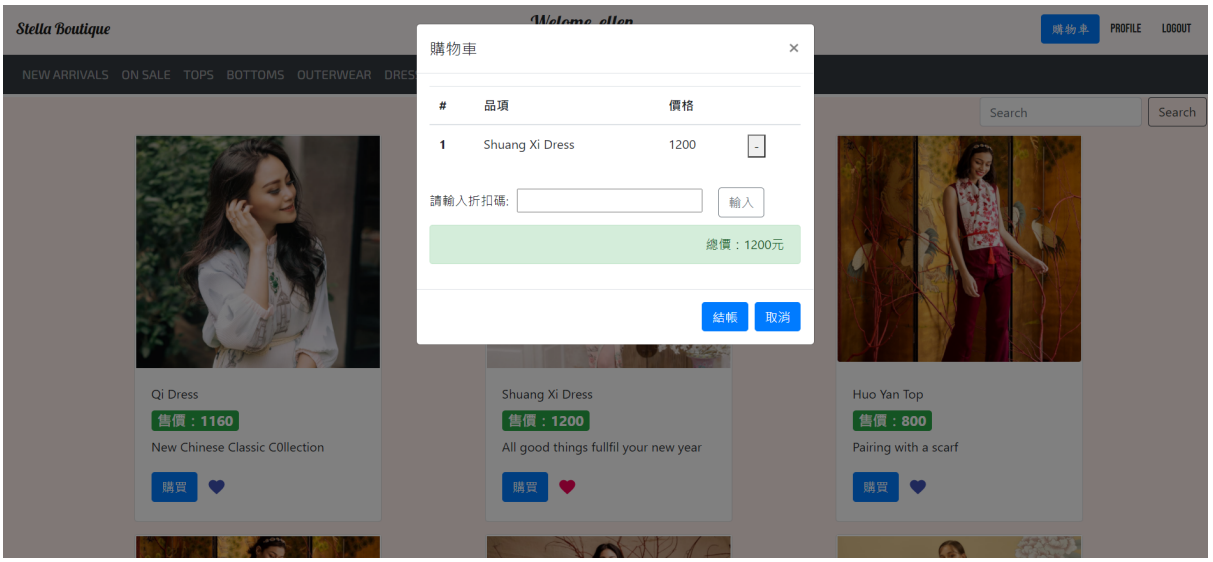
註冊會員畫面



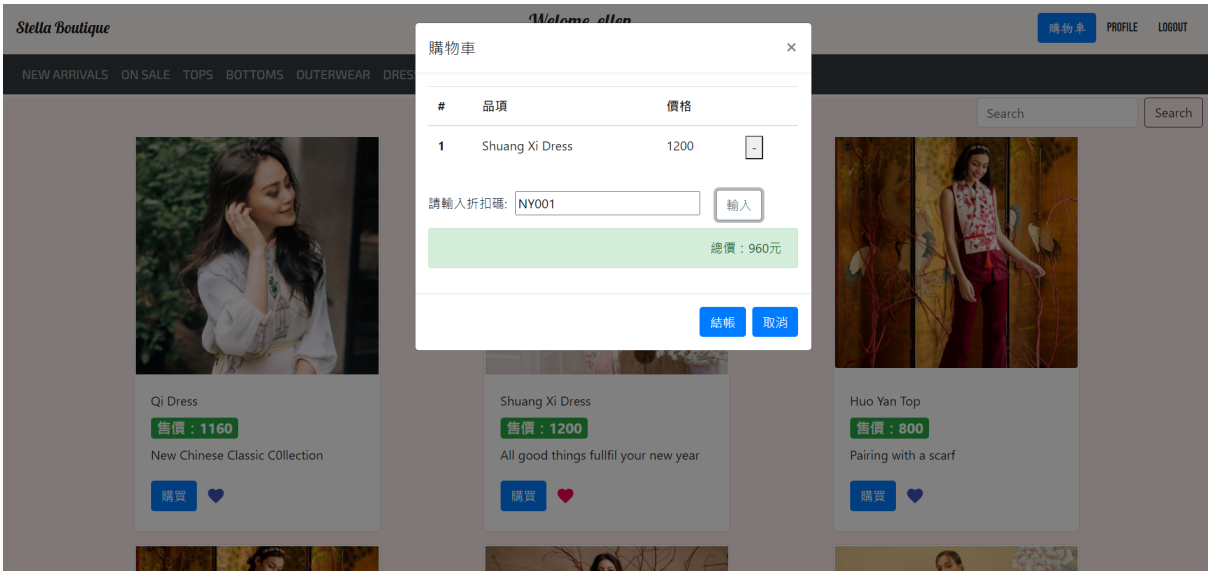
顧客加入商品到購物車



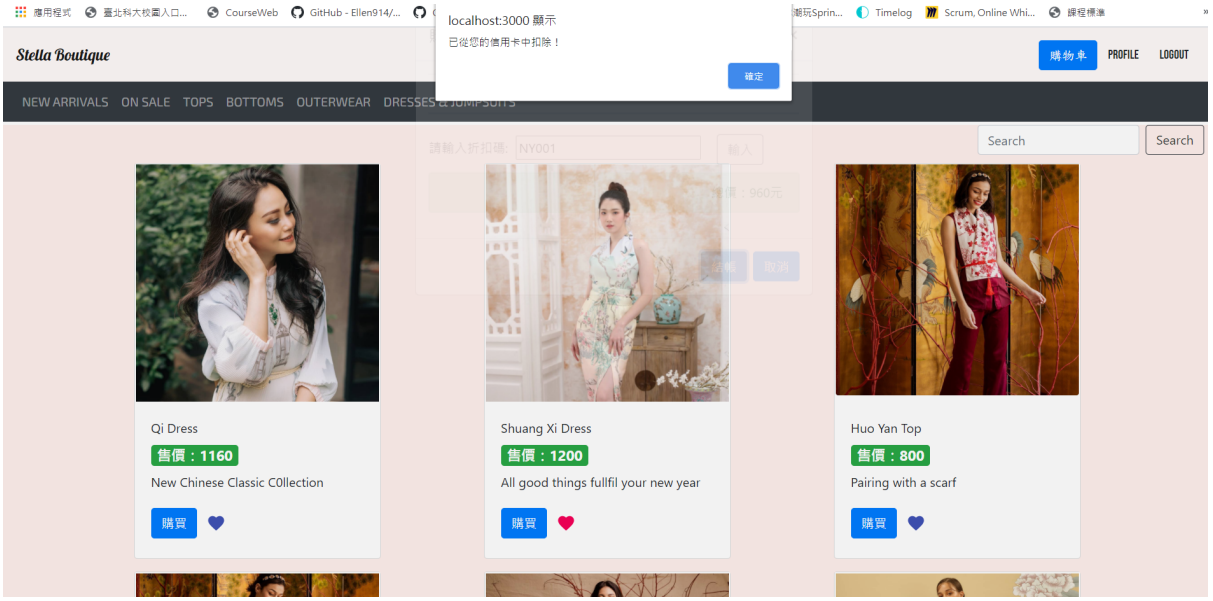
顧客打開購物車



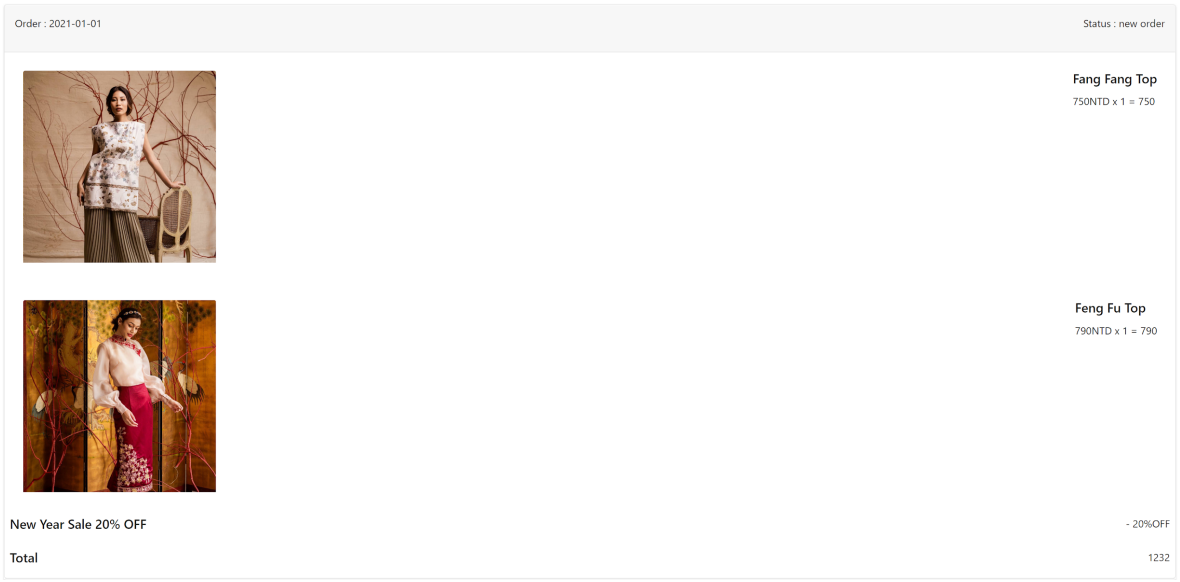
在購物車中使用折扣碼



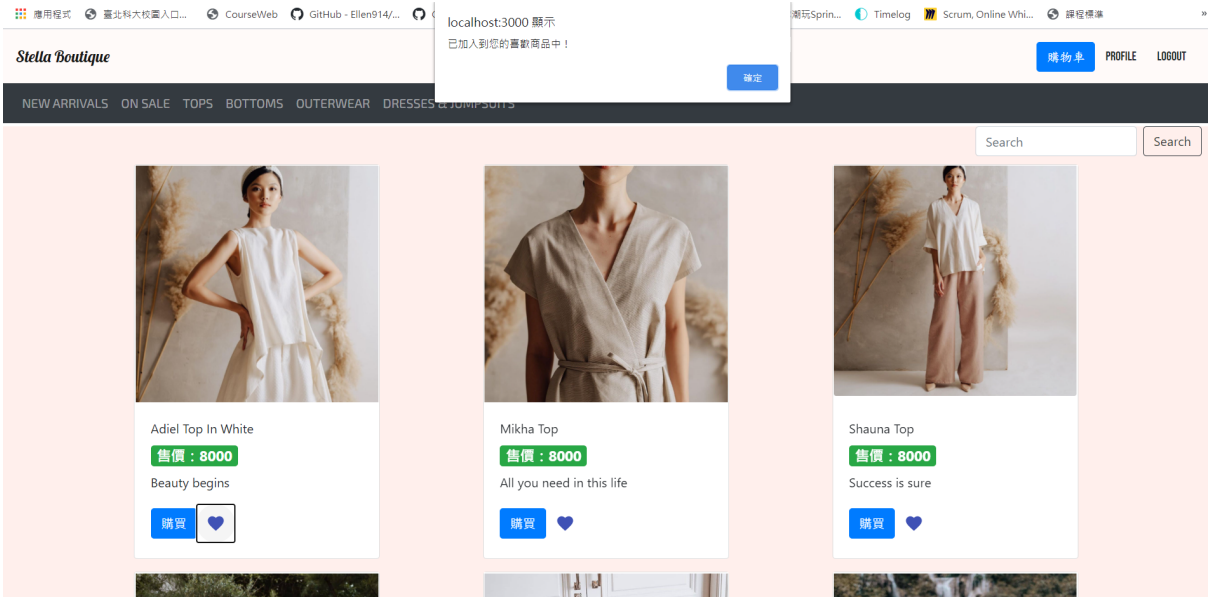
按下結帳即可完成購物，結帳完成畫面。若是按下取消則清空購物車。



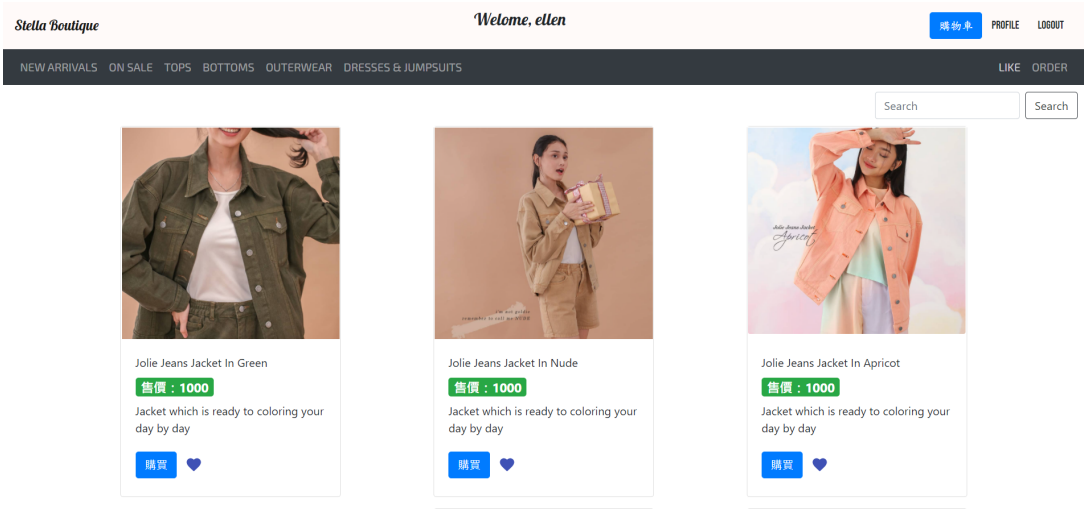
會員中心檢視顧客訂單清單，狀態轉為已寄送的話，買家可以確認取件



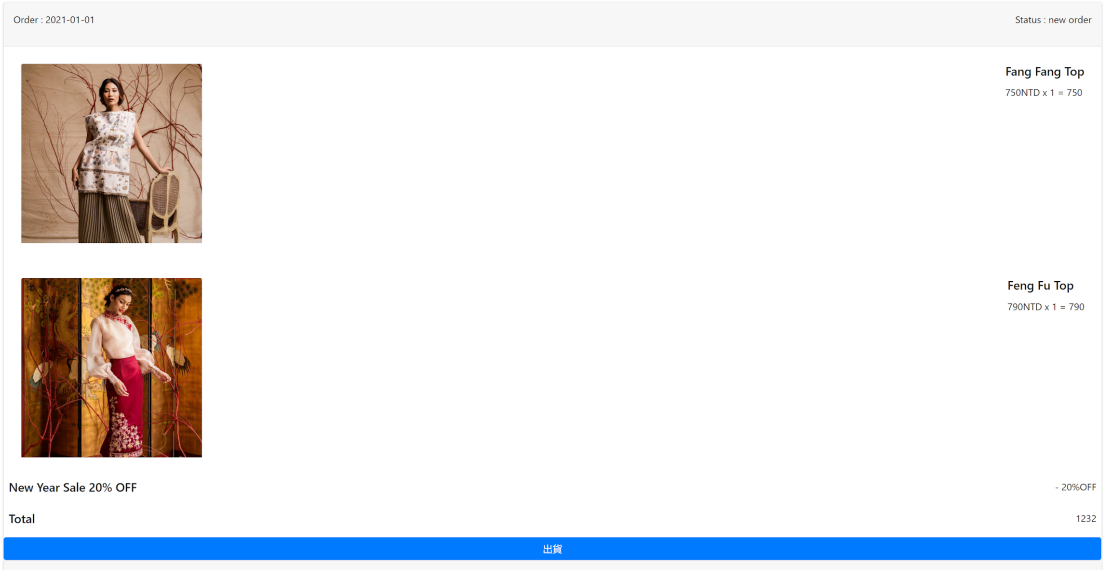
顧客加入商品到喜歡商品中



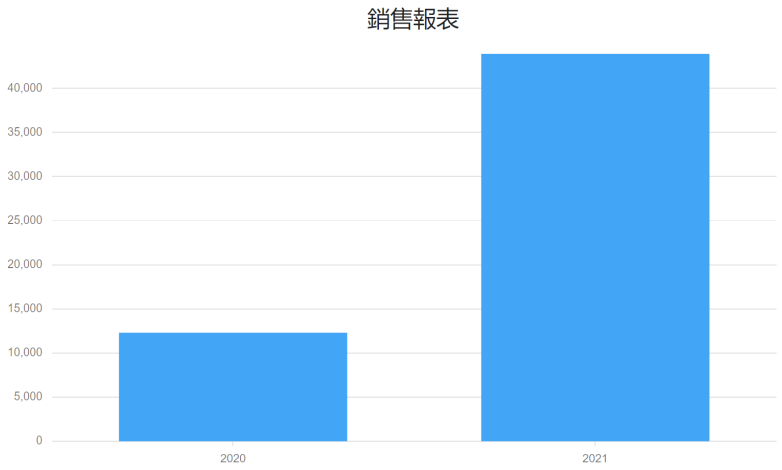
會員中心檢視顧客已喜歡商品，也可以在取消喜歡



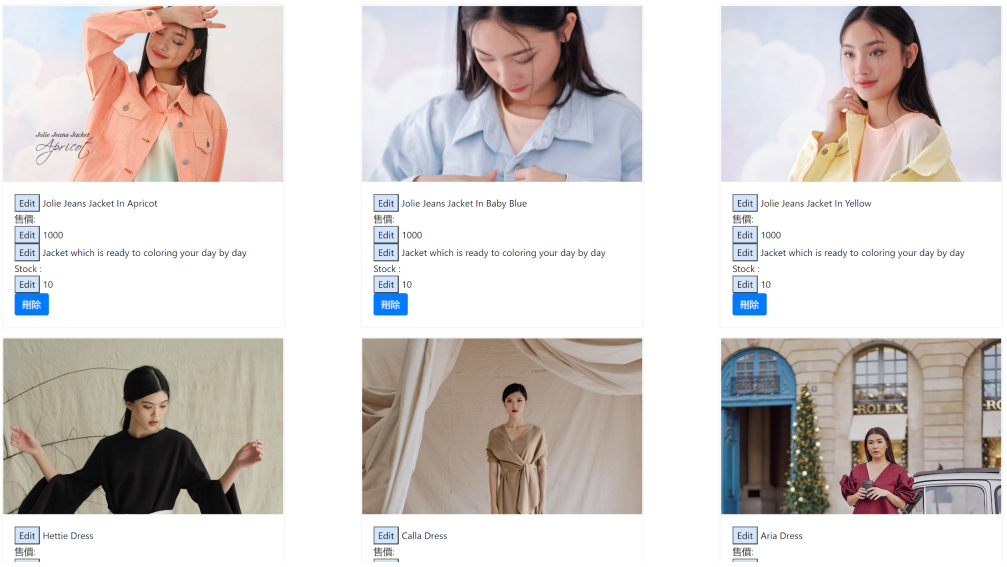
賣家管理訂單頁面，若顧客建立新建單，便可以更新狀態為出貨



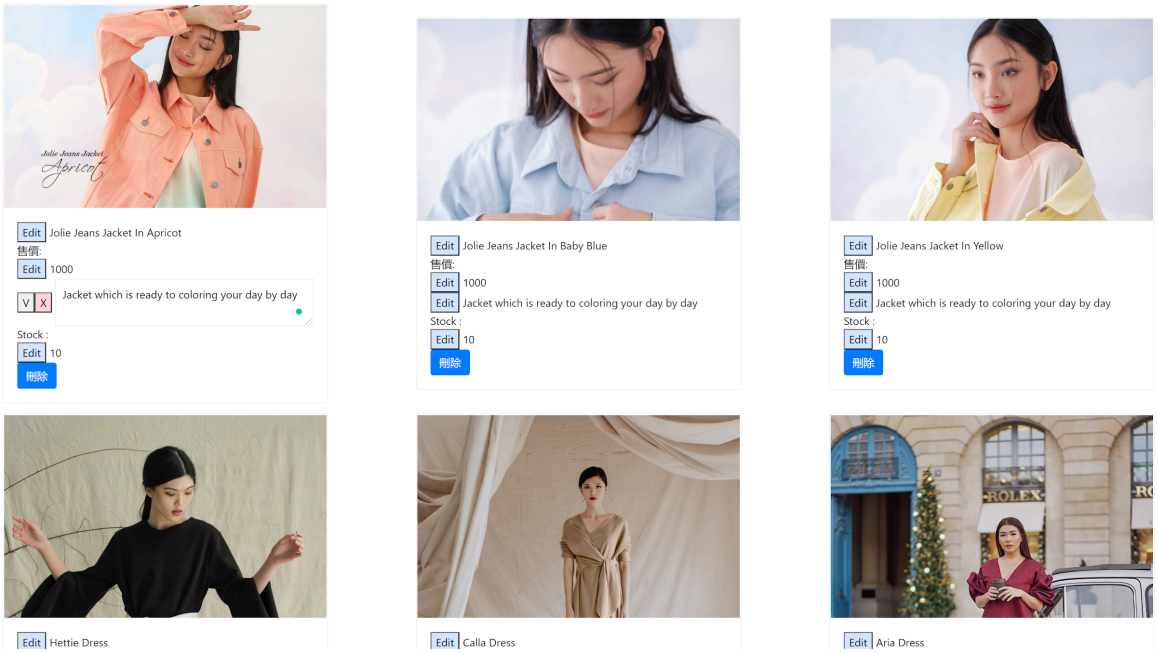
賣家銷售報表



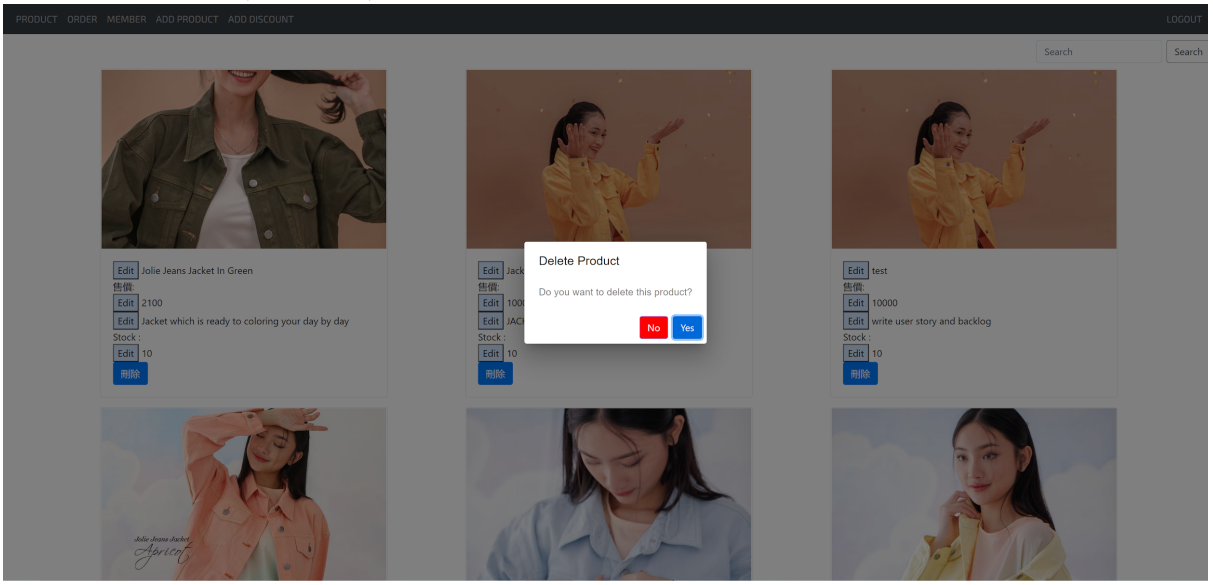
賣家管理商品頁面



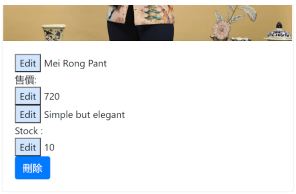
賣家可以編輯商品資訊

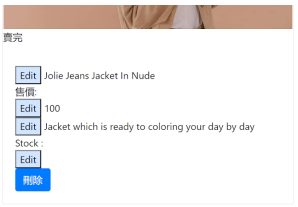


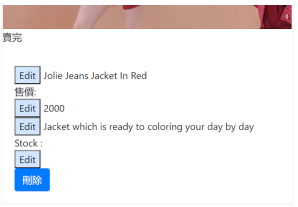
賣家選擇刪除商品(防呆畫面)

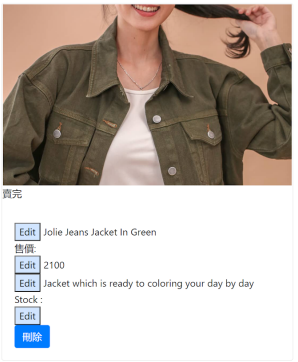


賣家刪除後的商品顯示狀態為賣完

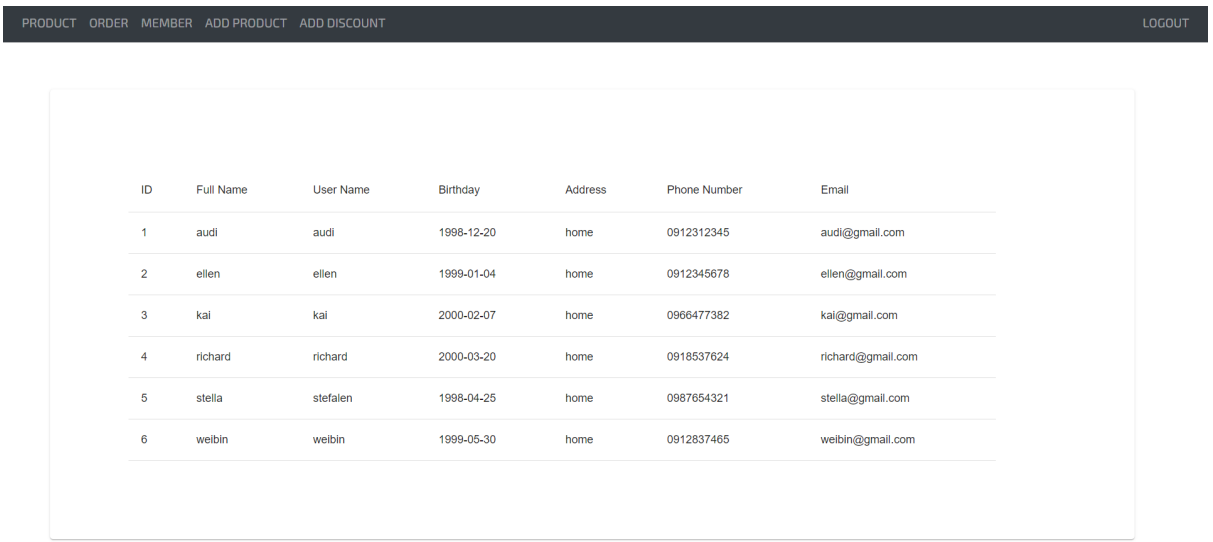




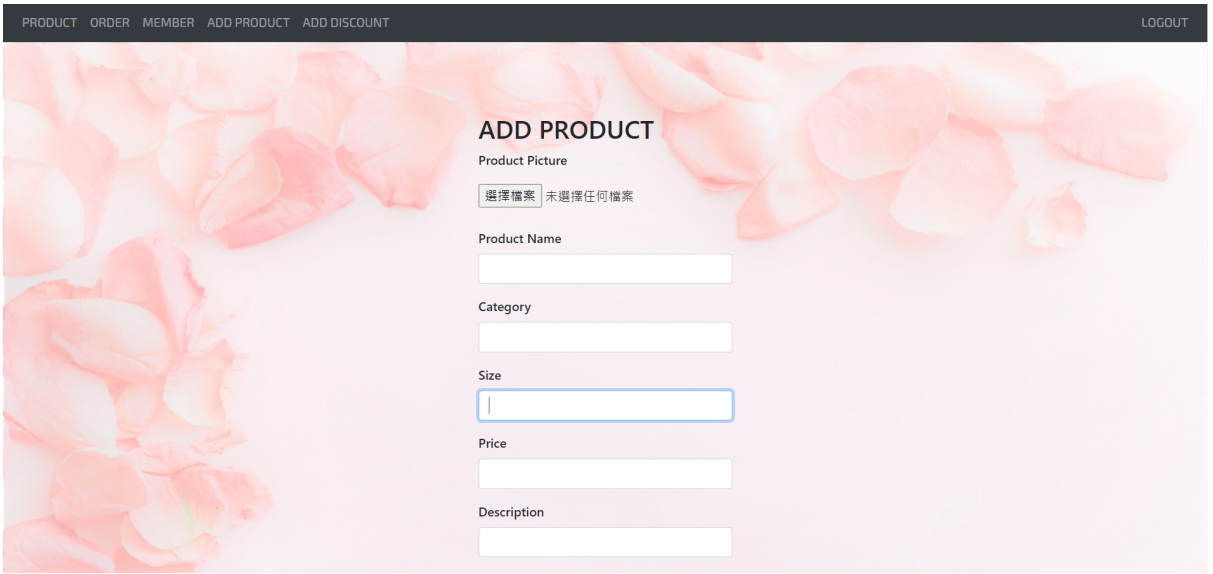




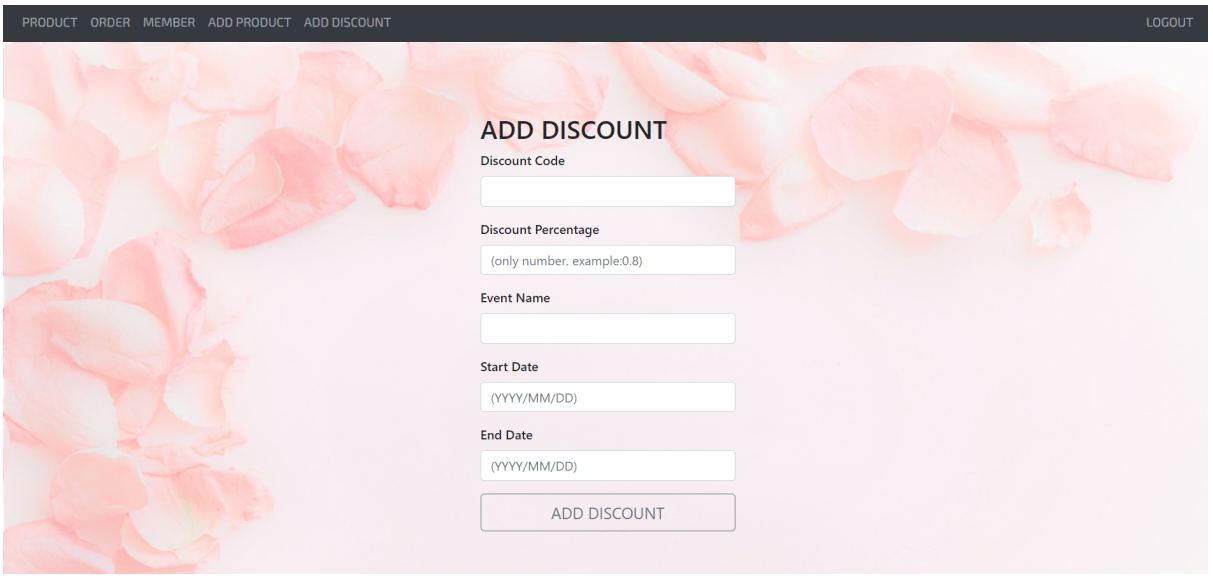
賣家會員檢視頁面



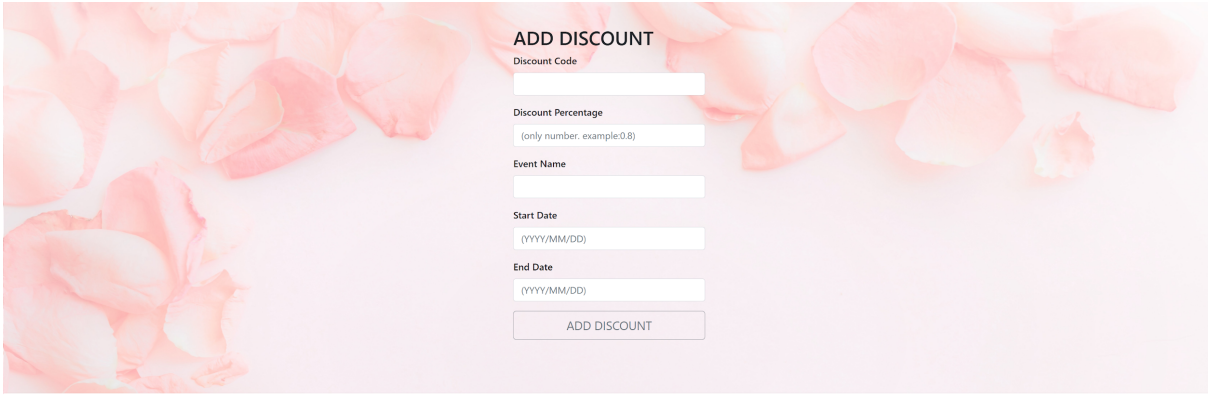
賣家新增商品頁面



賣家新增折扣頁面



賣家可以看到目前有的折扣清單



Discount Code	Discount Percentage	Event Name	Start Date	End Date
NY001	0.8	New Year Sale 20% OFF	2021/01/01	2021/01/30
CT001	0.5	Christmast Gift 50% OFF	2020/12/15	2020/12/30

Section 7 Suggestions on Database Turning

在搜索的部分資料量一旦變得龐大, 效率上就會有明顯的下降, 而且很可能出現許多類似或者名字相同的資料。針對這點我們可以通過建立index的方式來改善效能。在加入新的商品的時候, 可以自由加上數量不限的標簽例如:風格、顏色、類別、材料、舒適度、季節等等, 建立secondary index再去做query的動作, 這樣不僅可以提高系統的運作效能, 讓使用者有更多方式去篩選商品, 還能讓機器學習模型去做分群參考。

Section 8 Additional Queries and Views

8.1 Additional Queries

Guest :

Get Discount - >
SELECT * FROM `discount`

Get Product - >
SELECT * FROM `item` WHERE `quantity` != 0

Management :

Add New User - >
INSERT `user` (`password`, `fullname`, `username`, `birthday`, `address`, `phoneNumber`, `email`)
VALUES ({password}, {fullname}, {username}, {birthday}, {address}, {phoneNumber}, {email})

Login - >
SELECT * FROM `user`

Seller :

Add into Discount - >
INSERT INTO `discount` (`value`, `code`, `name`, `startDate`, `endDate`) VALUES({value}, {code}, {name}, {startDate}, {endDate})

Get Discount - >
SELECT * FROM `discount`

Get Order List - >
SELECT * FROM `order`

Get Item for Each Order - >

SELECT * FROM i`itemlist` il JOIN `item` i WHERE il.orderItemId = i.id AND il.orderID = {ordeID}

Update Order Status - >
UPDATE `order` SET `status`={newStatus} WHERE id={orderID}

Add into Item - >
INSERT `tem` (`name`, `quantity`, `category`, `size`, `price`, `description`, `pictureURL`) VALUES ({name}, {quantity}, {category}, {size}, {price}, {description}, {pictureURL})

Get Product - >
SELECT * FROM `item`

Update Item - >
UPDATE `Item` SET `name`={name}, `quantity`={quantity}, `price`={price}, `description`={description} WHERE `id`={itemID}

User :

Add into Like - >
INSERT `like` VALUES ({userID}, {itemID})

Get Like Item - >
SELECT * FROM i`like` JOIN `item` WHERE like.itemID = item.id AND like.userID = {userID}

Remove From Like - >
DELETE FROM `like` WHERE userID = {userID} AND itemID = {itemID}

Add into Order - >
INSERT `order`(`orderDate`, `orderUserID`) VALUES ({todayDate}, {userID})

Update Order Status to Cancel - >
UPDATE `order` SET `status`={newStatus} WHERE id={orderID}

Get Order List - >
SELECT * FROM `order` WHERE orderUserID = {userID}

Get Item for Each Order - >
SELECT * FROM i`itemlist` il JOIN `item` i WHERE il.orderItemId = i.id AND il.orderID = {ordeID}

Add item Into ItemList - >
INSERT `itemlist` VALUES({itemID}, {orderID}, {amount})

Remove Item From List - >
DELETE FROM `itemlist` WHERE orderID = {orderID} AND itemOrderID = {itemID}

Update Amount - >
UPDATE `listitem` SET `amount`={amount} WHERE orderID = {orderID} AND itemOrderID = {itemID}

Section 9 Conclusions and Future Work

9.1 Conclusion

相信在這次資料庫專題的設計和實作中，大家都學到了不少的東西。首先我們是第一次通過6人協作的方式來進行一項專題企劃，分工以及討論上的更需要花費心思與時間。打從設計系統的結構開始，資料庫的表格規劃以及子系統規劃就顯得非常複雜，很難想象我們有辦法從零開始實作出令人滿意且實用的系統。

這次的專案讓我了解到老師要表達的理念，資料庫設計的部分，先列出各項需求再到設計ER-diagram與Database Schema，最後才進行實作。實作的過程中遇到問題在重新回到起始點探討遇到的問題，才讓我們真正瞭解到課堂上提到的實際例子。

整體實作中，很慶幸的是我們剛好在開學前纍積了不錯的開發經驗，前後端分離的設計是我們的一大特色。成功必然的過程便是打好基礎，就好像蓋起大樓以前打的地基，雖然過程最爲艱辛，而且沒有人重視或注意這個過程，地基深埋地底也沒有人知道到底打了多深，但它卻是過程中最重要的部分，經過漫長的討論、設計等種種鋪成，我們順利且迅速地把前後端系統架設起來，并且與當初的計劃對比起來有非常高的完成度。

最後覺得比較可惜且不足的部分就是，資料庫作爲整個系統架構後端的核心，我們學習到有關於資料庫知識的時間點可能較爲不當，很難在一開始考慮得更加周全，必須在日後花費更多的時間去進行修改及調整。此外，在系統的效能以及實用性上缺乏實際的測試，系統在運行上可能存在很多bug，倘若沒有足夠的時間及設備進行測試並及時修復，未來可能因此需要花費很多時間進行排查。

9.2 Future Work

在這次的實作中，我們如期的完成當初所計劃的主要範圍，對於未來的展望，系統完整的測試是必要的，在功能方面，用戶管理子系統中的資料的收集還有可以完善的空間，同時加入機器學習的元素讓此平臺更加智能化。

前端的設計方面，動態的響應式網頁是目前主流的呈現方式，此系統還可以拓展到手機平臺，讓用戶在不同的作業系統上可以有更好的使用體驗。

References

Andy Tsai | React Maker | Medium
https://medium.com/@bbandydd
React 用來實作使用者界面的 JavaScript 函式庫
https://zh-hant.reactjs.org/
Material-UI:A popular React UI Framework
https://material-ui.com/
Tutorial | React.js and Spring Data REST
https://spring.io/guides/tutorials/react-and-spring-data-rest/
[Day14] – Spring Boot 與MySQL數據庫的應用教學 - iT 邦幫忙
https://ithelp.ithome.com.tw/articles/10217667
Getting Started | Accessing data with MySQL - Spring
https://spring.io/guides/gs/accessing-data-mysql/

Appendix

Appendix A

Guest :

Get Discount
SELECT * FROM `discount`
Get Product
SELECT * FROM `item` WHERE `quantity` != 0

Management :

Add New User
INSERT `user` (`password`, `fullname`, `username`, `birthday`, `address`, `phoneNumber`, `email`) VALUES ({password}, {fullname}, {username}, {birthday}, {address}, {phoneNumber}, {email})
Login
SELECT * FROM `user`

Seller :

Add into Discount
INSERT INTO `discount` (`value`, `code`, `name`, `startDate`, `endDate`) VALUES({value}, {code}, {name}, {startDate}, {endDate})
Get Discount
SELECT * FROM `discount`
Get Order List
SELECT * FROM `order`
Get Item for Each Order
SELECT * FROM i`itemlist` il JOIN `item` i WHERE il.orderItemId = i.id AND il.orderID = {ordeID}
Update Order Status
UPDATE `order` SET `status`={newStatus} WHERE id={orderID}
Add into Item
INSERT `tem` (`name`, `quantity`, `category`, `size`, `price`, `description`, `pictureURL`) VALUES ({name}, {quantity}, {category}, {size}, {price}, {description}, {pictureURL})
Get Product
SELECT * FROM `item`
Update Item
UPDATE `Item` SET `name`={name}, `quantity`={quantity}, `category`={category}, `size`={size}, `price`={price}, `description`={description}, `pictureURL`={pictureURL} WHERE `id`={itemID}

User :

Add into Like
INSERT `like` VALUES ({userID}, {itemID})
Get Like Item
SELECT * FROM i`like` JOIN `item` WHERE like.itemID = item.id AND like.userID = {userID}
Remove From Like
DELETE FROM `like` WHERE userID = {userID} AND itemID = {itemID}
Add into Order
INSERT `order`(`orderDate`, `orderUserID`) VALUES ({todayDate}, {userID})
Update Order Status to Cancel
UPDATE `order` SET `status`={newStatus} WHERE id={orderID}
Get Order List
SELECT * FROM `order` WHERE orderUserID = {userID}
Get Item for Each Order

SELECT * FROM i`itemlist` il JOIN `item` i WHERE il.orderItemId = i.id AND il.orderID = {ordeID}

Add item Into ItemList

INSERT `itemlist` VALUES({itemID}, {orderID}, {amount})

Remove Item From List

DELETE FROM `itemlist` WHERE orderID = {orderID} AND itemOrderID = {itemID}

Update Amount

UPDATE `listitem` SET `amount`={amount} WHERE orderID = {orderID} AND itemOrderID = {itemID}