Polaritique 試穿模擬購物平台

Group B

B11705017 陳芃宇

B11705024 謝友毅

B11705027 陳承妤

B11705034 蔡逸芃

B10705054 劉知微

B11801004 卓庭榛

GitHub連結:
https://github.com/BillyHsieh123a/113_1_Web_App_Final_Project

一、摘要

現今已有許多購物平台提供消費者能不必到場即可買到他們想要的東西，但若是衣物類的部分，則常常會遇到買來的尺寸與實際想像不合，而本平台則專注於提升使用者的購物體驗，透過串接現有api提供使用者虛擬試穿技術，讓顧客在購買服飾、配件等商品時，可以在虛擬環境中預覽穿戴效果。顧客可以利用自身的照片，模擬穿戴商品的效果，並根據身形與喜好，挑選最適合的衣著。整體來說，該平台旨在解決線上購物中無法試穿的問題，並增強顧客與品牌之間的互動。

二、前言

本篇報告中，我們分為五個段落，分別為摘要、前言、開發架構、問題、結果、討論，於後續的段落中，將討論小組成員於開發系統的過程中遇到的問題、挑戰，於開發方法說明專案之系統架構與流程，並於結果敘述我們所開發的主要功能。

三、開發架構

- **4.1** 系統架構

  前端：使用HTML、CSS 來製作前端頁面及美觀，並由 JavaScript 搭配 Flask處理串接。

  後端：使用 Flask 處理 API 請求，管理及處理數據庫。

  資料庫：採用 PostgreSQL 儲存商品資訊、用戶資料。

  AI 模型：基於 Hugging Face API，實現服裝試穿效果。

- **4.2** 系統流程圖

  [all]點擊 LOGO → 回到首頁(會回到原先的類別標籤) ([signin] 、[login] 會回到 [login])

  [all]點擊右上角人像→ 進入 [user_account_base]

  [all]點擊右上角愛心→ 進入 [favorite] 顯示喜愛清單

  [all]點擊右上角購物袋→ 進入 [bag] 顯示購物袋內容

  [favorite]Add to Bag→ 進入 [item] 顯示商品詳細資訊

  [favorite]各個物品右上角的垃圾桶標誌→ 刪除喜愛清單的該物品

  [bag]點擊物品右上角 X → 刪除購物袋內的該物品

  [bag]點擊 Check out → 進入 [checkout] 顯示購物袋大概內容並顯示購買資訊供填寫

  [checkout]填寫完資料點擊 Buy Now → 購買成功進入 [ordered] 顯示訂單資訊

  [checkout]點擊 Change → 修改訂單地址、姓名、電話、電子郵件等等資訊 → 點擊 save 儲存修改內容

  [category]點擊 M/F → 顯示過濾結果

  [category]查找商品 → 顯示該商品搜尋結果

  [category (點擊 M/F 之後)]點擊商品 → 進入[items]顯示商品詳細資訊

  [category (點擊 M/F 之後)]點擊類別 → 顯示過濾結果

  [item]點擊加入購物袋 → 將商品加入購物袋

  [item]點擊愛心 → 將商品加入喜愛清單

  [item]點擊返回 → 返回到前一頁

[item]點擊試穿 → 進入[try-on]

[try-on]上傳照片 → 呼叫 API → 顯示試穿結果

[try-on]點擊返回 → 返回到前一頁

[login]輸入帳戶資訊 → 登入成功跳回首頁／登入失敗回傳錯誤提示字

[signin]輸入帳戶資訊 → 註冊成功跳回首頁／註冊失敗回傳錯誤提示字

[user account base] 點擊 User Details → 跳到 [User Details] 頁面

[user account base] 點擊 My Orders → 跳到 [My Orders] 頁面

[user account base] 點擊 Sign Out → 跳回登入頁面

[user details] 顯示個人資訊 → 修改後點擊 save changes → 傳給後端修改資訊

[user orders] 有 In Progress／Received／Deleted／Returned 四項可選 → 點擊後回傳符合條件之資料

四、問題

- **3.1** 前端美觀與介面設計

  為了美觀和界面設計，這使得 CSS 調整和前端框架的設計變得更為繁瑣。具體來說，如何將功能合理地包裝到各個框架中是我們面對的一大挑戰。以及哪些元件是在每一個頁面都需要重複利用的都需要考慮。

- **3.2** 頁面跳轉

  當使用者點擊某個項目後，應該跳轉到相應的分類頁面以顯示搜尋結果。我們花了很長時間才意識到需要將搜尋項目嵌入到網址中（embed），以便實現正確的頁面跳轉和數據顯示。

- **3.3** 時間問題

  原先預期會開發後台管理模組，包括商品上架管理、訂單處理和用戶分析等功能。然而，經過評估後發現，時間上可能不夠，因此我們決定先跳過這一部分。我們認為，網站應該以使用者體驗為主，以節省開發時間。由於刪減了後台管理的前端頁面，我們得以將心思更放在優化前端頁面和確保系統穩定性，確保使用者端的功能完整且易於操作。

- **3.4** 試穿功能表現

由於是試衣網站，開發後端時自然專注於試衣模型的選擇上。發現不同的試衣模型普遍會有以下幾個問題：

Ø 幻覺（**Hallucination**）。會自行生成試衣者提供的照片與衣服以外不相關的物品。如：試穿褲子腳下之色塊、奇怪的手臂等。

Ø 無法辨別並試穿褲子。以下圖為例：誤將褲子穿在上衣上。

- **3.5 並行問題**

我們發現在使用者幾乎同時下單時，衣物的庫存量有可能變為負值。其他功能諸如加入最愛清單、加入購物袋等等皆有可能因為同一位使用者在不同地方登入而產生加入同一商品兩次的並行問題。

五、結果

- **5.1 功能介紹**

使用者：可進行登入、註冊、會員資料管理。

1. **登入：使用者需於登入後才可進行其他購物等功能，若使用者輸入錯誤或是有未填欄位則會跳出相關提示語提醒使用者。**

2. **註冊：若使用者尚未有帳號，則可以於與登入同界面處進行註冊功能。**

3. **會員資料管理：當登入後，於左上角處有人像可點擊，點擊後可選擇登出、查看自己的訂單、編輯個人資料。**

服裝商品模組：商品展示、AI 試穿模擬、購物車功能。

1. **商品展示：點擊商標至大廳首頁後，可於導航欄處選擇性別，並會出現與該性別相關之各商品分類，除了選擇性別查找商品外，亦可選擇直接查詢想買之商品。**

2. **AI試穿模擬功能：選定指定商品後，介面提供試穿功能，此使用者需提供自己的相片，便可透過AI API將所選商品套至圖像。**

3. **購物車功能：使用者可一次選擇多種喜愛的商品加入購物車，並可一鍵選擇結帳。**

六、討論

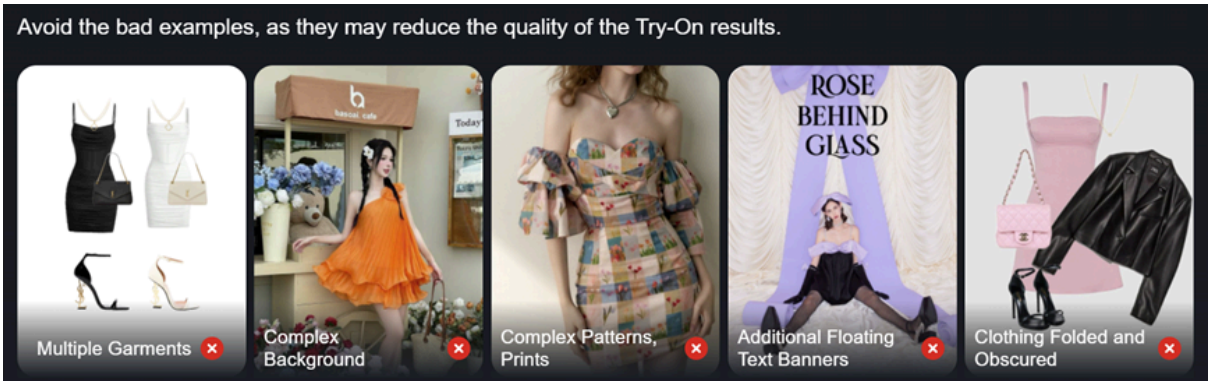
- **6.1 試穿功能：照片與模型選擇**

以照片的角度來看，試衣 API 的提供者提供了以下的建議：

Ø 背景不能複雜。

Ø 衣物不能有複雜的圖案與花紋、不能被遮擋、不能被褶起來、不能有多件套。

Ø 照片不能有印在衣物以外的文字。



對於衣物，我們可以選擇符合以上要求的照片，並要求使用者盡量於乾淨的背景拍攝個人照，然而，商業實務上是否總是有理想的衣物照片、是否會影響使用者的使用方便性與意願為一大疑慮。

因此，我們轉而專注於試衣模型的選擇。以下為我們找到的幾種試衣API之比較：

API	Kolors Virtual Try-On	IDM-VTON	Segmind Try-On Diffusion

上衣試穿	極佳	佳	普通
褲子試穿	佳, 自動判斷衣物部位	無法辨別	普通, 需要手動指定部位
收費	**HuggingFace Demo:** 免費	免費, 不過每小時只有試約兩到三次衣服的GPU額度	有嘗試約50次衣服之1美元額度
	API Calls: 新台幣136元 / 100次呼叫		
幻覺	偶有幻覺, 但不嚴重	偶有幻覺, 但不嚴重	會產生奇怪的手臂
產生速度	**HuggingFace Demo:** 極度不穩定, 時常因為使用者過多而失敗, 成功時30秒至90秒不等	30秒以內	30秒至1分鐘
	API Calls: 30秒至1分鐘		

權衡後, 我們認為 IDM-VTON 額度過低, 不利於測試與開發, 不能試穿褲子為一大痛點;Segmind Try-On Diffusion 的幻覺較嚴重且試穿品質不及 Kolors 與 IDM-VTON。因此, 我們在 **Kolors Virtual Try-On** 中直接使用 HuggingFace Demo 的免費版本與 API Calls 的付費版本中做選擇。

由於HuggingFace Demo的Gradio沒有開放API呼叫, 要使用就必須使用 selenium 套件自動化操作瀏覽器、上傳照片並抓取生成結果。實際實作後, 發現就算瀏覽器使用無頭模式, 載入 Demo 頁面仍需要一分鐘以上, 使用試穿服務的客戶可沒有這個耐心。因此, 還是決定以 136 元購入 100 次 API 呼叫的額度。

由於額度有限, 每次使用者試穿後的照片都會被暫存於 server 端, 使用者再次點入 try-on 頁面後就會直接顯示最近一次的試穿結果。在商業實務上也有助於業主節省 API 呼叫費用。

- **6.3 並行控制**

  使用者可能用多個裝置登入, 並執行相同操作, 這樣可能會造成資料不一致, 所以我們使用 PostgreSQL 的寫鎖達成並行控制。
  add to bag:

```
SELECT user_id
FROM BAG
WHERE user_id = %s AND clothes_id = %s AND color = %s AND size = %s
FOR UPDATE
```

add to favortie：

```
SELECT user_id
FROM FAVORITE
WHERE user_id = %s AND clothes_id = %s AND color = %s
FOR UPDATE
```

checkout_：

```
SELECT bag.clothes_id
FROM bag
WHERE bag.user_id = %s FOR UPDATE
```

- **6.4** 照片存儲方式

  照片存儲方式有二：使用 MongoDB 等 NoSQL 資料庫存除非結構化資料，或使用 PostgreSQL 等關聯式資料庫存放圖片的存儲相對位址。由於組員對 NoSQL 資料庫較不熟悉，且下單等操作需要遵守 ACID 原則，我們選擇以關聯式資料庫配合相對位址管理圖片。這衍生了一些問題——照片被刪除時，資料庫的相對位址沒有更新，會造成資料不一致（data inconsistency）。日後進行照片相關的系統實作時，應考慮使用 NoSQL 資料庫，將照片與日常交易的資料庫分開。