系統優化

- Author: B07902108 翁祖毅

安全性和錯誤處理

- 首先，任何資料庫都不應該將密碼存成明文。我的理想是在前端就使用雜湊演算法將密碼先行處理，因此我的後端沒有特別做這件事。這樣在 HTTP 傳輸時也更有保障，不會被直接看到內容。
- 然而，為了謹慎起見，我可以在後端再做一次 hash。我也看過某公司會用自行生成的 RSA public/private key 來對密碼輸入進行保護，這是很不錯的密碼學應用，也是可以優化的方向。
- 錯誤處理的部分，由於我自己本次寫作業的時間緊迫，沒有考慮例外的狀況，如輸入檢查等。錯誤的輸入會導致後端程式碼出現 bug，回傳 500 Error。實際上應該要針對輸入做檢查，出現錯誤時也需要妥善的處理 & 回傳適當的錯誤資訊。
- 即便如此，我該做 JWT token 驗證的地方還是有做，某些操作如果沒有帶 token 伺服器也會回傳錯誤。因此我認為至少在安全性上是有保障的。比如，只有賣家能夠 create product，也只有買家能夠下 order。
- 然而，我有點不確定 get order 的 JWT token 需要買家還是賣家的身分，因此這部分就沒有特別檢查。
- 總而言之，錯誤處理的部分，我認為仍可以再改進。

資料庫優化

- 資料庫的部分，我有進行第三正規化，因此 products 有使用 user_id 做為 foreign key，而 orders 也有使用 user_id 和 product_id 作為 foreign key。
- 此外，也有把 order 拆開，也就是說，同一個 order 的不同的 product 都是一個 row。但是這樣可能有一些缺點：
  - 比較難以生成遞增的 order id，因為資料庫的欄位只能採用或不採用嚴格遞增，很難創造一個 “group of rows” 有共同的 ID，然後下一個 group 又會遞增的欄位。我所採用的解決方法是 time 精細到小數點第六位，但這可能不是最好的，仍有極低的機率發生碰撞。
  - 查找的速率很慢，這會依據商品種類的數量而定。當訂單很多，而且每個訂單又有好幾種商品，就會降低查詢的速度。
- 一個可行的解法是不要求完全第三正規化，想辦法把 product 和 amount 集合在同一個 row。這樣或許就可以解決上述兩個問題。
- 另一個可行的解法是改用 NoSQL，以指標的想法來說，在 Get ALL 或 Get specific order 就會很快，因為龐大的訂單結構被物件包起來了，外面就只是一個指標的 list，查找應該蠻快的。