資料庫管理
期末專案書面報告
第 35 組

B09303131 馬松鐸、B10303091 洪榮盛、B11705039 李盈盈

1 系統分析

1.1 系統介紹

你的學生證又、又、又不見了嗎？你又、又、又在校園中撿到別人的學生證了嗎？快來「NTULOST」協尋和發布吧！

在台大學生的日常生活中，遺失物品（如學生證、鑰匙、書籍等）或撿到他人遺失的物品是一個常見的問題。目前大多數協尋需求都是透過臉書的「NTU 台大學生交流版」發布，但交流版本該是提供學生交流意見的平台，近年來層出不窮的協尋文和拾獲章往往佔據了交流版大多數的版面，尤其大概每五篇文章就會有一篇文章是在協尋丟失在校園某處的「學生證」。

然而，在交流版發佈的文章容易被淹沒、且管理效率不佳。因此，我們規劃一個專門的平台，提供台大生活圈（學生、教師、行政人員、校外人士）的人們協助尋找遺失物和回報拾獲物。系統還將提供懸賞機制、用戶之間的互動（如在貼文底下留言、私訊）、用戶回饋等功能，以及管理者對用戶的通知及系統維護管理功能。

根據不同的功能及掌控權限，「NTULOST」系統的用戶可以分為兩種身分，分別是 User 及 Admin。若作為一般使用者，身分別定義為 User。註冊為會員後，可以依照自身需求，選擇發佈協尋文或尋獲文。如果是要協尋的話，可以提供可能遺失時間、遺失地點、物品類別、是否要懸賞及懸賞的內容、圖片等資訊。如果是尋獲的話，可以提供尋獲時間、尋獲地點、物品類別、目前存放的地點、圖片等資訊。並且可以透過用戶間的訊息、在貼文下留言、用戶回饋等功能交流資訊，若非會員，則可以匿名方式發布，但能使用的功能就會有所限制。若作為管理者，身分別定義為 Admin。可以做到針對非會員發布的文章進行審核，並針對管理所有文章及用戶的基本資訊與歷史活動記錄，以及獲取各項統計數據及接受用戶回饋。

1.2 系統功能

1.2.1 相關設定

在「NTULOST」的相關設定中，平台會根據用戶的身份進行功能分配。用戶分為會員和非會員兩種類別。會員需要註冊並登入平台，才能享有完整的功能，包括發布協尋和拾獲通報、設置懸賞、與其他用戶互動、接收通知、管理通報進度等。這些功能針對台大生活圈中的學生、教職員和校外人士，旨在促進協尋效率並確保資訊的時效性和精確性。

非會員則以訪客身份使用平台，適合那些不常使用平台或不願意註冊的校內人員以及校外人士。非會員可以提交拾獲物通報，由管理者審核後代為發布，但無法使用留言、私訊和自定義通知功能。這樣的設計是為了簡化非會員的使用流程，同時也能維持平台的互動品質。

通報發布和管理的功能設計，主要是幫助用戶更方便地報告和尋找遺失或拾獲的物品。用戶在發布協尋通報時，需提供詳細的物品資訊，包括名稱、特徵描述、遺失地點與時間、聯絡方式，並可以選擇設置懸賞來提高找回物品的機率。拾獲通報則需要提供尋獲地點、時間和物品特徵，並上傳照片幫助確認。

為了提高搜尋的效率，平台設置了多種篩選條件，包括物品類型、地點和時間範圍等。用戶可以利用這些條件篩選或直接輸入關鍵字來搜尋相關的協尋或拾獲通報，以便快速找到目標資訊。此外，平台會持續收集用戶的回饋，根據需求進行系統優化和功能改進，如添加新的篩選條件或通知方式，讓平台更符合用戶的需求。

1.2.2 給 User 的功能

在本系統中，User 可以執行以下功能（有些功能非會員無法使用）：

1. 遺失物/拾獲物通報發布
    a. 用戶可以在平台上發布遺失物或拾獲物的通報，並填寫詳細的資訊（如物品名稱、特徵描述、遺失/拾獲的地點與時間、物品照片等）。
    b. 用戶可選擇是否設置懸賞來提升協尋效率，懸賞內容由用戶自行設定。
2. 協尋資訊瀏覽與搜尋
    a. 平台提供多種篩選條件（如物品類型、地點、時間）來幫助用戶快速找到相關的協尋資訊。
3. 用戶之間的互動
    a. 提供留言功能，讓用戶可以在協尋或拾獲通報下留言，進行公開的訊息交流。

b. 支援私訊功能，用戶可選擇直接私下聯繫通報者，以獲取更多詳細資訊或討論物品歸還事宜。

1.2.3 給 Admin 的功能

在本系統中，Admin 可以執行以下功能：

1. 資訊管理
    a. 管理者可對所有協尋/拾獲通報進行審核和管理，確保平台內容的準確性與適當性。
2. 用戶管理
    a. 管理者可以查看並管理用戶的基本資訊與歷史活動記錄，處理違規行為或異常狀況。
    b. 管理者可發送公告或通知，提醒用戶重要的系統更新或協尋進展。
3. 系統監控與維護
    a. 進行系統運行狀況的監控，定期檢查伺服器負載及資料庫狀態，確保系統穩定性。
    b. 定期備份資料庫，防止資料遺失。
    c. 接收用戶回饋，並根據需要進行功能優化和系統升級。
4. 統計資料
    a. 管理者可透過「統計數據」頁面來獲取一般使用者無法觸及的統計資料，以便進行市場分析。

2 系統設計

2.1 ER Diagram

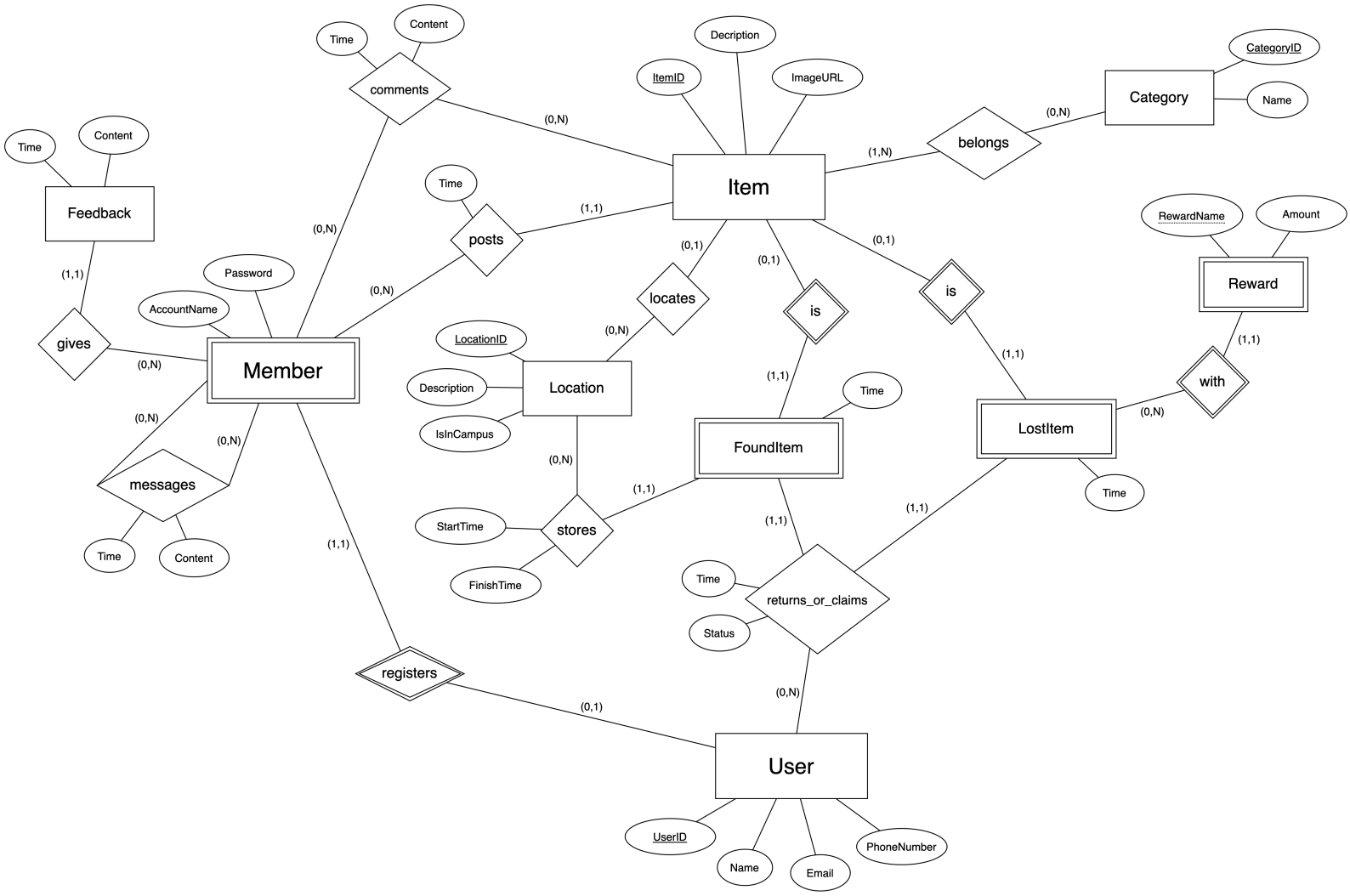

圖 1: 「NTULOST」的 ER Diagram

此 ERD 中，物品（Item）與使用者（User）為較主要的實體（entity），又分別衍生出會員（Member）與尋獲物、遺失物（FoundItem, LostItem）兩組弱實體（weak entity）。由圖中可以得知，「發文（posts）」、「留言（comments）」、「回饋（gives Feeback）」與「私訊（messages）」是僅限於會員能使用的功能，剩餘的「歸還或認領（returns_or_claims）」則對所有使用者皆開放。

除此之外，地點（Location）為剩餘實體中相對較複雜的實體，它與物品（Item）以「位在（locates）」連結，紀錄物品被找到或（可能）被遺失的地點（當然也有可能失主並沒有辦法提供可能遺失的地點，所以物品到地點之間的關聯為 (0,1)）；而它還與尋獲物（FoundItem）以「存放（stores）」連結，紀錄遺失物被存放的地點，因遺失物（LostItem）並不需要有這方面的紀錄，此部分就沒有與其有所連結。

另外，懸賞（Reward）只有在失主想懸賞遺失物時才會需要，因此只與遺失物（LostItem）連結，且可能一個遺失物會有多種懸賞品（若需要改變懸賞品的數量而不是類型，可直接修改數量（Amount），不需要額外建立一份懸賞），若將它們以編號全部分開又意義不大，因此最後是將懸賞設定為遺失物衍生的弱實體。

最後值得一提的是，同一個物品（Item）可能會同時屬於（belongs）多個類別（Category）的特性，我們並不會限制一個物品只會屬於一個類別，也能使關聯搜尋系統更為有效。

2.2 Relational Database Schema Diagram

根據前頁的 ERD 可以轉換成下頁的 Database Schema，共含有 16 個關聯（relation）。

大部分較清楚（PK 只有一兩個）的關聯就不在此多做描述，可以從圖中看到，「歸還或認領（RETURNS_OR_CLAIMS）」、「留言（COMMENTS」、「私訊（MESSAGE）」與「回饋（FEEDBACK）」皆以一個或多個 ID 與時間（Time）組成 PK，我們認為這樣做比新增編號、流水號更為有意義，因此不選擇額外新增編號、流水號作為 PK，以利資料的閱讀與簡潔。

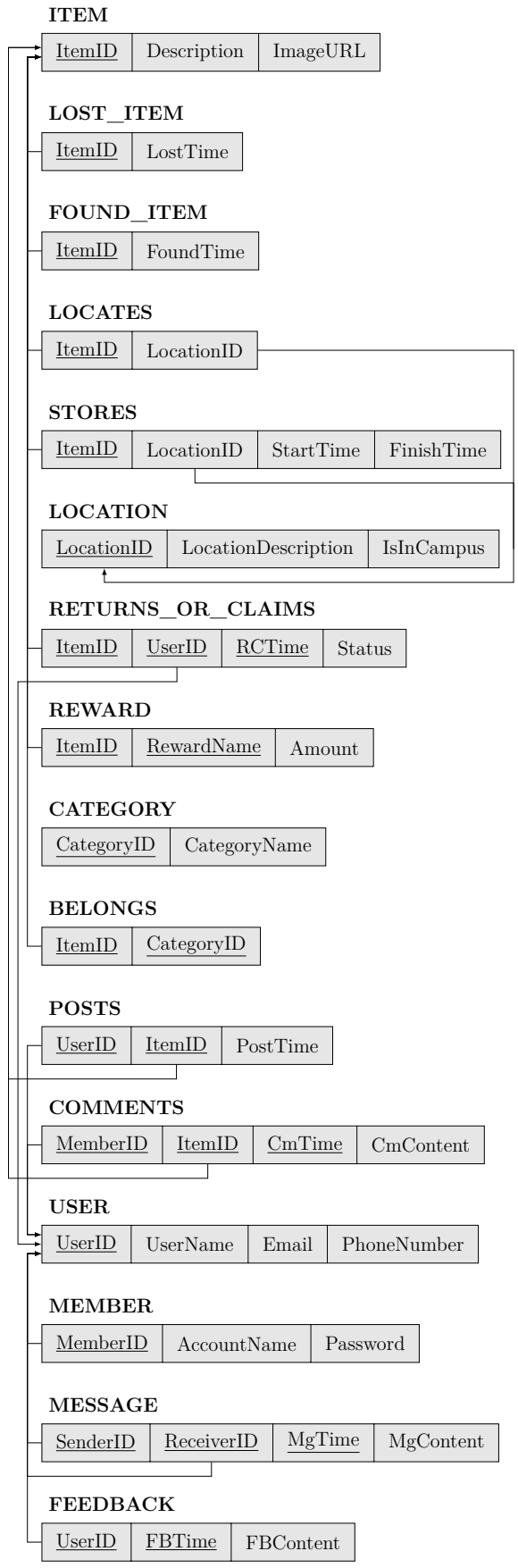

圖 2: 「NTULOST」的 Relational Database Schema Diagram

2.3 Data Dictionary

「NTULOST」的資料表共有圖 2 所示的 16 個，各個資料表的欄位相關資訊依序呈現在表 1 到表 16。

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK	Not Null, Unique	
Description	物品描述	text		Not Null	
ImageURL	物品圖片連結	text			

表 1: 資料表 ITEM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null, Unique	
LostTime	遺失時間	timestamp		Not Null	
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		

表 2: 資料表 LOST_ITEM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null, Unique	
FoundTime	尋獲時間	timestamp		Not Null	
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		

表 3: 資料表 FOUND_ITEM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null, Unique	
LocationID	地點編號	char(10)	FK: LOCATION(LocationID)	Not Null	
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		
LocationID: LOCATION(LocationID)		Cascade	Cascade		

表 4: 資料表 LOCATES 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null	
LocationID	地點編號	char(10)	FK: LOCATION(LocationID)	Not Null	
StartTime	開始存放時間	timestamp		Not Null	
FinishTime	結束存放時間	timestamp			
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		
LocationID: LOCATION(LocationID)		Cascade	Cascade		

表 5: 資料表 STORES 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
LocationID	地點編號	char(10)	PK	Not Null, Unique	
LocationDescription	地點描述	text		Not Null, Unique	
IsInCampus	是否在校園內	boolean		Not Null	{true, false}

表 6: 資料表 LOCATION 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null	
UserID	使用者編號	char(10)	PK, FK: USER(UserID)	Not Null	
RCTime	歸還或認領時間	timestamp	PK	Not Null	
Status	歸還或認領狀態	char(1)		Not Null	{P: 進行中, S: 完成, C: 取消, F: 失敗}
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		
UserID: USER(UserID)		Cascade	Cascade		

表 7: 資料表 RETURNS_OR_CLAIMS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null	
RewardName	懸賞物品名稱	varchar(15)	PK	Not Null	
Amount	數量	int		Not Null	
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		

表 8: 資料表 REWARD 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
CategoryID	類別編號	char(10)	PK	Not Null, Unique	
CategoryName	類別名稱	varchar(10)		Not Null, Unique	

表 9: 資料表 CATEGORY 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null	
CategoryID	類別編號	char(10)	PK, FK: CATEGORY(CategoryID)	Not Null	
Referential triggers		On Delete	On Update		
ItemID: ITEM(ItemID)		Cascade	Cascade		
CategoryID: CATEGORY(CategoryID)		Cascade	Cascade		

表 10: 資料表 BELONGS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
MemberID	會員編號	char(10)	PK, FK: USER(UserID)	Not Null	
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null, Unique	
PostTime	發文時間	timestamp		Not Null	
Referential triggers		On Delete	On Update		
MemberID: USER(UserID)		Cascade	Cascade		
ItemID: ITEM(ItemID)		Cascade	Cascade		

表 11: 資料表 POSTS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
MemberID	會員編號	char(10)	PK, FK: USER(UserID)	Not Null	
ItemID	物品編號	char(10)	PK, FK: ITEM(ItemID)	Not Null	
CmTime	留言時間	timestamp	PK	Not Null	
CmContent	留言內容	text		Not Null	
Referential triggers		On Delete	On Update		
MemberID: USER(UserID)		Cascade	Cascade		
ItemID: ITEM(ItemID)		Cascade	Cascade		

表 12: 資料表 COMMENTS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserID	使用者編號	char(10)	PK	Not Null, Unique	
UserName	使用者姓名	varchar(20)			
Email	電子郵件	text			
PhoneNumber	電話號碼	varchar(10)		Not Null	

表 13: 資料表 USER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
MemberID	會員編號	char(10)	PK, FK: USER(UserID)	Not Null, Unique	
AccountName	帳號名稱	varchar(30)		Not Null	
Password	密碼	varchar(20)		Not Null	
Referential triggers		On Delete	On Update		
MemberID: USER(UserID)		Cascade	Cascade		

表 14: 資料表 MEMBER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
SenderID	傳送者編號	char(10)	PK, FK: USER(UserID)	Not Null	
ReceiverID	接收者編號	char(10)	PK, FK: USER(UserID)	Not Null	
MgTime	私訊時間	timestamp	PK	Not Null	
MgContent	私訊內容	int		Not Null	
Referential triggers		On Delete	On Update		
SenderID: USER(UserID)		Cascade	Cascade		
ReceiverID: USER(UserID)		Cascade	Cascade		

表 15: 資料表 MESSAGE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
MemberID	會員編號	char(10)	PK, FK: USER(UserID)	Not Null	
FBTime	回饋時間	timestamp	PK	Not Null	
FBContent	回饋內容	text		Not Null	
Referential triggers		On Delete	On Update		
MemberID: USER(UserID)		Cascade	Cascade		

表 16: 資料表 FEEDBACK 的欄位資訊

2.4 正規化分析

當設計關聯式資料庫時，我們可以檢視資料庫綱目（database schema）是否滿足正規化（normalization）條件，因此我們將依序從第一正規式（1NF）到第四正規式（4NF）來說明「NTULOST」的關聯是如何滿足這些規則。

在 1NF 方面，如果每個關聯的屬性都是 simple 且 single-valued，換句話說，在關聯中沒有任何一個屬性是 composite 或 multi-valued，則滿足 1NF。從我們的 Schema 中，可以看到所有資料都是已經被簡化成 simple 且 single-valued，確保其滿足 1NF。

在 2NF 方面，如果關聯中的所有非鍵屬性（non-prime attribute）都完全功能相依（fully functional dependency）於任一候選鍵（candidate key），也就是沒有出現部分功能相依性（partial functional dependency），且此關聯滿足 1NF，則滿足 2NF。從我們的 Schema 中，可以看到我們將所有可能會違反 2NF 的資料都獨立成新的 Relations，確保其滿足 2NF。

在 3NF 方面，如果一個關聯中的非鍵屬性都沒有遞移相依（transitively dependency）於主鍵，則滿足 3NF。從我們的 Schema 中，可以看到我們將所有可能會違反 3NF 的資料都獨立成新的 Relations，確保其滿足 3NF。

在 BCNF 方面，要求關聯中的每一個功能相依的箭頭左方都要是超級鍵（superkey），也就是要確保 $X \rightarrow Y$ 的 X 一定是超級鍵。我們的 schema 也符合 BCNF。

最後是 4NF，由於「NTULOST」的所有關聯都不存在多值相依（multi-valued dependency），因此滿足 4NF 的條件。

2.5 前端系統設計與前後端系統整合

在本專案中，前端採用純文字介面設計，主要針對使用者的操作流程提供指引和輸出結果的展示。以下簡要說明系統提供的功能及每種角色的操作步驟。

使用者進入系統後，首先會看到一個簡單且直觀的初始選單，提供「註冊」、「登入」、「訪客註冊」以及「訪客登入」等多種選項。若為首次使用的用戶，可選擇「註冊」或「訪客註冊」來建立帳戶，並依照提示填寫必要的資料完成註冊流程；已經擁有帳戶的用戶則可直接選擇「登入」或「訪客登入」進入系統。登入後，根據使用者的身份，系統會導引至對應的首頁，分為「會員」、「訪客」與「管理者」三種界面。進入首頁後，使用者可依照選單中的指引進行各項操作。例如，發文功能會引導會員進入一個編輯界面，填寫物品描述、分類、時間等資訊，並可附加地點或圖片等細節；檢視貼文功能則列出所有公開的貼文，並允許用戶根據分類、地點等進行篩選；留言功能讓會員能與其他用戶互動，透過留言分享意見或討論話題。

在本專案中，我們採用 Python 作為主要的後端程式語言，並利用 PostgreSQL 資料庫來儲存和管理系統所需的資料。後端系統負責處理業務邏輯，進行資料的查詢、

更新與儲存，確保資料一致性與系統穩定性。前端則使用終端機介面模擬 client 與系統的互動。在這個架構中，client 可以透過終端機執行各種交易、查詢等操作，系統會即時地將操作結果以文字形式回饋至終端機，確保 client 能直觀地獲取相關資訊。例如，當使用者提交查詢請求後，系統會經由後端連接資料庫並執行 SQL 指令，將結果以結構化的方式呈現在終端機上，提升資料可讀性與使用體驗。此外，client 的身份會被後端系統記錄並儲存，系統會根據資訊提供相對應的回應。透過這種設計，我們有效實現了 client-server 架構，使前端與後端可以藉由終端機模擬的操作流程進行無縫的溝通與資料交換。

3 系統實作

3.1 資料建置

在本專案中，我們利用 Python 隨機生成各類測試資料，涵蓋了使用者個人資料、貼文、留言、回饋、私訊等多種類型的數據，這些資料的生成旨在幫助我們進行系統功能的測試與模擬。我們共生成了：使用者約 3000 位（包含會員與非會員）、物品約 1000 項（包含遺失物與尋獲物）、回饋約 1000 則、留言約 1.5 萬則、私訊約 200 萬則。

在生成測試資料的過程中，我們特別注意了資料的完整性與合理性。例如，每位使用者的個人資料（如帳號名稱、電子郵件）均獨立生成且無重複，確保測試結果的真實性與可靠性。此外，系統還模擬了各種可能的互動場景，從基本的資料檢索到複雜的交互功能，都能進行全面的驗證。這些測試資料讓我們能夠有效地模擬實際應用場景，確保系統在各種情境下的功能完整性與穩定性，同時也為後續的功能優化提供了可靠的數據基礎。

3.2 系統功能與其 SQL

在此系統中，SQL 查詢多處使用 `%s` 來接收外部輸入參數，主要目的是防範 SQL 注入攻擊。透過參數化查詢，所有外部輸入會被自動轉譯為安全的資料，不會被解釋為 SQL 語法，從而避免惡意操作。這種方式不僅提高了系統的安全性，也使查詢邏輯更清晰易懂，增強了程式碼的可維護性。

3.2.1 發文

```
WITH items_with_numbers AS (
    SELECT ItemID,
        CAST(SUBSTRING(ItemID FROM 3) AS INTEGER) AS
           number_part
    FROM item
```

```
    WHERE ItemID LIKE 'IT%'
)
SELECT 'IT' || LPAD((MAX(number_part) + 1)::TEXT, 8, '0')
   AS next_ItemID
FROM items_with_numbers;
```

首先我們會利用上面的 SQL，替新的物品設定開頭為 IT 的流水號 ID，接著再依據使用者是否輸入某些資訊，將輸入的資訊 INSERT 到它們該在的表中。由於此操作涉及到許多表的更動，因此在最前面會先輸入 BEGIN; 表示開始交易，最後若沒有任何錯誤產生便可以傳入 COMMIT; 確認交易完成並執行更動，反之則傳入 ROLLBACK; 以保證資料的一致性。

3.2.2 檢視貼文

檢視貼文分為兩種排序的方式，分別為「依照留言數排序」與「依照時間順排序」，以下僅呈現「依照留言數排序」的 SQL。

```
WITH CommentCounts AS (
SELECT
    p.ItemID,
    COUNT(cm.CmContent) AS CommentCount
FROM POSTS p
LEFT JOIN COMMENTS cm ON p.ItemID = cm.ItemID
GROUP BY p.ItemID
),
UniqueClaims AS (
    SELECT
        ItemID,
        MAX(Status) AS ClaimStatus
    FROM RETURNS_OR_CLAIMS
    GROUP BY ItemID
)
SELECT
    CASE
        WHEN p.UserID LIKE 'US%' THEN '匿名'
        ELSE m.AccountName
    END AS AccountName,
    c.CategoryName,
    i.Description,
    lo.LocationDescription,
    r.RewardName,
    r.Amount,
    cc.CommentCount,
```

```
    uc.ClaimStatus,
    i.ItemID
FROM POSTS p
LEFT JOIN MEMBERS m ON p.UserID = m.MemberID
JOIN ITEM i ON p.ItemID = i.ItemID
JOIN LOCATES l ON p.ItemID = l.ItemID
JOIN LOCATIONS lo ON l.LocationID = lo.LocationID
JOIN Belongs b ON p.ItemID = b.ItemID
JOIN CATEGORY c ON b.CategoryID = c.CategoryID
LEFT JOIN REWARD r ON p.ItemID = r.ItemID
JOIN CommentCounts cc ON p.ItemID = cc.ItemID
LEFT JOIN UniqueClaims uc ON p.ItemID = uc.ItemID
ORDER BY cc.CommentCount DESC;
```

這句 SQL 的目的是從資料庫中取得貼文的詳細資訊，並依據留言數由多到少進行排序，以便分析最受關注的貼文。首先，透過子查詢計算每篇貼文的留言數，將其結果加入主要查詢中；同時，利用另一個子查詢確定每項物品的唯一申請狀態，避免資料重複或混淆。在主要查詢中，綜合貼文的發布者資訊（如會員名稱或匿名標示）、物品描述、地點資訊、懸賞細節、留言數量及申請狀態等多項欄位，最終按留言數排序，方便呈現受關注程度的貼文清單。

3.2.3 檢視貼文

```
SELECT c.memberid, u.username, c.itemid, i.description, c.
   cmtime, c.cmcontent
FROM COMMENTS AS c
JOIN USERS AS u ON c.memberid = u.userid
JOIN ITEM AS i ON c.itemid = i.itemid
WHERE i.itemid = %s
ORDER BY c.cmtime DESC
LIMIT 10;
```

這句 SQL 的目的是查詢特定物品（以 `i.itemid = %s` 指定）的最新留言內容，並限制結果最多顯示 10 筆。查詢中，將留言表 (COMMENTS) 與使用者表 (USERS) 以及物品表 (ITEM) 進行連結，取得留言者的會員 ID 和用戶名、物品描述、留言時間以及留言內容等資訊。結果按照留言時間 (c.cmtime) 進行降序排列，確保最新的留言優先顯示，適合用於呈現某件物品相關的最新互動記錄。

3.2.4 私訊

```
SELECT
    CASE
```

```
        WHEN m.senderid = %s THEN u_r.username
        ELSE u_s.username
    END AS the_username,
    MAX(m.mgtime) AS last_message_time
FROM MESSAGE AS m
JOIN USERS AS u_s ON m.senderid = u_s.userid
JOIN USERS AS u_r ON m.receiverid = u_r.userid
WHERE m.senderid = %s OR m.receiverid = %s
GROUP BY the_username
ORDER BY last_message_time DESC;

SELECT m.senderid, u_s.username AS senderName, m.receiverid
   , u_r.username AS receiverName, m.mgtime, m.mgcontent
    FROM MESSAGE AS m
    JOIN USERS AS u_s ON m.senderid = u_s.userid
    JOIN USERS AS u_r ON m.receiverid = u_r.userid
    WHERE
        (senderid = %s AND u_r.username = %s)
        OR (u_s.username = %s AND receiverid = %s)
    ORDER BY m.mgtime DESC;
```

第一句 SQL 的功能是從私訊表中找出與指定使用者（以 %s 代表的 senderid）有互動的所有對象，並顯示每位對象的用戶名稱以及最後一則訊息的時間。查詢通過檢查訊息的發送者和接收者來篩選相關記錄，並利用 CASE 判斷當前使用者的私訊對象。最終結果依照最後一次訊息的時間由近到遠排序，方便使用者快速查看最新的私訊。

第二句 SQL 的功能是檢索與特定對象的訊息記錄，包括訊息的發送者名稱、接收者名稱、訊息時間以及訊息內容。查詢會根據雙方的身份關係進行條件篩選，確保只返回符合條件的訊息記錄。結果按照訊息的時間戳從最新到最舊排序，方便使用者查看完整的對話歷史。

結合兩句 SQL，使用者便可以查看他與其他使用者的私訊紀錄。

3.2.5 搜尋

分為搜尋「遺失物」與「尋獲物」兩個版本，因多處重疊，因此下面只敘述「遺失物」版本的。

```
SELECT
c.categoryname, i.description, lt.locationdescription, p.
   posttime
FROM posts p
JOIN lost_item li ON li.itemid = p.itemid
JOIN locates lo ON lo.itemid = p.itemid
```

```
JOIN locations lt ON lt.locationid = lo.locationid
JOIN belongs b ON b.itemid = p.itemid
JOIN category c ON c.categoryid = b.categoryid
LEFT JOIN members m ON m.memberid = p.userid
JOIN item i ON i.itemid = p.itemid
WHERE 1=1
```

這句 SQL 用來從資料庫中提取與遺失物相關的貼文資料，主要檢索的欄位包括物品的類別名稱、描述、遺失地點以及發文時間。資料是通過多個表格進行連結，查詢最初是固定的，沒有依賴用戶的輸入條件。

之後根據用戶提供的資訊（如物品類別、時間範圍、遺失地點）在 SQL 中動態插入搜尋的條件。系統會向使用者詢問物品類別、搜尋的時間範圍（開始時間和結束時間）、以及遺失地點等選項，根據使用者的輸入，系統會動態地修改 SQL 查詢，附加適當的條件並將對應的值加入到查詢的參數中。

3.2.6 分析熱門地點

```
SELECT
    L.locationdescription,
    COUNT(DISTINCT F.itemid) AS found_count,
    COUNT(DISTINCT LST.itemid) AS lost_count
FROM locations L
LEFT JOIN locates LO ON L.locationid = LO.locationid
LEFT JOIN found_item F ON LO.itemid = F.itemid
LEFT JOIN lost_item LST ON LO.itemid = LST.itemid
GROUP BY L.locationdescription
ORDER BY (COUNT(DISTINCT F.itemid) + COUNT(DISTINCT LST.
   itemid)) DESC
limit 10;
```

這句 SQL 旨在找出遺失物和尋獲物數量最多的前十個地點。透過將各表的連結，計算每個地點的尋獲物和遺失物的數量。最後，根據尋獲物和遺失物的總和對地點進行排序，並選出前十個地點。查詢結果顯示每個地點的名稱，以及該地點的尋獲物和遺失物數量。

3.2.7 分析用戶參與程度

```
SELECT
    U.userid,
    U.username,
    COUNT(DISTINCT P.itemid) AS posts_count
```

```
FROM users U
LEFT JOIN posts P ON U.userid = P.userid
GROUP BY U.userid, U.username
ORDER BY posts_count DESC
LIMIT 10;
```

這句 SQL 分析用戶的參與度，具體來說是通過統計每個用戶發佈的貼文數量來衡量。它通過 users 表和 posts 表的連結，對每個用戶統計不同物品 ID 的數量（即發佈的貼文數量），結果按貼文數量降序排序，並回傳前十個用戶及其發佈的貼文數量。

3.2.8 分析物品類別趨勢

```
SELECT
    L.locationdescription,
    COUNT(DISTINCT F.itemid) AS found_count,
    COUNT(DISTINCT LST.itemid) AS lost_count
FROM locations L
LEFT JOIN locates LO ON L.locationid = LO.locationid
LEFT JOIN found_item F ON LO.itemid = F.itemid
LEFT JOIN lost_item LST ON LO.itemid = LST.itemid
GROUP BY L.locationdescription
ORDER BY (COUNT(DISTINCT F.itemid) + COUNT(DISTINCT LST.
   itemid)) DESC
limit 10;
```

這句 SQL 用於分析物品類別的趨勢，它對每個類別進行分組，計算每個類別中的遺失物和尋獲物數量，結果按遺失物數量降序排列，若有相同數量，則根據尋獲物數量進行次級排序。

3.2.9 分析懸賞效益

```
SELECT
    R.rewardname,
    R.amount,
    COUNT(DISTINCT LST.itemid) AS
       number_of_lost_with_reward
FROM reward R
LEFT JOIN lost_item LST ON R.itemid = LST.itemid
GROUP BY R.rewardname, R.amount
ORDER BY number_of_lost_with_reward DESC
limit 10;
```

這句 SQL 分析各種懸賞對遺失物品的吸引效果，計算每個懸賞項目對應的遺失物品數量，結果按遺失物數量降序排列，並返回前十個回報項目，顯示回報名稱、金額以及對應的遺失物品數量。

3.2.10 分析熱門時段

```
SELECT
    hour AS time_hour,
    SUM(lost_count) AS total_lost_count,
    SUM(found_count) AS total_found_count,
    SUM(lost_count + found_count) AS total_items_count
FROM (
    SELECT
        EXTRACT(HOUR FROM losttime) AS hour,
        COUNT(DISTINCT L.itemid) AS lost_count,
        0 AS found_count
    FROM lost_item L
    GROUP BY hour

    UNION ALL

    SELECT
        EXTRACT(HOUR FROM foundtime) AS hour,
        0 AS lost_count,
        COUNT(DISTINCT F.itemid) AS found_count
    FROM found_item F
    GROUP BY hour
) AS combined_data
GROUP BY time_hour
ORDER BY total_items_count DESC;
```

這句 SQL 旨在分析一天中各個小時內，遺失物和尋獲物的數量分佈情況。它分別從 `lost_item` 和 `found_item` 表中提取遺失物和尋獲物的時間，並使用 `EXTRACT(HOUR FROM ...)` 來提取小時。將兩者的數據合併後，按小時總結遺失物和尋獲物的數量，並計算總物品數量（遺失物和尋獲物的總和）。結果按總物品數量降序排列，以顯示最活躍的小時段。

3.3 SQL 指令優化

因為私訊的量大，在處理私訊時通常會花費很多時間，所以我們對 `MESSAGE` 表中的 `senderid` 與 `receiverid` 分別建立了 index，沒有 index 時預計 cost 為 42742.06、實

際花費的時間為 340.212 ms，而建立 index 後預計 cost 為 5665.70、實際花費的時間為 7.487 ms，有明顯的效率提升。除了建立 index 外，我們也利用兩種不同的 SQL 寫法來比較效率（程式碼與 cost 的比較在 index.ipynb 檔案中）。

不好的 SQL 查詢經常因為使用嵌套子查詢而導致效率低下。例如，`SELECT username FROM USERS` 的嵌套子查詢會對每一行數據執行一次操作，導致大量冗餘計算。每次需要從 `USERS` 表中查找對應的 `userid` 時，都會進行一次表掃描，表掃描的次數因此遠超出必要範圍。此外，嵌套子查詢無法有效利用索引（如果存在索引），因為它們作為動態查詢處理單行數據，而非進行批量操作，這樣的查詢方式進一步增加了計算開銷。而當查詢中的 `CASE` 與子查詢結合時，查詢優化器難以合併操作，最終導致整個 `GROUP BY` 和排序都在完成子查詢後執行，進一步增加了排序的計算量。

相比之下，優化後的 SQL 查詢使用直接的 `JOIN` 操作，將 `MESSAGE` 表與 `USERS` 表通過 `JOIN` 關聯起來，避免逐行查詢 `USERS` 表的操作。`JOIN` 允許關聯操作以批量方式處理，大幅減少了多次表掃描的開銷。如果 `MESSAGE.senderid` 和 `USERS.userid` 上設置了索引，`JOIN` 操作能直接利用這些索引，快速定位匹配行。這使得查詢優化器能夠更高效地對 `JOIN` 的排序和過濾進行優化。同時，查詢的複雜度也明顯降低，單次掃描即可完成 `CASE` 判斷，避免了多次動態執行子查詢的情況。優化後的查詢執行計劃更加高效，查詢優化器能更好地合併 `WHERE` 和 `GROUP BY` 條件，從而減少排序和聚合操作的計算開銷，提升整體性能。

交易管理與併行控制在上節已有所提及便不再贅述。

4 分工資訊與專案心得

4.1 分工資訊

馬松鐸：系統分析、系統設計、系統程式碼撰寫、DEMO 展示、影片拍攝
洪榮盛：系統分析、系統設計、系統程式碼撰寫、Proposal 撰寫、期末報告撰寫
李盈盈：系統分析、系統設計、系統程式碼撰寫、簡報製作

4.2 專案心得

馬松鐸

在這次期末專案中，我參與了「NTULOST」系統的分析、設計，以及部分功能實作與效能最佳化，這段過程讓我學到了許多實用技能與知識，也深刻體會到設計出一個完整系統的挑戰與不容易。

從一開始的點子發想到具體的系統分析，再到精細的系統設計，接著進入功能實作與效能調整，每一個階段都充滿了需要解決的問題與不斷學習的過程。系統分析需

要將抽象的想法具體化，厘清需求與用戶流程；系統設計則要求將功能結構化，並規劃合理的資料庫架構；到了實作階段，更需要兼顧功能的完整性與系統的運行效率。尤其在效能優化的過程中，我對資料庫運行的內部機制有了更清晰的認識。從索引的使用到查詢語句的調整，這些細節雖然看似微小，卻直接影響了整個系統的流暢度與穩定性，也讓我明白資料庫設計與效能平衡的重要性。

這次專案讓我不僅提升了技術能力，也更理解了一個系統從無到有的艱辛與成就感。完成一個能解決實際問題的系統，是一個需要創意、規劃與執行力的完整體驗，這段經歷對我的學習與未來發展意義深遠。

洪榮盛

這次的專案又帶給我全新的體驗，之前自己建網站的時候有用過 php 連接資料庫，這次則是用 python，對於完全不熟悉 python 的我算是一大挑戰 (?)，一直都在寫 C 的我在寫這次專案的過程中常常會有語法寫錯的時候，還好現在有 ChatGPT 可以幫我更有效率地抓出並修改錯誤。

還有合作寫程式真的比自己寫難上不少，當我認為我們前期已經討論地很有共識，直到開始做之後才發現還有很多細節沒有事先討論清楚，導致後期需要花一些時間整合大家寫出來的結果，這也給予我一個良好的經驗，讓我之後與他人合作寫程式時能考慮得更周全。

最後，有一點覺得稍嫌可惜，就是我原先希望能在終端機上做出很精緻的畫面，本來預計是利用 curses 資源包建立完整的頁面，再寫程式把圖片像素化印在終端機上，很可惜後來沒什麼時間因而作罷，但希望以後還有機會實現這個有趣的操作。

李盈盈

在實作過程中，我獲得了許多寶貴的知識與經驗，這段旅程充滿挑戰，但也讓我成長良多。從分析需求（讓用戶依不同排序方式瀏覽貼文）到撰寫程式碼，深刻體會到細化需求的重要性！具體到邏輯細節，例如過濾已認領貼文、動態顯示分頁數等，每一部分的實現都需要經過反覆思考與驗證。在設計功能時，我逐漸培養了以用戶需求為核心的思維。例如，分析用戶參與度的功能不僅能為管理者提供數據支持，還能引導更多用戶積極參與協尋與拾獲活動，讓整個平台更具價值。

雖然部分工作是獨立完成，但與其他同學的交流讓我更了解合作的重要性。我們針對專案功能需求進行了深入討論，彼此分工並確保整體系統的一致性，這讓我認識到溝通與協作是成功的關鍵。

這次的經驗啟發了我對未來功能改進的思考，例如加入更細化的篩選條件（依地點、類別等）或更人性化的提醒機制。我希望未來能進一步強化這些設計，讓系統變得更加智慧更高效。