

113-1 資料庫期末專案

My Accounting Assistant

隊名：資科營小視窗

<u>111703013 黃蓉容</u>	111703005 唐湘怡	111703023 沈思妤
111703029 劉 白	111703052 林冠儀	111703055 吳佩萱



需求分析

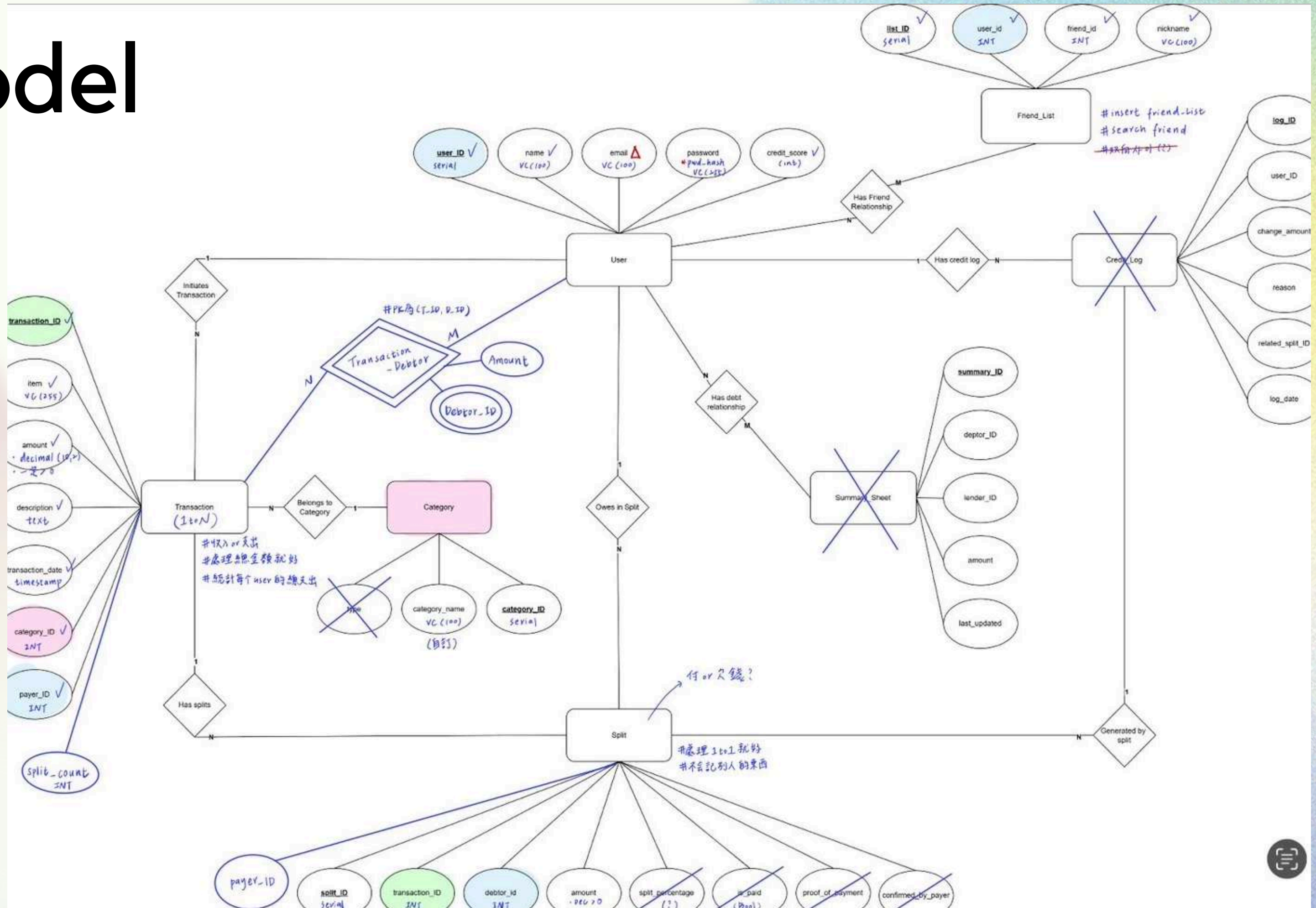
1. 背景與問題

- 在日常生活中需要處理**多人間的費用結算**，並且希望能夠快速記錄交易、並處理分賬。
- 傳統的分賬方式無法有效**跟蹤朋友之間的交易記錄**和**信用狀況**。

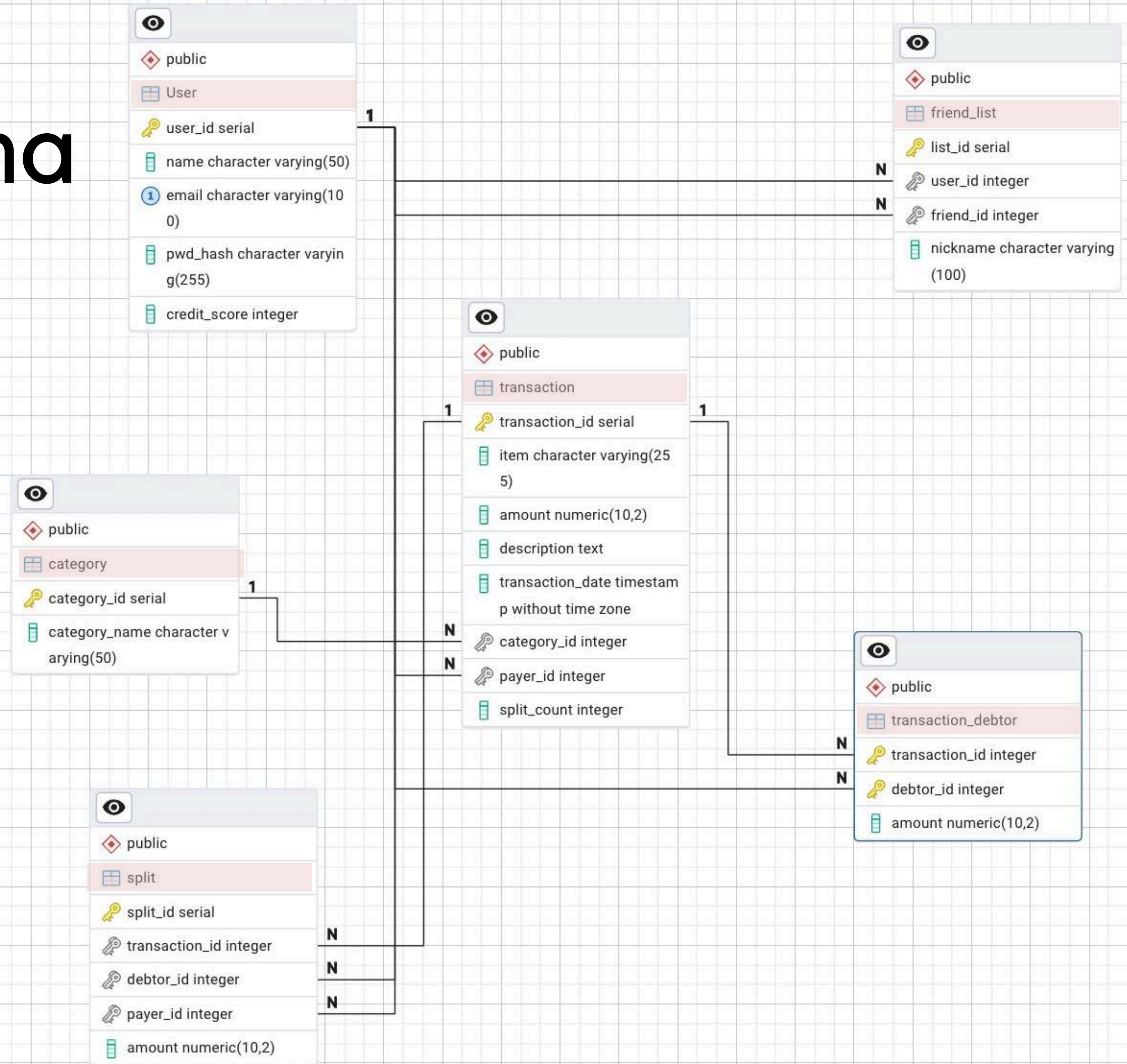
2. 核心需求

- 用戶註冊、登入
- 建立個人帳本
- 即時分帳並記錄
- 好友清單
- 個人信用積分、積分排行榜

ER-Model



relational schema



Relational Schema

-- User 表: 存儲使用者資訊

```
CREATE TABLE "user" (  
    user_ID SERIAL PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    pwd_hash VARCHAR(255) NOT NULL,  
    credit_score INTEGER  
);
```

-- "friend_list" 表: 存儲使用者之間的好友關係

```
CREATE TABLE "friend_list" (  
    list_ID SERIAL PRIMARY KEY,  
    user_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    friend_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    nickname VARCHAR(100),  
    -- Constraint: Ensure no duplicate friendships between users  
    CONSTRAINT unique_user_friend UNIQUE (user_ID, friend_ID)  
);
```

-- Transaction 表: 存儲交易資訊

```
CREATE TABLE "transaction" (  
    transaction_ID SERIAL PRIMARY KEY,  
    item VARCHAR(255) NOT NULL,  
    amount DECIMAL(10, 2) NOT NULL CHECK (amount > 0),  
    description TEXT,  
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    category_ID INTEGER REFERENCES "category" (category_ID) ON DELETE SET NULL,  
    payer_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    split_count INTEGER NOT NULL  
);
```

-- split 表: 記錄更詳細的分帳資訊

```
CREATE TABLE "split" (  
    split_ID SERIAL PRIMARY KEY,  
    transaction_ID INTEGER NOT NULL REFERENCES "transaction" (transaction_ID) ON DELETE CASCADE,  
    debtor_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    payer_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    amount DECIMAL(10, 2) NOT NULL CHECK (amount > 0)  
);
```

-- Transaction_Debtor 關聯表: 處理交易與債務人的多對多關係

```
CREATE TABLE "transaction_debtor" (  
    transaction_ID INTEGER NOT NULL REFERENCES "transaction" (transaction_ID) ON DELETE CASCADE,  
    debtor_ID INTEGER NOT NULL REFERENCES "user" (user_ID) ON DELETE CASCADE,  
    amount DECIMAL(10, 2) NOT NULL CHECK (amount > 0),  
    PRIMARY KEY (transaction_ID, debtor_ID)  
);
```

-- Category 表: 存儲交易類別資訊

```
CREATE TABLE "category" (  
    category_ID SERIAL PRIMARY KEY,  
    category_name VARCHAR(50) NOT NULL  
);
```


系統架構

1. 前端

- 使用 **React.js + Vite** 開發

2. 後端

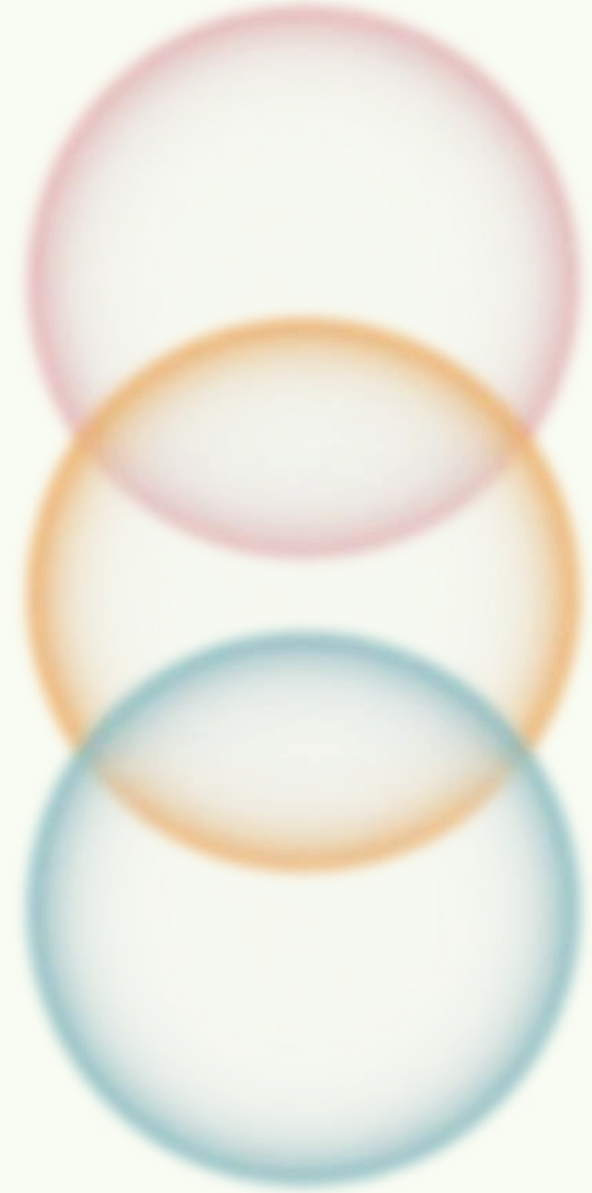
- 使用 **Flask** 開發 API，包括處理用戶註冊、登入、好友管理、交易記錄查詢等功能。
- 採用 **Blueprint 模組化設計**與 **Axios** 進行資料傳遞。

3. 資料庫

- 使用 **PostgreSQL**，依據 ER Model 設計資料表。

4. 容器化部署

- **Docker** 容器化後端、前端及資料庫，方便同步開發環境。
- Docker Compose 用於多服務（前端、後端、資料庫、Redis）整合。

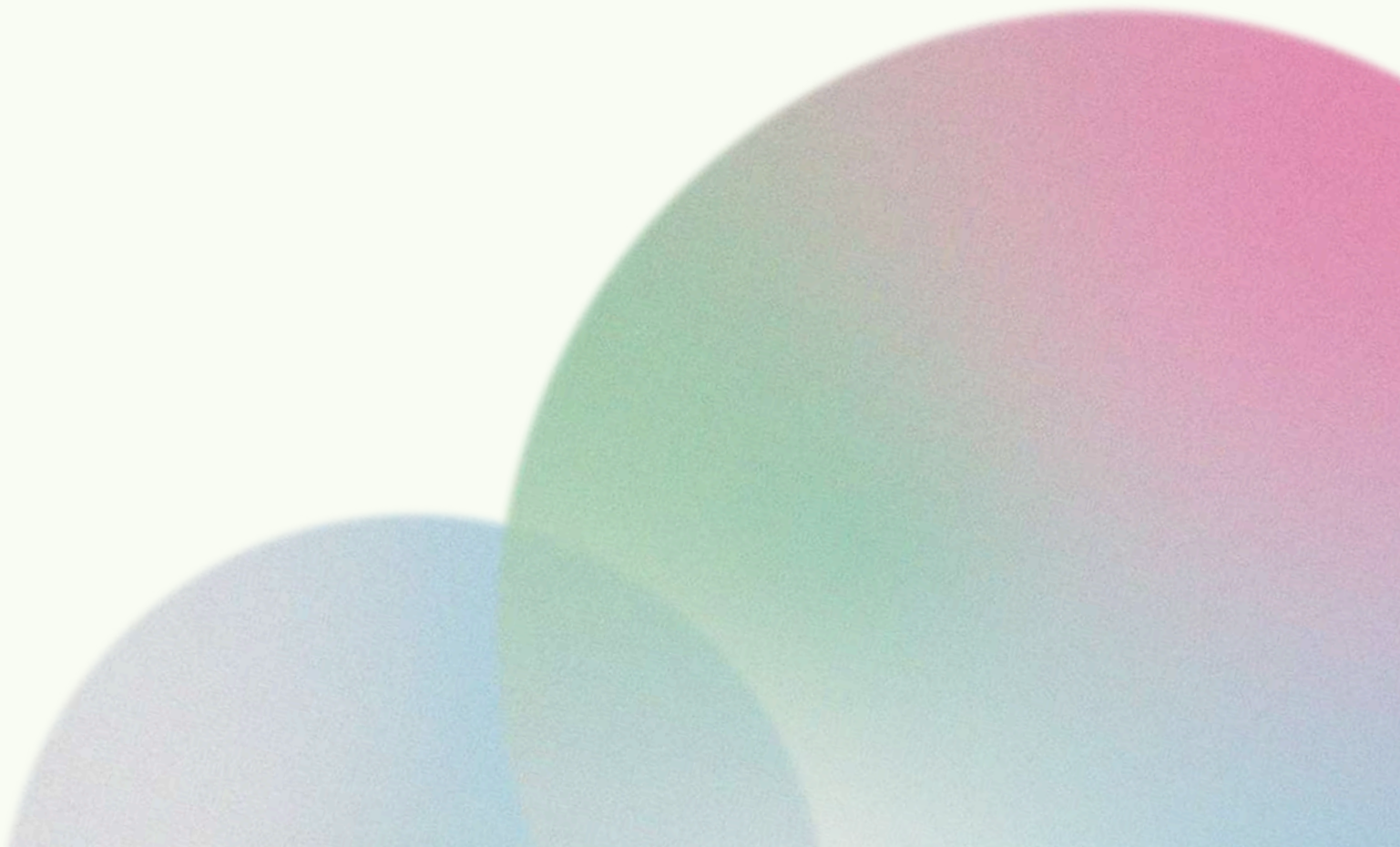


核心功能

登入/註冊 (Login/Sign Up)

- 使用者能註冊帳號，輸入基本資料（姓名、電子郵件、密碼）。
- 使用者可安全登入系統，系統需哈希加密儲存密碼並驗證輸入資料的正確性。

建立帳務記錄 (Create Transaction)

- 使用者可**新增交易記錄**，包括項目**名稱、金額、描述、類別、交易日期**等。
 - 支援**交易分類**與標記付款人（需為好友關係）。
 - 支援將交易金額分配給多位參與者。
- 



分帳邏輯 (Split Logic)

- 支援將交易金額分配給多位債務人。
- 自動計算每位債務人的應付金額。

好友清單 (Friend List)

- 使用者可管理好友清單，包括**新增**、**刪除**好友
- 支援好友**暱稱**
- 驗證好友關係的唯一性，避免重複添加

個人積分 (Credit Score)

- 系統追蹤用戶**信用積分**，根據支付行為進行**評分**。
- 提供積分查詢功能，提醒用戶保持良好的信用紀錄。
- 定期同步 **Redis** 資料至資料庫



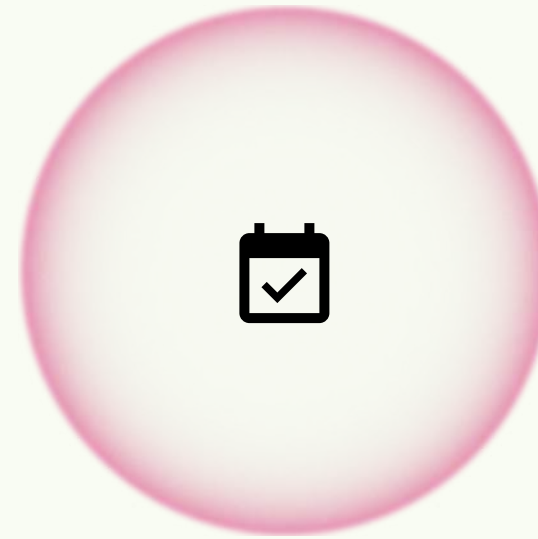
demo

心得與收穫



團隊協作

- 團隊溝通與分工的重要性
- 完成多模組整合並解決開發過程中的衝突



全端設計

- 前後端、API的整合
- 資料庫的設計



問題解決

- 遇到挑戰時找到有效的解決方案並優化系統性能
- 使用Docker解決跨平台的環境配置問題，確保團隊開發與測試的一致性

分工表

負責人	功能/ 任務
黃蓉容 (組長)	前端：HomePage、SplitPage、WelcomePage 後端：HomePage GitHub merge(專案整合)、環境建置、專案分工
唐湘宜	前端：Transaction
沈思妤	SignUp/Login前後端
劉 白	Friendlist前後端、建db基底、簡報製作
林冠儀	ScorePage前後端、建db基底、GitHub merge(專案整合)
吳佩萱	前端：Transaction 後端：Transaction、SplitPage、SignUp、Login GitHub merge(專案整合)、建db基底
大家一起	功能分析、畫 ER Model

Thank you!