**報告文件-第九組 美食清單**

這次作業透過實作一個「餐廳推薦系統」，實際演練完整的全端開發流程，培養系統設計、問題解決及安全管理能力。

在資訊爆炸的時代，使用者往往被過多的選擇所淹沒。本專案精準鎖定現代都市生活中的兩大核心痛點，並提出針對性的技術解決方案：

**面對的問題與解決方案**

* **選擇困難 / 決策效率低**
  + **問題**：面對眾多餐廳選項，使用者容易出現決策疲勞與猶豫。
  + **解法**：借鏡 Tinder 的滑動介面，設計直覺的卡片式滑動互動，左滑代表「無感」，右滑代表「喜歡」，降低使用者的心理負擔，提升互動體驗。
* **資訊不透明**
  + **問題**：許多餐廳的線上資訊，如菜單與價格，經常是零散、過時甚至不存在的。
  + **解法**：建立由社群或開發者維護的平台，透過結構化的資料庫設計，確保每家餐廳都能呈現清晰、完整的菜單項目與價格，並提供高品質的實物照片，實現資訊的透明化。

為此，我們精心打造了以下系統特色：

* **Tinder 式滑動互動介面**：利用 framer-motion 函式庫，提供媲美原生 App 的流暢物理動畫，讓每一次滑動都充滿回饋感，有效提升使用者黏著度。
* **安全防護機制**：導入 JWT 雙 Token（Access/Refresh）策略，結合儲存於 HttpOnly Cookie 中的 Refresh Token 與 CSRF 權杖，在確保便利性的同時，最大化地防禦了 XSS 與 CSRF 兩大主流 Web 攻擊。
* **雲端圖片直傳架構**：採用 Cloudflare R2 物件儲存，並實作 Presigned URL 技術。此設計讓客戶端（使用者瀏覽器）能直接將圖片上傳至雲端，伺服器僅需負責簽發一次性授權，極大降低了後端伺服器的頻寬與運算負擔，是現代化高流量應用的最佳實踐。
* **RESTful API 後端**：基於輕量而強大的 Flask 3 框架，搭配 SQLAlchemy 2.0 的非同步 ORM 能力，並整合 Supabase 提供的企業級 PostgreSQL 資料庫服務，確保了資料處理的穩定性與可擴展性。
* **次世代前端體驗**：採用 Next.js 15 App Router 與 React Server Components (RSC) 架構，搭配以 shadcn/ui 為核心的元件庫。打造出順暢且具備高度互動性的使用者介面。

**2. 背景資料**

**技術棧簡介**

* **後端技術**：Flask 3、SQLAlchemy 2.0（ORM）、PostgreSQL（Supabase 雲端服務）。
* **安全性技術**：JWT 雙 Token（Flask-JWT-Extended）、CSRF 防護、flask-cors。
* **前端技術**：Next.js 15、TypeScript、shadcn/ui、Radix UI、TailwindCSS、framer-motion。
* **雲端服務**：Cloudflare R2 雲端圖片儲存，採用 Presigned URL 技術提升效能。

**3. 環境介紹**

| **類別** | **工具與技術** |
| --- | --- |
| 開發語言 | Python 3　TypeScript |
| 後端框架 | Flask 3 |
| 前端框架 | Next.js 15 (App Router) |
| UI 套件 | shadcn/ui　TailwindCSS |
| 動畫控制 | framer-motion |
| ORM | SQLAlchemy 2.0 |
| 資料庫 | Supabase PostgreSQL |
| 雲端圖片儲存 | Cloudflare R2 |
| 權限控管 | Flask-JWT-Extended (JWT 雙 Token) |
| 安全防護 | CSRF Token　flask-cors |
| Cookie 設定 |  |
| 作業系統 | Linux Ubuntu 24.01 |
| 環境管理 | Python: uv　Next.js: bun |

**4. 實作方法**

**系統功能實作**

* **會員系統：JWT 雙權杖認證流程**
  + 使用者註冊/登入後，後端簽發 Access Token（短效）與 Refresh Token（長效），分別存放於前端記憶體與 HttpOnly Cookie 中。
  + 前端在 API 請求中夾帶 Access Token，後端驗證其有效性。
  + 當 Access Token 過期時，前端自動使用 Refresh Token 向後端請求新的 Token，實現無縫體驗。
  + 引入 CSRF 權杖，防範跨站請求偽造攻擊。
  + 註冊：/api/auth/register
  + 登入：/api/auth/login
  + Refresh Token：/api/auth/refresh
  + 登出：/api/auth/logout
* **餐廳與菜單管理：結構化資料庫與 RESTful API**
  + 設計資料表：users、restaurants restaurant\_menu，透過外鍵定義實體間的關聯。
  + 提供 RESTful API，如：
    - 新增餐廳：/api/restaurant/add
    - 查詢所有餐廳：/api/restaurant/all
    - 新增菜單：/api/restaurant-menus/add/<restaurant\_id>
    - 查詢菜單：/api/restaurant-menus/get/<restaurant\_id>
* **圖片上傳管理：基於 Presigned URL 的高效直傳方案**
  + 取得上傳 URL：/api/images/presigned/upload
  + 前端透過 Presigned URL 直接上傳至 Cloudflare R2
  + 上傳完成後回傳 image\_key 儲存至資料庫
* **Tinder 互動滑動推薦**
  + **動畫實現**：利用 framer-motion 的 useDragControls 和 motion 元件，監聽使用者的拖曳手勢。我們設定了物理動畫參數（如彈性、阻尼），讓卡片滑動與回彈的效果極度逼真。
  + **邏輯觸發**：在 onDragEnd 事件中，我們檢查卡片的最終 offset。若 X 軸位移超過預設閾值（例如螢幕寬度的 30%），則判定為一次有效的滑動。
  + **後端通訊**：根據滑動方向（左或右），觸發對應的 like 或 dislike 動作，並非同步呼叫 /api/swipes 端點，將結果記錄至後端。
  + **即時反饋**：無論成功與否，卡片都會以流暢的動畫移出畫面，同時下一張推薦卡片會自動補上，確保了不間斷的互動流程。
* **中介層權限驗證**
  + 利用 Next.js Middleware，對需要保護的路由進行攔截。
  + 使用 jose 函式庫，在 Middleware 中解析並驗證 Access Token，驗證失敗則重導向至登入頁面

**資料流程**

**註冊驗證流程**

使用者填入 username & Email & 密碼→ 前端

驗證格式->發送 POST 請求 /api/auth/register→後端

驗證是否重覆→密碼加密→將資料寫進資料庫→前端

註冊成功

**登入驗證流程**

使用者填入 Email & 密碼 → 前端驗證格式 → 發送 POST 請求 /api/auth/login

→ 後端驗證帳密 → 產生 Token 寫入 Cookie → 前端登入成功

**圖片上傳流程**

使用者選擇圖片 → 取得上傳 URL → 直接上傳至 Cloudflare R2 → 回傳 image\_key 存入資料庫

**Tinder 滑動互動流程**

前端載入卡片 → 使用者滑動 → 判斷 Like/Dislike → 更新卡片堆疊 → 結束後可重新開始

**中介層權限驗證流程**

訪問保護頁面 → Middleware 驗證 Cookie 內 JWT → 驗證成功允許進入，失敗則導向登入頁

**5. 實作心得**

* **Presigned URL 的啟示**：實作 Presigned URL 讓我們深刻體會到，現代雲端架構的核心在於「責任分離」。將檔案傳輸的重擔從應用程式伺服器轉移至專用的雲端服務，不僅是技術上的優化，更是架構設計上的一次躍進，讓我們的前後端能更專注於自身的核心業務邏輯。
* **安全性的深度權衡**：成功整合 JWT 雙 Token 與 CSRF 防護，讓我們理解到，系統安全並非單點防禦，而是一個體系化的工程。我們在「使用者無感刷新」的便利性與「權杖洩漏風險」的安全性之間找到了精妙的平衡點，這是從書本理論邁向工程實踐的關鍵一步。
* **中介層的戰略價值**：Next.js Middleware 的應用，讓我們對請求生命週期的理解更加全面，也意識到在分散式系統中，越早處理請求，就能獲得越大的效能與安全增益。
* **ORM 的開發者體驗革命**：SQLAlchemy 2.0 搭配其完整的 Type Hinting 支援，徹底改變了我們的開發模式。它不僅是資料庫的操作工具，更是我們程式碼的「靜態文檔」。編譯器（Linter/Type Checker）能在執行前就捕捉到潛在的型別錯誤，這讓重構變得極為自信，也大幅提升了長期維護的程式碼品質。
* **前端工程化的專業追求**：選擇 Next.js 15 與 shadcn/ui，代表我們對前端工程化的態度。我們不再滿足於僅僅實現功能，而是追求更合理的元件架構、更極致的效能、以及更具擴展性的設計模式。這份專案的 UI/UX 成果，證明了現代化前端框架與設計理念的結合，足以打造出具備商業水準的產品，這也將是我們作品集中最亮眼的一筆。

**6. 附註**

**團隊分工明細**

* 系統架構與後端開發：成員 蘇惟鈞
* 前端開發與互動介面設計：成員 蘇惟鈞 , 哖芯瑜
* 資料庫設計與安全性驗證：成員 蘇惟鈞
* 專案文件撰寫與整合：成員 蘇惟鈞 , 哖芯瑜

**參考資源**

* Next.js 15 官方文件
* Flask 官方文件
* SQLAlchemy ORM 官方文件
* Supabase 官方文件
* Cloudflare R2 官方說明
* Flask-JWT-Extended 官方文件
* framer-motion 官方指南
* shadcn/ui 官方文件