資安事件通報系統

本作品為一個完整的資安事件通報系統，基於 Flask 框架開發，旨在為普通民眾、管理人員與資安人員提供一個安全、高效的平台，用於通報、處理與管理資安事件。系統採用基於角色的存取控制（RBAC）模型，確保不同使用者僅能存取其職責範圍內的功能。

我們的設計目標包括權限控制系統設計、資安系統開發、全端整合開發以及透過賽博龐克風格的 UI 設計提升使用者體驗。透過這個系統，我們期望提升資安事件處理效率，建立標準化通報流程，並確保資料隔離與權限控制。

設計目標與預期效益

權限控制系統設計

實作基於角色的存取控制，學習多層級權限管理。

資安系統開發

確保系統安全性，防止未授權存取或資料洩露。

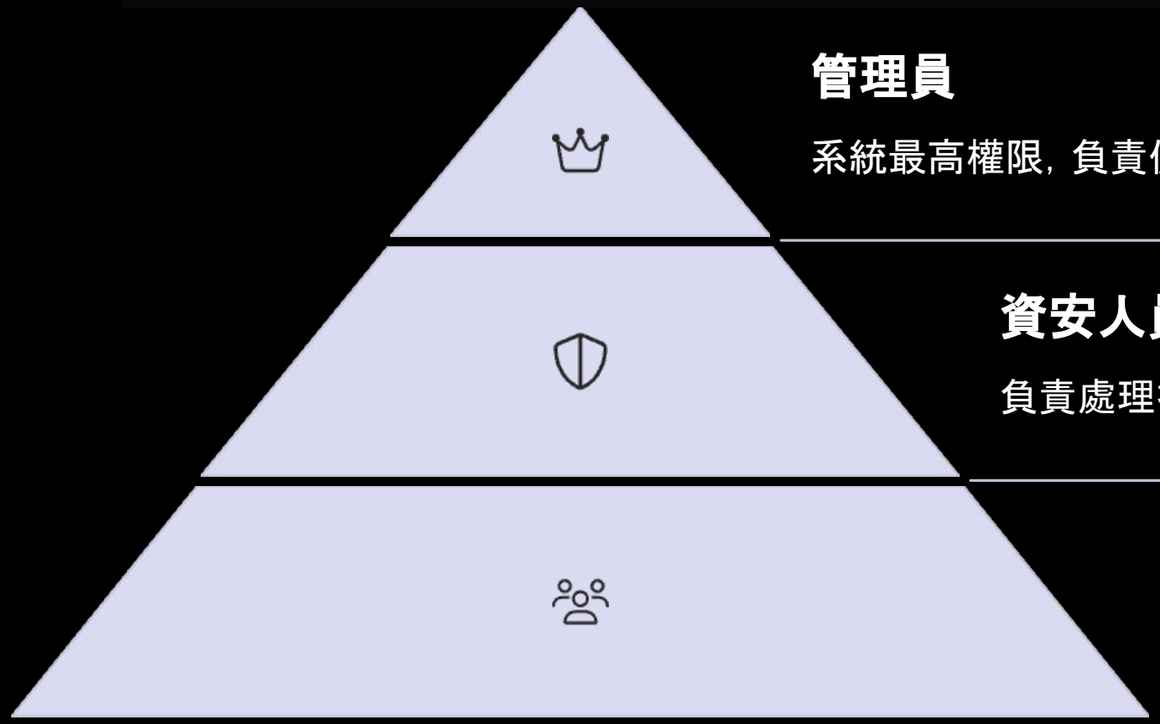
全端整合開發

實現前後端與資料庫的完整整合。

使用者體驗

透過賽博龐克風格的 UI 設計提升現代感改善使用者體驗。

基於角色的存取控制（RBAC）概述



管理員

系統最高權限，負責使用者管理和事件指派

資安人員

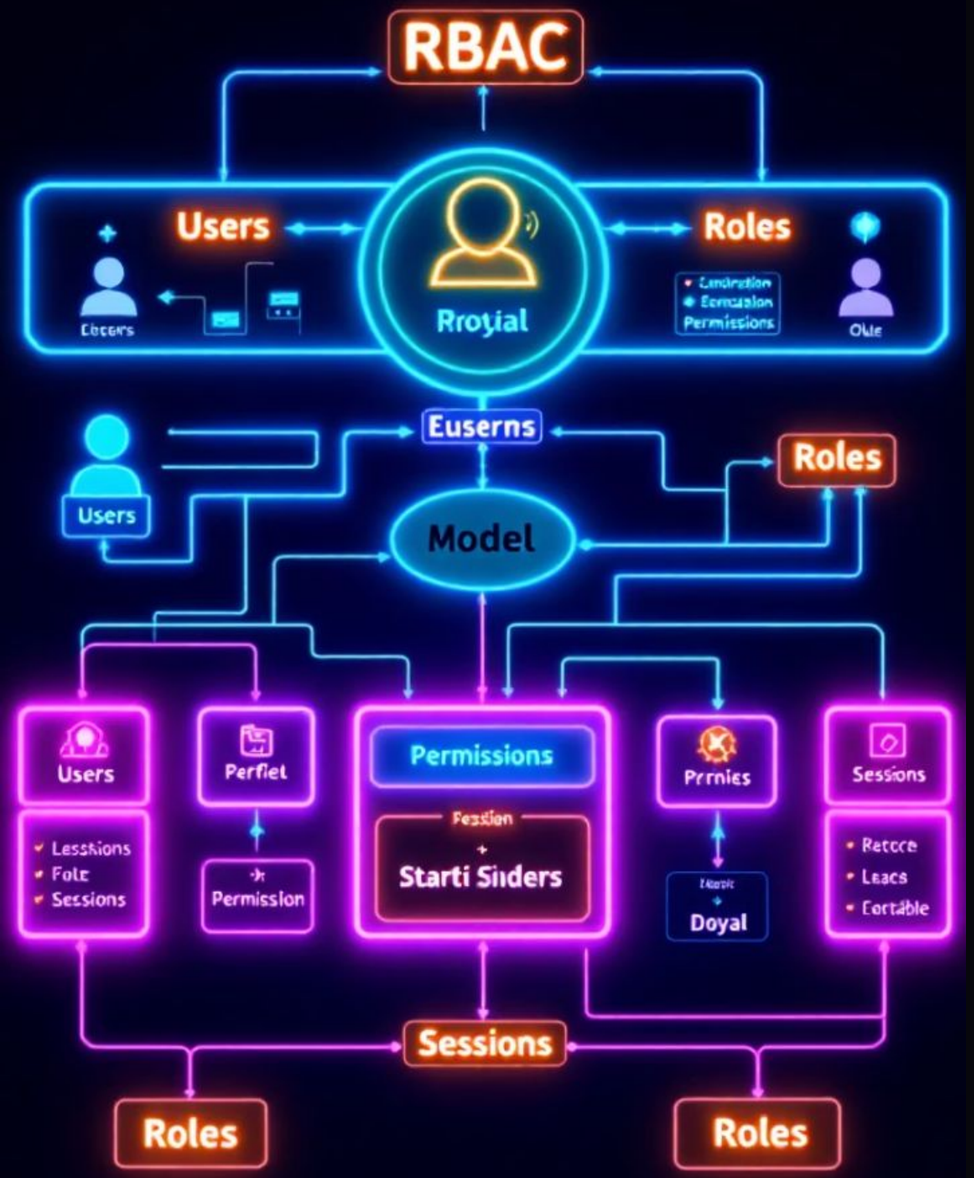
負責處理指派的資安事件，更新處理狀態

一般使用者

僅能查看和管理自己提交的事件

角色與權限控制是資安系統的核心組成部分，旨在確保只有授權的使用者能夠存取特定資源或執行特定操作。基於角色的存取控制（RBAC）是一種廣泛應用的存取控制模型，通過將權限分配給角色而非單個使用者，實現靈活且可管理的權限控制。

RBAC 的核心理念是將使用者的職責與系統功能對應，減少權限管理的複雜性，同時提升系統安全性。這種模型的優勢包括簡化管理、增強安全性、提高可擴展性，並符合資安法規的要求。



RBAC 模型的組成

使用者（Users）

系統的實際操作者，例如普通民眾、資安人員或管理員。每個使用者根據其職責被分配到特定角色。

角色（Roles）

定義一組職責或功能集合，例如本系統中的「一般使用者」、「資安人員」和「管理員」。角色是權限分配的中介。

權限（Permissions）

對系統資源或操作的存取能力，例如提交事件、指派事件或修改使用者帳號。權限與角色相關聯，而非直接與使用者關聯。

會話（Sessions）

使用者登入系統時的活動狀態，系統根據會話中的角色動態應用權限，確保安全存取控制。

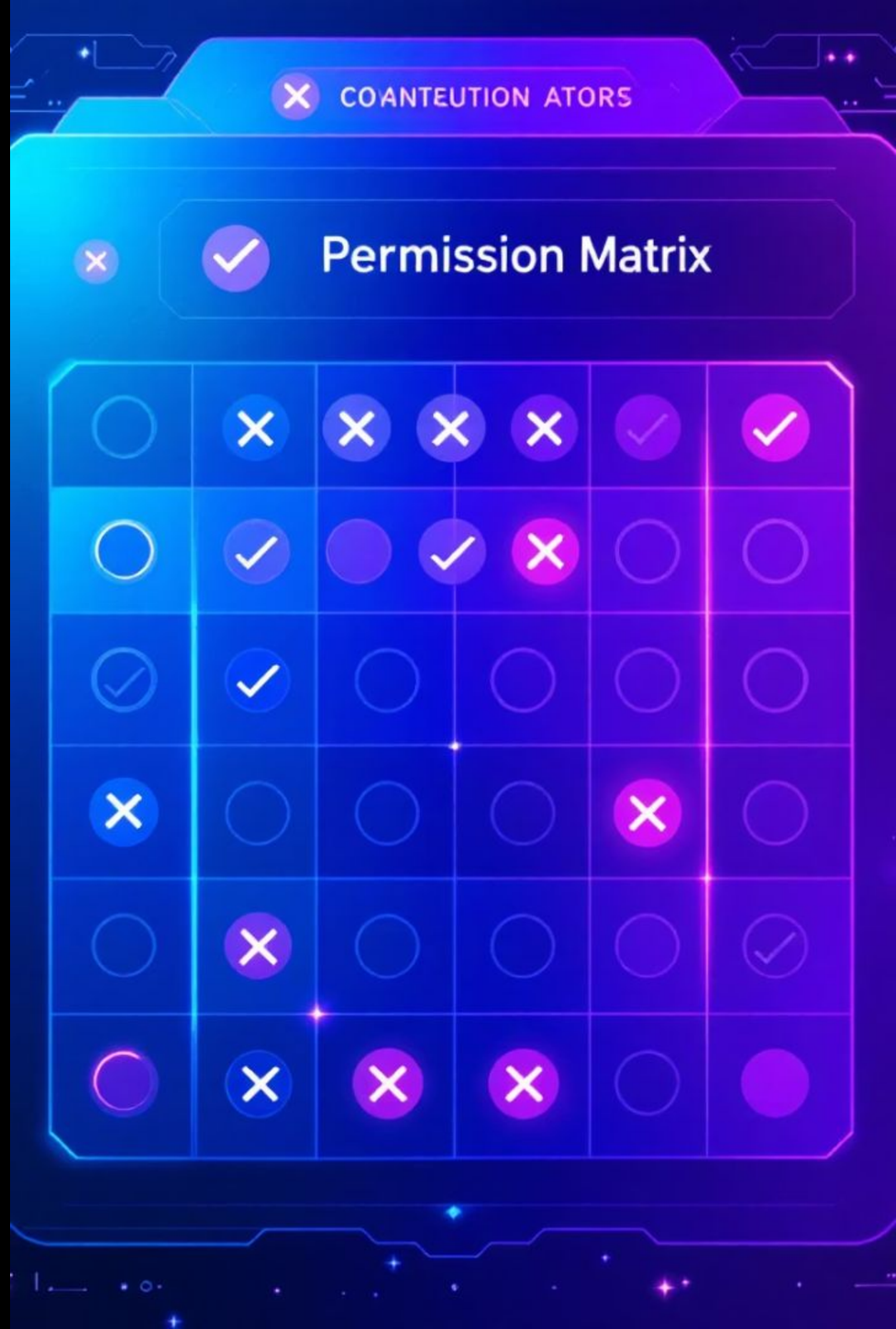
本系統採用三層 RBAC 架構，根據角色職責設計權限。一般使用者僅能提交資安事件並查看自己提交的事件記錄；資安人員負責處理指派的事件，更新事件狀態並記錄處理歷程；管理員擁有最高權限，可指派事件、管理使用者帳號、設置系統參數和匯出統計報表。

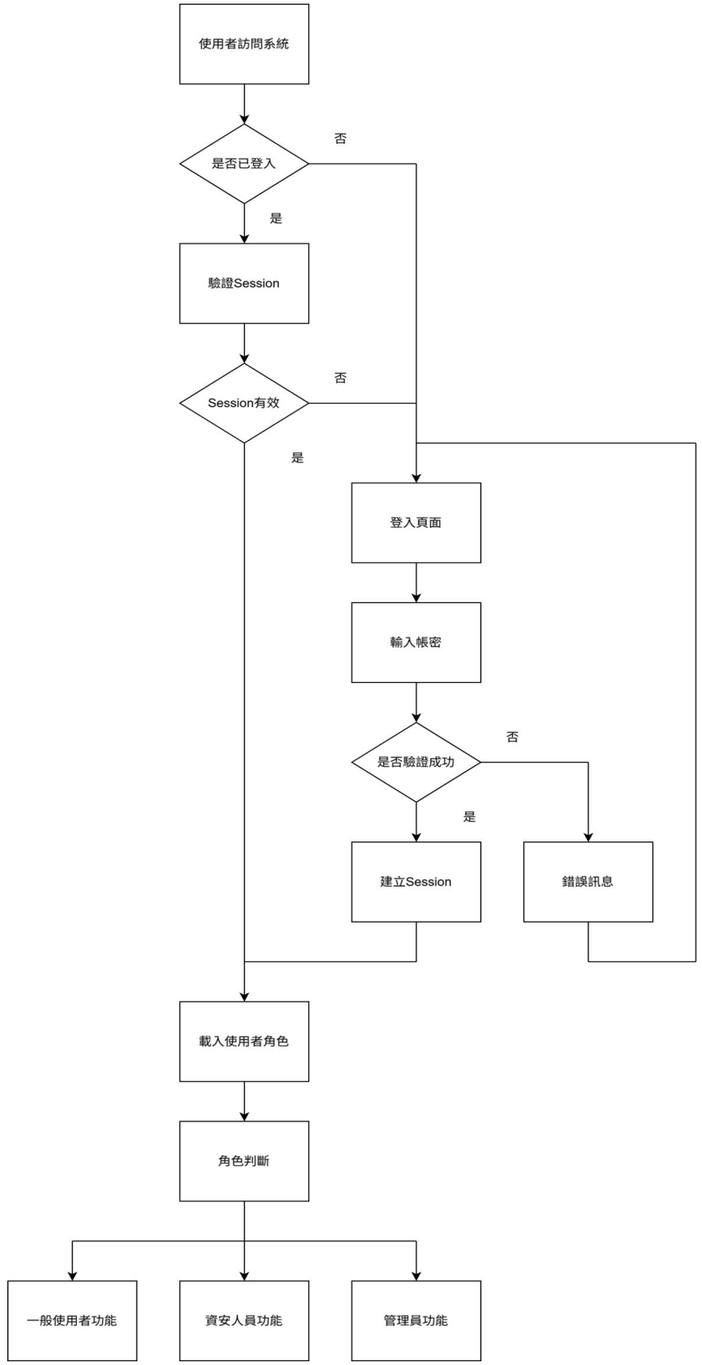
權限矩陣

功能 / 角色	普通民眾	管理人員	資安人員
提交資安通報	✅	❌	❌
查詢自己通報紀錄	✅	❌	❌
查看所有通報紀錄	❌	✅	✅（僅限指派案件）
指派通報事件	❌	✅	❌
處理事件並紀錄結果	❌	❌	✅

權限矩陣清晰展示了不同角色在系統中的操作權限。普通民眾主要負責提交資安通報並 查詢自己的通報紀錄；管理人員擁有較高權限，可 查看所有通報紀錄、指派事件、修改使用者帳號及設置系統參數；資安人員則專注於處理被指派的事件，更新狀態並記錄處理結果。

這種權限設計確保了職責分離原則，每個角色只能執行其職責範圍 內的操作，有效防止權限濫用和未授權存取。





系統流程圖

使用者註冊與登入

建立帳號並依角色分配權限

事件通報

一般使用者提交資安事件

事件指派

管理員將事件指派給資安人員

事件處理與解決

資安人員處理並記錄解決方案

技術架構

前端技術

- HTML5/CSS3：結構化網頁內容與響應式設計
- Jinja2 模板引擎：動態內容渲染與模板繼承

後端技術

- Flask 框架：輕量級 Python Web 框架
- SQLite 資料庫：嵌入式關聯式資料庫
- bcrypt 加密：密碼雜湊與驗證機制
- Session 管理：使用者狀態維護

資料庫設計

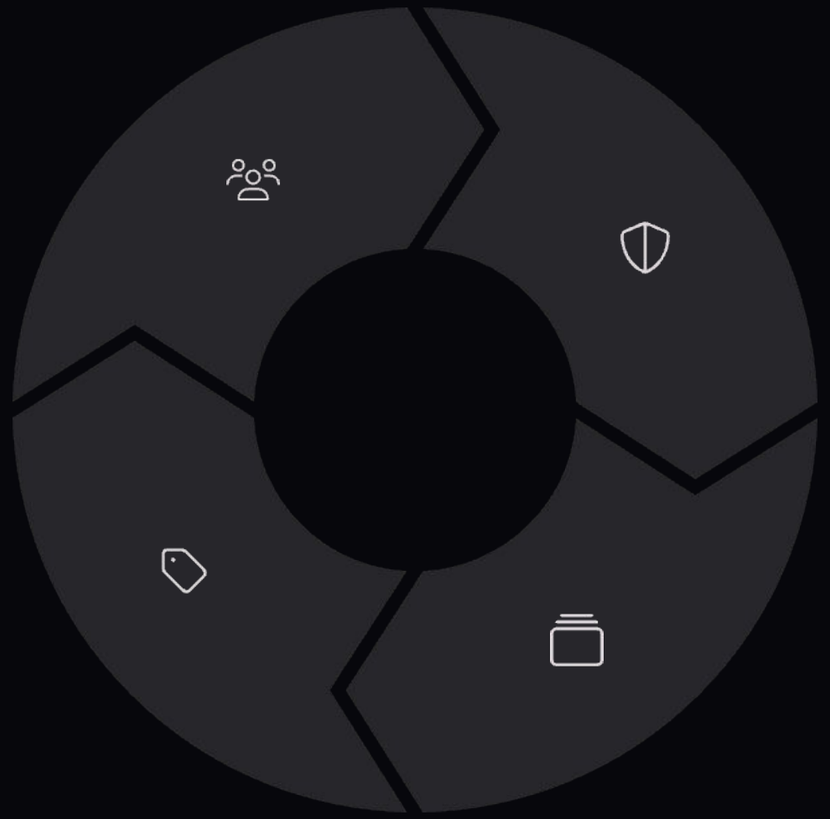
- 使用者表格（users）：儲存帳號資訊與角色
- 資安事件表格（security_incidents）：事件詳細資訊
- 事件記錄表格（incident_logs）：處理歷程追蹤
- 事件標籤表格（incident_tags）：威脅分類標籤

我們的系統採用了前後端分離的架構設計，前端使用 HTML5/CSS3 和 Jinja2 模板引擎實現動態內容渲染，後端則基於 Flask 框架開發，結合 SQLite 資料庫和 bcrypt 加密技術確保系統安全性。這種架構設計不僅提高了開發效率，也增強了系統的可維護性和擴展性。

資料庫設計

使用者表（users）

儲存使用者資訊，包括 UUID 主鍵、使用者名稱、電子郵件、加密密碼、角色和帳號狀態



資安事件表（security_incidents）

儲存事件詳細資訊，包括事件 UUID、標題、描述、通報者、嚴重程度、狀態和指派人員

事件標籤表（incident_tags）

儲存事件標籤，以複合主鍵確保唯一性，支援威脅分類

事件記錄表（incident_logs）

追蹤事件操作歷程，包含記錄 UUID、關聯事件、操作者、記錄類型和操作內容

資料庫採用 SQLite 作為嵌入式關聯式資料庫，輕量且適合本專案需求。資料庫初始化由 DatabaseManager 類別負責，自動創建表格並設置外鍵約束和索引以確保資料完整性和查詢效率。所有 SQL 操作使用參數化查詢，防止注入攻擊，確保資料安全。

資料庫設計遵循正規化原則，減少資料冗餘並提高查詢效率。表格間的關聯通過外鍵實現，確保資料一致性和完整性。

運用技巧與安全措施

權限檢查

系統使用 Flask 的 session 管理使用者登入狀態，並通過自定義 login_required 裝飾器檢查是否登入。角色權限驗證在每個路由中實現，確保使用者只能存取其權限範圍內的功能。

資料隔離

資料隔離通過 SQL 查詢的條件過濾實現。例如，為一般使用者添加 WHERE reporter_id = :user_id 條件，資安人員則檢查 assigned_to 欄位。所有資料庫操作均使用參數化查詢，防止 SQL 注入攻擊。

附件安全

未來版本將加入附件上傳功能，設計時將限制檔案類型，並使用唯一檔案名稱儲存於隔離目錄。附件存取將遵循 RBAC 模型，結合防毒掃描確保安全性。

通知系統

系統目前模擬電子郵件通知。未來計劃整合 smtplib 模組實現真實郵件通知，當事件狀態更新時，自動發送郵件給相關人員。儀表板中的通知清單將透過 AJAX 動態更新。

這些安全措施和技術實現確保了系統的穩定性、安全性和可擴展性。通過嚴格的權限控制和資料隔離，我們有效防止了未授權存取和資料洩露風險，同時提供了良好的使用者體驗和系統性能。

專案結構

主要檔案

app.py、requirements.txt、README.md、LICENSE

模板目錄

base.html、login.html、dashboard.html 等頁面模板

靜態資源

style.css 及其他前端資源

實作心得與未來展望

技術成長

掌握 Flask 框架應用與 RBAC 權限控制實現

協作經驗

學習團隊分工與版本控制管理

未來展望

整合更多安全功能與使用者體驗優化

透過 RBAC 實現權限控制，讓我們更理解資安系統中角色分工的重要性。RBAC 模型不僅簡化了權限管理，還確保了系統的安全性和可擴展性，讓不同角色能專注於其職責範圍內的工作。

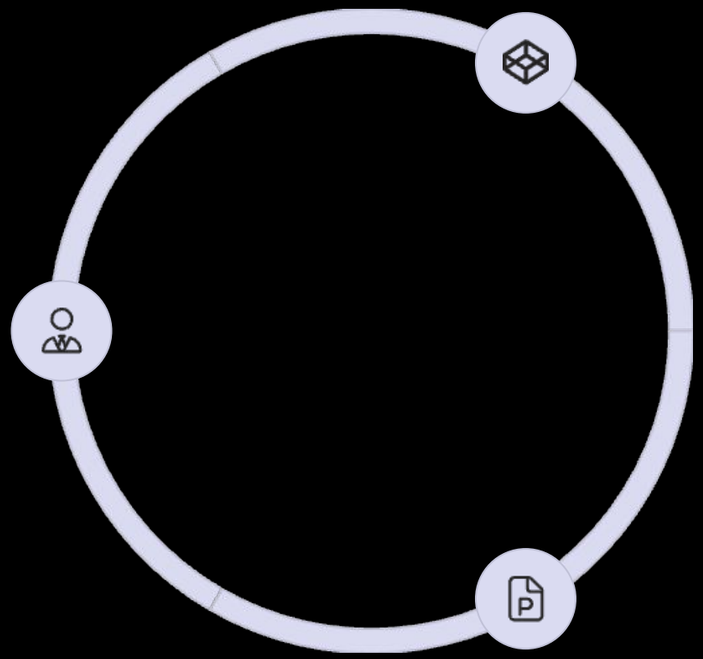
本專案讓我們掌握了 Flask 框架的應用，從路由設計到模板渲染，再到資料庫操作。我們學會了如何使用 bcrypt 加密密碼、實現 RBAC 權限控制，並透過 SQLite 管理複雜的事件資料。使用 GitHub 進行版本控制，讓我們學會了分支管理、衝突解決和程式碼審查。

未來，我們計劃擴展系統功能，包括實現真實的電子郵件通知、添加附件上傳功能、整合更多安全措施，並優化使用者界面，提供更好的使用者體驗。

工作分配

李萌家（組長）

- 專案企劃與架構設計
- AI 工具基礎框架建構
- 資料庫結構規劃



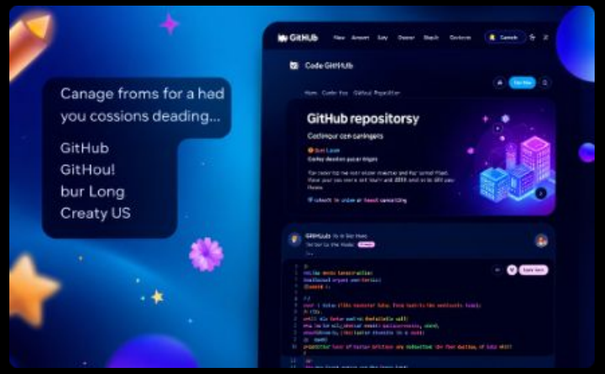
陳右承

- 後端系統核心功能開發
- 前端 UI/UX設計
- 期末報告上台負責人

陳柏勁

- 修正前後端缺失內容
- 製作PPT
- 書面報告內容修正

專案資源與參考



GitHub儲存庫

完整原始碼與說明文件

https://github.com/AmanoShizukikun/security_system

技術文件

Flask、RBAC 、SQLite 參考資料

https://flask.palletsproject
s.com/
https://csrc.nist.gov/projects/role-based-ac
cess-control
https://www.sqlite.org/doc
s.html

專案展示

系統功能與操作示範

https://youtu.be/FcvB6M-iEws?si=QUaJgNEo6l91XWcz