|  |  |
| --- | --- |
| 蜿蜒道路與樹木的圖片  資安事件通報系統  Web程式設計期末報告 | 國立勤益科技大學  資訊工程系  李萌家、陳右承、陳柏勁 |

## **資安事件通報系統 - 期末分組報告**

#### **系統概述**

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。

#### **設計目標**

* **權限控制系統設計：**實作基於角色的存取控制（RBAC），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

#### **預期效益**

* 提升資安事件處理效率，建立標準化通報流程
* 確保資料隔離與權限控制，維護系統安全性
* 透過角色設計學習資安系統的權限控制實務
* 建立完整的事件追蹤與處理記錄

**背景資料**

角色與權限相關知識

**基於角色的存取控制（RBAC）概述**  
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。

**RBAC 模型的組成**  
RBAC 模型包含以下核心元素：

* **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。
* **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」。
* **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。
* **角色分配（Role Assignment）**：將使用者與角色關聯，每個使用者可被分配一個或多個角色。
* **權限分配（Permission Assignment）**：將特定權限分配給角色，確保角色執行其職責所需的存取能力。
* **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。

**RBAC 的優勢**

* **簡化管理**：透過角色集中管理權限，減少逐一為使用者設置權限的工作量。
* **安全性**：嚴格的權限控制防止未授權存取，降低資料洩露風險。
* **可擴展性**：新增使用者或角色時，只需分配對應角色，免修改底層權限邏輯。
* **符合法規要求**：RBAC 支援資料隔離和最小權限原則，符合資安法規（如 GDPR、ISO 27001）的要求。

**本系統中的 RBAC 應用**  
本系統採用三層 RBAC 架構，根據角色職責設計權限，具體實現如下：

* **一般使用者（User）**：僅能提交資安事件（report\_incident）並查看自己提交的事件記錄（view\_my\_incidents）。程式碼中透過 WHERE reporter\_id = :user\_id 限制資料存取，確保資料隔離。
* **資安人員（Security Staff）**：負責處理指派的事件，更新事件狀態（update\_incident\_status）並記錄處理歷程（incident\_logs）。程式碼檢查 assigned\_to 欄位，確保資安人員只能操作其負責的事件。
* **管理員（Admin）**：擁有最高權限，可指派事件（assign\_incident）、管理使用者帳號（未來可擴展）、設置系統參數和匯出統計報表。程式碼中的 login\_required 裝飾器和角色檢查（user\_role == UserRole.ADMIN）確保管理員專屬功能的安全性。

**RBAC 實作細節**

* **資料庫支援**：users 表的 role 欄位儲存使用者角色，支援 RBAC 的角色分配。security\_incidents 表的 reporter\_id 和 assigned\_to 欄位實現資料隔離和權限控制。
* **程式碼實現**：Flask 路由使用 login\_required 裝飾器檢查登入狀態，並在每個功能（如 /assign\_incident、/update\_status）中驗證角色權限。例如，assign\_incident\_to\_staff 方法檢查 current\_user.role == UserRole.ADMIN，確保只有管理員能指派事件。
* **會話管理**：Flask 的 session 物件儲存 user\_id、username 和 user\_role，確保每次請求都能正確應用 RBAC 規則。
* **日誌追蹤**：所有權限相關操作（如事件創建、指派、狀態更新）記錄於 incident\_logs 表，確保操作可追溯，符合資安審計要求。

**挑戰與解決方案**

* **權限邊界**：不同角色間的權限邊界需明確定義。例如，資安人員只能更新指派給自己的事件，程式碼透過 assigned\_to != user\_id 檢查防止越權。
* **資料隔離**：一般使用者的資料存取受限於其 user\_id，透過參數化查詢（如 SELECT ... WHERE reporter\_id = ?）實現，防止 SQL 注入。
* **可擴展性**：RBAC 模型允許未來新增角色（如審計員）或權限（如事件審核），只需更新資料庫和程式碼中的角色檢查邏輯。

**參考標準**  
本系統的 RBAC 設計參考 NIST（美國國家標準與技術研究院）的 RBAC 標準，確保模型符合業界最佳實務。同時，結合 Flask 框架的 session 管理和 SQLite 的資料庫結構，實現高效且安全的權限控制。

**RBAC 模型設計**

本系統採用三層 RBAC 架構：

* **一般使用者（User）：**僅能查看和管理自己提交的事件
* **資安人員（SECURITY\_STAFF）：**負責處理指派的資安事件，更新處理狀態
* **管理員（Admin）：**擁有系統最高權限，負責使用者管理和事件指派

### **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 | ❌ | ❌ | ✅ |
| 修改使用者帳號或角色 | ❌ | ✅ | ❌ |
| 系統參數/通報類別設定 | ❌ | ✅ | ❌ |
| 匯出統計報表 | ❌ | ✅ | ❌ |
| 補充附件與說明（案件內） | ✅ | ❌ | ✅ |

#### **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
* **密碼重設表格（password\_reset\_codes）**：重設驗證碼

#### **環境介紹**

**開發環境**

* **作業系統**：Windows 11 Pro for Workstations
* **Python 版本**：3.10
* **開發工具**：Visual Studio Code
* **版本控制**：GitHub & GitHub Destop

#### **專案結構**

**資料庫結構**

security\_system/

├── app.py # 主應用程式

├── requirements.txt

├── README.md

├── LICENSE

├── templates/

│ ├── base.html # 基礎模板

│ ├── login.html # 登入頁面

│ ├── register.html # 註冊頁面

│ ├── dashboard.html # 儀表板

│ ├── report\_incident.html # 通報事件

│ ├── incident\_detail.html # 事件詳情

│ ├── forgot\_password.html # 忘記密碼

│ ├── reset\_password.html # 重設密碼

│ └── about.html # 關於系統

└── static/

└── style.css

#### **安裝方法**

**環境準備**

git clone https://github.com/AmanoShizukikun/security\_system.git

cd security\_system

**環境依賴：**安裝必要套件，定義於 requirements.txt，包括 Flask（Web 框架）、bcrypt（密碼加密）、sqlite3（資料庫操作）。執行以下命令安裝：

***pip install -r requirements.txt***

**Flask 配置：**在 app.py 中設置 Flask 應用，配置 secret\_key 用於 session 管理（程式碼中為 "security\_system\_secret"，建議正式環境使用環境變數儲存）。啟用 Flask 的 debug 模式以便開發：

***flask --debug run***

#### **資料庫設計**

**1. users（使用者表）**

CREATE TABLE users (

user\_id TEXT PRIMARY KEY, -- UUID 主鍵

username TEXT UNIQUE NOT NULL, -- 使用者名稱

email TEXT UNIQUE NOT NULL, -- 電子郵件

password\_hash TEXT NOT NULL, -- bcrypt 加密密碼

role TEXT NOT NULL, -- 使用者角色

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

is\_active BOOLEAN DEFAULT 1 -- 帳號啟用狀態

);

**2. security\_incidents（資安事件表）**

CREATE TABLE security\_incidents (

incident\_id TEXT PRIMARY KEY, -- 事件 UUID

title TEXT NOT NULL, -- 事件標題

description TEXT NOT NULL, -- 詳細描述

reporter\_id TEXT NOT NULL, -- 通報者 ID

severity TEXT NOT NULL, -- 嚴重程度

status TEXT NOT NULL, -- 處理狀態

assigned\_to TEXT, -- 指派人員

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

resolved\_at TIMESTAMP -- 解決時間

);

**3. incident\_logs（事件記錄表）**

CREATE TABLE incident\_logs (

log\_id TEXT PRIMARY KEY,

incident\_id TEXT NOT NULL, -- 關聯事件

user\_id TEXT NOT NULL, -- 操作人員

log\_type TEXT NOT NULL, -- 記錄類型

action TEXT NOT NULL, -- 執行動作

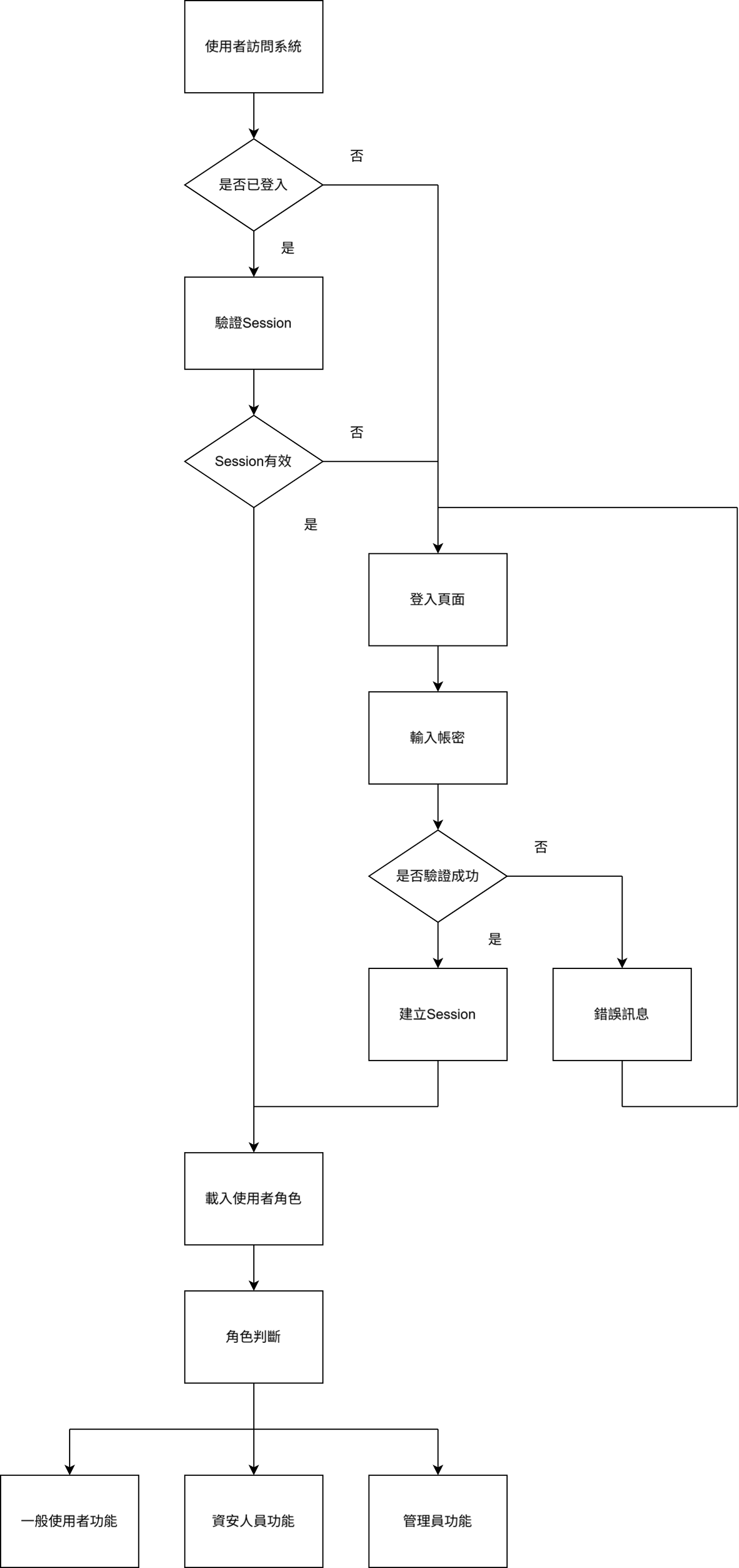
comment TEXT, -- 備註說明

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

#### **系統流程圖**

**主要系統流程**

****

**事件處理流程**

**一張含有 螢幕擷取畫面, 文字, 黑與白, 字型 的圖片

AI 產生的內容可能不正確。**

**權限驗證流程**

#### **一張含有 螢幕擷取畫面, 文字, 設計 的圖片 AI 產生的內容可能不正確。**

**系統相關說明**

1. **資料庫設計**：  
   資料庫採用 SQLite 作為嵌入式關聯式資料庫。資料庫初始化由 DatabaseManager 類別負責，自動創建以下五個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：

**users（使用者表）**：儲存使用者資訊。  
user\_id（UUID 主鍵）  
username（唯一）  
email（唯一）  
password\_hash（bcrypt 加密密碼）  
role（角色）  
created\_at（創建時間）  
is\_active（帳號狀態）  
索引 idx\_users\_user\_id

**security\_incidents（資安事件表）**：儲存事件詳細資訊。  
incident\_id（UUID 主鍵）  
title、description、reporter\_id（關聯使用者）  
severity（嚴重程度）  
status（狀態）  
assigned\_to（指派人員）  
created\_at、updated\_at 和 resolved\_at（時間）  
索引 idx\_incidents\_incident\_id

**incident\_logs（事件記錄表）**：追蹤事件操作歷程。  
log\_id（UUID 主鍵）  
incident\_id（關聯事件）  
user\_id（操作者）  
log\_type（記錄類型）  
action（操作內容）  
comment（備註）  
created\_at。（時間）

**incident\_tags（事件標籤表）**：儲存事件標籤。  
incident\_id（UUID主鍵）  
tag（標籤）  
  
以複合主鍵確保唯一性，支援威脅分類。

**password\_reset\_codes（密碼重設驗證碼表）**：儲存密碼重設資料。   
user\_id（主鍵）  
code（驗證碼）  
created\_at（時間）

初始化過程在程式啟動時自動執行，通過 DatabaseManager.init\_database 方法創建表格並設置外鍵約束。若資料庫已存在，則不會重複創建。所有 SQL 操作使用參數化查詢，防止注入攻擊。

1. **後端設定**：

**資料庫連線：**DatabaseManager 類別管理 SQLite 連線，預設資料庫檔案為 security\_incidents.db。連線採用上下文管理，確保資源正確釋放。

**權限控制：**使用自定義 login\_required 裝飾器檢查使用者登入狀態，結合 session 儲存 user\_id、username 和 user\_role，確保路由安全。

**路由設定：**定義多個路由（如 /login、/dashboard、/report\_incident），對應報告中的功能模組。路由使用 Jinja2 模板渲染前端頁面，並透過 flash 提供使用者反饋。

**安全措施：**密碼使用 bcrypt 加密，電子郵件和密碼輸入經過驗證（validate\_email 和 validate\_password），防止無效或惡意輸入。未來可整合 HTTPS 和 CSRF 保護增強安全性。

1. **前端設定**：  
   前端採用 HTML5、CSS3 和 Jinja2 模板引擎，實現響應式設計和動態內容渲染，具體設定如下：

**模板結構：**所有模板位於 templates/ 目錄，包含 base.html（基礎模板，提供共用佈局）、login.html、register.html、dashboard.html、report\_incident.html、incident\_detail.html、forgot\_password.html、reset\_password.html 和 about.html，與報告中的專案結構一致。使用 Jinja2 的模板繼承（{% extends "base.html" %}）確保一致的頁面佈局。

**樣式設計：**CSS 檔案位於 static/style.css，採用賽博龐克風格設計，包含霓虹色調、動態特效和響應式佈局，適配不同設備。CSS 使用 Flexbox 和 Grid 確保頁面元素排列靈活。

**動態渲染：**Jinja2 模板動態渲染事件列表（如 dashboard.html 中的事件表格）、通知訊息和表單選項（如嚴重程度和狀態下拉選單）。搜尋和篩選功能透過表單（method="POST"）提交，結合 AJAX（未來可擴展）實現即時更新。

**使用者體驗：**儀表板（dashboard.html）支援關鍵字搜尋、嚴重程度和狀態篩選，透過 search\_and\_filter\_incidents 方法動態生成事件列表。事件詳情頁（incident\_detail.html）顯示事件資訊和操作記錄，管理員可指派事件，資安人員可更新狀態。

**未來改進：**計劃整合 JavaScript 框架（如 Vue.js 或 jQuery）增強前端互動性，例如即時通知或動態圖表展示事件統計。

#### **運用技巧**

**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，例如 /assign\_incident 僅允許 ADMIN 角色訪問。程式碼中的 search\_and\_filter\_incidents 方法根據使用者角色動態過濾事件資料，確保一般使用者只能查看自己提交的事件，資安人員僅能處理指派的事件。這種基於 RBAC 的權限控制有效防止未授權存取。

**資料隔離：**  
資料隔離通過 SQL 查詢的條件過濾實現。例如，get\_incidents\_by\_user 方法為一般使用者添加 WHERE reporter\_id = :user\_id 條件，資安人員則檢查 assigned\_to 欄位。敏感資料（如密碼）使用 bcrypt 加密，儲存在 users 表的 password\_hash 欄位中。所有資料庫操作均使用參數化查詢，防止 SQL 注入攻擊，確保資料安全。

**附件安全：**  
目前系統尚未實作附件上傳功能，但計劃在未來版本中加入。設計時將限制檔案類型（僅允許 .txt、.pdf、.png 等），並使用唯一檔案名稱（UUID + 時間戳）儲存於隔離目錄。附件存取將遵循 RBAC 模型，僅允許事件相關人員（通報者、指派資安人員、管理員）下載，並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（如密碼重設碼發送）。未來計劃整合 smtplib 模組實現真實郵件通知，當事件狀態更新（如指派或解決）時，自動發送郵件給相關人員。通知內容將使用 Jinja2 模板生成，包含事件 ID、標題和操作連結。儀表板中的通知清單將透過 AJAX 動態更新，提升即時性。

### **實作心得**

* **想法**：

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。

* **心得**：
* 技術成長：

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。搜尋和篩選功能的實作（search\_and\_filter\_incidents）讓我們理解了動態查詢的設計原則。賽博龐克風格的 UI 設計增強了系統的現代感，也讓我們學會如何在前端和後端之間實現高效整合。

* 協作經驗：

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。團隊分工明確，例如後端開發、資料庫設計和前端 UI 各有專人負責。定期討論和反饋幫助我們快速迭代，解決了許多初期設計上的問題，提升了專案品質。

* **困難**：
* RBAC 權限邏輯：

初期在實現 RBAC 時，對於如何在 Flask 中動態檢查角色權限感到困惑，尤其是在多角色間的邊界處理（如資安人員只能更新指派事件）。最終透過 login\_required 裝飾器和角色檢查邏輯解決，但需要多次測試確保無漏洞。

* 資料庫查詢效能：

在測試大量事件資料時，search\_and\_filter\_incidents 的聯表查詢（涉及 security\_incidents 和 incident\_tags）效率較低。我們通過添加索引（idx\_incidents\_incident\_id）和優化查詢條件解決了問題，但這讓我們意識到資料庫設計的重要性。

* 前端動態更新：

實現儀表板的搜尋和篩選功能時，Jinja2 模板與 JavaScript 的整合遇到挑戰，例如事件列表的即時更新。我們使用 AJAX 技術動態載入資料，但需要反覆調試以確保跨瀏覽器相容性。

#### **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
* 資料庫結構規劃與後期 SQL 語法新功能添加
* 撰寫書面報告
* 安排小組工作

**陳右承：**

* 後端系統核心功能開發
* 前端 UI/UX設計
* 基礎資料庫建置與資料操作
* 基礎框架測試與除錯優化
* 使用者體驗測試與改善
* 技術文件撰寫與專案管理
* 書面報告內容修正
* 期末報告上台負責人
* 團隊協調與反饋需修改的缺失內容

**陳柏勁：**

* 修正前端跟後端缺失內容
* 製作PPT
* 書面報告內容修正

#### **參考文獻**

* Flask 文檔：<https://flask.palletsprojects.com/>
* RBAC 介紹：<https://csrc.nist.gov/projects/role-based-access-control>
* SQLite 文檔 - <https://www.sqlite.org/docs.html>
* NIST的 RBAC 標準<https://csrc.nist.gov/projects/role-based-access-control>

#### **專案連結**

* **GitHub 儲存庫：**[**https://github.com/AmanoShizukikun/security\_system**](https://github.com/AmanoShizukikun/security_system)
* **專案展示影片：**[**https://youtu.be/FcvB6M-iEws?si=QUaJgNEo6I91XWcz**](https://youtu.be/FcvB6M-iEws?si=QUaJgNEo6I91XWcz)