## 會員系統實作報告

### 第一章 緒論

#### 研究背景與動機

本報告詳述 C# 與 Windows Forms 框架建構會員管理系統之過程，展示程式設計概念應用。鑑於當代應用程式對高效、安全、可擴展用戶管理方案之需求日增，此專案旨在克服傳統挑戰，示範結構化程式設計典範之最佳實踐。

#### 研究目的

本研究旨在建構功能完備之會員管理系統，涵蓋用戶註冊、登入驗證、個人資料維護及管理員操作。系統將實現安全密碼儲存、精確身份驗證、彈性角色存取控制及響應式使用者介面。此外，專案亦透過 Windows Forms 應用程式，實踐物件導向程式設計（OOP）核心原則，為學習者提供實用範例。

#### 問題陳述

建構安全高效會員管理系統需審慎考量資料持久性、用戶身份驗證、基於角色之存取控制及使用者介面響應性。挑戰包括密碼安全管理（防範破解與字典攻擊）、系統漏洞防範（如 SQL 注入、跨站腳本攻擊），並確保計算密集型操作期間介面流暢，避免用戶不滿。

#### 預期貢獻

本專案作為綜合教育資源，闡明多元程式設計概念於應用系統中之有效整合。其為學習者提供實踐物件導向原則之平台，並為建構類似系統提供清晰藍圖。透過運用設計模式與 OOP 原則，本專案強調系統模組化、可維護性與可擴展性，為未來軟體開發奠定基礎。

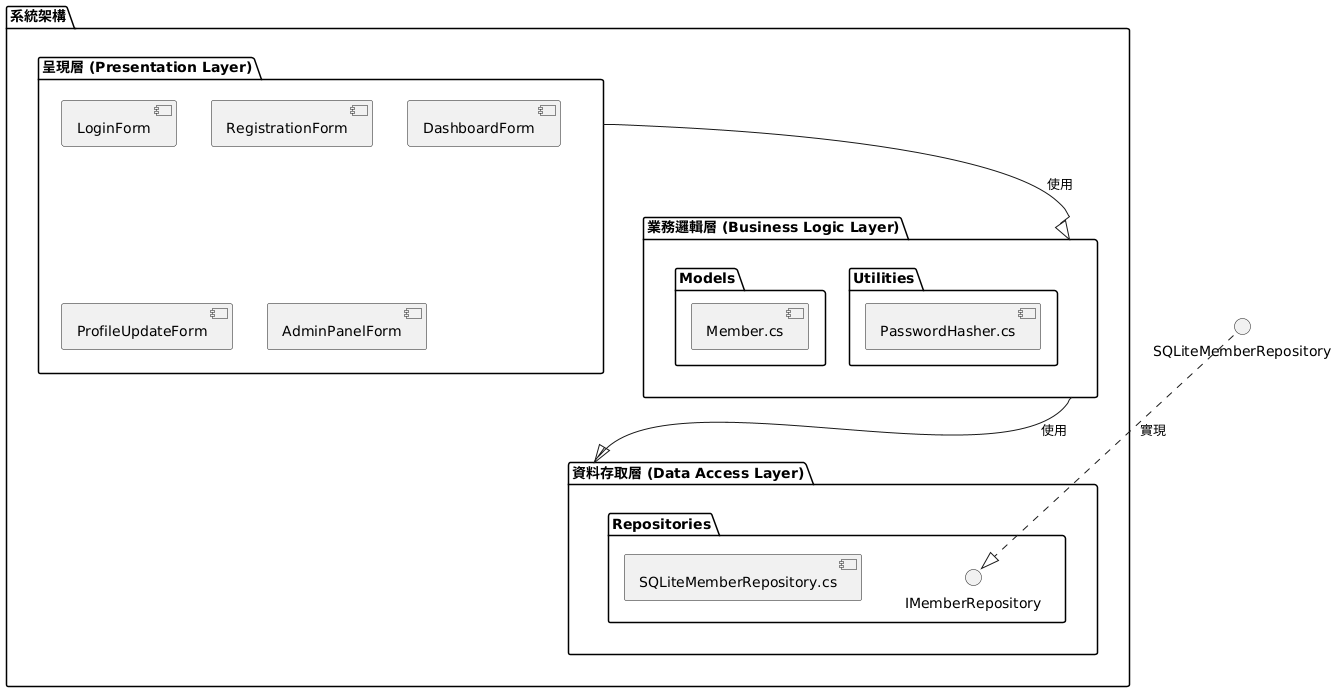
### 第二章 系統設計

#### 系統架構圖

本系統採分層架構，區分呈現層（表單）、業務邏輯層（模型、實用工具）及資料存取層（儲存庫）。此設計提升模組化、可維護性與可擴展性，各層透過明確介面互動，降低耦合度，實現獨立開發與維護。

* **呈現層：** 由 LoginForm、RegistrationForm、DashboardForm、ProfileUpdateForm 和 AdminPanelForm 五個 Form.cs 檔案構成，負責使用者互動與資訊呈現。
* **業務邏輯層：** 包含 Models 資料夾（Member.cs 定義會員實體）及 Utilities 資料夾（PasswordHasher.cs 處理密碼安全）。
* **資料存取層：** 由 Repositories 資料夾管理，SQLiteMemberRepository.cs 實作 IMemberRepository 介面，處理 SQLite 資料庫操作，並將細節與業務邏輯隔離。

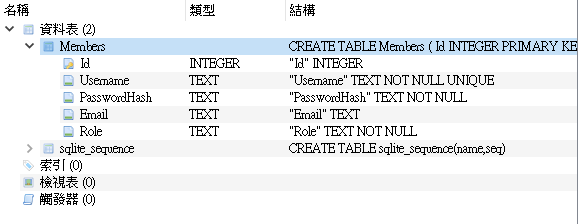
#### 功能模組說明

* **登入模組（LoginForm.cs）：** 處理用戶身份驗證，驗證憑證後依角色導向儀表板。整合多執行緒提升響應性，運用委派實現模組通訊。
* **註冊模組（RegistrationForm.cs）：** 允許新用戶註冊，具備嚴格輸入驗證，並與 PasswordHasher 整合確保密碼安全。
* **儀表板模組（DashboardForm.cs）：** 登入後主介面，顯示用戶資訊，依角色提供個人資料更新及管理員面板存取。
* **個人資料更新模組（ProfileUpdateForm.cs）：** 允許會員更新電子郵件及密碼，包含資料驗證，透過委派通知儀表板變更。
* **管理員面板模組（AdminPanelForm.cs）：** 僅限管理員存取，功能涵蓋會員資料檢視、新增與刪除，於 DataGridView 中呈現。

#### 資料庫設計

本系統採用輕量級 SQLite 資料庫，包含一個名為 Members 之資料表，簡化部署與管理。

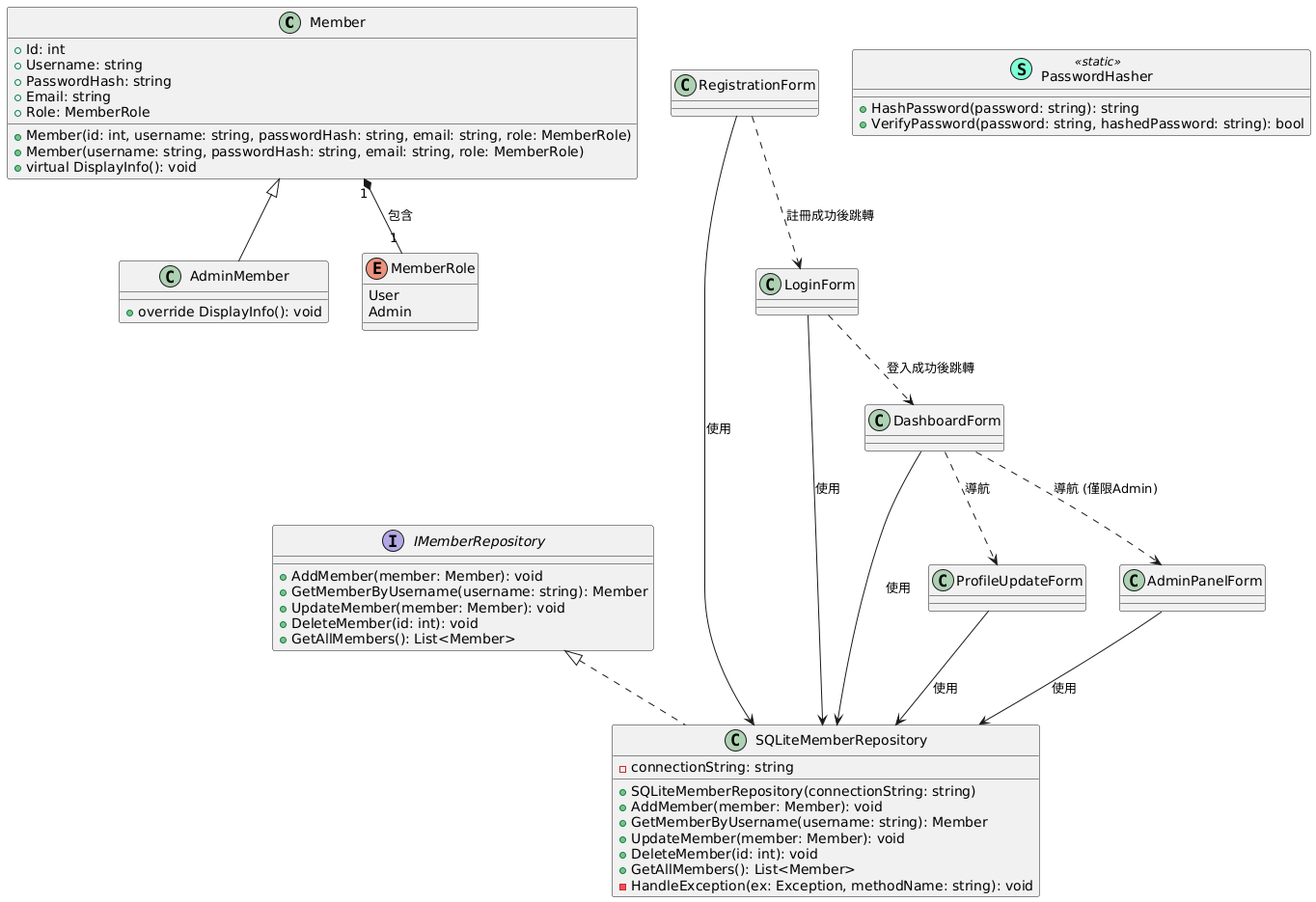
* **資料表名稱：** Members
* **欄位：**
  + Id (INTEGER PRIMARY KEY AUTOINCREMENT)：會員唯一識別符。
  + Username (TEXT UNIQUE NOT NULL)：會員用戶名，唯一且不可為空。
  + PasswordHash (TEXT NOT NULL)：儲存哈希處理之密碼，不可為空。
  + Email (TEXT)：會員電子郵件地址，允許為空。
  + Role (TEXT NOT NULL)：會員角色（「User」或「Admin」），不可為空。



#### 類別圖、循序圖等 UML 圖表

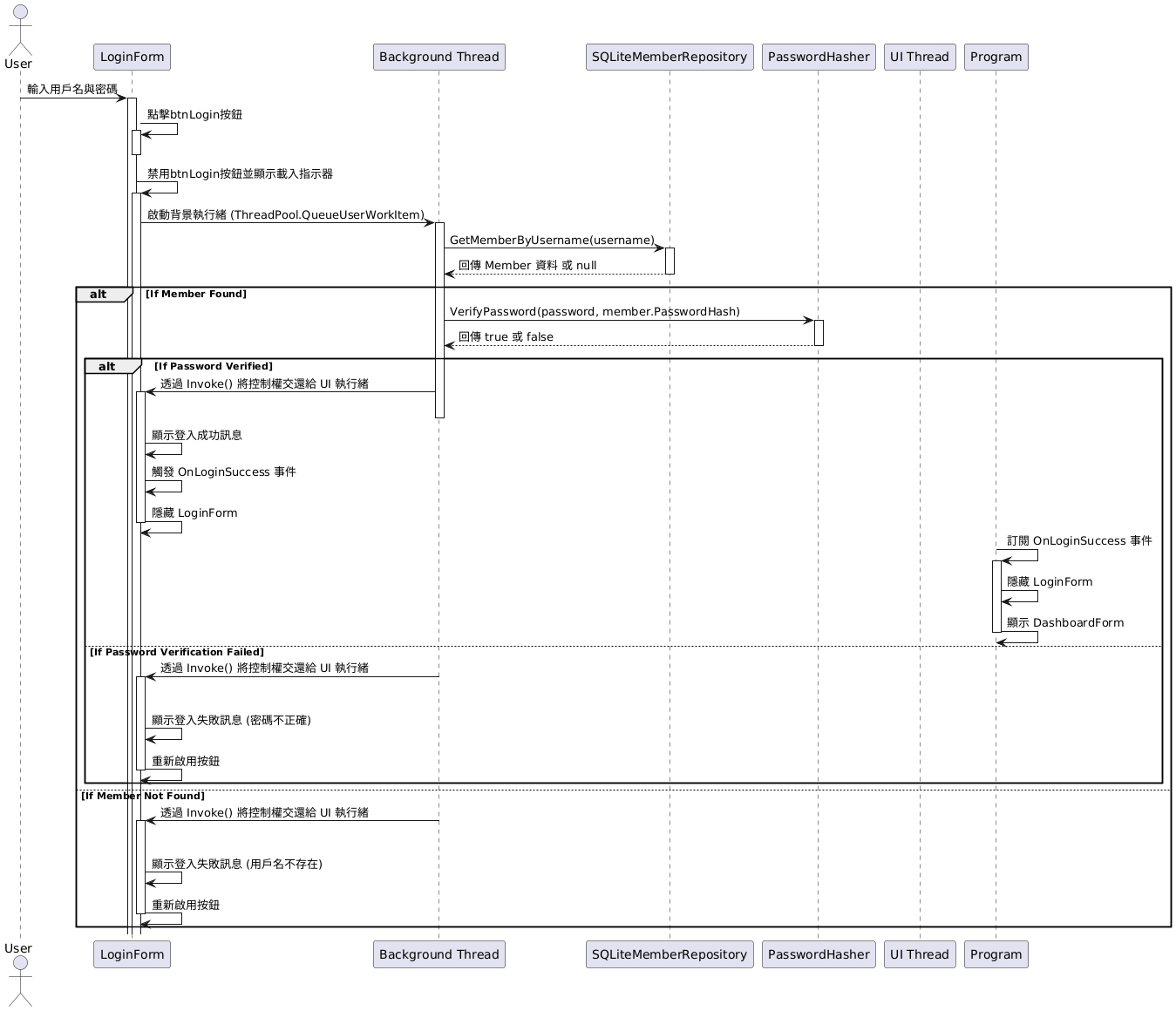
**類別圖（概念性描述）：**

* **Member 類別：** 會員實體基礎抽象，具備核心屬性、多載建構函數及虛擬 DisplayInfo() 方法。
* **AdminMember 類別：** 繼承自 Member，覆寫 DisplayInfo() 提供管理員專屬資訊，為多型應用。
* MemberRole **列舉：** 定義會員角色（User、Admin），提升程式碼清晰度。
* IMemberRepository **介面：** 定義資料庫操作標準契約，確保資料存取層抽象化。
* **SQLiteMemberRepository 類別：** 實作 IMemberRepository，處理 SQLite 資料庫互動並整合例外處理。
* PasswordHasher **類別：** 靜態工具類別，提供 SHA256 哈希及驗證方法，確保密碼安全。
* **表單類別：** 各表單與 SQLiteMemberRepository 互動，並透過委派與其他表單通訊，實現模組鬆散耦合。



**循序圖（登入過程 - 高層次描述）：**

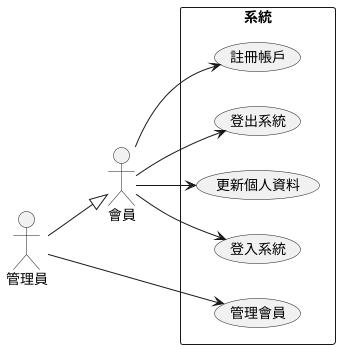
1. **使用者：** 於 LoginForm 鍵入用戶名與密碼。
2. LoginForm**：** 點擊登入按鈕後禁用按鈕，顯示載入指示器，並啟動背景執行緒處理登入驗證。
3. **背景執行緒：** 呼叫 \_memberRepository.GetMemberByUsername() 檢索會員資料，並透過 PasswordHasher.VerifyPassword() 驗證密碼。
4. **背景執行緒：** 若驗證成功，透過 LoginForm.Invoke 更新 UI 執行緒。
5. LoginForm**（UI 執行緒）：** 顯示成功訊息，觸發 OnLoginSuccess 事件並隱藏自身。
6. **Program.cs：** 訂閱 OnLoginSuccess，隱藏 LoginForm 並顯示 DashboardForm。
7. LoginForm**（UI 執行緒）：** 若登入失敗，顯示失敗訊息並重新啟用按鈕。



**用例圖（概念性描述）：**

用例圖將描繪系統與其參與者（Actors）之間的功能互動。主要參與者包括「會員」和「管理員」。

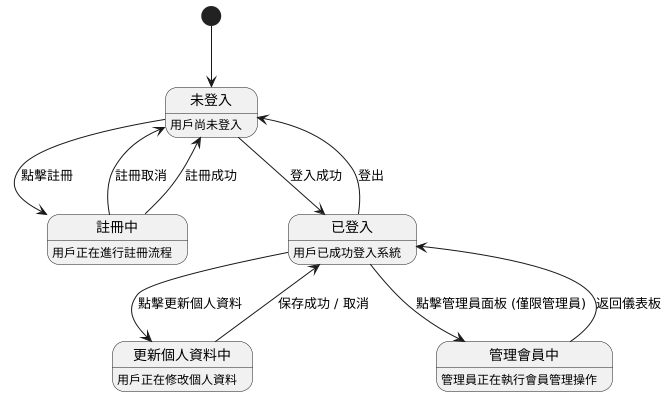
* **會員用例：**
* 註冊帳戶：新用戶建立帳戶。
* 登入系統：已註冊用戶進入系統。
* 更新個人資料：會員修改其電子郵件或密碼。
* 登出系統：會員安全退出應用程式。
* **管理員用例：**
* 管理會員：管理員查看、新增、編輯（透過更新個人資料功能）或刪除會員帳戶。
* （繼承自會員）登入系統：管理員登入系統。
* （繼承自會員）更新個人資料：管理員修改其個人資料。
* （繼承自會員）登出系統：管理員安全退出應用程式。



**狀態圖（概念性描述）：**

狀態圖將展示系統中關鍵物件（例如「會員會話」）在其生命週期中可能經歷的狀態及其之間的轉換。

* **會員會話狀態：**
* **未登入：** 初始狀態，用戶尚未登入。
* 轉換至「註冊中」：觸發條件為「點擊註冊」。
* 轉換至「已登入」：觸發條件為「登入成功」。
* **註冊中：** 用戶正在進行註冊流程。
* 轉換至「未登入」：觸發條件為「註冊取消」。
* 轉換至「已登入」：觸發條件為「註冊成功」。
* **已登入：** 用戶已成功登入系統。
* 轉換至「更新個人資料中」：觸發條件為「點擊更新個人資料」。
* 轉換至「管理會員中」（僅限管理員）：觸發條件為「點擊管理員面板」。
* 轉換至「未登入」：觸發條件為「登出」。
* **更新個人資料中：** 用戶正在修改個人資料。
* 轉換至「已登入」：觸發條件為「保存成功」或「取消」。
* **管理會員中（僅限管理員）：** 管理員正在執行會員管理操作。
* 轉換至「已登入」：觸發條件為「返回儀表板」。



**活動圖（概念性描述）：**

活動圖將描繪特定業務流程（例如「用戶註冊流程」）中的步驟序列和決策點。

* **用戶註冊流程：**

1. 開始。
2. 顯示註冊表單。
3. 輸入用戶名、密碼、確認密碼、電子郵件。
4. [決策點] 驗證輸入：

* [否] 顯示錯誤訊息，返回「輸入用戶名、密碼...」。
* [是] 檢查用戶名是否已存在。

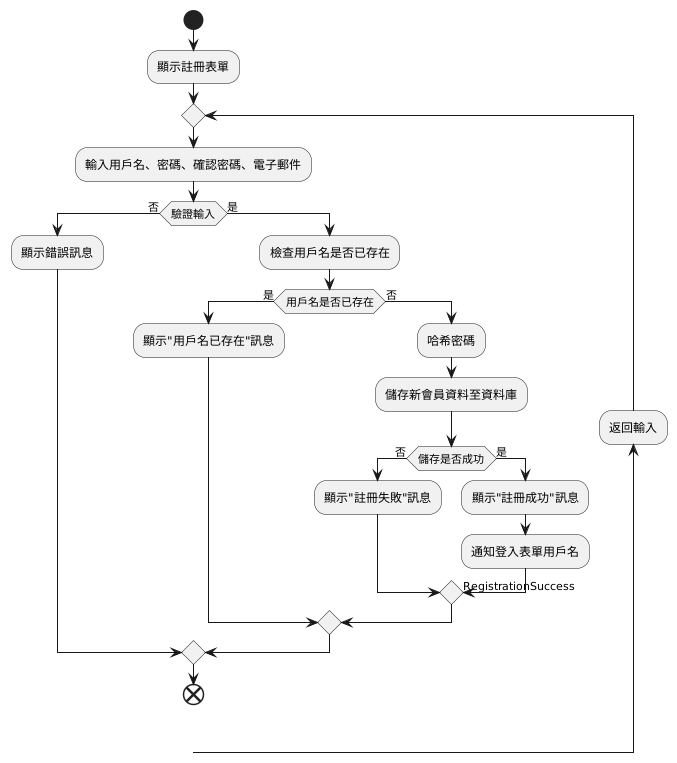
1. [決策點] 用戶名是否已存在：

* [是] 顯示「用戶名已存在」訊息，返回「輸入用戶名、密碼...」。
* [否] 哈希密碼。

1. 儲存新會員資料至資料庫。
2. [決策點] 儲存是否成功：

* [否] 顯示「註冊失敗」訊息，返回「輸入用戶名、密碼...」。
* [是] 顯示「註冊成功」訊息。

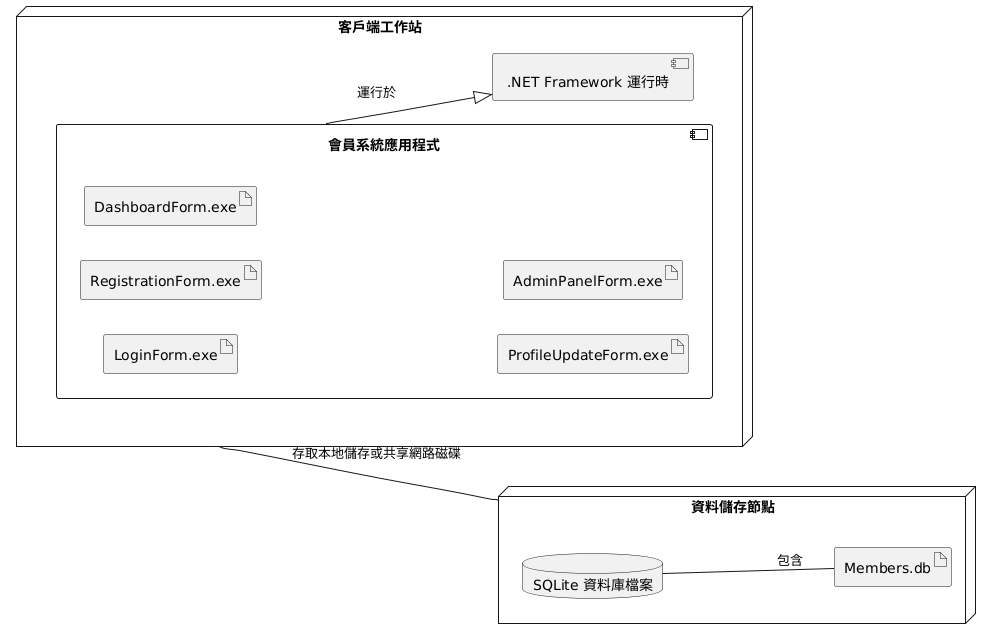
1. 通知登入表單用戶名。
2. 結束。



**部署圖（概念性描述）：**

部署圖將展示系統的物理部署架構，包括硬體節點和部署在這些節點上的軟體組件。

* **部署環境：**
* **客戶端工作站（Client Workstation）：**
* 節點：用戶個人電腦。
* 部署組件：
* 會員系統應用程式（Windows Forms App）：包含所有呈現層表單（LoginForm.exe、DashboardForm.exe 等）。
* .NET Framework 運行時：提供應用程式運行環境。
* **資料儲存節點（Data Storage Node）：**
* 節點：客戶端工作站的本地儲存空間（或共享網路驅動器）。
* 部署組件：
* SQLite 資料庫檔案（Members.db）：儲存所有會員數據。



### 第三章 系統實現

本章節詳述系統實現細節，包含開發環境配置、核心功能程式碼解析及關鍵技術實作。

#### 開發環境

* **Visual Studio：** 採用 Visual Studio 作為 IDE，提供開發工具與調試功能。
* **.NET Framework：** 系統基於 .NET Framework 平台開發，使用 8.0 。
* **NuGet 套件：** System.Data.SQLite 用於資料庫連接，其輕量級與嵌入式特性適合中小型應用。

#### 核心功能代碼解析

* **Member.cs（類別與物件、過載、多型）：** 定義 Member 類別為核心實體，具備關鍵屬性。擁有多個建構函數（過載）提供彈性初始化。virtual DisplayInfo() 方法實現多型，允許子類覆寫。AdminMember 繼承 Member，覆寫 DisplayInfo() 彰顯多型應用。
* **IMemberRepository.cs（介面）：** 定義資料存取標準契約，確保程式碼一致性並啟用依賴反轉原則，提升可測試性與可擴展性。
* **SQLiteMemberRepository.cs（類別、實作介面、例外處理、多型）：** 實作 IMemberRepository，提供 SQLite 互動方法，包含 InitializeDatabase() 自動建立資料庫。廣泛運用 try-catch 實現強健例外處理。根據 Role 動態實例化 Member 或 AdminMember，體現多型。
* **PasswordHasher.cs（實用工具）：** 靜態類別，提供 SHA256 哈希及驗證方法，將密碼哈希邏輯封裝，確保安全性。

#### 關鍵技術實現說明

* **委派與事件：** 委派定義事件簽名，事件基於委派實現，促進表單間鬆散耦合通訊，提升模組化與可維護性。
* **執行緒：** ThreadPool.QueueUserWorkItem 於 btnLogin\_Click 中將登入驗證卸載至背景執行緒，避免 UI 凍結。this.Invoke 確保 UI 更新於主 UI 執行緒執行，預防跨執行緒問題。

#### 界面設計與使用者操作流程

1. **應用程式啟動：** Program.cs 啟動 LoginForm。
2. **登入/註冊：** 用戶於 LoginForm 登入或點擊「註冊」開啟 RegistrationForm。成功註冊後，RegistrationForm 關閉並自動填入用戶名至 LoginForm。
3. **儀表板存取：** 成功登入後，LoginForm 隱藏，顯示 DashboardForm。DashboardForm 動態呈現用戶資訊，並依管理員權限顯示「管理員面板」按鈕。
4. **個人資料管理：** 從 DashboardForm 開啟 ProfileUpdateForm，允許更新電子郵件及密碼，透過委派通知 DashboardForm。
5. **管理員功能：** 管理員可從 DashboardForm 存取 AdminPanelForm，檢視、新增、刪除會員。
6. **登出：** 點擊 DashboardForm 上「登出」按鈕，關閉表單並重新開啟 LoginForm。

### 第四章 測試與驗證

本章節闡述系統測試與驗證策略，涵蓋單元測試、系統整合測試及效能評估。

#### 單元測試案例

* Member **類別：** 測試建構函數過載及 DisplayInfo() 方法之多型實作。
* PasswordHasher **類別：** 測試哈希及驗證方法之正確性。
* **SQLiteMemberRepository 類別：** 測試資料庫初始化、增刪改查操作，包括唯一約束處理及錯誤捕捉。

#### 系統整合測試

* **登入流程：** 驗證正確憑證導向、錯誤提示及並發處理。
* **註冊流程：** 測試成功註冊、用戶名重複、密碼不匹配及強度不足情境。
* **個人資料更新流程：** 測試更新電子郵件、密碼更改及確認，驗證變更反映。
* **管理員面板流程：** 測試會員資料載入、刪除、新增功能，驗證資料庫變更及權限限制。
* **表單導航：** 確保各表單間轉換流暢無礙。
* **執行緒安全：** 驗證背景操作期間 UI 響應性及跨執行緒更新之正確性。

#### 效能評估

* **登入/註冊響應時間：** 測量用戶提交請求至系統響應之時間，尤其於模擬網路延遲下。
* **資料庫操作速度：** 衡量核心資料庫操作（增刪改查）之執行速度，考量會員數量增加情境。
* **使用者介面響應性：** 監控大量操作或背景任務期間 UI 響應性，確認多執行緒有效防止介面凍結。

### 第五章 結論與未來工作

#### 研究成果總結

本專案成功實作基於 C# Windows Forms 與 SQLite 之會員管理系統。其有效整合 OOP 核心概念（類別、繼承、介面、過載、多型），並應用強健例外處理。委派促進表單間鬆散耦合通訊，多執行緒確保 UI 響應性，顯著提升用戶體驗。

#### 專題限制

* **安全性：** 密碼哈希可進一步強化，如引入加鹽及更強金鑰派生函數。
* **UI/UX：** 介面設計相對基礎，可應用進階原則提升視覺與可用性。
* **可擴展性：** SQLite 適用中小型應用，大規模部署需更強大資料庫，但介面設計允許替換。
* **錯誤報告：** 雖有例外處理，可實作更詳盡日誌記錄及用戶友善錯誤訊息。
* **管理員功能：** 目前提供基本 CRUD，未來可增補用戶搜尋、角色管理、審計日誌等。

#### 未來改進方向

* **安全性強化：** 實作密碼加鹽、採用 bcrypt/scrypt 等強哈希演算法，並導入雙因素身份驗證。
* **進階使用者介面框架：** 探索 WPF、Avalonia UI 或 MAUI，實現更豐富介面及跨平台兼容性。
* **非同步操作：** 擴展 async/await 應用於所有潛在阻塞 I/O 操作，確保 UI 持續響應。
* **日誌記錄：** 整合 NLog 或 Serilog 等專業日誌框架，提升錯誤追蹤與應用程式監控。
* **依賴注入：** 實作 DI 容器（如 Autofac），提升程式碼可測試性、模組化及可維護性。
* **單元測試框架：** 整合 NUnit 或 xUnit 等專用框架，實現結構化自動測試，確保品質。
* **基於角色之存取控制：** 實作更細緻權限管理機制，定義多種角色及其操作權限。
* **使用者體驗增強：** 增補「忘記密碼」、「記住我」及即時輸入驗證等功能。