SCAF

設計文件

專案名稱	SCAF 開發輔助工具
撰寫日期	2022 / 11 / 21
發展者	簡蔚驊 鄧暐宣 余威霆 林佳何 唐劭賢

1. 系統模型與架構 (System Model/System Architecture)

描述系統之架構。可用 C4 model 的 Container diagram + Component diagram、UML 之 Component diagram 或單純的 Block diagram (方塊圖) 表達此系統包含的模組，以及與模組之間的關係。
架構圖應與 SRS 一致，但可特別強調介面 (Interface) 與實際佈署之環境。
放 C4 所有的圖片
鄧

2. 介面需求與設計 (Interface Requirement and Design)

2.1 會員子系統

2.1.1 外部介面

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-01	登入帳號	Account Management Module	User
連結方式	輸入資料		輸出資料
	email, password		登入成功後跳轉頁面
對應介面之要求			
接收 email 及 password，交由 firebase 驗證。驗證成功後，回傳 token 到前端。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-02	註冊帳號	Account Management Module	Any
連結方式	輸入資料		輸出資料
	email, password		註冊成功後跳轉頁面
對應介面之要求			
接收 email 及 password，交由 firebase 註冊。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-03	忘記密碼	Account Management Module	User
連結方式	輸入資料		輸出資料
	email		驗證完後跳回登入頁
對應介面之要求			
接收 email，交由 firebase 驗證，並由 firebase 寄送驗證信到使用者上。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-04	修改個人資料	Account Management Module	User
連結方式	輸入資料		輸出資料
	avatar, nickname, bio		顯示新的個人資料
對應介面之要求			
接收收 avatar、nickname、bio，將資料交給 firebase 儲存。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-05	修改密碼	Account Management Module	User
連結方式	輸入資料		輸出資料
	old password, new password		跳到登入畫面
對應介面之要求			
接收 old password、new password，old password 與 firebase 驗證過後，將 new password 交給 firebase 做更新。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-06	Google 日曆授權	Account Management Module	User
連結方式	輸入資料		輸出資料
			授權結果
對應介面之要求			
透過 Google 日曆 API 管理授權			

介面編號	介面名稱	介面提供者	介面使用者
AMM-EI-07	取得通知	Account Management Module	User
連結方式	輸入資料		輸出資料
			顯示通知訊息
對應介面之要求			
系統上顯示傳給使用者的通知			

2.1.2 內部介面

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-01	/signin	Account Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”email”: ”Email (string)”,
	”password”: ”Password (string)”
	}
			{
			”status”: ”Status (string, 200)”,
			”JWT”: ”JWT token (string)”
			}

對應介面之要求			
將登入資訊傳給 firebase 進行資料驗證，接收回傳結果			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-02	/signup	Account Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”email”: ”Email (string)”,
	”password”: ”Password (string)”
	}
			{
			”status”: ”Status (string, 200)”,
			”JWT”: ”JWT token (string)”
			}

對應介面之要求			
將註冊資訊傳給 firebase，驗證後加入資料庫			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-03	/forget	Account Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”email”: ”Email (string)”,
	}
			{
			”status”: ”Status (string, 200)”
			}

對應介面之要求			
傳送 email，交給 firebase 寄信重設密碼			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-04	/person	Account Management Module	User
連結方式	輸入資料		輸出資料

PUT	{
	”JWT”: ”JWT token (string, cookie)”,
	”avatar”: ”Avatar (base64)”,
	”nickname”: ”Nickname (string)”,
	”bio”: ”bio (string)”
	}
			{
			”status”: ”Status (string, 200)”
			}

對應介面之要求			
將個人資訊傳給 firebase，根據 JWT 找到帳戶進行儲存，回傳狀態			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-05	/person	Account Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	”JWT”: ”JWT token (string, cookie)”,
	”email”: ”Email (string)”,
	}
			{
			”status”: ”Status (string, 200)”,
			}

對應介面之要求			
從 firebase 獲得使用者資料			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-06	/person/reset	Account Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”JWT”: ”JWT token (string, cookie)”,
	”oldPassword”: ”Password (string)”,
	”newPassword”: ”Password (string)”,
	}
			{
			”status”: ”Status (string, 200)”,
			}

對應介面之要求			
Firebase 確認 oldPassword 後，更新 Firebase 中 password 資訊。			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-07	/person/calendar	Account Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”JWT”: ”JWT token (string, cookie)”,
	}
			{
			”status”: ”Status (string, 200)”,
			}

對應介面之要求			
透過 Google 日曆 API 管理授權			

介面編號	介面名稱	介面提供者	介面使用者
AMM-II-08	/notify	Account Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	”JWT”: ”JWT token (string, cookie)”,
	}
			{
			”status”: ”Status (string, 200)”,
			}

對應介面之要求			
系統上顯示傳給使用者的通知			

2.2 專案管理子系統

2.2.1 外部介面

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-01	列出所有專案	Project Management Module	User
連結方式	輸入資料		輸出資料
			列出所有專案的資料
對應介面之要求			
根據登入的帳戶，回傳對應的專案資訊			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-02	建立新專案	Project Management Module	User
連結方式	輸入資料		輸出資料
	Project Name, Development Mode		新增完專案後，跳到專案畫面。
對應介面之要求			
接收 Project Name、Development mode，將資料丟給 firebase 儲存，並跳轉頁面			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-03	新建 Readme	Project Management Module	User
連結方式	輸入資料		輸出資料
			建立 Readme 檔案
對應介面之要求			
當有選則要建立 Readme 時，會自動建立 Readme			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-04	專案設定修改	Project Management Module	User
連結方式	輸入資料		輸出資料
	Project Name, Development Mode		修改完專案設定後，跳到專案畫面。
對應介面之要求			
接收 Project Name、Development Mode，交給 firebase，跳轉頁面			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-05	邀請加入專案	Project Management Module	User
連結方式	輸入資料		輸出資料
	User Mail		邀請成功後，跳出成功邀請通知。
對應介面之要求			
透過 Firebase 寄送訊息給 User Mail			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-06	移除成員	Project Management Module	User
連結方式	輸入資料		輸出資料
	User Mail		移除成功後，跳出成功移除通知。
對應介面之要求			
接收 User Mail 並交給 Firebase，Firebase 移除專案中成員。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-EI-07	刪除專案	Project Management Module	User
連結方式	輸入資料		輸出資料
			刪除完專案，跳回到專案列表。
對應介面之要求			
刪除使用者點擊的專案，並跳回專案列表。			

2.2.2 內部介面

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-01	/projects	Project Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	"JWT": "JWT token (string, cookie)"
	}
			{
			"status" : "Status (string)",
			"projects": " 專案列表 (Array<string, projectid>)"
			}

對應介面之要求			
根據登入的帳戶，取得 firebase 回傳的專案列表。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-02	/project	Project Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	"JWT": "JWT token (string, cookie)"
	"Name": "project name (string)",
	"DevMode": "development mode (string)",
	"DevTools": "development tools (string)",
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
傳 projectname、developmentmode 給 firebase，firebase 新增專案資訊。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-03	/project/readme	Project Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	"JWT": "JWT token (string, cookie)",
	"ProjectID": " 專案 ID (string)",
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
若有選擇加入 readme，則在 firebase 中建立 readme 資訊			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-04	/project	Project Management Module	User
連結方式	輸入資料		輸出資料

PUT	{
	"JWT": "JWT token (string, cookie)"
	"Name": "project name (string)",
	"DevMode": "development mode (string)",
	"DevTools": "development tools (string)",
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
傳 projectname、developmentmode 給 firebase，firebase 更新專案資訊。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-05	/project/member	Project Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	"JWT": "JWT token (string, cookie)",
	"ProjectID": " 專案 ID (string)",
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
獲取專案成員列表			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-06	/project/member	Project Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	"JWT": "JWT token (string, cookie)",
	"ProjectID": " 專案 ID (string)",
	"mail": "User mail (string)",
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
傳 mail 給 firebase，firebase 發送邀請決定。，是否讓 firebase 新增專案成員資訊。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-07	/project/member	Project Management Module	User
連結方式	輸入資料		輸出資料

DELETE	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	”mail”: ”User mail (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
根據 usermail，刪除 firebase 專案中成員資訊。			

介面編號	介面名稱	介面提供者	介面使用者
PMM-II-03	/project	Project Management Module	User
連結方式	輸入資料		輸出資料

DELETE	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
刪除專案資訊，並刪除 firebase 專案資訊。			

2.3 Repo 管理子系統

2.3.1 外部介面

介面編號	介面名稱	介面提供者	介面使用者
RMM-EI-01	列出所有 REPO	Repo Management Module	User
連結方式	輸入資料		輸出資料
			列出所有 REPO
對應介面之要求			
根據專案資訊，列出專案中所有 REPO			

介面編號	介面名稱	介面提供者	介面使用者
RMM-EI-02	新增 REPO	Repo Management Module	User
連結方式	輸入資料		輸出資料
	reponame, repourl		顯示新增成功，並回到 REPO 列表頁面
對應介面之要求			
接收 reponame, repourl，新增 REPO，回傳 repoid 到前端。			

介面編號	介面名稱	介面提供者	介面使用者
RMM-EI-02	刪除 REPO	Repo Management Module	User
連結方式	輸入資料		輸出資料
	repoid		顯示刪除成功，並回到 REPO 列表頁面
對應介面之要求			
接收 repoid，刪除 REPO。			

2.3.2 內部介面

介面編號	介面名稱	介面提供者	介面使用者
RMM-II-01	/project/repos	Repo Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”,
			”repos”: ”REPO 列表”
			}

對應介面之要求			
接收 projectid，傳 projectid 給 firebase，並接收 repos			

介面編號	介面名稱	介面提供者	介面使用者
RMM-II-02	/project/repo	Repo Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”JWT”: ”JWT token (string, cookie)”,
	”projectid”: ”Project ID (string)”
	”reponame”: ”REPO 名稱 (string)”
	”repourl”: ”REPO URL (string)”
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”,
			”repoid”: ”REPO ID (string)”
			}

對應介面之要求			
傳 projectid 給 firebase 找到對應的 project，並把 reponame, repourl 給 firebase 儲存，並回傳 repoid			

介面編號	介面名稱	介面提供者	介面使用者
RMM-II-03	/project/repo	Repo Management Module	User
連結方式	輸入資料		輸出資料

PUT	{
	”JWT”: ”JWT token (string, cookie)”,
	”projectid”: ”Project ID (string)”
	”repoid”: ”REPO ID (string)”
	”reponame”: ”REPO 名稱 (string)”
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”,
			”repoid”: ”REPO ID (string)”
			}

對應介面之要求			
傳 projectid 給 firebase 找到對應的 project，並把 reponame, repourl 給 firebase 更新，並回傳 repoid			

介面編號	介面名稱	介面提供者	介面使用者
RMM-II-02	/project/repo	Repo Management Module	User
連結方式	輸入資料		輸出資料

DELETE	{
	”JWT”: ”JWT token (string, cookie)”,
	”projectid”: ”Project ID (string)”
	”repoid”: ”Project ID (string)”
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊 息 (string)”,
			}

對應介面之要求			
傳 projectid、repoid 給 firebase 找到對應的 repo，最後刪除 repo			

2.4 文件管理子系統

2.4.1 外部介面

介面編號	介面名稱	介面提供者	介面使用者
DMM-EI-01	列出所有文件	Doc Management Module	User
連結方式	輸入資料		輸出資料
			列出所有文件
對應介面之要求			
firebase 根據 projectid 找到對應的 project，並回傳所有文件			

介面編號	介面名稱	介面提供者	介面使用者
DMM-EI-02	新增文件	Doc Management Module	User
連結方式	輸入資料		輸出資料
	文件類型		顯示新增成功，並顯示新開的文件
對應介面之要求			
根據文件類型，開啟新的文件			

介面編號	介面名稱	介面提供者	介面使用者
DMM-EI-03	編輯文件	Doc Management Module	User
連結方式	輸入資料		輸出資料
	文件類型		顯示編輯成功，並顯示編輯後的文件
對應介面之要求			
firebase 根據文件類型找到對應的文件，並更新內容。			

介面編號	介面名稱	介面提供者	介面使用者
DMM-EI-04	刪除文件	Doc Management Module	User
連結方式	輸入資料		輸出資料
	文件類型		顯示刪除成功，並回到文件列表。
對應介面之要求			
firebase 根據文件類型找到對應的文件，並刪除後更新頁面。			

2.4.2 內部介面

介面編號	介面名稱	介面提供者	介面使用者
DMM-II-01	/project/doc	Doc Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	”docid”: ” 文件 ID (string)”
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
docid 為各個文件的 id，根據 id 找到對應的文件			

介面編號	介面名稱	介面提供者	介面使用者
DMM-II-03	/project/doc	Doc Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	"JWT": "JWT token (string, cookie)",
	"ProjectID": " 專案 ID (string)",
	"doctype": " 文件類型 (string)"
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
doctype 為文件類型，根據 doctype 建立對應的位置，並新增一個文件			

介面編號	介面名稱	介面提供者	介面使用者
DMM-II-04	/project/doc	Doc Management Module	User
連結方式	輸入資料		輸出資料

PUT	{
	"JWT": "JWT token (string, cookie)",
	"ProjectID": " 專案 ID (string)",
	"docid": " 文件 ID (string)"
	}
			{
			"status": "Status (string)",
			"message": " 訊息 (string)"
			}

對應介面之要求			
根據 docid 找到對應的文件，並更新文件內容			

介面編號	介面名稱	介面提供者	介面使用者
DMM-II-05	/project/doc	Doc Management Module	User
連結方式	輸入資料		輸出資料

DELETE	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	”docid”: ” 文件 ID (string)”
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
根據 docid 找到對應的文件，並刪除文件			

2.5 看板子系統

2.5.1 外部介面

介面編號	介面名稱	介面提供者	介面使用者
KMM-EI-01	顯示所有任務	Kanban Management Module	User
連結方式	輸入資料		輸出資料
			列出所有任務
對應介面之要求			
顯示所有任務，包含任務名稱、任務負責人、任務結束時間、任務描述			

介面編號	介面名稱	介面提供者	介面使用者
KMM-EI-02	新增任務	Kanban Management Module	User
連結方式	輸入資料		輸出資料
	taskname, taskowner, taskend, taskdescription		顯示新增成功，並顯示新增的任務
對應介面之要求			
接收 taskname, taskowner, taskend, taskdescription 後丟到 firebase 新增			

介面編號	介面名稱	介面提供者	介面使用者
KMM-EI-03	移動任務	Kanban Management Module	User
連結方式	輸入資料		輸出資料
	oldstate, newstate, taskid		顯示移動成功，並顯示移動後的任務
對應介面之要求			
拖曳之後自動變更 task 所屬狀態，和位置			

2.5.2 內部介面

介面編號	介面名稱	介面提供者	介面使用者
KMM-II-01	/project/kanbans	Kanban Management Module	User
連結方式	輸入資料		輸出資料

GET	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
顯示所有任務，包含任務名稱、任務負責人、任務結束時間、任務描述			

介面編號	介面名稱	介面提供者	介面使用者
KMM-II-02	/project/kanban	Kanban Management Module	User
連結方式	輸入資料		輸出資料

POST	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	”name”: ” 任務名稱 (string)”,
	”owner”: ” 任務負責人 (string)”,
	”end”: ” 任務結束時間 (string)”,
	”description”: ” 任務描述 (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊 息 (string)”
			}

對應介面之要求			
新增任務，並且顯示新增的任務			

介面編號	介面名稱	介面提供者	介面使用者
KMM-II-03	/project/kanban	Kanban Management Module	User
連結方式	輸入資料		輸出資料

PUT	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	”name”: ” 任務名稱 (string)”,
	”owner”: ” 任務負責人 (string)”,
	”end”: ” 任務結束時間 (string)”,
	”description”: ” 任務描述 (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊 息 (string)”
			}

對應介面之要求			
編輯任務，並且顯示編輯後的任務			

介面編號	介面名稱	介面提供者	介面使用者
KMM-II-02	/project/kanban	Kanban Management Module	User
連結方式	輸入資料		輸出資料

DELETE	{
	”JWT”: ”JWT token (string, cookie)”,
	”ProjectID”: ” 專案 ID (string)”,
	}
			{
			”status”: ”Status (string)”,
			”message”: ” 訊息 (string)”
			}

對應介面之要求			
刪除任務			

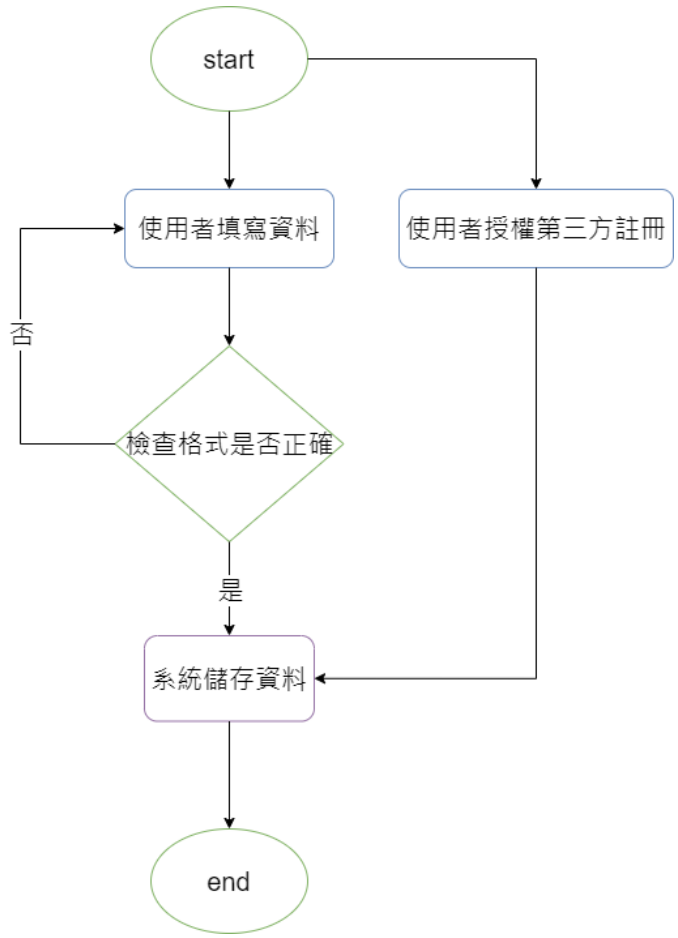
2.6 Q&A 子系統

2.6.1 外部介面

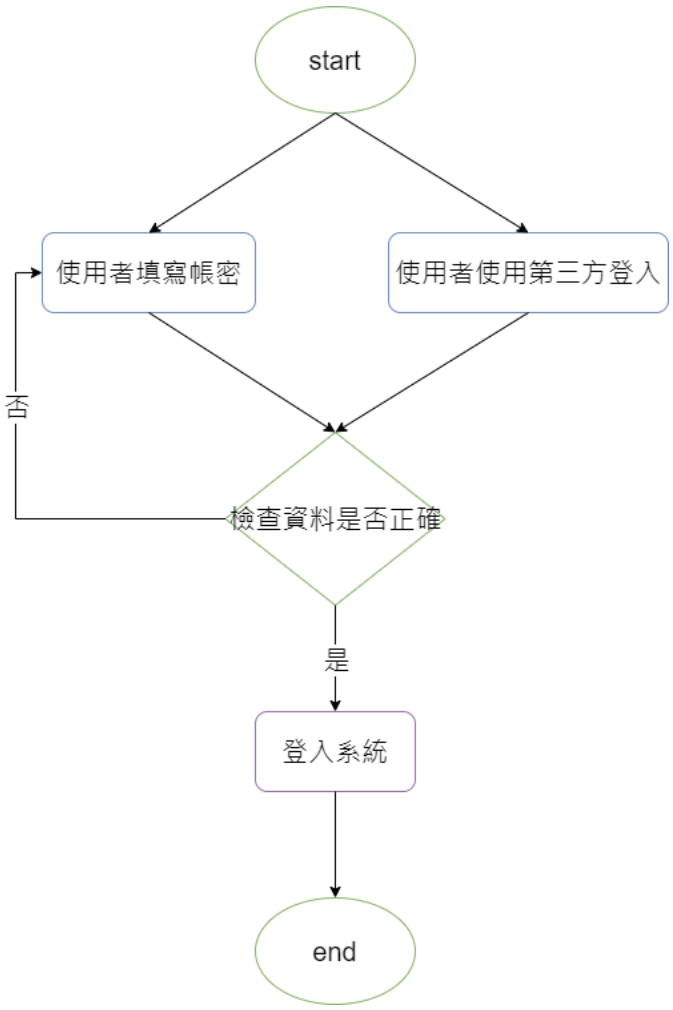
介面編號	介面名稱	介面提供者	介面使用者
QMM-EI-01	顯示所有任務	Q&A Management Module	User
連結方式	輸入資料		輸出資料
			列出所有 Q&A
對應介面之要求			
使用 gitbook 列出所有 Q&A			

3. 流程設計 (Process Design)

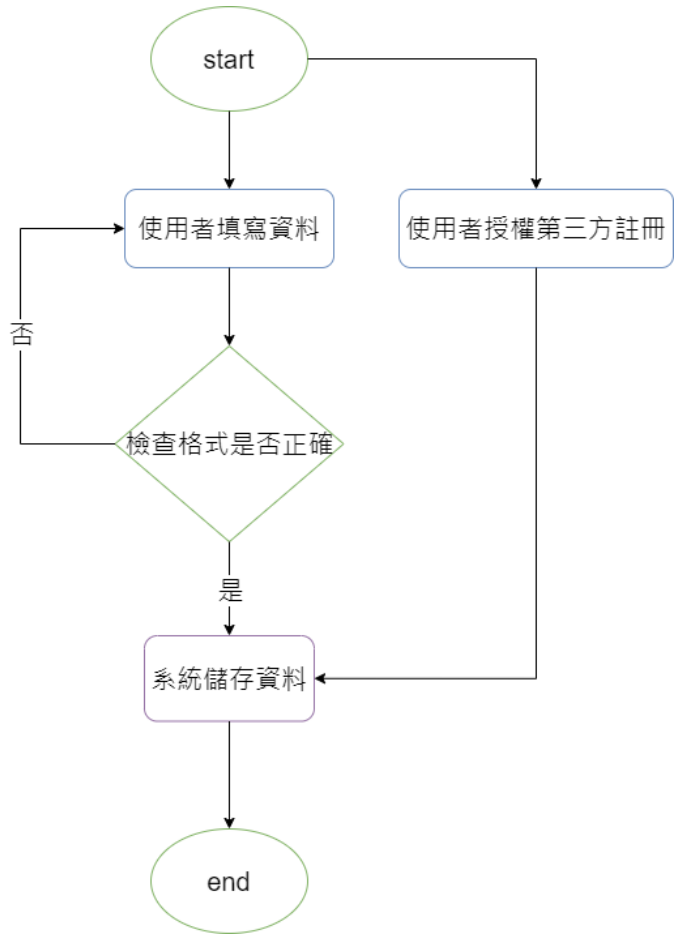
(A) 會員註冊



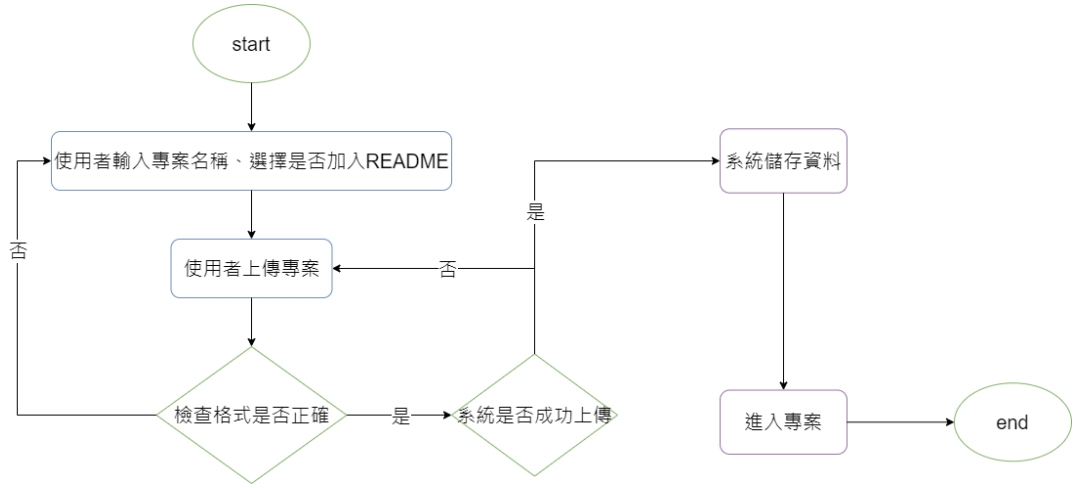
(B) **會員登入**



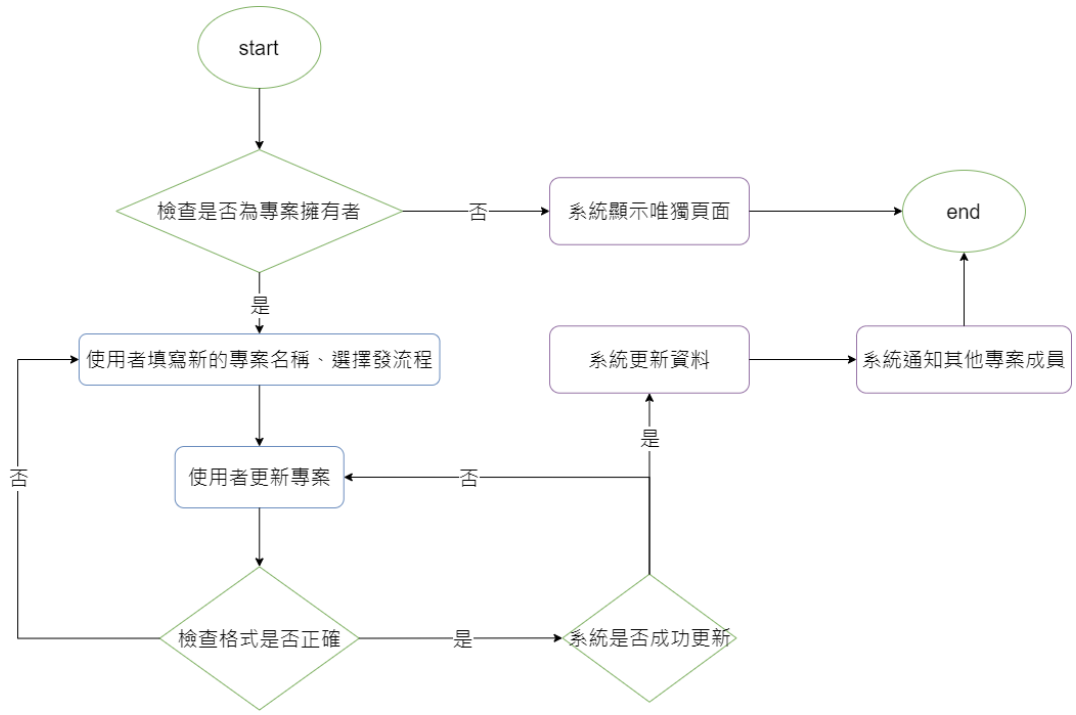
(C) 重設密碼


start
使用者填寫資料
使用者授權第三方註冊
否
檢查格式是否正確
是
系統儲存資料
end

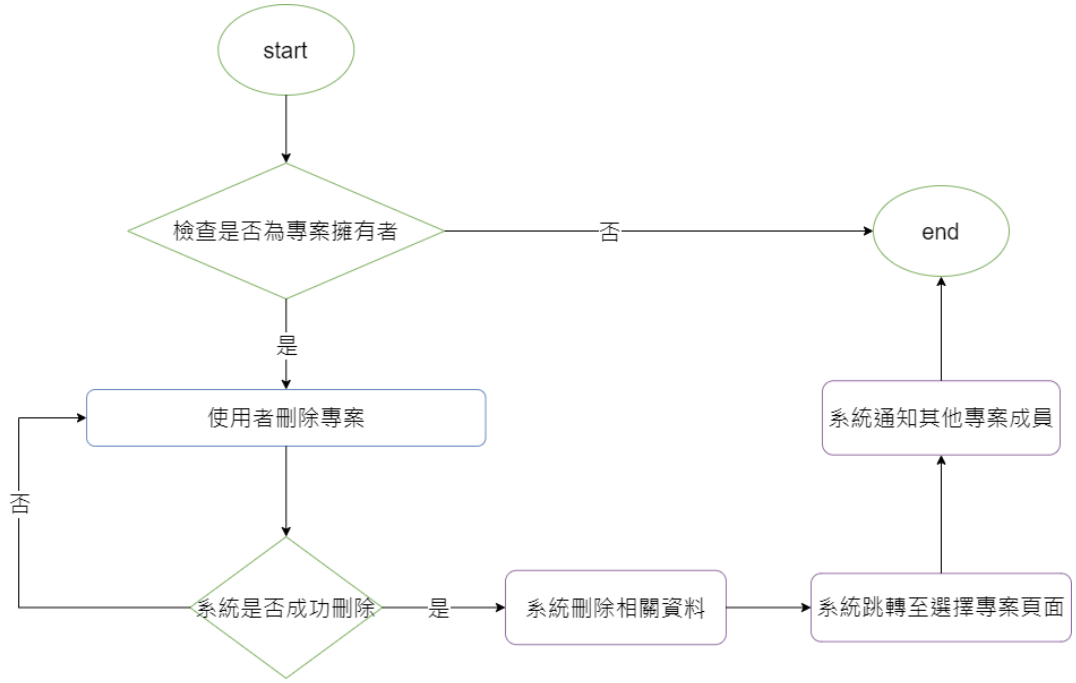
(D) 建立專案


start
使用者輸入專案名稱、選擇是否加入README
否
使用者上傳專案
否
是
檢查格式是否正確
是
系統是否成功上傳
是
系統儲存資料
進入專案
end

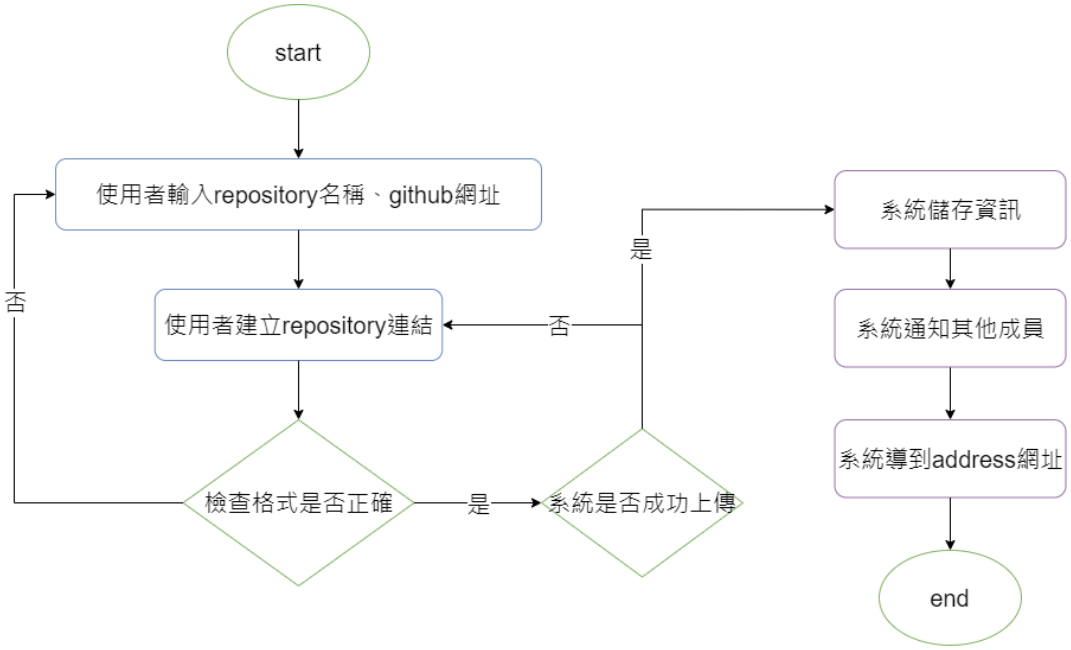
(E) 編輯專案



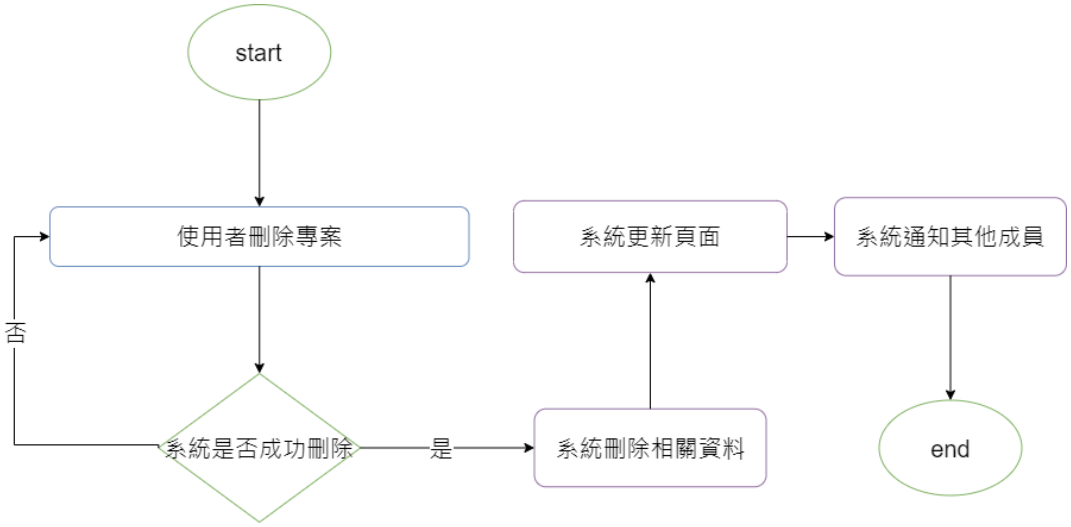
(F) 刪除專案



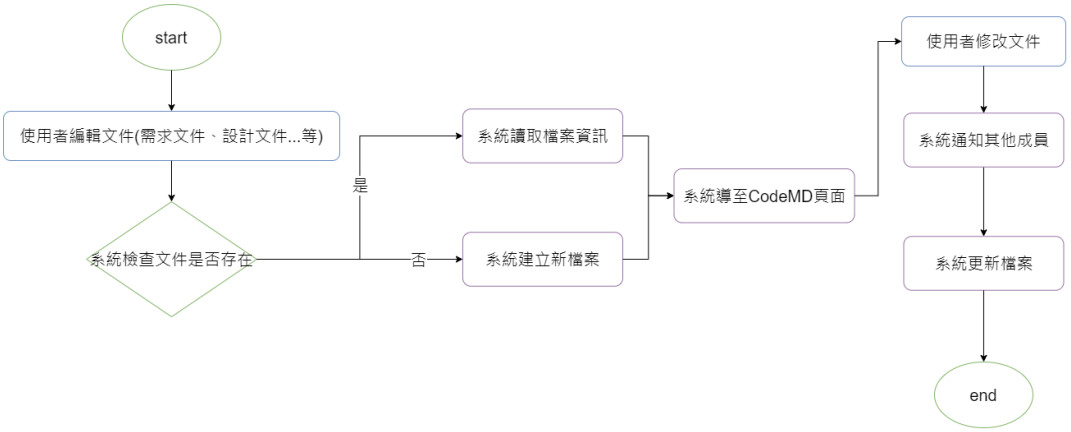
(G) 建立 repository



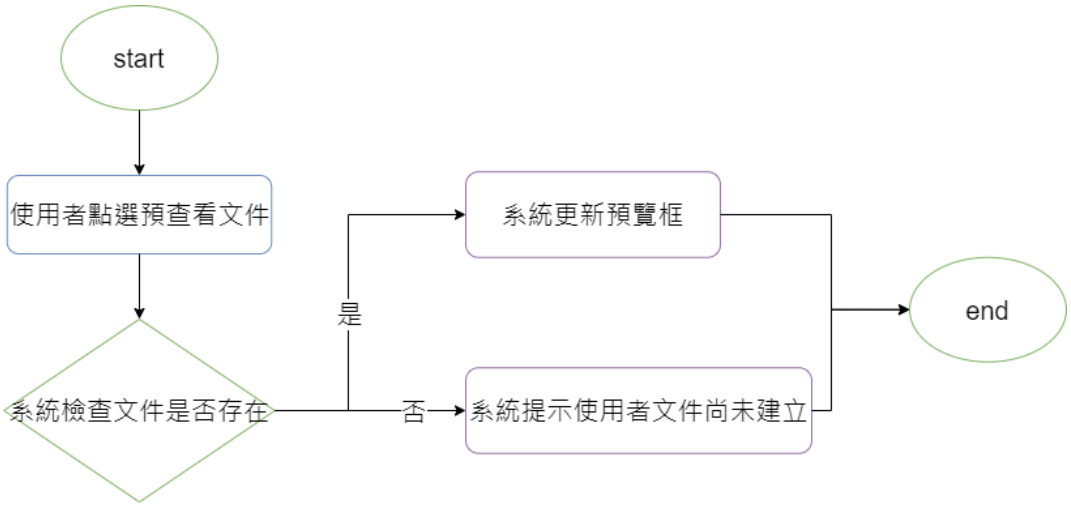
(H) 刪除 repository



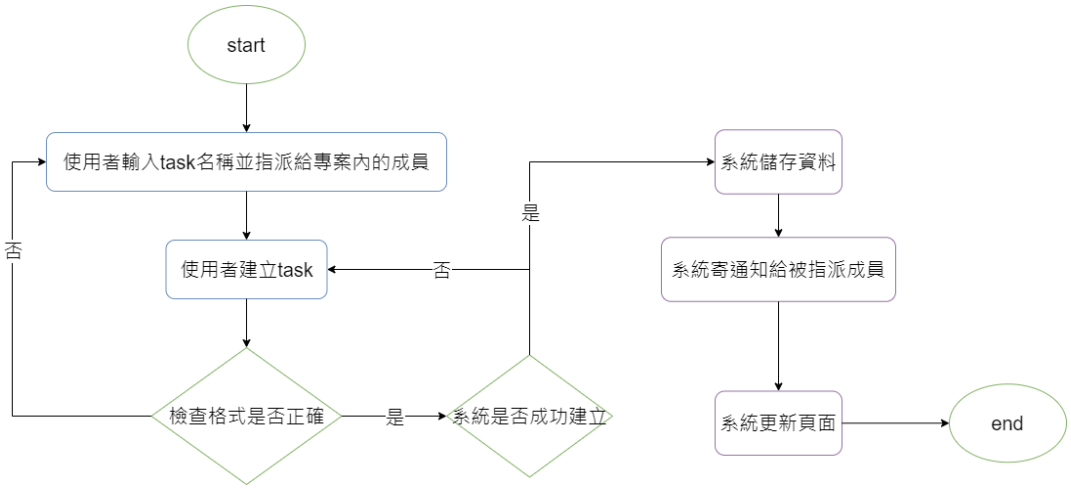
(I) 編輯文件



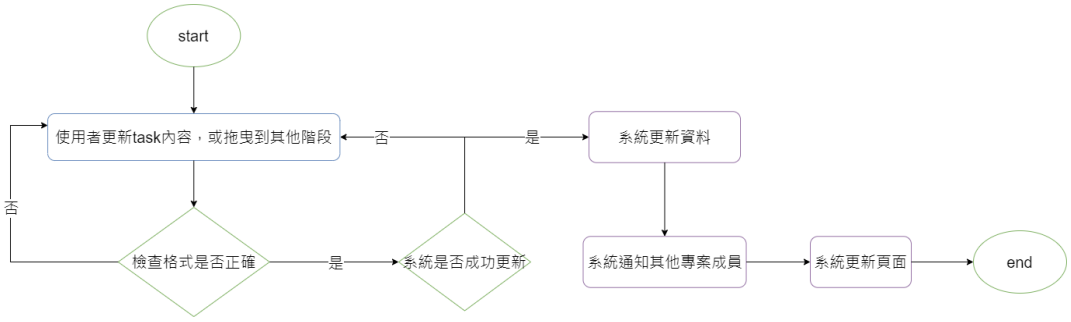
(J) 預覽文件



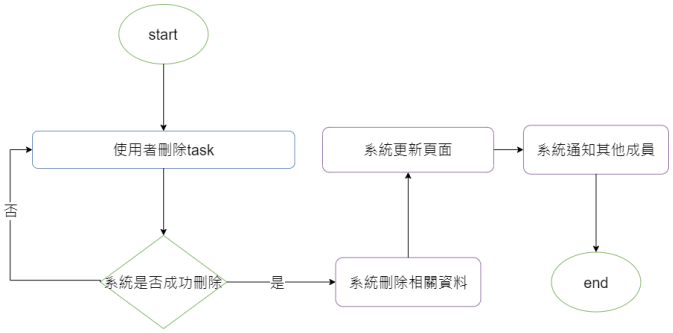
(K) 看板新增任務



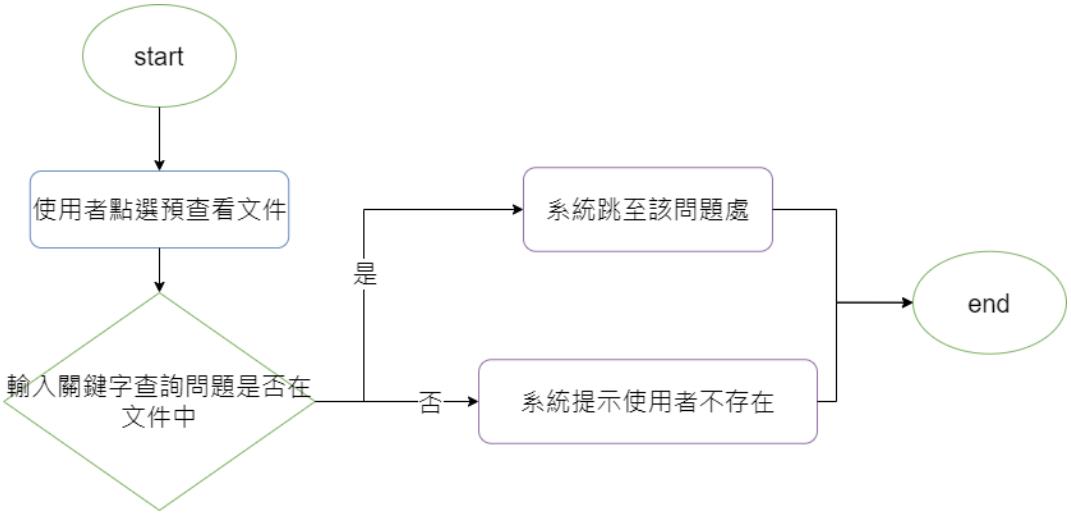
(L) 看板編輯任務



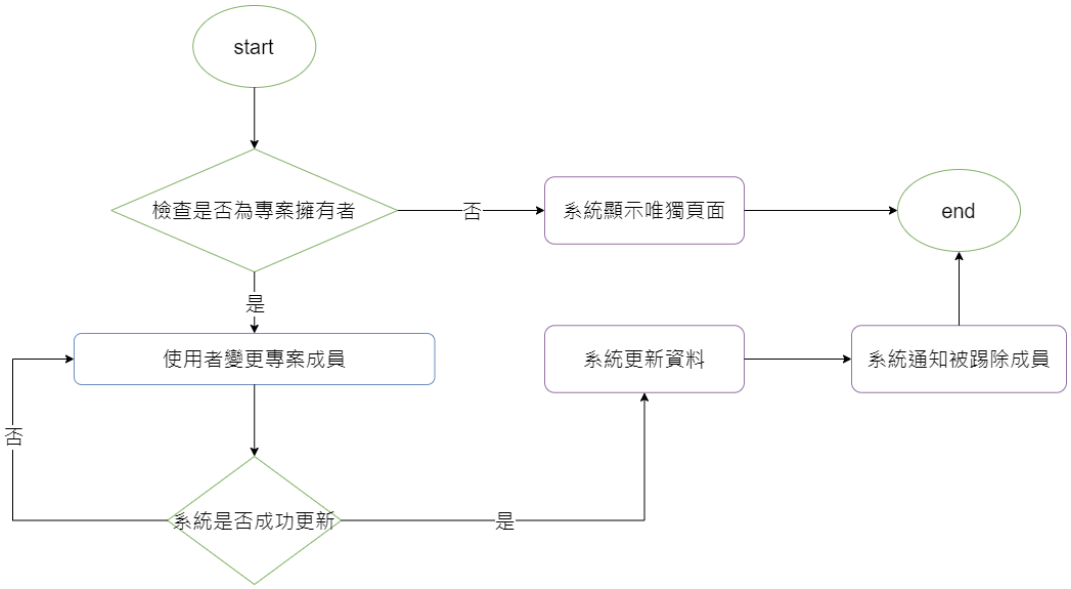
(M) 看板刪除任務



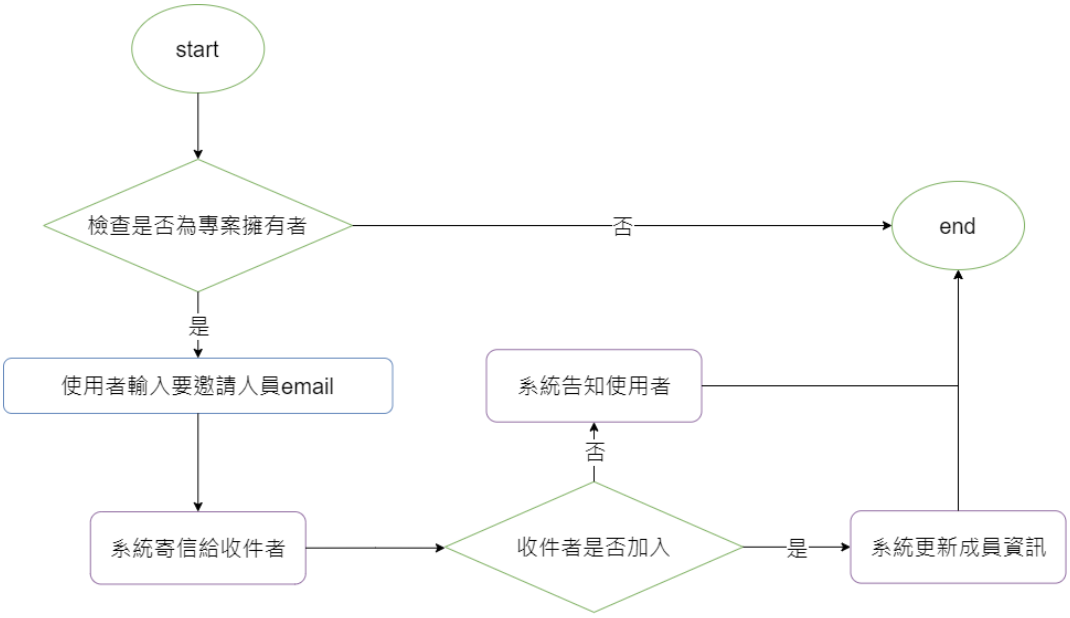
(N) Q&A 文件



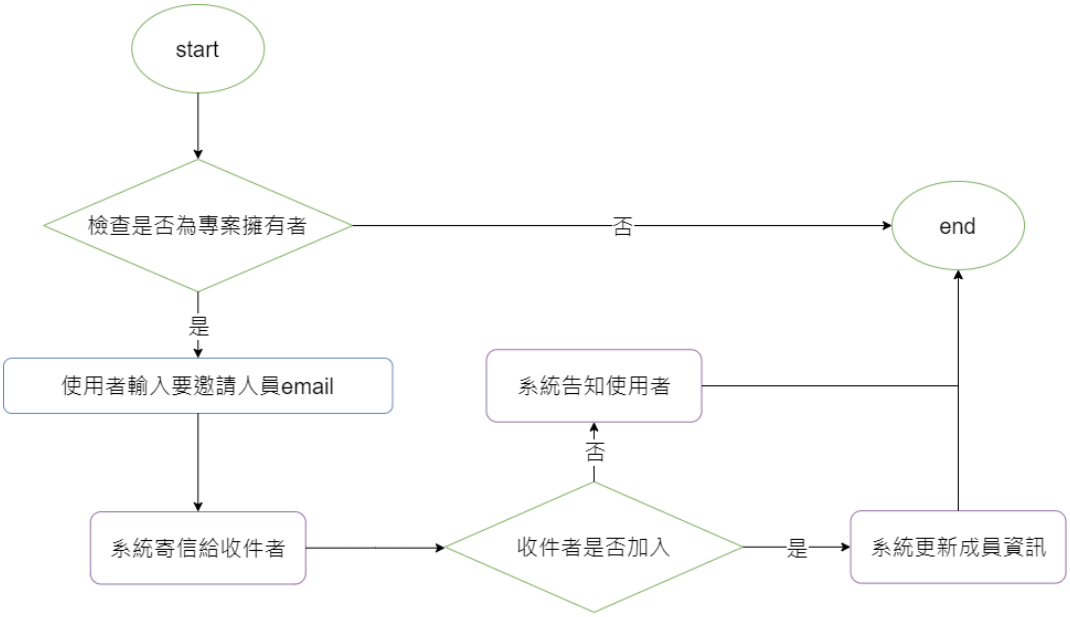
(O) 變更成員



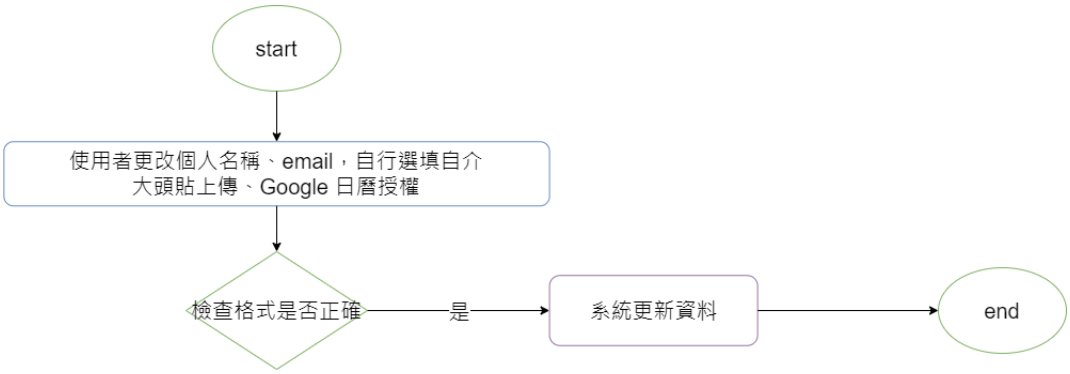
(P) 邀請其他人



(Q) 邀請其他人



(R) 設定個人資料



4. 使用者畫面設計 (User Interface Design)

可用各種畫面設計方式，如發展網頁 prototype、繪製 PowerPoint 投影片等，去設計你的主要系統畫面。此部份應是 SRS 中使用者介面分析的進階設計版本。
線框圖? 包含所有 UI 設計版型顏色等等
CLI 工具 UI 設計
何

5. 資料設計 (Data Design)

1. User

欄位代碼	欄位名稱	欄位內容	欄位型態
email	電子郵件	User 的電子郵件	string
nikename	姓名	User 的姓名	string
avatar	大頭貼	User 的大頭貼	string
bio	自我介紹	User 的自我介紹	string
projects	專案	User 的全部專案 ID	Array

2. Project

欄位代碼	欄位名稱	欄位內容	欄位型態
id	專案 ID	專案的 ID	string
name	專案名稱	專案的名稱	string
members	專案成員	專案的成員 ID	Array<User>
repositories	專案 repository	專案的 repositoryID	Array<Repo>
documents	專案文件	專案的文件 ID	Array<Doc>
tasks	專案任務	專案的任務 ID	Array<Tasks>

3. Repo

欄位代碼	欄位名稱	欄位內容	欄位型態
id	repositoryID	repository 的 ID	string
name	repository 名稱	repository 的名稱	string
description	repository 描述	repository 的描述	string
url	repository url	repository 的 url	Array<string>

4. Doc

欄位代碼	欄位名稱	欄位內容	欄位型態
id	文件 ID	文件的 ID	string
name	文件名稱	文件的名稱	string
content	文件內容	文件的內容	string

5. Tasks

欄位代碼	欄位名稱	欄位內容	欄位型態
id	任務 ID	任務的 ID	string
name	任務名稱	任務的名稱	string
description	任務描述	任務的描述	string
status	任務狀態	任務的狀態	string
deadline	任務截止日期	任務的截止日期	string
assignee	任務指派人	任務的指派人 ID	User

6. 類別圖設計 (Class Diagram)

發展 UML 之 class diagram (可對應 C4 model 中的 Code diagram)，明確設計系統中包含哪些類別，以及這些類別之間的關係。
若遇到非一般物件類別或特殊物件類別，如 HTML、JavaScript、JSP、Servlet 等，可用 stereotype 表達，如 «JavaScript»、«Servlet»。
可把重點放在定義程式結構：訂出所有類別名稱、拉出所有類別關係，再加入重要的 attibute/operation 即可。而特殊的類別職責設計可加上文字說明。
StarUML
前後端都要

7. 實作方案 (Implementation Languages and Platforms)

平台	網站	ClI
前端	JavaScript	Go
後端	Go	Go
前端框架	Vue	Cobra
後端框架	Gin	Gin
資料庫	Firebase	Firebase
前後端溝通方式	RESTful API	RESTful API

8. 設計議題 (Design Issue)

1. 議題一
A. 問題描述: Golang 沒有 Firebase 的客戶端 SDK，無法使用 Firebase。

B. 解決方案：a. 自己寫一個 library，b. 將後端改成 node.js。
C. 最終定案：a. 自己寫一個 library。
D. 原因：換另一個後端寫成本太高。
2. 議題二
A. 問題描述: 前端網頁的 UI 使用 html 手刻太過於花時間。
B. 解決方案：a. 慢慢刻到底，b. 使用 element-ui。
C. 最終定案：b. 使用 element-ui。
D. 原因：使用 element-ui，可以減少開發時間，並且可以配合 Vue。
@ 唐劭賢交給你排版力
議題內容:
在本地端的一個專案應該用什麼方式呈現?
可能解決方案:
1. 用一個檔案存所有資料
2. 用一個資料夾存所有資料
3. 直接讀取遠端的資料
最後解決方案:
用一個資料夾存一個專案的所有資料
理由:
1. 讓本機即使是離線狀態也能使用
2. 使用類似 git 的方式來管理專案
3. 方便使用者控制專案的檔案
議題內容:
在本地端使用 SCAF 時，需要與遠端同步，應該使用什麼策略?
可能解決方案:
1. 使用 git 進行版本控制，並利用 git 的合併功能，將本地端與遠端同步。
2. 使用類似 google 文件的方式，版本只會是線性的，以最新修改的版本為主。但會有不同本地端的版本衝突的問題。
最後解決方案:
使用者執行同步指令時，將遠端與本地端強制更新成其中的最新版本。
理由:
1. 實作簡單
2. 雖然無法合併不同版本，但能解決版本衝突問題
3. 未來可以再加入合併功能
議題內容:
在 CLI 工具中新建專案時，會有必要與非必要設定 (如名稱與開發流程)，每個設定使用者都必須輸入一次選項，怎麼讓使用者的體驗更好?
可能解決方案:
1. 使用者在新建專案時，只會有必要設定，非必要設定可以在專案建立後再設定。
2. 使用者在新建專案時，需要輸入所有設定，包含必要與非必要設定。

最後解決方案:
使用者在新建專案時，若使用 –default 參數，則只會有必要設定，非必要設定會設定為預設值，建立完成再去個別設定。
若沒有使用 –default 參數，則需要輸入所有設定，但也可以在輸入時指定為預設值。
理由:
1. 使用者有彈性，可以選擇是否要使用預設值。
2. 能讓使用者知道有哪些設定可以調整。
議題內容:
在 CLI 工具中，是否要使用指令來編輯 README 檔案?
可能解決方案:
1. 使用指令編輯 README 檔案。
2. 使用編輯器編輯 README 檔案。
最後解決方案:
使用編輯器直接編輯 README 檔案。
理由:
1. 使用者可以用自己習慣的編輯器。
2. 使用者不會被指令綁住，較為彈性。
3. 減少指令的複雜度。
議題內容:
在 CLI 工具中，專案開發工具 (如 Kanban) 的啟用要當作一般設定，或是獨立成一個子指令?
可能解決方案:
1. 獨立成一個子指令，如 scaf devtools 可以進入開發工具的設定畫面。
2. 當作一般設定，如 scaf config set project.kanban true 可以啟用 Kanban。
最後解決方案:
獨立成一個子指令，讓使用者可以一覽 (甚至是搜尋) 所有的開發工具，並決定要啟用哪些。
理由:
1. 未來專案開發工具可能會有更多的選項 (如 CI/CD、甘特圖)，讓設定變得複雜。
2. 與套件管理器 (如 Python 的 pip) 的概念類似，都把套件管理的任務獨立出來。
3. 其他設定 (如專案名稱、開發流程) 都不會像是開發工具有那麼多種選項。
議題內容:
如何讓專案中被邀請的使用者可以接受/拒絕邀請?
可能解決方案:
1. 透過電子郵件傳送邀請。
2. 使用通知功能，並在通知中讓使用者選擇接受或拒絕邀請。
最後解決方案:
使用通知功能，並在通知中讓使用者選擇接受或拒絕邀請。
理由:
1. 對發送邀請的做一個獨立的功能太多餘。
2. 通知功能可以在未來新增其他種類的通知，泛用性很高。

3. 即使是用通知實作邀請，未來也能加入電子郵件通知功能。