Project Name

Github 專案分析系統

GRAS

(GitHub Repository Analysis System)

Software Design Document (SDD)

Version: 1.6

Team#2

Name	ID	E-mail
劉文揚	110598096	t110598096@ntut.org.tw
謝狄烽	110598087	t110598087@ntut.org.tw
林妤潔	110598055	t110598055@ntut.org.tw
曹智凱	110598085	t110598085@ntut.org.tw

Department of Computer Science & Information Engineering
National Taipei University of Technology

12/10/2021

Table of Contents

Section 1 Introduction

1.1 Scope of the System

Github 專案分析系統(GitHub Repository Analysis System, GRAS)令使用者可以使用本平台來分析專案狀態，並且可透過 SonarQube 找出專案的 Code Smell、Bugs、Vulnerabilities 等等關於專案品質的指標，使得使用者得以快速的瞭解專案目前的狀況。

1.2 Purpose of the Document

此文件主要是介紹 Github 專案分析系統(GitHub Repository Analysis System, GRAS)的研究背景與動機以及系統的設計、規格、流程，使得閱覽此文件的開發人員得以迅速的瞭解此系統的設計規格與架構。

1.3 Overview of the Document

- 第一章 Introduction
- 第二章 描述系統需求，並以 User Story 的方式呈現出來
- 第三章 設計限制和解決方案
- 第四章 系統架構
- 第五章 更詳細的系統設計細節

Section 2 System Requirements

2.1 Functional Requirements

2.1.1 User Stories and Acceptance Criteria

使用者故事(User Stories):

需求編號	使用者故事
GRAS-U-01	為了減少管理多個平台帳號， 身為一個平常有在用 Github 的 Leader， 我想要用 GitHub 帳號登入分析平台。
GRAS—02	為了得知系統潛在的問題， 身為一個剛接手他人系統的工程師， 我想要一個可以分析系統中 bugs 的分析平台。
GRAS—03	為了保護使用者的資料， 身為一個資安工程師， 我想要一個可以分析系統中資安疑慮的分析平台。
GRAS—04	為了減少維護成本， 身為一個開發系統的工程師， 我想要一個分析系統中 Code Smells 的分析平台。

驗收準則(Acceptance Criteria):

需求編號	驗收準則
GRAS--01	1. 可以使用 Github 帳號登入系統。 2. 可以將 Github 帳號中的 Project 自動連結至本分析平台。
GRAS--02	1. 可以有一個 buttons，跳到顯示掃描結果的頁面。 2. 可以有一個頁面顯示程式碼經過 SonarQube 掃描後的 bugs。
GRAS--03	1. 可以有一個 buttons，跳到顯示掃描結果的頁面。 2. 可以有一個頁面顯示程式碼經過 SonarQube 掃描後的 Vulnerabilities。
GRAS--04	1. 可以有一個 buttons，跳到顯示掃描結果的頁面。 2. 可以有一個頁面顯示程式碼經過 SonarQube 掃描後的 Code Smells。

2.2 Non-Functional Requirements

需求編號	需求描述
GRAS-NF-01	重新設計界面。
GRAS-NF-02	對使用者密碼進行加密。
GRAS-NF-03	防止SQL Injection。
GRAS-NF-04	使用RWD 設計令不同尺寸裝置都可正常顯示頁面。

Section 3 Design Constraints and Solutions

3.1 Technical Solution Criteria

3.1.1 Language

- 易學性：考慮選擇的應用軟體是否容易上手。
- 安全性：考慮應用軟體設計上的安全保密性。
- 熟悉度：考慮組員有無該語言的開發經驗。
- 擴充性：考慮後續的擴充是否容易。
- 維護性：考慮後續的維護是否容易。
- 遷移性：考慮從舊專案遷移至新專案是否容易。

3.1.2 Frontend Framework

- 易學性：考慮選擇的應用軟體是否容易上手。
- 安全性：考慮應用軟體設計上的安全保密性。
- 生態圈大小：考慮該框架的社群大小。
- 擴充性：考慮後續的擴充是否容易。
- 維護性：考慮後續的維護是否容易。
- 遷移性：考慮從舊專案遷移至新專案是否容易。
- 熟悉程度：考慮組員有無該框架的開發經驗。

3.1.3 Database

- 擴充性：考慮後續的擴充是否容易。
- 熟悉度：考慮組員有無該資料庫的使用經驗。
- 花費　：考慮使用資料庫的成本。
- 遷移性：考慮從舊專案遷移至新專案是否容易。

3.2 Alternative Solutions

3.2.1 Language

	Java	Python	C++
易學性	中 該語言對於新手來說較難上手，規範與限制較 C++少一點。	高 自由度高，規範與限制較少。	低 該語言對於新手來說較難上手，規範與限制較多。
安全性	較安全。	較不安全。	較安全。
熟悉度	每個組員都會使用。	每個組員都會使用。	較少組員精通。
擴充性	套件較 C++豐富。	套件及其生態系很豐富。	套件較少。
維護性	資料封裝程度最完整。	細節難以處理，不好維護。	需考慮 MemoryLeak 的問題。
遷移性	與舊專案使用相同語言，因此相對容易。	與舊專案使用不同語言，因此遷移不易。	與舊專案使用相同語言，因此遷移不易。

3.2.2 Frontend Framework

	Vue	React	Angular
易學性	高 很自由，很好上手，限制少。	中 提供了一個入門指南，文件非常的完整，有問題都可以在裡面找到解決方案。	低 對於新手入門較不友善，學習周期較長，適合用於開發大型網站，為 google 推出的框架。
安全性	沒有安全性需求		
生態圈大小	大	大	較小
擴充性	高	高	高
維護性	高 有瀏覽器專用開發工具，資料流雙向綁定	高 有瀏覽器專用開發工具，資料流雙向綁定	中 頁面元件化
遷移性	與舊專案使用不同框架，因此遷移不易。	與舊專案使用不同框架，因此遷移不易。	與舊專案使用相同框架，因此相對容易。
熟悉程度	成員們都沒學過		

3.2.3 Database

	MySQL	MongoDB	SQLServer
擴充性	中，採用傳統關聯式資料庫。	高，因採用 NoSQL 技術，水平擴充能力較傳統 SQL 好。	中，採用傳統關聯式資料庫。
熟悉度	較高，大部分組員接觸。	低，組員皆未接觸過。	中，部分組員接觸過。
花費(分數高→便宜)	高	高	低
遷移性	與舊專案使用相同 DB，因此相對容易。	與舊專案使用不同 DB，因此遷移不易。	與舊專案使用不同 DB，因此遷移不易。

3.3 Selected Solution

3.3.1 Language

	Java	Python	C++
易學性	☆☆	☆☆☆	☆
安全性	☆☆☆	☆	☆☆
熟悉度	☆☆	☆☆☆	☆
擴充性	☆☆	☆☆☆	☆
維護性	☆☆☆	☆	☆☆
遷移性	☆☆☆	☆	☆
加總	15	12	8

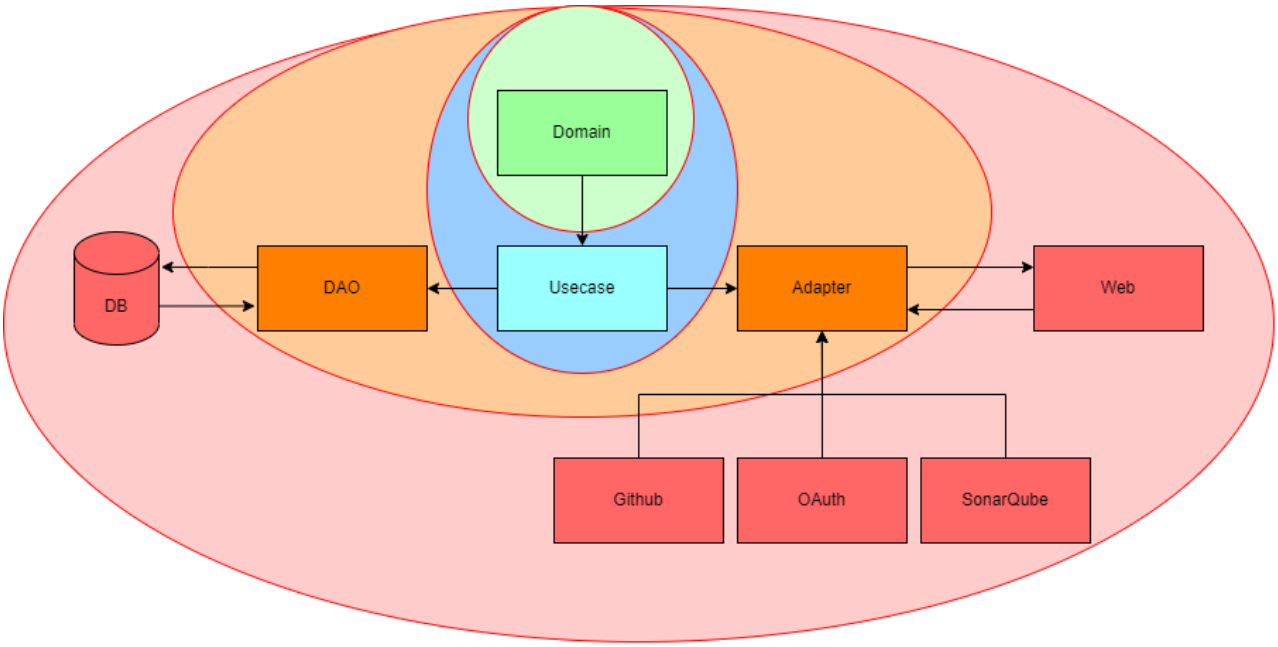
3.3.2 Frontend Framework

	Vue	React	Angular
易學性	★★★	★★	★★
安全性	★	★	★★★
生態圈大小	★★	★★★	★
擴充性	★★	★★	★★
維護性	★★	★★★	★★
遷移性	★	★	★★★
熟悉程度	★	★	★★★
加總	12	13	16

3.3.3 Database

	MySQL	MongoDB	SQLServer
安全性	★	★★★	★★★
擴充性	★★	★★★	★★
熟悉度	★★★	★	★
花費(分數高→便宜)	★★★	★	★★
遷移性	★★★	★	★
加總	9	8	8

Section 4 System Architecture



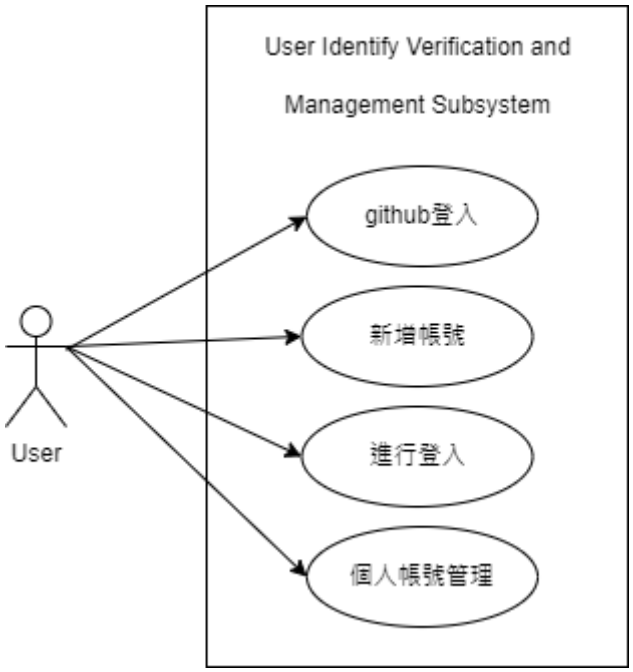
Layer	
Domain	Account GitRepository Project
UseCase	account gitrepository project GithubRepositoryAccessor OAuth
Adapter	Account Gitrepository Project Servlet sonarqube
Database	GithubRepositoryAccessor

Section 5 Detailed System Design

5.1 System Design Models

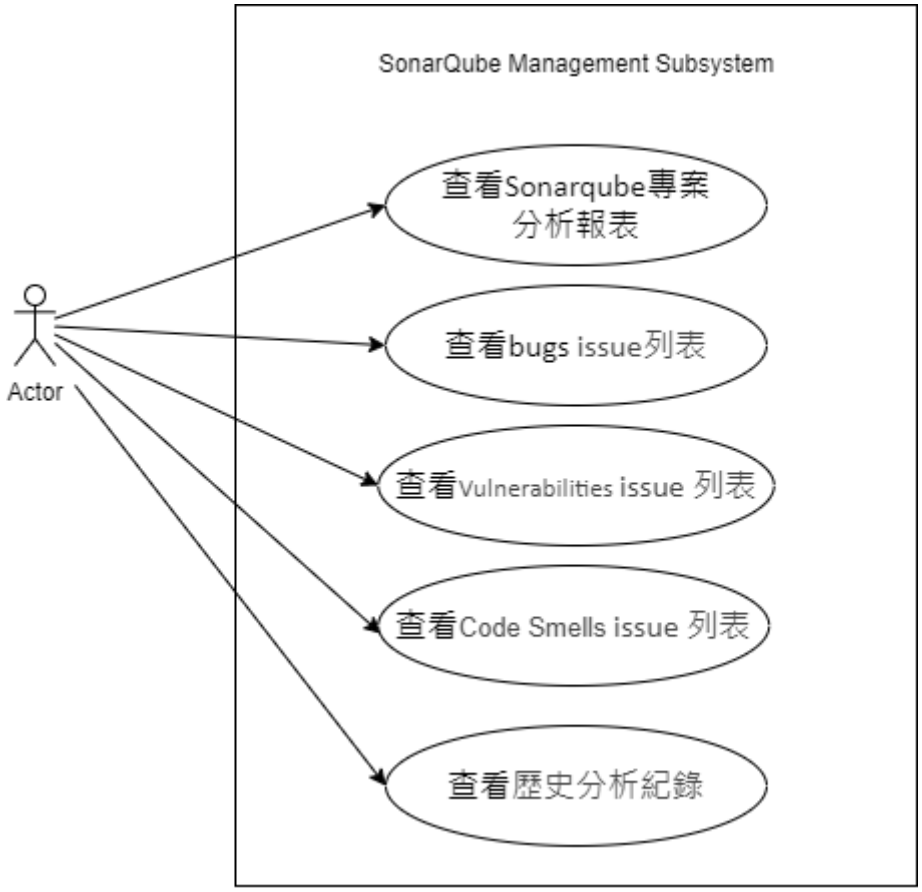
5.1.1 Use Case Model

- **User Identity Verification and Management Subsystem**



No	UIVMS-UC01
Use Case	Github登入。
Summary	使用者可以使用github帳號登入。
Actors	All User。
Preconditions	連上 Internet，擁有github帳號。
Description	1. 輸入系統網址。 2. 按下使用github帳號登入的按鈕。 3. 在github登入頁面進行登入。 4. 成功登入。
Extensions	None
Exceptions	None
Postconditions	進入系統首頁。

- **SonarQube Management Subsystem**



No	SQMS-UC01
Use Case	查看Sonarqube專案分析報表。
Summary	查看專案於Sonarqube分析完的結果。
Actors	All User。
Preconditions	使用者必須登入此系統，並且專案已於SonarQube上分析完成。
Description	1. 按下Sonarqube分析的按鈕。 2. 進入Sonarqube分析頁面，並查看專案的bugs、Vulnerabilities、Code Smells
Extensions	None
Postconditions	顯示專案的bugs、Vulnerabilities、Code Smells。

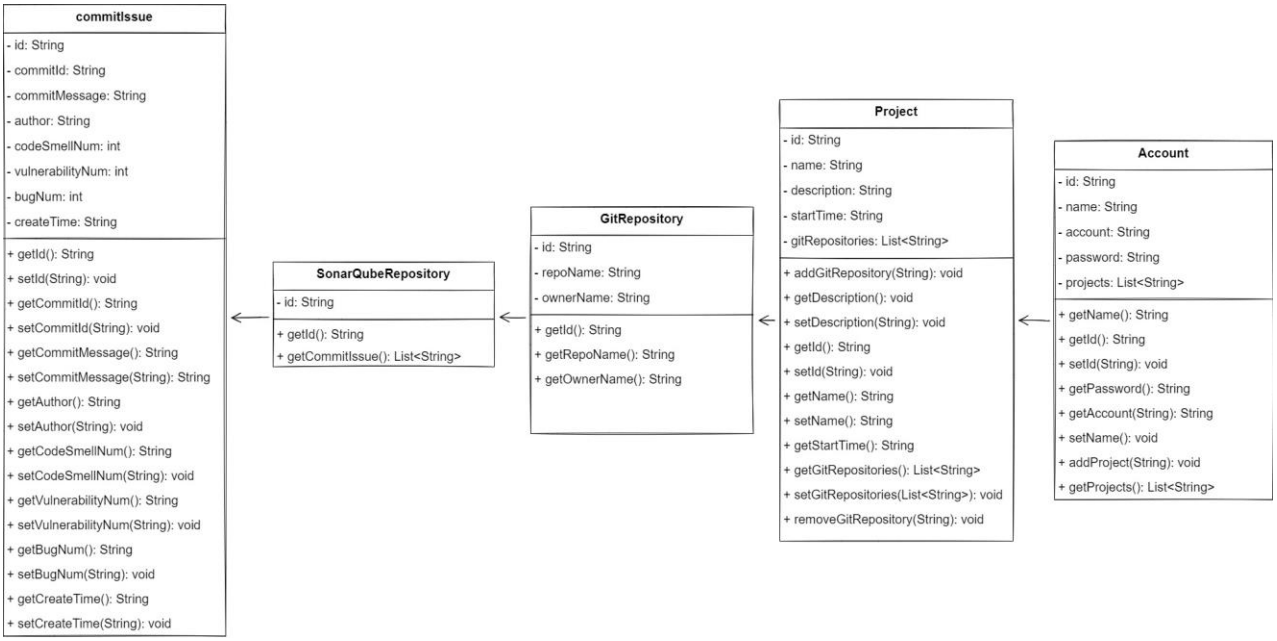
No	SQMS-UC02
Use Case	查看bugs。
Summary	查看專案bugs。
Actors	All User。
Preconditions	使用者必須登入此系統，並且專案已於SonarQube上分析完成。
Description	1. 按下SonarQube的按鈕。 2. 進入SonarQubeOverview頁面，並查看專案的bug issue列表。
Extensions	None
Postconditions	顯示專案的bug issue列表。

No	SQMS-UC03
Use Case	查看Vulnerabilities。
Summary	查看專案Vulnerabilities。
Actors	All User。
Preconditions	使用者必須登入此系統，並且專案已於SonarQube上分析完成。
Description	1. 按下SonarQube的按鈕。 2. 進入SonarQubeOverview頁面，並查看專案的Vulnerabilities issue列表。
Extensions	None
Postconditions	顯示專案的Vulnerabilities issue列表。

No	SQMS-UC04
Use Case	查看Code Smells。
Summary	查看專案Code Smells。
Actors	All User。
Preconditions	使用者必須登入此系統，並且專案已於SonarQube上分析完成。
Description	1. 按下SonarQube按鈕。 2. 進入SonarQubeOverview頁面，並查看專案的Code Smells issue列表。
Extensions	None
Postconditions	顯示專案的Code Smells issue列表。

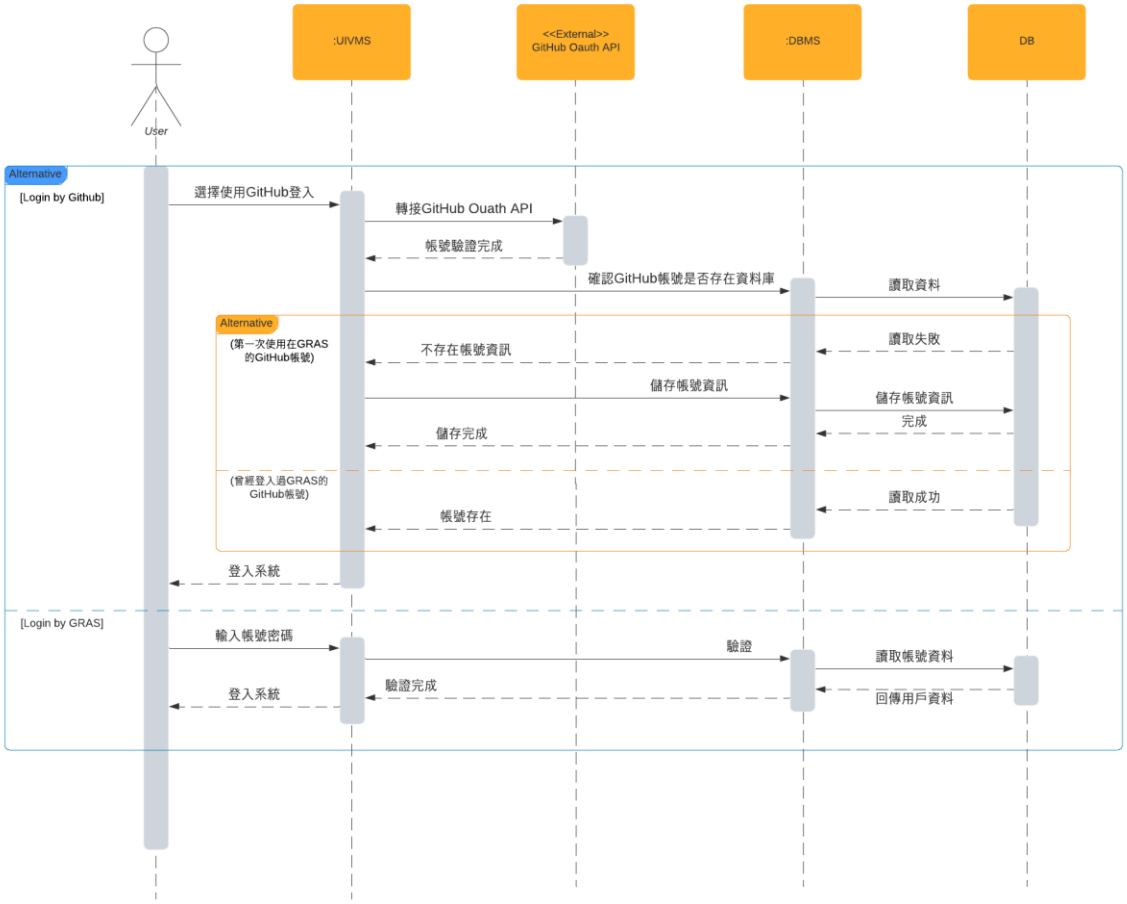
No	SQMS-UC05
Use Case	查看歷史分析紀錄。
Summary	查看專案歷史分析紀錄。
Actors	All User。
Preconditions	使用者必須登入此系統，並且專案已於SonarQube上分析完成。
Description	1. 按下SonarQube按鈕。 2. 進入SonarQubeOverview頁面，按下History按鈕。 3. 進入SonarQubeHistory頁面，並查看專案的歷史分析紀錄。
Extensions	None
Postconditions	顯示專案的歷史分析紀錄。。

5.1.2 Static Models (Structural Models)

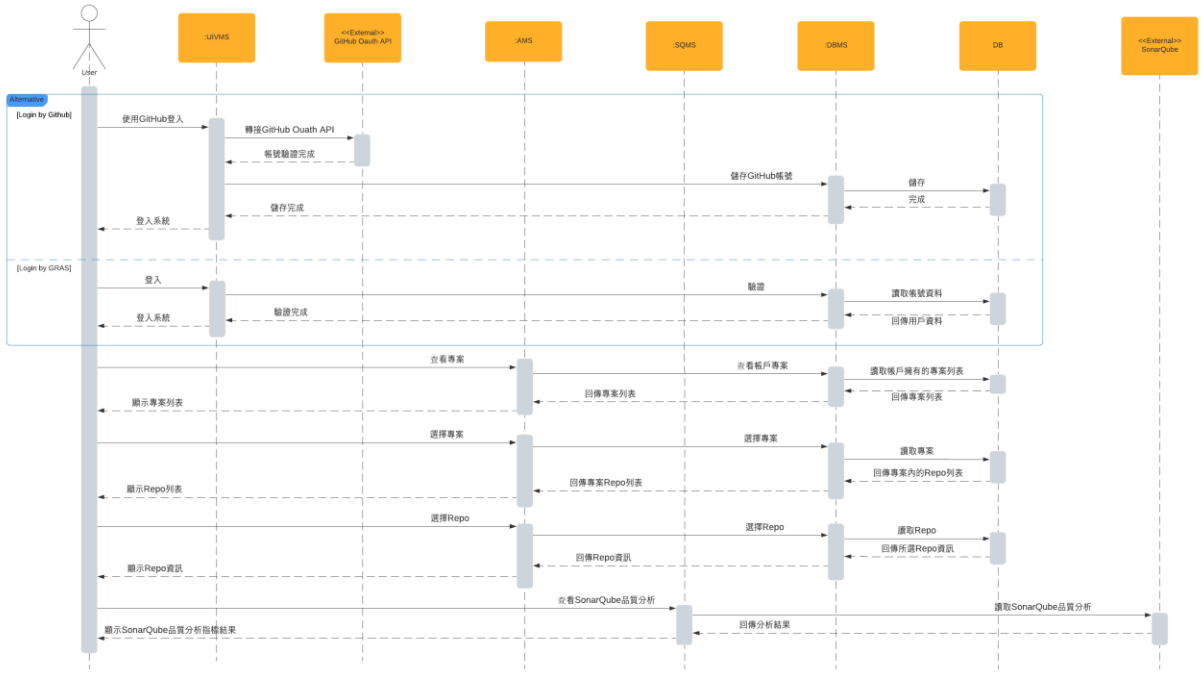


5.1.3 Dynamic Models (Behavioral Models)

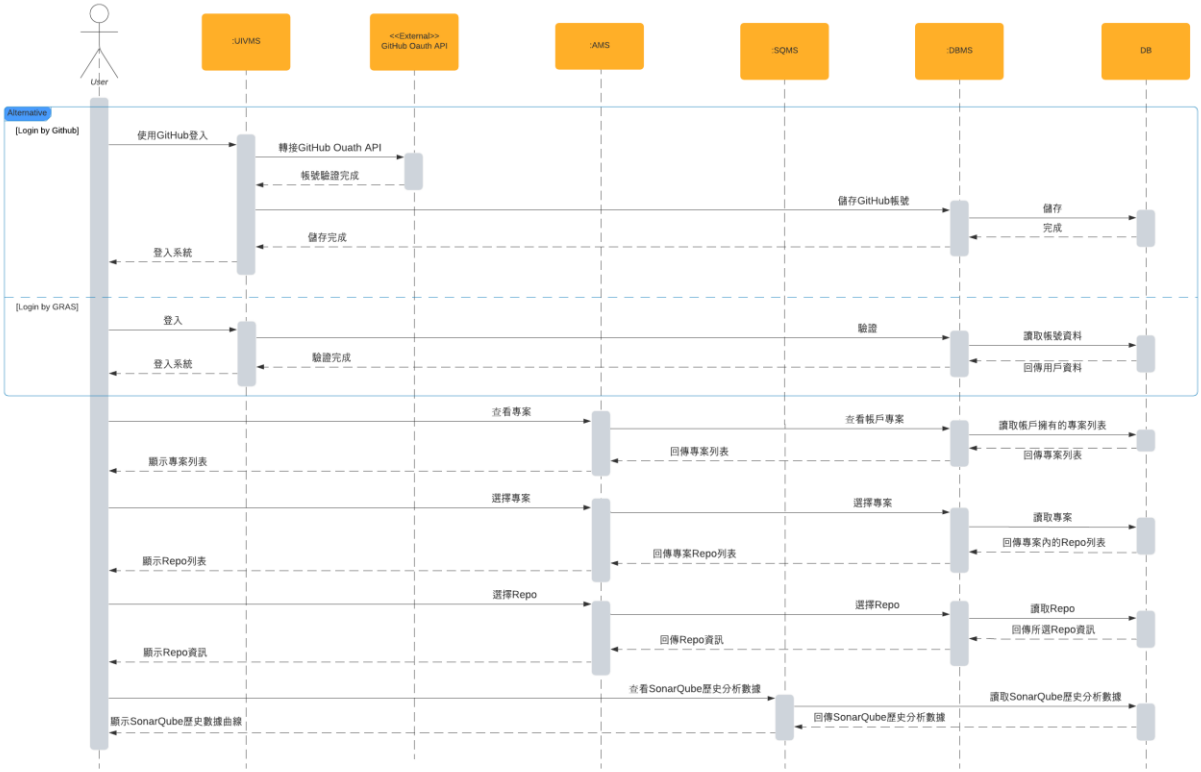
5.1.3.1　使用者選用 github 登入：



5.1.3.2 查看 sonarqube 品質分析：



5.1.3.3 查看 sonarqube 分析歷史紀錄



5.2 Interface Design

5.2.1 API Design

API Name	Method	Request param	Response
login	POST	account:string password:string	isSuccess:bool
signUp	POST	username: string account: string password: string	isSuccess:bool
addGitRepository	POST	repoOwnerName:string repoName:string githubUrl:string	isSuccess:bool
commit	POST	owner:string repo:string	weeks_stats:<list> user_name:string total_deletions:int total_additions:int total_commits:int
createProject	POST	projectName:string projectDescription:string	userId:string
deleteProject	POST	userId:string projectId:string	isSuccess:bool
getProjectGitRepositories	POST	projectId:string	ownerName:string repoName:string
getUserProject	POST	userId:string	projectId:string projectName:string projectDescription:string projectStartTime:string gitRepoCount:int
issueWithQuery	POST	owner:string repo:string	issues_info:<list>
verifyUrl	POST	githubUrl:string	isUrlVaild: bool
repoInfo	POST	repoId:string	description:string contributorCount:string
Bug	POST	Component:string	Count:int
CodeSmell	POST	Component:string	Count:int
Vulnerability	POST	Component:string	Count:int
SonarIssue	POST	Component:string	bugInfo:<list> codesmellInfo:<list> vulnerabilitiesInfo:<list>

SonarHistory	POST	Component:string	historyInfo:<list>

5.2.2 Internal Interface Design

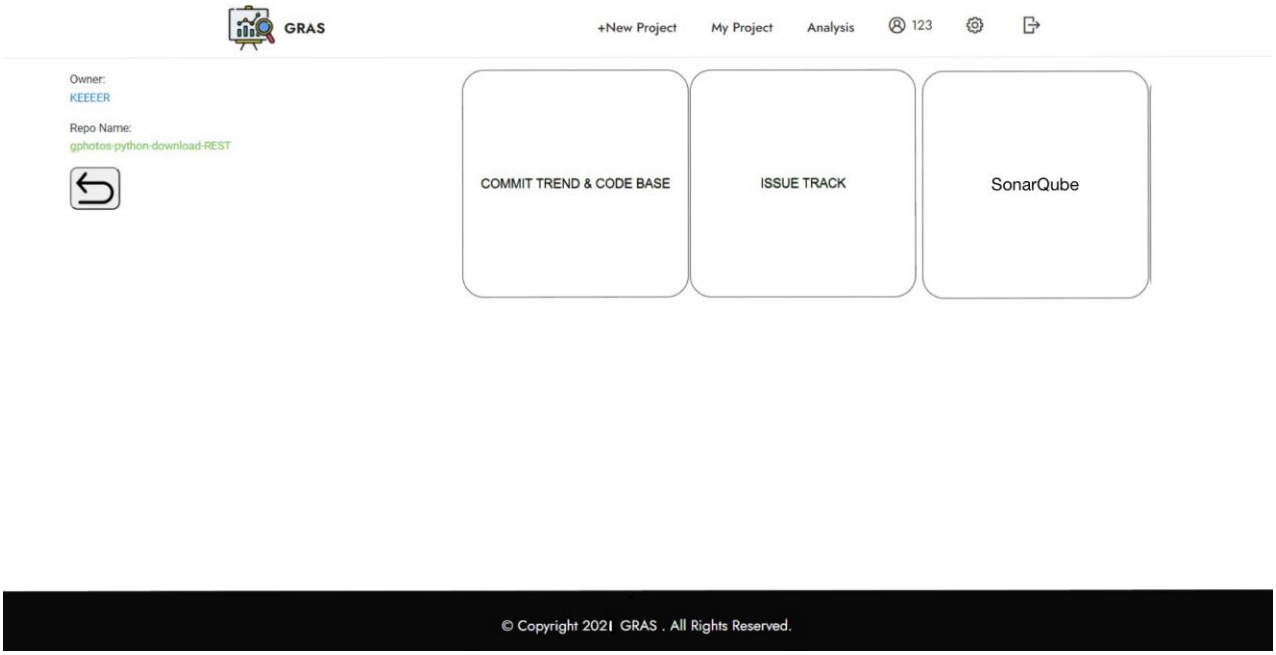
Name	Method	param
addProject	add	id:string
addGitRepository	add	id:string

5.2.3 External Interface Design

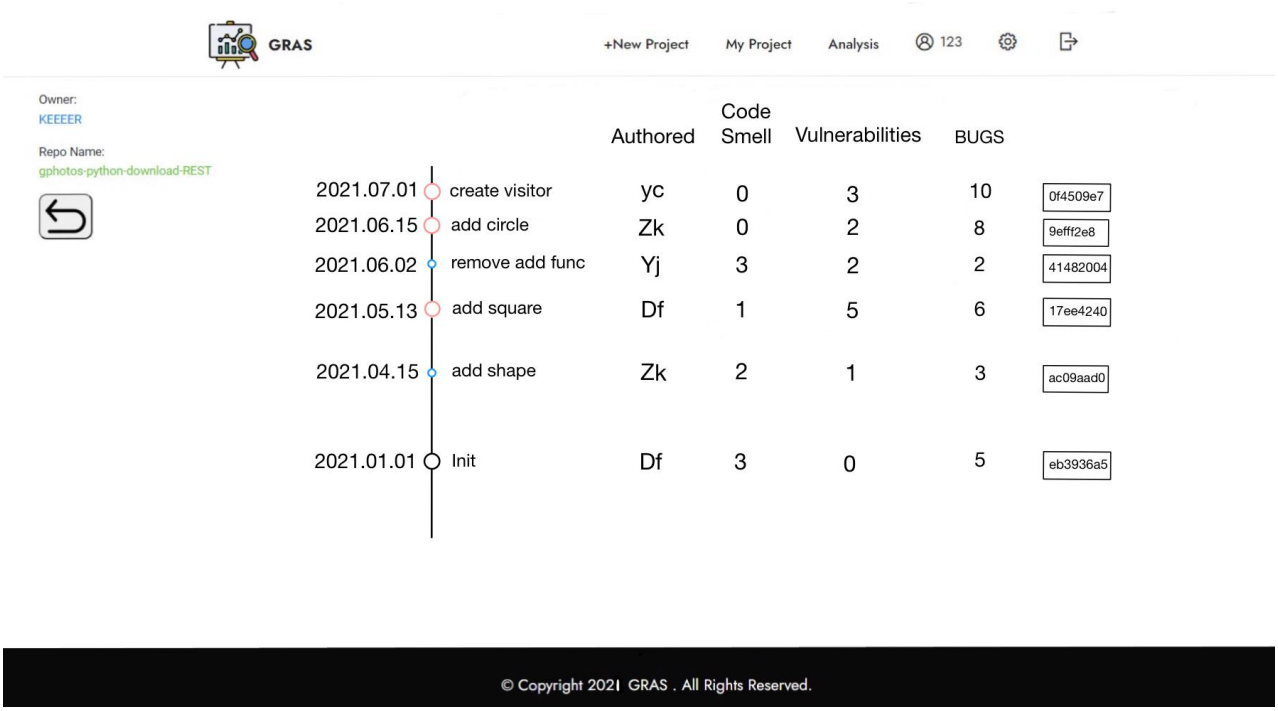
API Name	Method	Request param	Response
api/components/search	GET	component:string	componentList : list
Api/issues/search	GET	componentKeys:string	componentList : list

5.2.4 User Interface Design

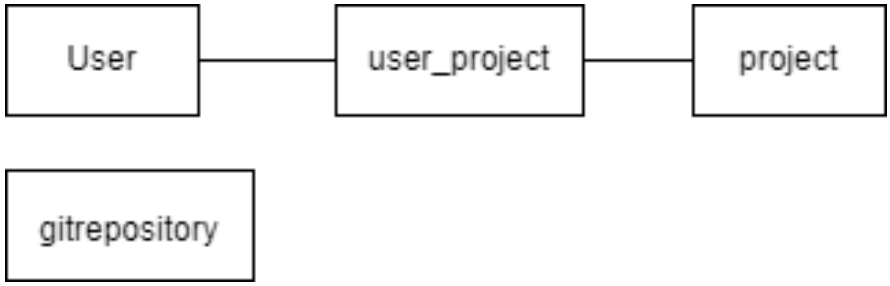
5.2.4.1 增加進入 SQMS 的按鈕



5.2.4.2 SQMS 分析紀錄



5.3 Database Design



5.4 Traceability Matrix – Requirements vs Components

	UIVMS	RMS	SQMS	DBMS
GRAS-U-01	●			
GRAS-U-02			●	
GRAS-U-03			●	
GRAS-U-04			●	
GRAS-NF-01		●		
GRAS-NF-02	●			
GRAS-NF-03				●
GRAS-NF-04		●		

Glossary

References

[1] Erich Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

Appendices

A. Tracebility Matrix (Use Cases v.s. Classes or User Stories v.s. Classes)

B. Tracebility Matrix (Classes v.s. Classes)