

# **Github 專案分析系統**

## **Software Design Document (SDD)**

**Version: 1.2**

**TEAM #9**

<b>Name</b>	<b>ID</b>	<b>E-mail</b>
王泓翔	108598076	t108598076@ntut.org.tw
林育德	109598073	t109598073@ntut.org.tw
諸政安	109598087	t109598087@ntut.org.tw
溫志嘉	109598037	t109598037@ntut.org.tw

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

**12/28/2020**

## Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>Section 0. 版次變更紀錄 (Change Log) .....</b>	<b>3</b>
0.1 Decomposition of the Functional Requirement to Subsystem Requirement and Interfaces.....	4
1.1.1 Functional Requirement.....	4
GRAS 的 Functional Requirement 如下: .....	4
1.1.2 System Interfaces.....	5
1.2 Establish Technical Solution Criteria.....	5
1.3 Describe Alternative Solution.....	5
<b>Section 2. Design Issues and Solutions .....</b>	<b>7</b>
2.1 User Identity Verification and Management Subsystem.....	7
2.1.1 Subsystem Characteristics .....	7
2.1.2 Establish Technical Solution Criteria.....	7
2.1.3 Selected Subsystem Solution.....	7
2.1.4 Error Detection and Recovery .....	8
2.2.1 Subsystem Characteristics:.....	8
利用資料庫進行專案編輯子系統描述 (Project Management Subsystem Description)，進行專案分析資訊之功能為從資料庫中讀取 Repository 資料，並用這些資料來製作相關專案圖表分析。 .....	8
專案分析資訊圖標功能包含:時間軸 codebase 顯示功能、Repo 個別成員 commit 顯示功能、Repo 開發時間軸顯示功能、小組一周內時段 Commit 狀況顯示功能、Repo issue 顯示功能。 .....	8
2.2.2 Establish Technical Solution Criteria.....	9
2.2.3 Selected Subsystem Solution.....	9
2.2.4 Error Detection and Recovery .....	9
2.3.1 Subsystem Characteristics .....	10
2.3.2 Establish Technical Solution Criteria.....	10
2.3.3 Selected Subsystem Solution.....	10
2.3.4 Error Detection and Recovery .....	11
<b>3 Detailed of Subsystem and Interface Description.....</b>	<b>12</b>
3.1.1 Use Cases Analysis.....	12
3.1.2 User Interfaces Analysis 使用者介面的部分透過 Prototype 工具 AdobeXD 進行了設計，頁面的邏輯實際開發時會延續對應的設計邏輯，以下為使用者會使用到的頁面: 1. 使用者登入介面(prototype) .....	23
2. 使用者註冊介面(prototype).....	23

3. 使用者功能-專案總覽頁面 .....	24
5. 使用者功能-選取欲分析頁面(比較多個 Repo 為 Optional 功能).....	25
7. 使用者功能-查看專案團隊 CommitTrend 分析頁面(查看 codebase 頁面為相同顯示邏輯).....	26
.....	26
8. 使用者功能-查看專案個人 CommitTrend 分析頁面(查看 codebase 頁面為相同顯示邏輯).....	26
.....	26
9. 使用者功能-查看專案 ISSUE 頁面，並根據 label Filter 查看關注的 Issue..	27
10. 使用者功能-查看 Commit 頁面，並根據 label Filter 查看關注的 Commit...	27
.....	27
3.1.3 Static Model.....	27
3.1.4 Dynamic Models.....	32
<b>Glossary .....</b>	<b>38</b>

## Section 0. 版次變更紀錄 (Change Log)

### Revisions

Verison	Primary Author(s)	Description of Version	Date Completed
1.0	王泓翔、諸政安、林育德、溫志嘉	SDD 規劃討論、大綱完成	2020/11/25
1.1	王泓翔、諸政安、林育德、溫志嘉	SDD 完成。	2020/12/08
1.2	王泓翔、諸政安、林育德、溫志嘉	加入 Increment2 調整內容，錯字修改	2020/12/28

## Section 1. Introduction

### 0.1 Decomposition of the Functional Requirement to Subsystem Requirement and Interfaces

#### 1.1.1 Functional Requirement

Github 專案分析系統(GitHub Repository Analysis System, GRAS)提供團隊使用者一個可以查看及分析專案狀態的管理平台，透過將專案的各類活動指標進行視覺化，方便團隊間快速了解專案進度，並且保管這些歷程以供團隊後續回顧的功能。

此系統包含三個子系統：

1. 身分驗證子系統(User Identity Verification and Management Subsystem, UIVMS)，負責帳號管理以及登入方式管理的相關流程。
2. 專案管理子系統(Repository Management Subsystem, RMS)，負責 Repository 的導入以及資訊分析功能。
3. 資料庫管理子系統(Database Management Subsystem, DBMS)，負責處理資料、程式的存取。

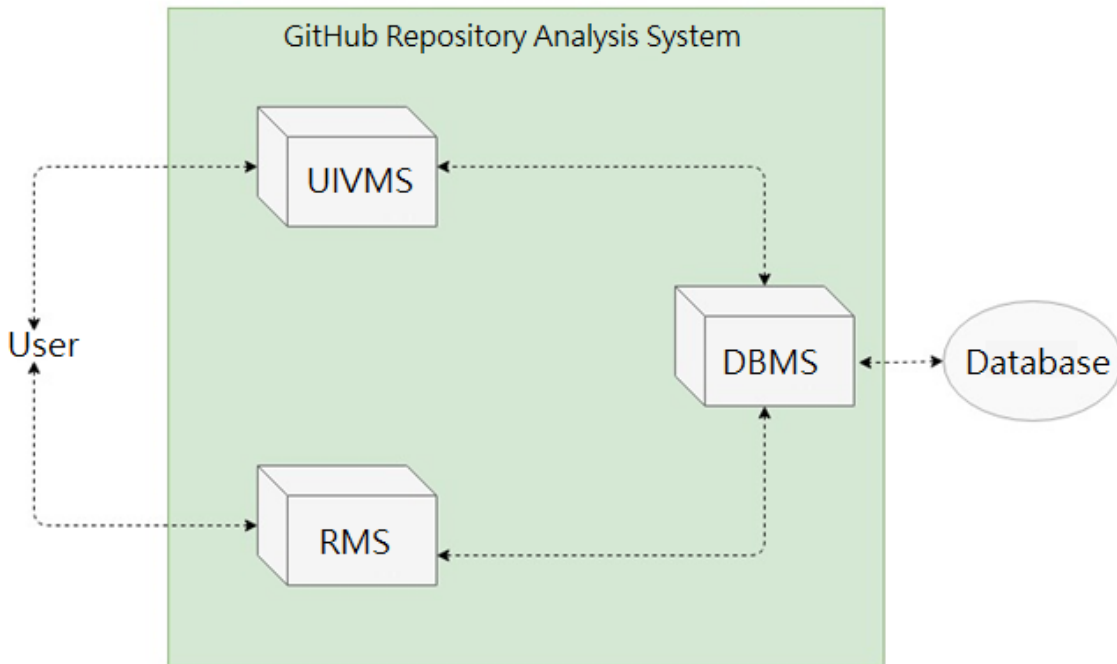
透過整合這三個系統以完成 GRAS 的系統開發。

GRAS 的 Functional Requirement 如下：

需求編號	優先順序	需求描述
GRAS-F-01	1	提供使用者驗證功能。
GRAS-F-02	1	提供專案各項指標分析
GRAS-F-03	2	提供不同專案比較功能
UIVMS-F-01	1	提供 Administrator 建立帳號功能。
UIVMS-F-02	1	提供使用者登入，進行身份辨識。
UIVMS-F-03	1	提供 Administrator 帳號管理功能，包括帳號的新增、修改、刪除和查詢。
UIVMS-F-04	1	使用者可以設定/修改自己的個人資料。
UIVMS-F-05	1	設定帳戶權限。
RMS-F-01	1	提供專案分析資訊-時間軸 codebase 顯示功能。
RMS-F-02	1	提供專案分析資訊-Repo 個別成員 commit 顯示功能。
RMS-F-03	1	提供專案分析資訊-Repo 開發時間軸顯示功能。
RMS-F-04	1	提供專案分析資訊-小組一周內時段 Commit 狀況顯示功能。
RMS-F-05	1	提供專案分析資訊-Repo issue 顯示功能。
DBMS-F-01	1	提供 UIVMS 使用者帳號及基本資料的新增、修改、刪除、查詢等功能。
DBMS-F-02	1	提供 RMS 對專案資訊的新增、修改、刪除、查詢、產生分析結果等功能。

### 1.1.2 System Interfaces

圖 1-1 為 GRAS 的系統架構圖，說明了 GRAS 與其包含的數個子系統間的互動關係。



主系統分成三個部分，分別為身分驗證子系統(User Identity Verification and Management Subsystem, UIVMS)、專案管理子系統(Repository Management Subsystem, RMS)、資料庫管理子系統(Database Management Subsystem, DBMS)。

### 1.2 Establish Technical Solution Criteria

有關 Solution Criteria 這方面，專案針對 GRAS 擬定了將來會遇到的各種限制，包括：

- 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。
- 擴充性：考慮後續的擴充是否容易。
- 廠商支持程度：考慮是否有廣泛的使用者。
- 維護性：考慮後續的維護是否容易。

上述的限制問題，基本上與系統所用的軟體是否有關。

### 1.3 Describe Alternative Solution

在本系統設計前，要考慮各種可能影響系統架構的因素，包括系統架構、程式設計架構、資料庫軟體系統、與程式語言等的選擇。

(一) 由本專案的系統架構，開發者提供了可行的系統架構，分別為單機系統、主從系統、與網頁系統架構，以下列出與上述限制比較表：

	單機系統	主從系統	網頁系統
易學性	高，開發時間較短	低，開發較長時間	低，開發較長時間
可攜性	低，必需因應每個平台而修改	中，必需額外開放使用	高，使用 Browser 連上即可

安全性	高，單機資料保護較容易	中，有被盜取資料的風險	中，有被盜取資料的風險
擴充性	彈性小，不能完全符合使用者需求	彈性大，可完全符合使用者需求	彈性大，可完全符合使用者需求
維護性	高，由開發者自行維護，且發展也極為不易。安裝較為簡單	高，由開發者自行維護。安裝較為複雜	低，由供應商提供維護作業。安裝容易

(二) 可用來開發本系統的程式語言約有 Python、Java、NodeJS：

Priorities {Scale: 1= Worst, 3= Best, 0= Not necessary}

	Python	Java	NodeJS
易學性	3	2	3
可攜性	2	2	3
安全性	2	2	2
擴充性	2	3	2
維護性	2	3	2

(三) 系統的 User Interface比較表：

	AngularJS	React	Vue. js
易學性	3	2	2
可攜性	3	2	2
安全性	3	2	2
擴充性	3	2	2
維護性	3	2	2

本系統將 JS 來採用網頁系統的架構，並使用 AngularJS開發 GUI 介面。

## Section 2. Design Issues and Solutions

### 2.1 User Identity Verification and Management Subsystem

#### 2.1.1 Subsystem Characteristics

- 帳號管理(Account Management)之功能為建立、修改、刪除所有使用者的登入帳戶。
- 登入管理(Login Management)的主要功能為驗證系統使用者是否已註冊，若是以註冊則已登入。

#### 2.1.2 Establish Technical Solution Criteria

使用者驗證方法比較表：

使用者的驗證機制有「以角色為基礎的存取控制 (Role-based access control, RBAC)」，此部分會串接Github API提供的OAuth2 token和直接實作兩種方法，以下為其特性。

	RBAC	直接實作登入驗證機制
易學性	低，RBAC使用較為複雜	高，直接實作角色權限不需另外學習
可攜性	中，必須處理安置SessionKey的儲存位置	高，沒有需要額外負擔的處理
安全性	高，RDBC控制權是直接賦予角色較安全	低，直接實作控制權限，漏洞很多
擴充性	中，必須依靠RBAC內建功能來擴充	高，不足部份直接實作即可
廠商支持程度	高，由美國國家標準局提出的一種新的存取控制機制	中，自行撰寫的權限控制方法
維護性	高，RDBC的角色權限較易維護	中，定義一個新的角色必須撰寫新的權限功能

#### 2.1.3 Selected Subsystem Solution

使用者驗證方法選擇：

	RBAC	直接實作登入驗證機制
易學性	2	4
可攜性	3	5
安全性	5	2
擴充性	3	5
廠商支持程度	4	3
維護性	4	3
加總	21	22

Priorities {Scale : 1 = worst (difficult), 5 = best (easy) }



由於專案開發時間較短，而整體功能都是與 GitHub API 相關的分析，因此我們會先採用較安全且功能完善的 Role-based access control 方法當作 KBS 的登入、權限驗證機制。

#### 2.1.4 Error Detection and Recovery

以下針對第一次與第二次 Increment/里程碑為使用 Github OAuth2 token，帳密輸入錯誤的提示交由 Github API 來進行提示，其餘偵測的對應錯誤與提示如下。

Error	Detection & Recovery
使用者Github帳戶登入授權失敗	跳出警告標語，並提示使用者請再次登入。
使用者輸入資料錯誤(EX: 導入專案網址無法辨認)	於使用者輸入資料錯誤的項目顯示訊息以供更正。
使用者帳戶登入授權失敗	跳出警告標語，並提示使用者請再次登入。
資料不一致(資料庫端)	本系統的主程式將於使用者新增資料的時候自動檢查 (1)帳號資料與群組資料的一致性。 (2)專案資料與開發人員帳號的一致性。 當系統偵測出不一致性時，會顯示相關訊息，且使用者無法新增任何資料。
資料不一致(使用者輸入端)	於前端的資料送至伺服端的應用程式時，再進行檢查。當子系統一偵測出不一致性時，相關的訊息將由本子系統由前端通知使用者。
資料庫損毀	資料庫的偵錯：使資料庫具偵錯機制，並定期產生LOG檔，供系統管理員查核。
	資料庫的備份：定期備份所有資料，確認資料庫安全不被入侵。
	資料庫的復原：系統發生問題、資料損毀時，可由之前備份的資料進行還原修復。

## 2.2 Repository Management Subsystem

### 2.2.1 Subsystem Characteristics:

利用資料庫進行專案編輯子系統描述 (Project Management Subsystem Description)，進行專案分析資訊之功能為從資料庫中讀取 Repository 資料，並用這些資料來製作相關專案圖表分析。

專案分析資訊圖標功能包含:時間軸 codebase 顯示功能、Repo 個別成員 commit 顯示功能、Repo 開發時間軸顯示功能、小組一周內時段 Commit 狀況顯示功能、Repo issue 顯示功能。

### 2.2.2 Establish Technical Solution Criteria

前端的產生可產生使用(HTML+JS+CSS)語法或圖片

	HTML+JS+CSS	產出固定圖片
易學性	中，需控制HTML+JS+CSS	低，必須學習使用撰寫產生圖片報表的Library
可攜性	中，可能因為Browser不同，而使表格變形。	高，圖片不會因Browser而改變
安全性	無，無安全性考量	無，無安全性考量
擴充性	高，可根據不同指標修改對應顯示圖表	低，使用者修改報表困難
廠商支持程度	高，較容易配合HTML表格	中，較不易配合圖片
維護性	高，修改HTML表格較容易	中，必須重新配置圖片資料

### 2.2.3 Selected Subsystem Solution

前端呈現的選擇：

	HTML+JS+CSS	圖片
易學性	4	1
可攜性	3	4
擴充性	5	3
廠商支持程度	5	4
加總	17	12

Priorities {Scale: 1= worst(difficult), 5= best(easy), 0= Not necessary}

根據上表評估，決定採用 HTML+JS+CSS 作為 前端報表呈現方式。

### 2.2.4 Error Detection and Recovery

以下針對第一次與第二次 Increment/里程碑所預設完成的核心功能進行設計。

Error	Detection & Recovery
報表產生速度過慢	重新審視子系統中模組的取得資料庫資料的方式並加以改良。
使用者未登入	沒有客戶端的授權，不執行使用者操作提示請登入畫面。
未有導入專案	提示請先導入專案
使用者登入錯誤	提示請輸入正確帳號或密碼

## 2.3 Database Management Subsystem

### 2.3.1 Subsystem Characteristics

DBMS 提供資料存取模組與 API。

- API:提供其他子系統統一接口提出資料存取需求。
- 資料存取模組:針對存取要求對應到存取模組進行處理，並將結果傳回 API。

### 2.3.2 Establish Technical Solution Criteria

資料庫系統儲存的比較表：

目前資料庫常見的有 Microsoft SQL Server 與 MySQL。Microsoft SQL Server 與 MySQL 都屬於較大型的資料庫系統，兩者差別主要是在 MySQL 是免費的，而 Microsoft SQL Server 不是。

	Microsoft SQL Server	MySQL
易學性	中，原來的使用者已有基礎，但仍需再訓練	高，原來的使用者已有基礎
可攜性	低，MS SQL 與.NET 整合，必須使用 .NET FrameWork才能使用到完整的功能	高，為企業用免費的系統，容易安裝
安全性	高，MS會定期提供安全性更新	中，My SQL安全性目前無重大漏失
擴充性	高，資料量大，其SQL Server 在重負載下表現特別突出	中，儲存量適於中小型的資料量儲存
廠商支持程度	低，需付費	高，免費

### 2.3.3 Selected Subsystem Solution

使用者驗證方法選擇：

資料庫系統儲存的選擇：

	Microsoft SQL Server	MySQL
易學性	2	4
可攜性	4	3
安全性	4	3
擴充性	4	3

廠商支持程度	1	5
加總	15	18

Priorities {Scale : 1 = worst (difficult), 5 = best (easy) }

根據上表評估，決定採用 My SQL 作為 DBMS 的儲存資料庫。

### 2.3.4 Error Detection and Recovery

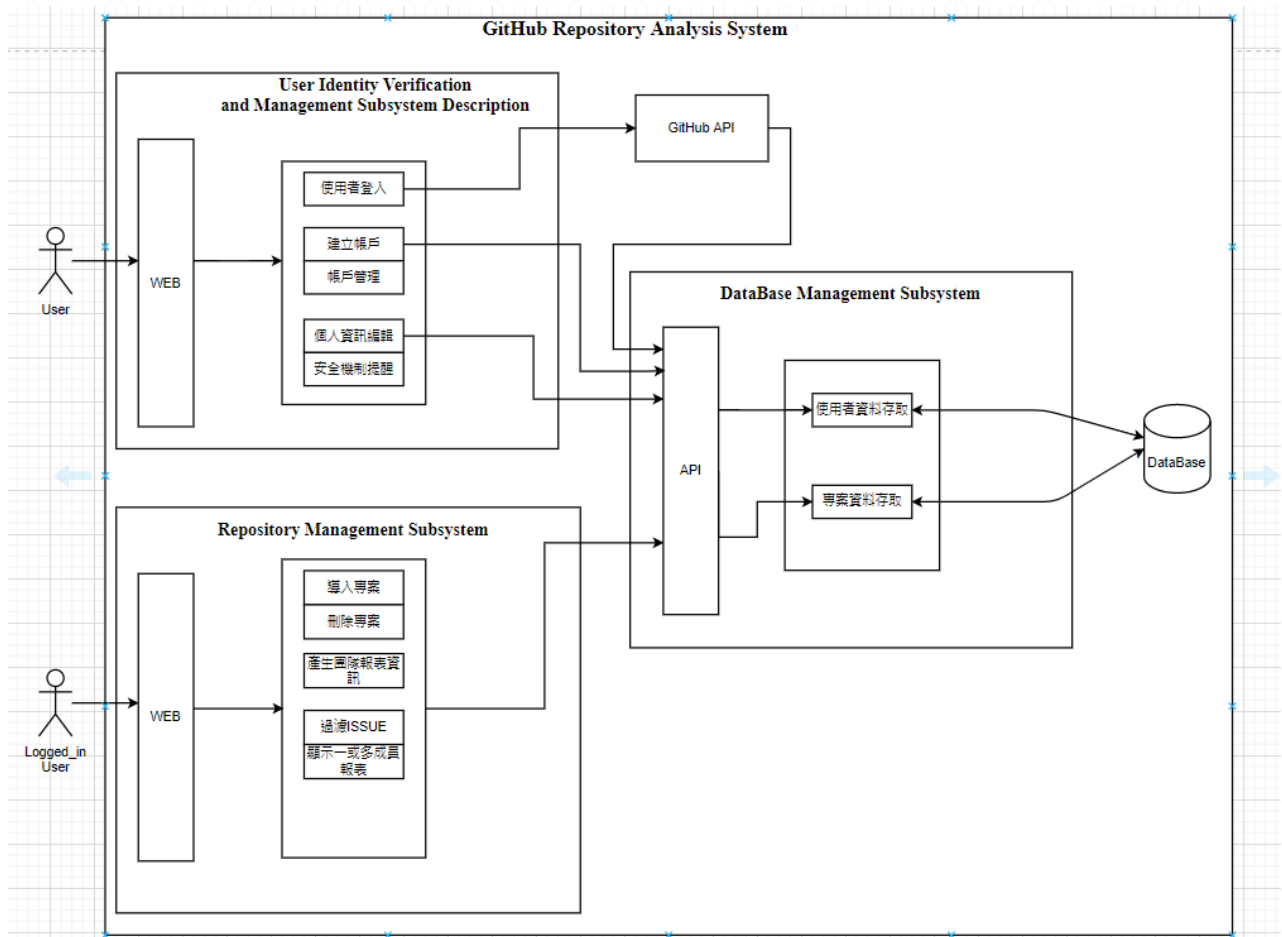
以下針對第一次 Increment/里程碑為使用 My SQL 作為 DBMS，對應錯誤與提示如下。

Error	Detection & Recovery
存取導入專案失敗	提示使用者所導入專案檔案有異，標示Repo異常。
操作或修改超過期限的專案	顯示警告視窗提醒使用者專案已過期
資料庫損毀	資料庫偵錯機制:定期產生log檔供系統管理員查詢
	資料庫備份:定期備份資料庫資料
	資料庫復原發生問題及異常時，使用之前備份的資料庫資料復原資料庫

### 3 Detailed of Subsystem and Interface Description

#### 3.1 Detailed System

本系統因為其特性所以採用網頁系統的架構來實作本系統，並且使用 UML Diagram 來描述其中的細部設計。

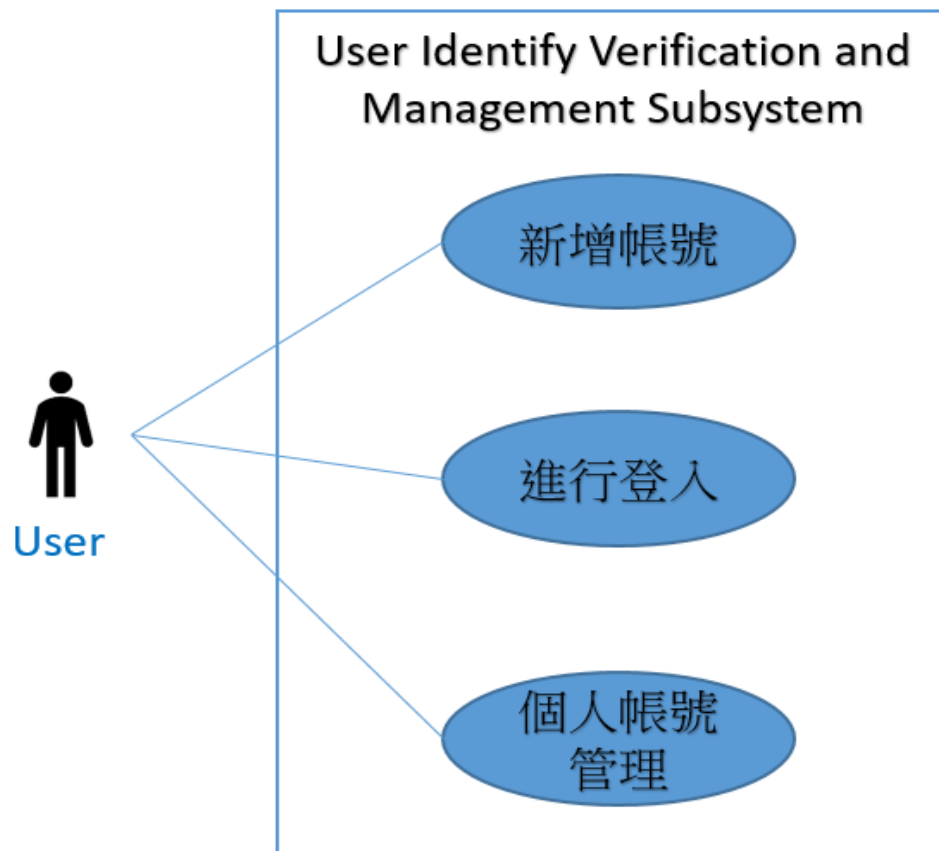


操作流程示意圖

##### 3.1.1 Use Cases Analysis

在這小節中我們將以 Use case 來描述此子系統中各個元件的使用流程和反應。

- **User Identity Verification and Management Subsystem**

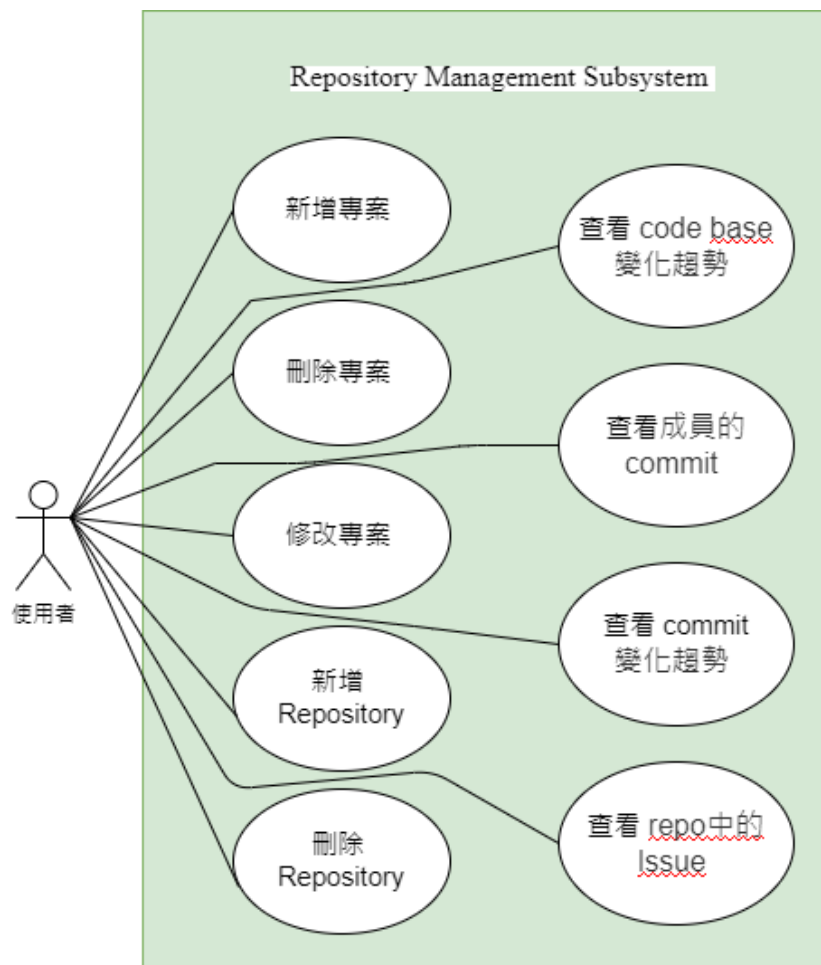


<b>No</b>	UIVMS-UC01
<b>Use Case</b>	新增帳號。
<b>Summary</b>	使用者可以創建帳號。
<b>Actors</b>	All User。
<b>Preconditions</b>	連上 Internet。
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 輸入系統網址。</li> <li>2. 進入系統申請帳號頁面。</li> <li>3. 輸入姓名、帳號、密碼等個人資訊。</li> <li>4. 點選"建立帳號"按鈕。</li> <li>5. 成功創建帳號。</li> </ol>
<b>Extensions</b>	None
<b>Exceptions</b>	<ol style="list-style-type: none"> <li>1. 系統中帳號已存在。</li> <li>2. 密碼與確認密碼不一致。</li> </ol>
<b>Postconditions</b>	進入系統首頁。

No	UIVMS-UC02
Use Case	進行登入。
Summary	使用者可以登入系統。
Actors	All User。
Preconditions	連上 Internet。
Description	<ol style="list-style-type: none"> <li>1. 輸入系統網址。</li> <li>2. 進入系統登入頁面。</li> <li>3. 輸入帳號。</li> <li>4. 輸入密碼。</li> <li>5. 點選登入。</li> <li>6. 成功登入系統。</li> </ol>
Extensions	None。
Exceptions	<ol style="list-style-type: none"> <li>1. 帳號不存在。</li> <li>2. 密碼錯誤。</li> </ol>
Postconditions	進入系統首頁。

No	UIVMS-UC03
Use Case	個人帳號管理。
Summary	使用者可以修改本身的帳號資料。
Actors	擁有帳號之使用者。
Preconditions	UIVMS-UC02
Description	<ol style="list-style-type: none"> <li>1. 進入帳號管理頁面。</li> <li>2. 點選修改個人資料。</li> <li>3. 進入修改帳號資料頁面。</li> <li>4. 輸入帳號密碼。</li> <li>5. 進行個人資料修改。</li> <li>6. 點選完成。</li> <li>7. 個人資料修改完成。</li> </ol>
Extensions	None。
Exceptions	None。
Postconditions	回到帳號管理頁面。

● Repository Management Subsystem



<b>No</b>	RMS-UC-01
<b>Use Case</b>	新增專案
<b>Summary</b>	新增一個新的專案
<b>Actors</b>	專案管理者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選"新增專案"按鈕</li> <li>2. 使用者輸入欲新增的專案名稱、專案敘述、欲加入的 GitHub Repository 網址</li> <li>3. 使用者點選"建立專案"按鈕</li> <li>4. 將欲新增之專案資訊更新至資料庫</li> </ol>



	5. 系統回到專案列表
<b>Extensions</b>	5a. 自動更新專案列表頁面
<b>Exceptions</b>	2a. 使用者未填入專案名稱之必要條件 2b. 使用者輸入之名稱與現有專案同名 將該填入之資訊填入並且未與現有專案同名 4a. 更新至資料庫失敗 提示使用者更新失敗 要求使用者重新新增資料
<b>Postconditions</b>	專案資料庫上新增一筆專案資訊

<b>No</b>	RMS -UC-02
<b>Use Case</b>	刪除專案
<b>Summary</b>	刪除一個存在之專案
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表上已有專案存在
<b>Description</b>	1. 使用者點選"刪除專案"按鈕 2. 使用者點選欲刪除之專案 3. 使用者點選"確定"按鈕 4. 將欲刪除之專案資訊更新至資料庫 5. 系統回到專案列表畫面，並顯示刪除專案後之專案列表
<b>Extensions</b>	1a. 列出目前所有專案列表 5a. 自動更新專案列表頁面

<b>Exceptions</b>	4a 更新至資料庫失敗 提示使用者更新失敗 要求使用者重新新增資料
<b>Postconditions</b>	專案資料庫上刪除一筆專案資訊

<b>No</b>	RMS -UC-03
<b>Use Case</b>	修改專案
<b>Summary</b>	修改一個存在之專案
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在
<b>Description</b>	1.使用者點選"修改專案"列表按鈕 2.使用者點選欲修改之專案 3.使用者對專案名稱、專案敘述等欄位進行修改 4.使用者點選"確定"按鈕 5.將欲修改之專案資訊更新至資料庫 6.系統回到專案列表畫面，並顯示修改專案後之專案列表
<b>Extensions</b>	1a. 列出目前所有專案列表 6a. 自動更新專案列表頁面
<b>Exceptions</b>	3a 使用者填入空白至欄位中 3b 使用者修改之名稱與現有專案同名 檢查欄位是否填入空白並且未與現有專案同名  5a 更新至資料庫失敗 提示使用者更新失敗 要求使用者重新新增資料

<b>Postconditions</b>	專案資料庫上修改一筆任務資料
-----------------------	----------------

<b>No</b>	RMS -UC-04
<b>Use Case</b>	新增 Repository
<b>Summary</b>	新增一個新的 Repository
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選"新增 Repository "按鈕</li> <li>2. 使用者輸入欲加入的 GitHub Repository 網址</li> <li>3. 使用者點選"建立 Repository "按鈕</li> <li>4. 將欲新增之 Repository 資訊更新至資料庫</li> <li>5. 系統回到 Repository 列表，並顯示已新增之 Repository 資訊</li> </ol>
<b>Extensions</b>	5a. 自動更新 Repository 列表頁面
<b>Exceptions</b>	4a. 更新至資料庫失敗 提示使用者更新失敗 要求使用者重新新增資料
<b>Postconditions</b>	Repository 資料庫上新增一筆 Repository 資訊

<b>No</b>	RMS -UC-05
<b>Use Case</b>	刪除 Repository
<b>Summary</b>	刪除一個存在之 Repository
<b>Actors</b>	使用者

<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在，並且專案中已有 Repository 存在
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選"刪除 Repository" 按鈕</li> <li>2. 使用者點選欲刪除之 Repository</li> <li>3. 使用者點選"確定"按鈕</li> <li>4. 將欲刪除之 Repository 資訊更新至資料庫</li> <li>5. 系統回到 Repository 列表畫面</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>1a. 列出目前所有 Repository 列表</li> <li>5a. 自動更新 Repository 列表頁面</li> </ol>
<b>Exceptions</b>	<ol style="list-style-type: none"> <li>4a 更新至資料庫失敗 提示使用者更新失敗 要求使用者重新新增資料</li> </ol>
<b>Postconditions</b>	專案資料庫上刪除一筆 Repository 資訊

<b>No</b>	RMS -UC-06
<b>Use Case</b>	查看 code base 變化趨勢
<b>Summary</b>	查看 code base 隨時間變化的趨勢
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在，並且專案中已有 Repository 存在
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選"code base" 按鈕</li> <li>2. 系統進入 code base 頁面，並查看根據時間軸 code base 變化趨勢</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2a. 自動更新 code base 頁面</li> </ol>

<b>Exceptions</b>	2a 從資料庫撈取資料失敗
<b>Postconditions</b>	顯示 code base 變化趨勢

<b>No</b>	RMS -UC-07
<b>Use Case</b>	查看成員的 commit
<b>Summary</b>	查看 repo 中每一個成員的 commit
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在，並且專案中已有 Repository 存在
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選“commit log” 按鈕</li> <li>2. 系統進入 commit log 頁面，並查看 repo 中每一個成員的 commit 時間、commit 數和 commit 行數</li> </ol>
<b>Extensions</b>	2a. 自動更新 commit log 頁面
<b>Exceptions</b>	2a 從資料庫撈取資料失敗
<b>Postconditions</b>	顯示 repo 中每一個成員的 commit

<b>No</b>	RMS -UC-08
-----------	------------

<b>Use Case</b>	查看 commit 變化趨勢
<b>Summary</b>	在時間軸上查看 commit 變化趨勢
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在，並且專案中已有 Repository 存在
<b>Description</b>	<ol style="list-style-type: none"> <li>1. 使用者點選“commit trend” 按鈕</li> <li>2. 系統進入 commit trend 頁面，並顯示成員的 commit 變化趨勢</li> </ol>
<b>Extensions</b>	2a. 自動更新 commit trend 頁面
<b>Exceptions</b>	2a 從資料庫撈取資料失敗
<b>Postconditions</b>	在時間軸上顯示 commit 變化趨勢

<b>No</b>	RMS -UC-09
<b>Use Case</b>	查看 repo 中的 Issue
<b>Summary</b>	查看 repo 中的 Issue 的 Label (Open、assigned、closed)、計算 Lead time、顯示其他細節
<b>Actors</b>	使用者
<b>Preconditions</b>	使用者必須擁有可登入此系統的帳號、密碼並且已登入，並且專案列表已有專案存在，並且專案中已有 Repository 存在

<b>Description</b>	1. 使用者點選"issue track" 按鈕 2. 系統進入 issue track 頁面，並顯示 issue 的 Label、計算 Lead time、顯示其他細節
<b>Extensions</b>	2a. 自動更新 issue track 頁面
<b>Exceptions</b>	2a 從資料庫撈取資料失敗
<b>Postconditions</b>	顯示 repo 中的 Issue

### 3.1.2 User Interfaces Analysis

使用者界面的部分透過 Prototype 工具 AdobeXD 進行了設計，頁面的邏輯實際開發時會延續對應的設計邏輯，以下為使用者會使用到的頁面：

#### 1. 使用者登入介面(prototype)

GitHub專案分析管理系統

[關於我們](#)[如何運用](#)[團隊介紹](#)[開發日志](#)[客服支援](#)

Sign In

Sign Up

## 會員登入

Name

JohnCena@mail.com


Password

\*\*\*\*\*

×

☐ Remember me

LOGIN

 使用GitHub帳號登入

#### 2. 使用者註冊介面(prototype)

GitHub專案分析管理系統

[關於我們](#)[如何運用](#)[團隊介紹](#)[開發日志](#)[客服支援](#)

Sign In

Sign Up

## 建立帳號

Full Name

John Cena

Oralase Email

JohnCena@mail.com

Password

\*\*\*\*\*

Confirm Password

\*\*\*\*\*

Personal Information

Date of Birth

09/03/1997

▼

Country/Region

Taiwan

▼

Agency

NTUT\_SISo




建立帳號




### 3. 使用者功能-專案總覽頁面

GitHub專案分析管理系統

[關於我們](#)[如何使用](#)[團隊介紹](#)[開發日誌](#)[客服支援](#)



 John Cena





**John Cena**  
J.C

**個人資訊**  
Company : WWE  
Job Title : Wrestler  
Location : USA  
self introduction:  
ARE YOU SURE ABOUT THAT?



WWE2020  
(專案內容敘述)


 3  3month

WWE2019  
(專案內容敘述)

 1  1month

WWE2018  
(專案內容敘述)




 1  1hour


 ADD NEW PROJECT

### 4. 使用者功能-專案導入頁面

GitHub專案分析管理系統

[關於我們](#)[如何使用](#)[團隊介紹](#)[開發日誌](#)[客服支援](#)

 John Cena



**John Cena**  
J.C

**個人資訊**  
Company : WWE  
Job Title : Wrestler  
Location : USA  
self introduction:  
ARE YOU SURE ABOUT THAT?


Project Name


Project Description (optional)


All roads lead to Rome.  
All work and no play makes Jack a dull boy.  
A man is known by his friends.  
A miss is as good as a mile.  
A rolling stone gathers no moss.  
A straight foot is not afraid of a crooked shoe.


Add To Project

FROM URL:


GitHub.COM 

GitHub.COM 

GitHub.COM 

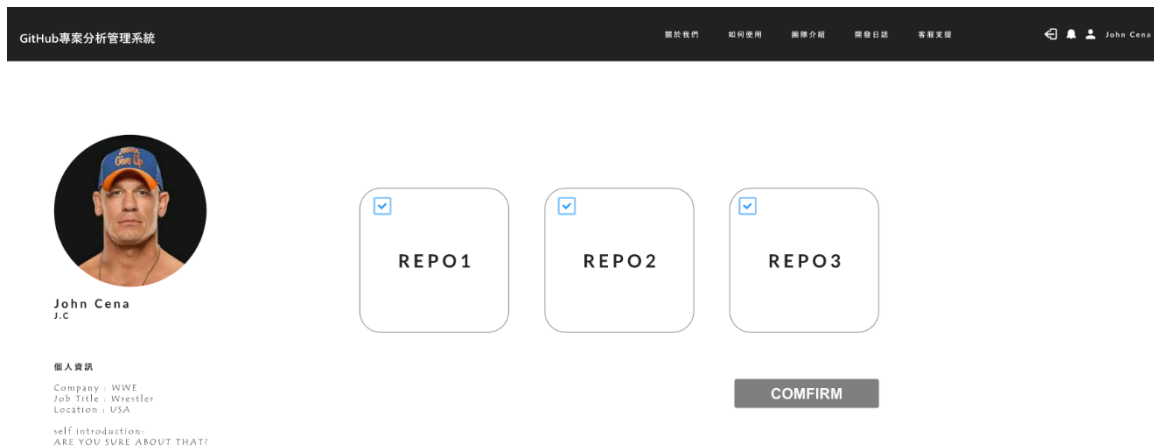


FROM GITHUB  
(Should Link First)

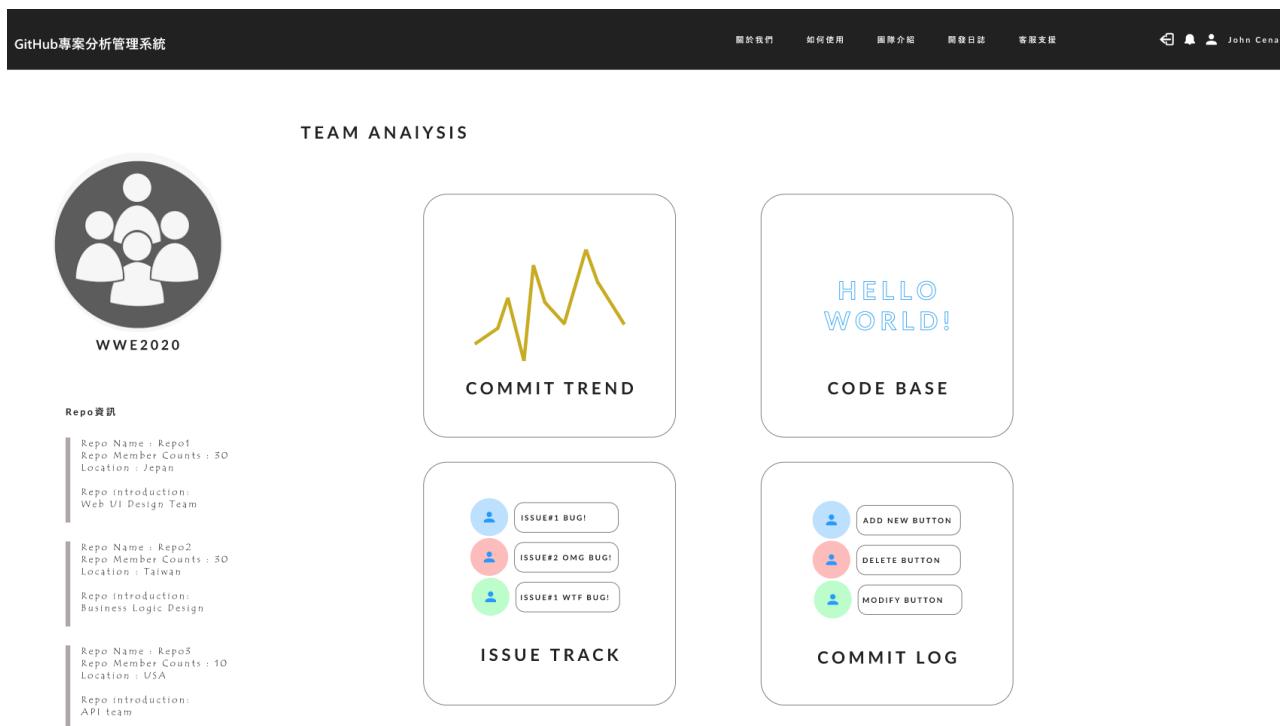
 **GitHub**

24

## 5. 使用者功能-選取欲分析頁面(比較多個 Repo 為 Optional 功能)



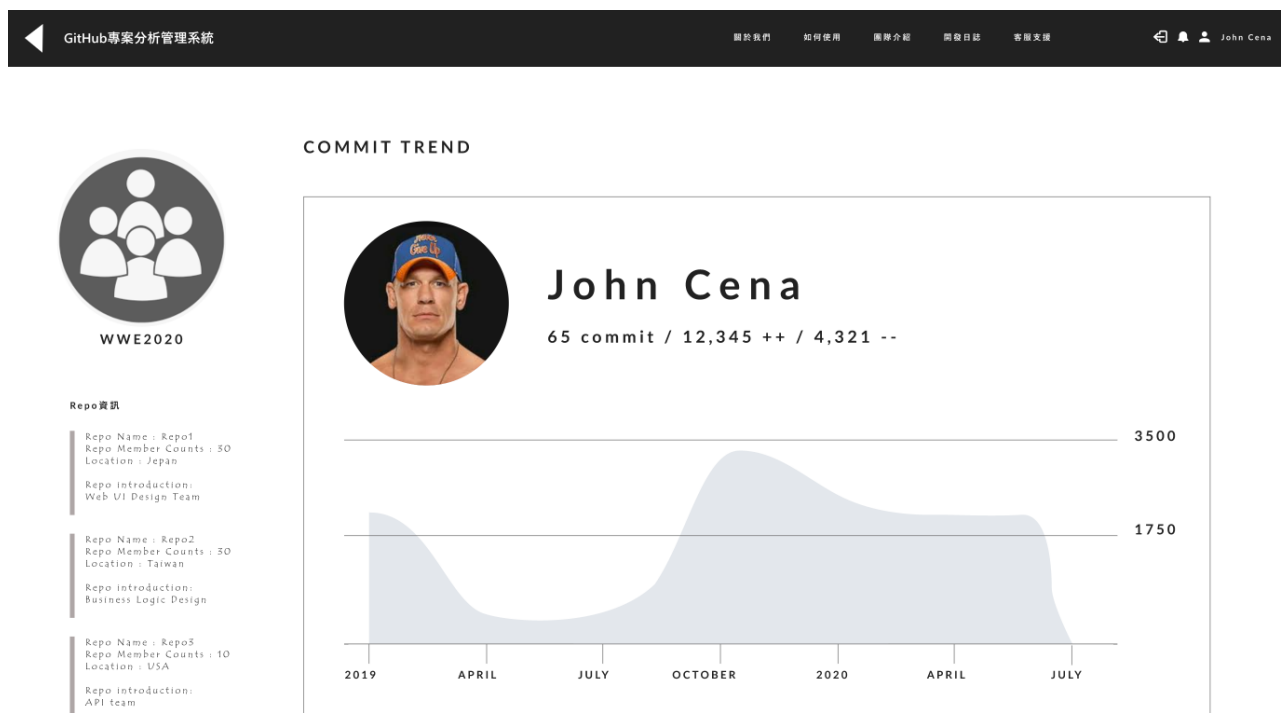
## 6. 使用者功能-專案分析頁面



7. 使用者功能-查看專案團隊 CommitTrend 分析頁面(查看 codebase 頁面為相同顯示邏輯)



8. 使用者功能-查看專案個人 CommitTrend 分析頁面(查看 codebase 頁面為相同顯示邏輯)



## 9. 使用者功能-查看專案 ISSUE 頁面，並根據 label Filter 查看關注的 Issue

The screenshot displays the 'Focusing Issues' section of the GitHub Project Analysis Management System. On the left, a sidebar shows the 'Repo 資訊' (Repo Info) for 'WWE2020', listing three repositories: Repo1 (Japan), Repo2 (Taiwan), and Repo3 (USA). The main content area features a 'Filter' dropdown menu and a list of issues. The 'Focusing Issues' section highlights two issues: 'Air conditioner is broken' by BOB and 'Purchase Function Broken' by Victor. Each issue card includes a description, start date, close date, and type. Below this, the 'Recent Modified Issues' section is partially visible.

**Filter**

**Focusing Issues**

- Air conditioner is broken**  
description: It is SO HOT !!!!!  
Start Date : 2020/01/01  
Close Date :Not Yet  
Type : other problem
- Purchase Function Broken**  
description: fall everytime.  
Start Date : 2020/01/01  
Close Date :Not Yet  
Type : tech problem

**Recent Modified Issues**

## 10. 使用者功能-查看 Commit 頁面，並根據 label Filter 查看關注的 Commit

The screenshot displays the 'COMMIT LOG' section of the GitHub Project Analysis Management System. On the left, the same 'Repo 資訊' sidebar for 'WWE2020' is visible. The main content area features a 'Filter' dropdown menu and a list of commits. The 'COMMIT LOG' section shows commits from September 7, 2020, and September 4, 2020. Commits include 'I'm Randy Orton' by Randy Orton, 'Welcome Kevin Owens' Show' by Kevin Owens, 'Believe that, Believe in The Shield' by Roman Reigns, 'Rest in peace !' by The Undertaker, 'You can't see me !' by John Cena, and 'Welcome to Ambrose Asylum !' by Dean Ambrose. Each commit card includes the commit message, the author's name, the commit date, and the commit hash.

**COMMIT LOG**

**Filter**

**Sep 7, 2020**

- I'm Randy Orton**  
Randy Orton committed on 7 Sep  
e76bf50
- Welcome Kevin Owens' Show**  
Kevin Owens committed on 7 Sep  
7c4c296

**Sep 4, 2020**

- Believe that, Believe in The Shield**  
Roman Reigns committed on 4 Sep  
a66625a
- Rest in peace !**  
The Undertaker committed on 4 Sep  
be23b31
- You can't see me !**  
John Cena committed on 4 Sep  
57aa75d
- Welcome to Ambrose Asylum !**  
Dean Ambrose committed on 4 Sep  
9f2efc0

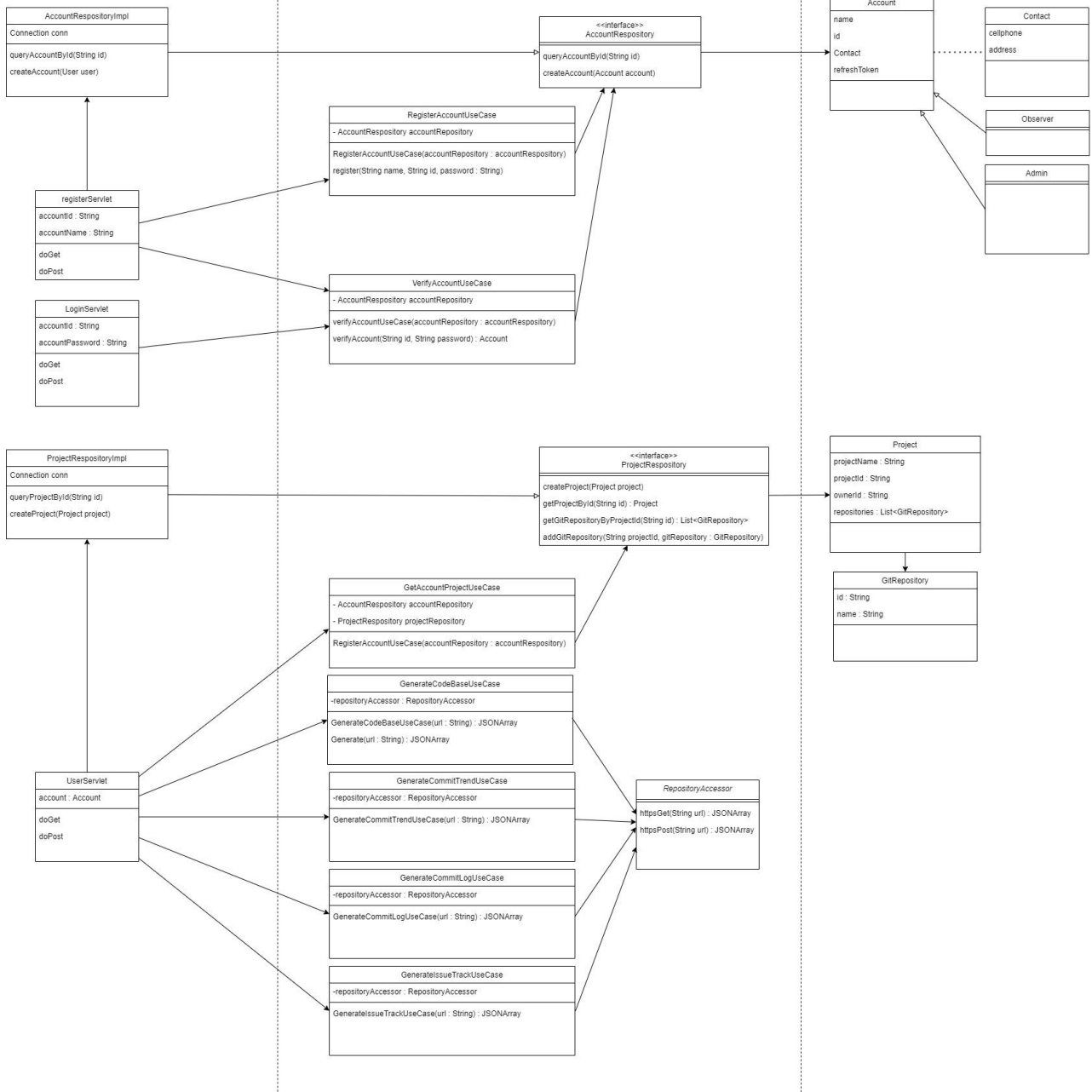
### 3.1.3 Static Model

#### 第一次 Increment

# Controller

# UseCase

# Entity

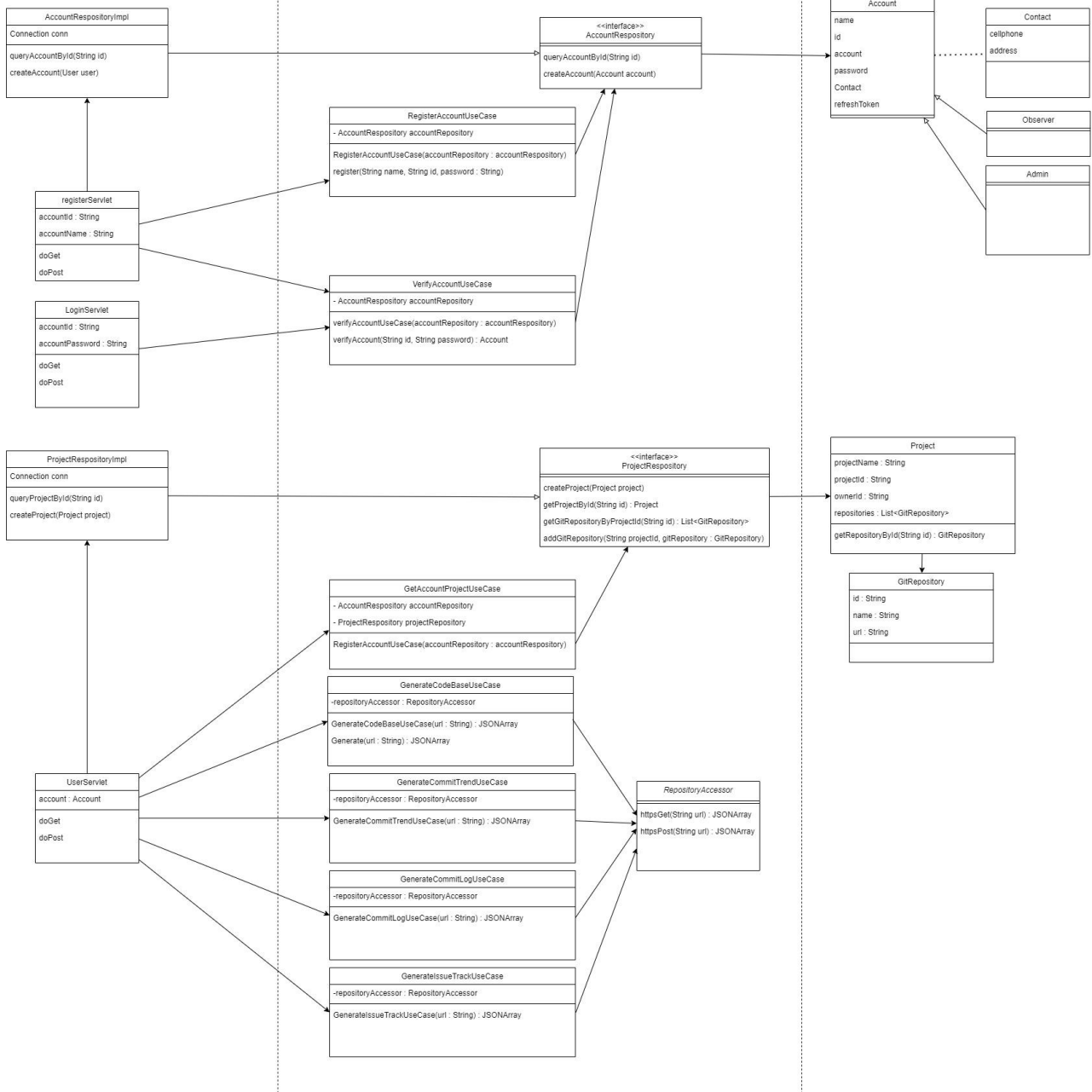


第二次 Increment，對資料庫欄位的部分進行了調整

## Controller

## UseCase

## Entity



Class No.	
Class Name	registerServlet
Responsibility	負責處理使用者註冊行為的請求轉送
Collaboration	AccountRespositoryImpl、RegisterAccountUseCase、

	VerifyAccountUseCase
<b>Related Subsystem</b>	UIVMS Module

<b>Class No.</b>	
<b>Class Name</b>	LoginServlet
<b>Responsibility</b>	負責處理使用者登入行為需求轉送
<b>Collaboration</b>	VerifyAccountUseCase
<b>Related Subsystem</b>	UIVMS Module

<b>Class No.</b>	
<b>Class Name</b>	UserServlet
<b>Responsibility</b>	負責處理使用者專案管理的需求轉送
<b>Collaboration</b>	ProjectRespositoryImpl、GetAccountProjectUseCase、 GenerateCodeBaseUseCase、GenerateCommitTrendUseCase、 GenerateCommitLogUseCase、GenerateIssueTrackUseCase
<b>Related Subsystem</b>	UIVMS Module

<b>Class No.</b>	
<b>Class Name</b>	AccountRespositoryImpl
<b>Responsibility</b>	註冊帳戶的資料庫操作
<b>Collaboration</b>	AccountRespository
<b>Related Subsystem</b>	UIVMS Module

<b>Class No.</b>	
<b>Class Name</b>	RegisterAccountUseCase
<b>Responsibility</b>	註冊帳戶
<b>Collaboration</b>	AccountRespository
<b>Related Subsystem</b>	UIVMS Module

<b>Class No.</b>	
<b>Class Name</b>	VerifyAccountUseCase
<b>Responsibility</b>	使用者登入驗證
<b>Collaboration</b>	AccountRespository

<b>Related Subsystem</b>	TMS Module
--------------------------	------------

<b>Class No.</b>	
<b>Class Name</b>	ProjectRespositoryImpl
<b>Responsibility</b>	專案的資料庫操作
<b>Collaboration</b>	ProjectRespository
<b>Related Subsystem</b>	RMS Module

<b>Class No.</b>	
<b>Class Name</b>	GetAccountProjectUseCase
<b>Responsibility</b>	獲取帳戶的相關Repo資料
<b>Collaboration</b>	ProjectRespository
<b>Related Subsystem</b>	RMS Module

<b>Class No.</b>	
<b>Class Name</b>	GenerateCodeBaseUseCase
<b>Responsibility</b>	獲取相關Repo 的CodeBase資料
<b>Collaboration</b>	<i>RepositoryAccessor</i>
<b>Related Subsystem</b>	RMS Module

<b>Class No.</b>	
<b>Class Name</b>	GenerateCommitTrendUseCase
<b>Responsibility</b>	獲取相關Repo 的Commit趨勢資料
<b>Collaboration</b>	<i>RepositoryAccessor</i>
<b>Related Subsystem</b>	RMS Module

<b>Class No.</b>	
<b>Class Name</b>	GenerateCommitLogUseCase
<b>Responsibility</b>	獲取相關Repo 的CommitLog資料
<b>Collaboration</b>	<i>RepositoryAccessor</i>
<b>Related Subsystem</b>	RMS Module

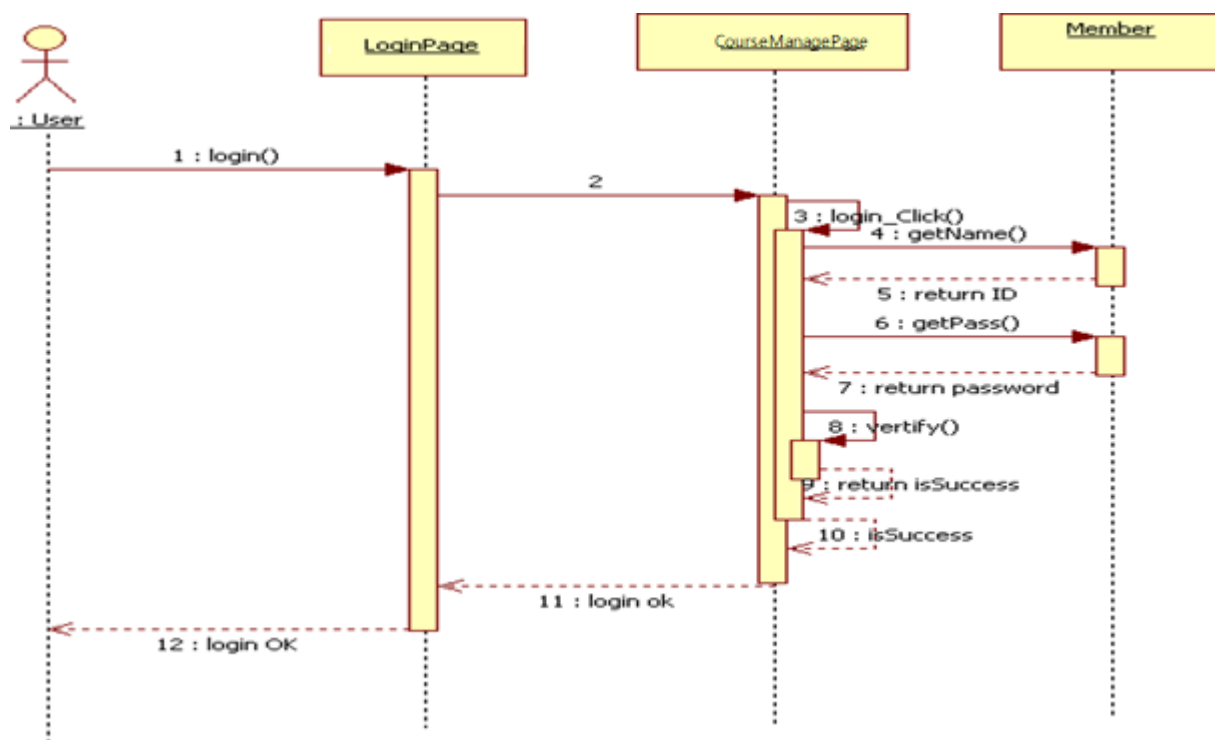


<b>Class No.</b>	
<b>Class Name</b>	GenerateIssueTrackUseCase
<b>Responsibility</b>	獲取相關Repo 的IssueTrack資料
<b>Collaboration</b>	<i>RepositoryAccessor</i>
<b>Related Subsystem</b>	RMS Module

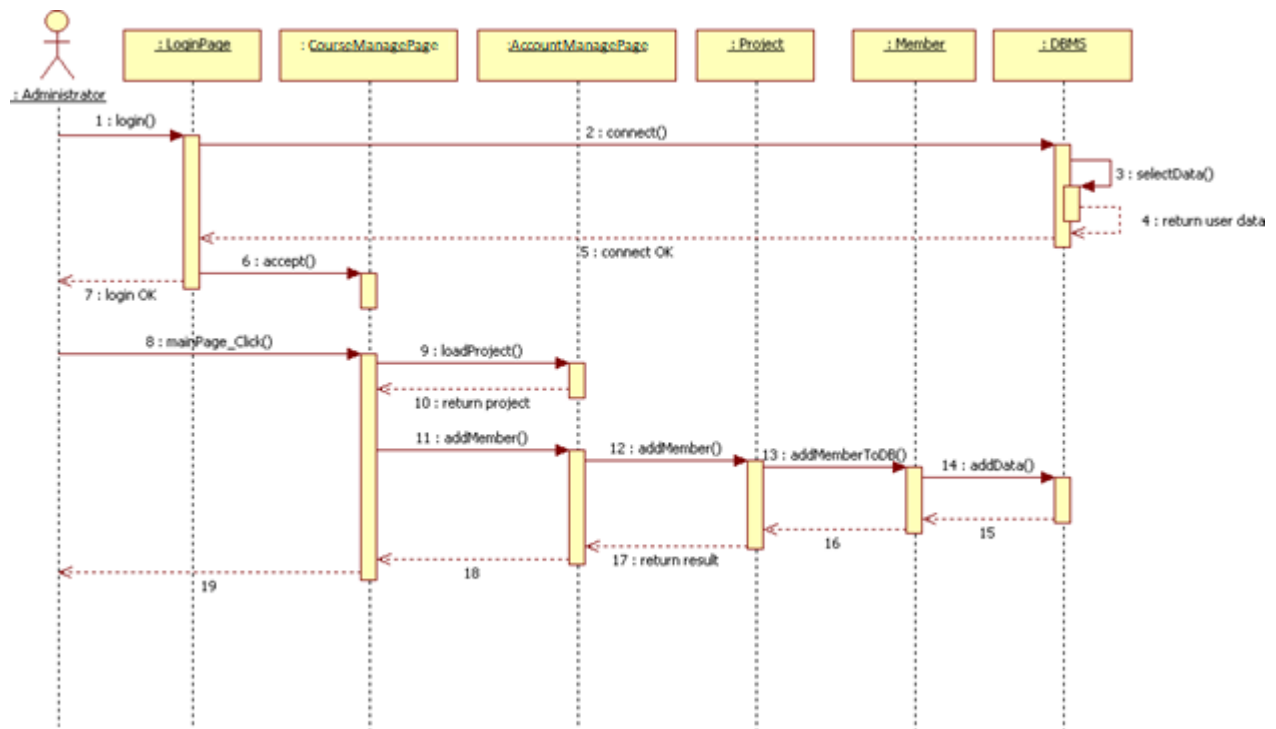
### 3.1.4 Dynamic Models

#### User Identity Verification and Management Subsystem

使用者登入：

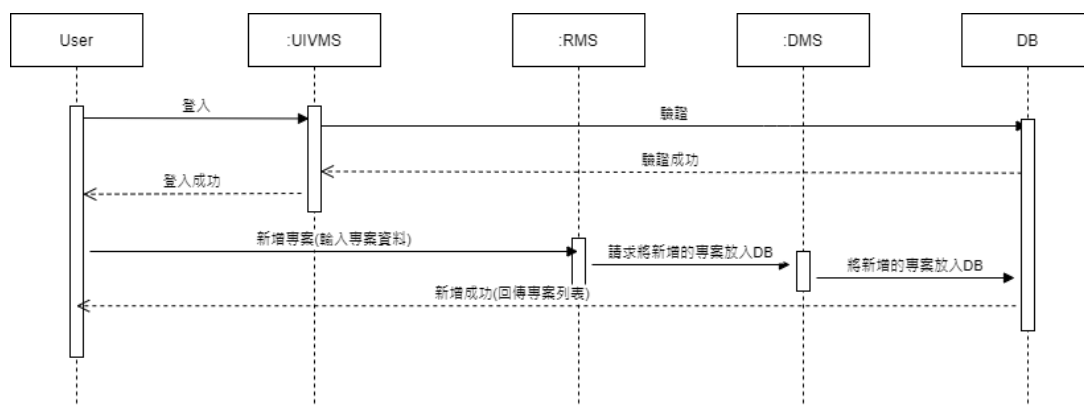


新增使用者：

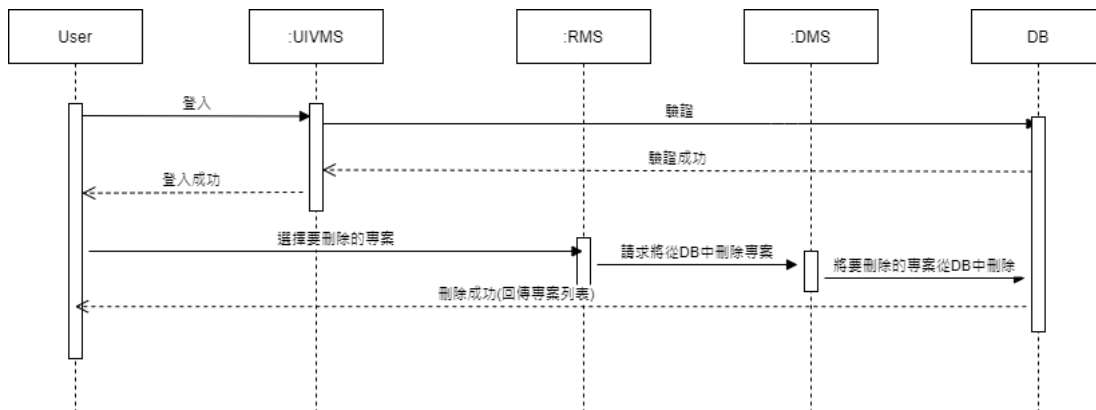


## Repository Management Subsystem

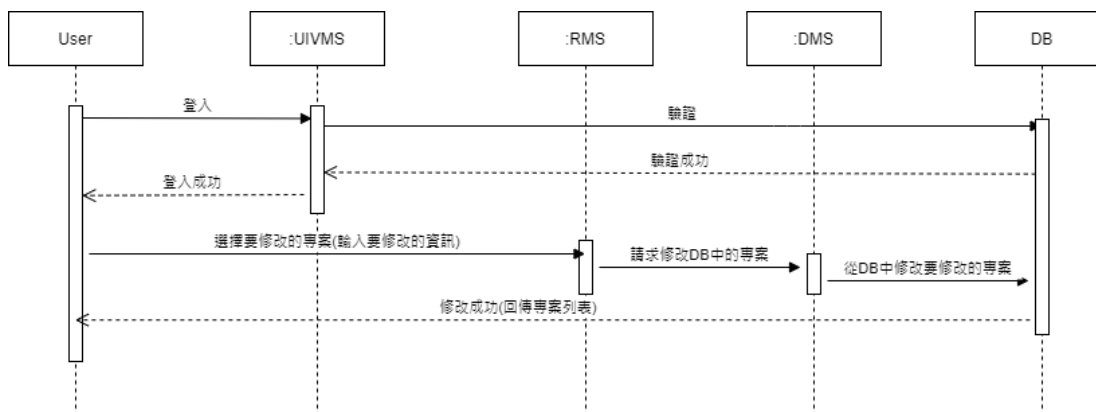
新增專案：



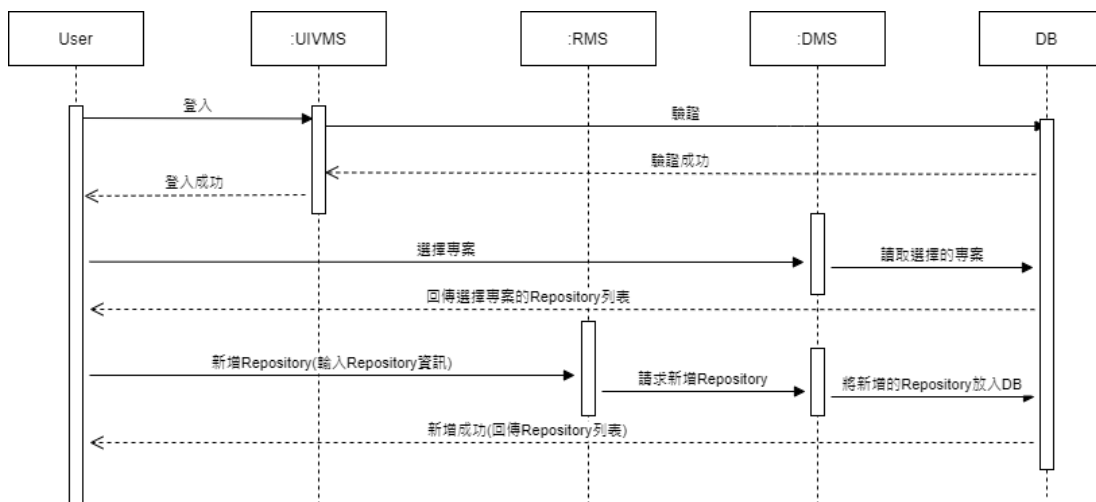
### 刪除專案:



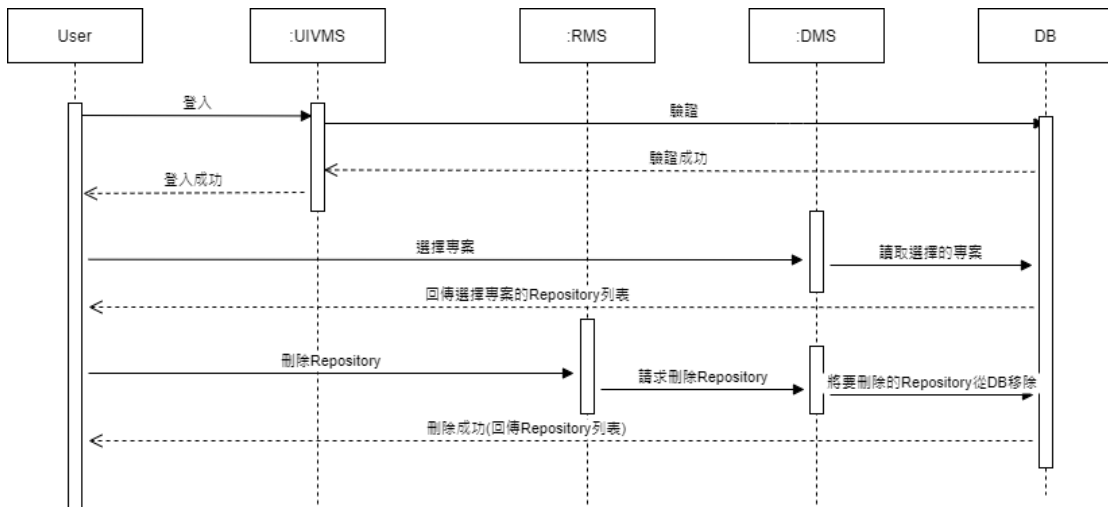
### 修改專案:



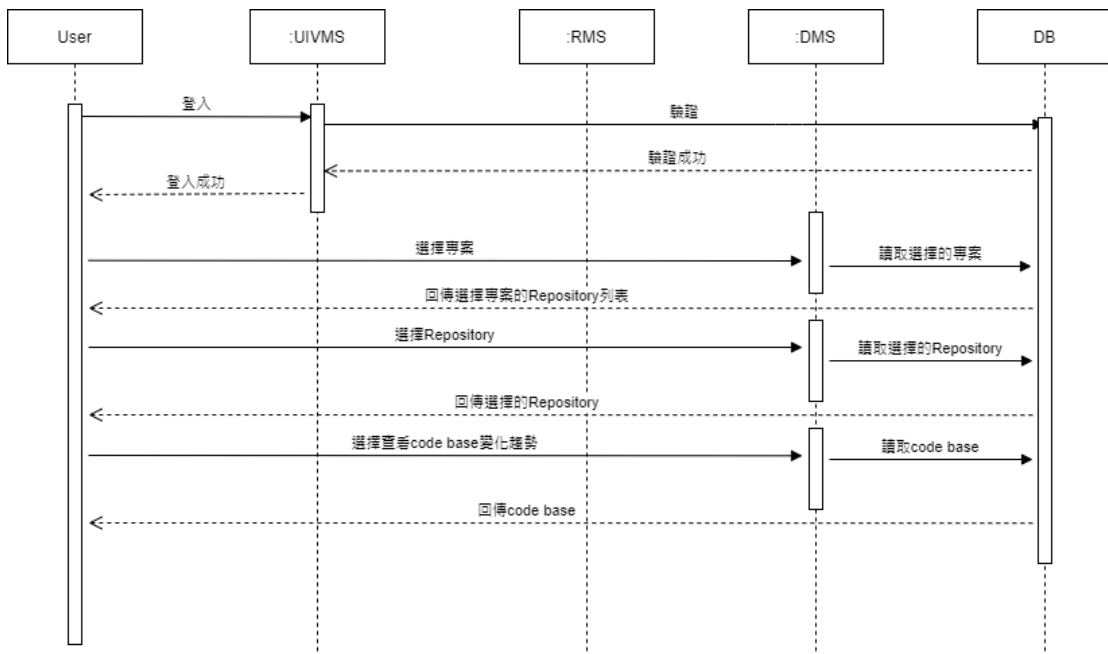
### 新增 Repository:



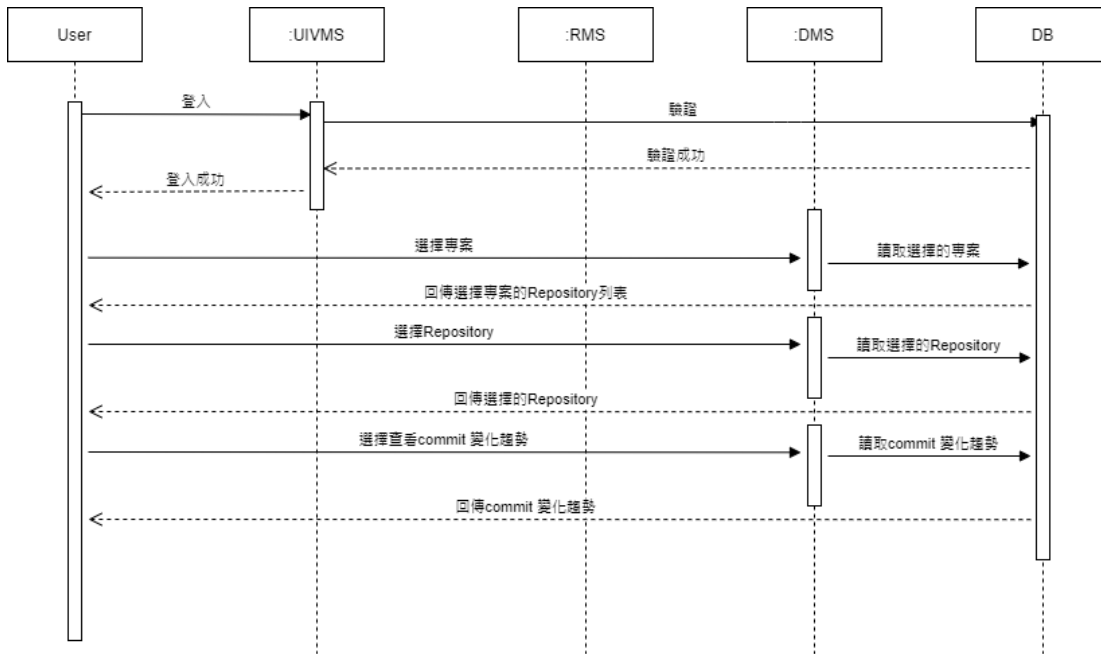
## 刪除 Repository:



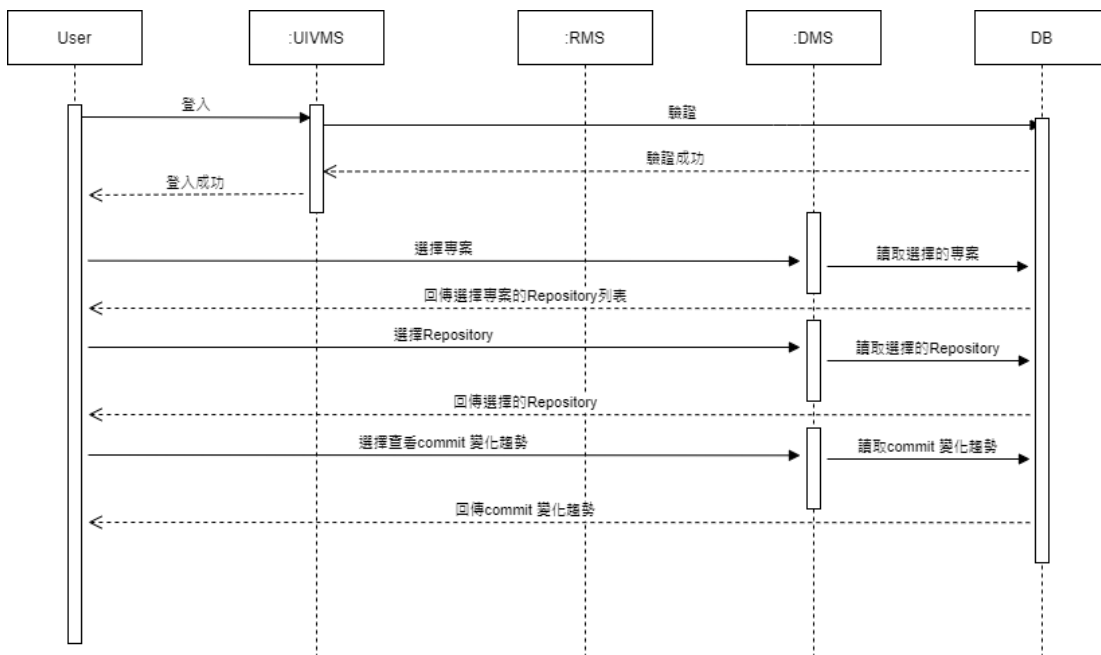
## 查看 code base 變化趨勢:



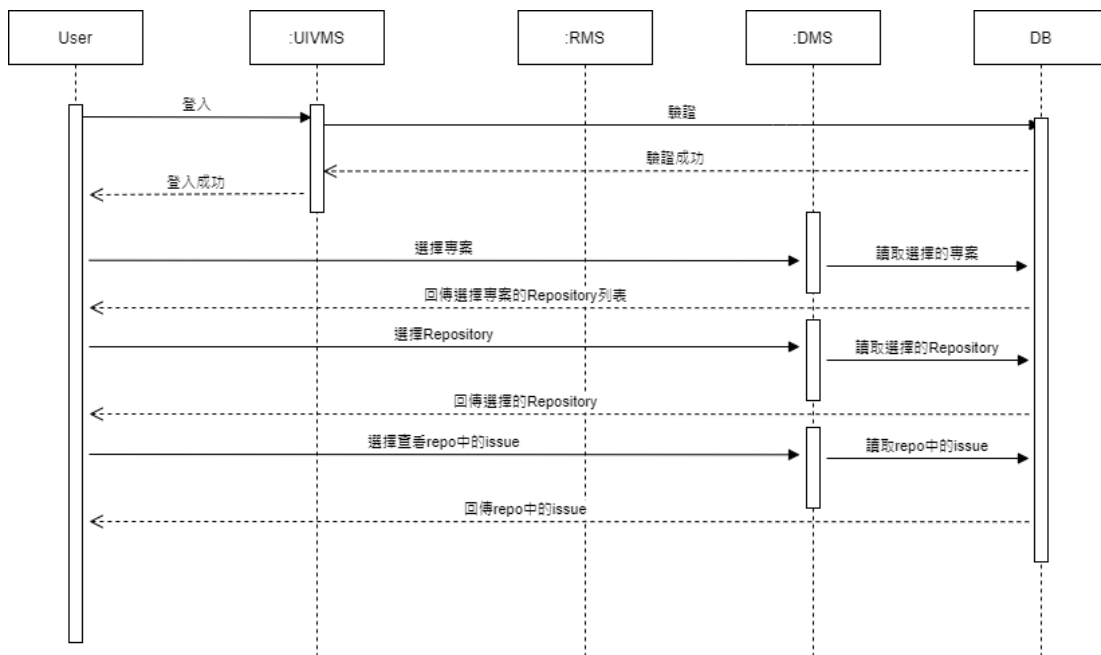
## 查看成員的 commit:



查看 commit 變化趨勢:



查看 repo 中的 issue:



## Glossary

Database	記錄資料的地方，提供新增、刪除、更新記錄的功能。
JAVA	Java 程式語言的風格十分接近 C++語言。繼承了 C++ 語言物件導向技術的核心，Java 捨棄了 C++語言中容易引起錯誤的指標、運算符過載、多重繼承等特性，增加了垃圾回收器功能用於回收不再被參照的物件所佔據的內部記憶體空間。Java 伴隨著網際網路的迅猛發展而發展，逐漸成為重要的網路程式語言。
MySQL	MySQL 是一個開放原始碼的關聯式資料庫管理系統，由於性能高、成本低、可靠性好，已經成為最流行的開源資料庫，被廣泛地應用在 Internet 上的中小型網站中。隨著 MySQL 的不斷成熟，它也逐漸用於更多大規模網站和應用。
SQL	Structured Query Language，關連式查詢語言，用來定義資料庫的結構，或是利用 SQL 對資料庫執行一些查詢、增加、刪除、更新記錄，到目前為止，SQL 是第一個，也是唯一的標準資料庫語言，受到廣泛的接受。
UI	用戶介面(User Interface)是介於使用者與硬體而設計彼此之間互動溝通相關軟體，目的在使得使用者能夠方便有效率地去操作硬體以達成雙向之互動，完成所希望借助硬體完成之工作，用戶介面定義廣泛，包含了人機互動與圖形使用者介面，凡參與人類與機械的信息交流的領域都存在著用戶介面。
Jsp	全稱是 Java Server Page，JSP 的面向是處理 View 的部份，也就是實際把動態產生的內容呈現給客戶端。這樣的做法是我們能夠把邏輯和顯示層分開，達到更容易維護。
Angular	是一個基於 TypeScript 的開源 Web 應用框架由 Google 的 Angular 團隊以及社群共同領導。Angular 是由 AngularJS 的同一個開發團隊完全重寫的。
UI	用戶介面(User Interface)是介於使用者與硬體而設計彼此之間互動溝通相關軟體，目的在使得使用者能夠方便有效率地去操作硬體以達成雙向之互動，完成所希望借助硬體完成之工作，用戶介面定義廣泛，包含了人機互動與圖形使用者介面，凡參與人類與機械的信息交流的領域都存在著用戶介面。