SunbirdDCIM - test automation PR Tool

專題編號：111-CSIE-S018
執行期限：110 年第 1 學期至 111 年第 1 學期
指導教授：郭忠義
專題參與人員： 108590001 李文至
108590003 黃明萱
108590050 李浩銘

一、 摘要

本專題預計使用 Python 程式開發一個適用於 SunbirdDCIM - test automation 的 Pull Request Tool，並以網頁方式呈現，讓使用者可以更方便、更快速、更簡單地建立 Pull Request (PR)。

關鍵詞：GitHub Rest API、Pull Request (PR)、Python

二、 緣由與目的

本專題是在 Sunbird 公司實習時碰到的問題，在完成測試案例後，把測試腳本上傳至 GitHub 專案，並且需要通過 PR 審核後才能正式合併至專案內。現階段每完成一個測試案例後需要手動建立 PR，逐一加入每位 reviewers 以及 assignees ，所以會浪費很多的時間。

除此之外，每次準備 sprint review 文件需要確認待開發、開發中和已完成的測試案例，這時候會到 GitHub 確認目前開發中的 PR 是否有被 merge 到主分支，而且同一組團隊中不只有一個組員會發布 PR，所以要逐一搜尋 User Name 看目前的狀態。

我們想到可以使用 GitHub 的 API 來解決建立 PR 的效率問題。因此，本專題預計開發一個網頁，讓使用者可以更方便、更快速、更簡單地建立 PR。

三、 研究範圍

熟悉 GitHub 頁面及 GitHub REST API 操作、Python Flask 的實作深入研究重點有以下這些列表：

表一、使用到的 GitHub API

API 名稱	HTTP method	End point
Create a pull request	POST	/repos/{owner}/{repo}/pulls
Request reviewers for a pull request	POST	/repos/{owner}/{repo}/pulls /{pull_number}/requested_reviewers
Add assignees to an issue	POST	/repos/{owner}/{repo} /issues/{issue_number}/assignees
Add labels to an issue	POST	/repos/{owner}/{repo} /issues/{issue_number}/labels
Get repository content	GET	/repos/{owner}/{repo} /contents/{path}
List branches	GET	/repos/{owner}/{repo}/branches
Get a branch	GET	/repos/{owner}/{repo}/branches /{branch}
List organization members	GET	/orgs/{owner}/members
List label	GET	/repos/{owner}/{repo}/labels
List pull requests	GET	/repos/{owner}/{repo}/pulls
Get a pull request	GET	/repos/{owner}/{repo}/pulls /{pull_number}

四、 工具說明

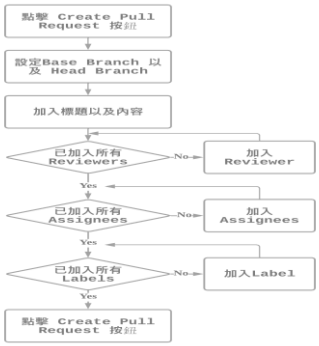
1. GitHub REST API[1]：用 URL 來操作包括取得、建立、修改和刪除，並對應到 HTTP 協定提供的 GET、POST、

PUT 和 DELETE 方法。

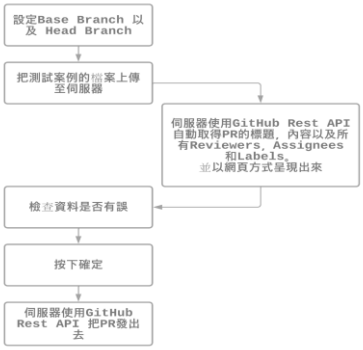
2. Python Flask[2]：使用 Python Flask Web 應用框架來架設一個輕量級的網頁伺服器。
3. SQLite[3]：遵守 ACID，實現了大多數 SQL 標準。它作為嵌入式資料庫，是應用程式，如網頁瀏覽器，在本地/客戶端儲存資料的常見選擇。

五、 使用技術方法

從圖一可見，建立 PR 的流程中有三個迴圈。因此，可以整合這個部分以節省時間，我們利用 Python Flask 實作一個網頁，把 Reviewers、Assignees、Labels 使用資料庫儲存起來，然後在建立 PR 時把資料讀取出來，最後使用 GitHub 的 API 建立 PR。



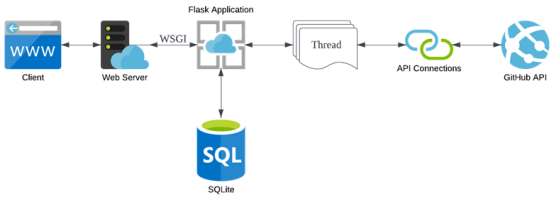
圖一、Pull Request 流程圖



圖二、簡化後，Pull Request 流程圖

六、 開發流程架構圖

我們採用 Python Flask 建立一個輕量級的網頁伺服器。資料庫是選擇使用 SQLite，把建立 PR 所需要的 Reviewers、Assignees 和 Labels 儲存在資料庫內，再通過 multithreading 請求 GitHub 的 Restful API，就可以自動抓取上述的資料，最後把資料呈現在頁面上回傳給使用者。



圖三、系統架構圖

七、 結論與展望

經過實際操作與計時整個使用過程，建立 PR 結果使用 GitHub 的網頁，建立時間約 3 分鐘；反之使用我們開發的系統，第一次不會有 Reviewers、Assignees、Labels 的記錄資料，所以建立時間約 2 分鐘，節省了約 33%的時間。創建過後，系統自動會把上述的資料記錄起來，後續建立的時間約 40 秒，等同於節省了約 83%的時間。除此之外，一開始會組別分類，可以直接查看同組 PR 列表，並且在 closed 部分會按照被 merge 的時間進行排序，以便於準備 sprint review 文件。雖然我們開發的系統可以大量節省建立 PR 的時間，但是在其他的功能上還是有很多不足的地方。未來希望可以新增更多的功能，像是以下功能：

1. 系統網頁上回覆 PR。
2. 系統網頁上 Pull 主分支以確保無衝突。
3. 結合 Jenkins，並顯示專案的測試結果，可直接在系統網頁確認。
4. 改善 GUI 讓畫面美觀與直觀。

八、 參考文獻

[1] GitHub Rest API
docs.github.com/en

[2] Python Flask
flask.palletsprojects.com

[3] SQLite
wikipedia.org/SQLite