Object-Oriented Analysis and Design

IoT相容性測試系統

HW#4 Domain Model

資工碩一 106598039 林亮勳

資工碩一 106598006 林照晟

目錄

[Requirement Document 4](#_Toc513747349)

[1.1 Change History 4](#_Toc513747350)

[1.2 Problem Statement 4](#_Toc513747351)

[1.3 System Context Diagram 5](#_Toc513747352)

[1.4 Summary of System Features 5](#_Toc513747353)

[1.5 Use Case Diagram 5](#_Toc513747354)

[1.6 Use Cases 6](#_Toc513747355)

[1.7 Non-functional Requirements and Constraints 10](#_Toc513747356)

[1.8 Glossary 10](#_Toc513747357)

[1.9 Software Environments 11](#_Toc513747358)

[Domain model 12](#_Toc513747359)

[2.1 Domain class diagram showing only concepts 12](#_Toc513747360)

[2.2 Add associations 15](#_Toc513747361)

[2.3 Add attributes 16](#_Toc513747362)

[Design 17](#_Toc513747363)

[3.1 Logical Architecture 17](#_Toc513747364)

[3.2 Use-Case Realizations with GRASP Patterns 18](#_Toc513747365)

[3.2.1 System sequence diagram 18](#_Toc513747366)

[3.2.2 Contract 20](#_Toc513747367)

[3.2.3 Sequence diagram 22](#_Toc513747368)

[3.3 Design Class Model 26](#_Toc513747369)

[Implementation Class Model 27](#_Toc513747370)

[4.1 Implementation class diagram 27](#_Toc513747371)

[4.2 Difference between implementation class model and design class model 28](#_Toc513747372)

[4.3 Calculate Line of Code 29](#_Toc513747373)

[Programming 29](#_Toc513747374)

[5.1 Snapshots of system execution 29](#_Toc513747375)

[5.2 Source Code Listing 31](#_Toc513747376)

[5.2.1 RequestHandler.js 31](#_Toc513747377)

[5.2.2 TestVersion.js 35](#_Toc513747378)

[5.2.3 TestVersionManager.js 35](#_Toc513747379)

[5.2.4 TestCombination.js 36](#_Toc513747380)

[5.2.5 TestMission.js 37](#_Toc513747381)

[5.2.6 TestMissionManager.js 40](#_Toc513747382)

[Unit Testing 41](#_Toc513747383)

[6.1 Snapshots of testing result 41](#_Toc513747384)

[6.2 Unit Testing Code Listing 42](#_Toc513747385)

[6.2.1 test/testVersion.js 42](#_Toc513747386)

[6.2.2 test/testVersionManager.js 42](#_Toc513747387)

[6.2.3 test/testMissionAndCombination.js 44](#_Toc513747388)

[6.2.4 test/testdata.js 48](#_Toc513747389)

# Requirement Document

## 1.1 Change History

|  |  |  |
| --- | --- | --- |
| Revision | Description | Date |
| Iteration I | | |
| 1 | Problem statement | May 9, 2018 |
| 2 | System Context Diagram  Summary of System Features  Use Case Diagram  Use Cases  Non-functional Requirements and Constraints  Glossary  Software Environments | May 20, 2018 |
| 3 | Revision#2 revise  Domain model | April 2, 2018 |
| 4 | Revision#3 revise  Package Diagram  System Sequence Diagram & Contract  Sequence Diagram  Design Class Diagram | May 2, 2018 |
| 5 | Revision#4 revise  Implement class diagram  Implement program | May 10, 2018 |

## 1.2 Problem Statements

由於IoT相關產品經常需要在不同版本、不同環境、不同參數等狀況下確保能夠正確執行，多重配對測試往往是其不可或缺的一環。但是，多重配對測試不僅操作繁複、操作枯燥且會產生大量檢測報告，因而需要一個系統整合測試環境簡化操作流程及整合結果報表，加快測試速度，遂為本系統之開發目的。

本系統將提供給開發人員及測試人員使用，開發人員可手動新增測試項目諸如作業系統版本、瀏覽器版本、裝置版本、韌體版本等等以及該項目底下所提供之測試版本，爾後測試人員即可直接透過網頁添加欲測試之測試項目並勾選測試版本，自動化進行多重交互配對測試並瀏覽配對測試之整合測試結果。

## 1.3 System Context Diagram



圖1-1 System Context Diagram

## 1.4 Summary of System Features

|  |  |
| --- | --- |
| Feature ID | Feature Description |
| FEA-01 | 提供測試人員網頁介面，可於介面中選擇任意數量之測試項目及測試版本，並依所選項目新增測試任務至資料庫。 |
| FEA-02 | 系統透過API呼叫Jenkins伺服器至資料庫讀取測試任務並執行對應之測試。 |
| FEA-03 | Jenkins伺服器依據測試任務執行對應測試，測試完畢後將測試結果以SSH方式回傳至系統並更新資料庫中之測試任務狀態。 |
| FEA-04 | 網頁介面可讀取系統中之測試結果報表及資料庫中之測試任務紀錄，並將記錄統整後呈現至網頁上供測試人員瀏覽。 |
| FEA-05 | 開發人員可依據需求在網頁介面上新增測試版本。 |

## 1.5 Use Case Diagram



圖1-2 Use Case Diagram

## 1.6 Use Cases

|  |  |
| --- | --- |
| Use case ID | UC-01 |
| Use Case Name | 登入 |
| Scope | IoT相容性測試系統 |
| Level | User goal |
| Primary Actor | 使用者 |
| Stakeholders and Interests | 使用者：想要能夠準確快速的驗證身分進入到系統。  系統擁有者：希望非授權人員無法進入系統。 |
| Preconditions | 網路連線正常，資料庫內需有該人員之帳號。 |
| Success Guarantee | 人員依照所屬權限進入系統進行相關操作。 |
| Main Success Scenario | 1. 使用者進入到登入畫面。 2. 使用者輸入認證資料後點擊確認。 3. 系統認證身分。 4. 使用者進入到系統。 |
| Extensions | \*a. 網路連線失敗。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  1a. 系統無法取得登入頁面。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  3a. 系統認證失敗。   1. 提醒使用者所輸入的認證資料是否正確。 2. 重新載入系統。   3b. 使用者忘記或遺失認證資料。   1. 提供系統管理員聯絡資訊，要求使用者聯繫系統管理員取得認證資料。   4a. 系統無法取得系統主頁面。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。 |
| Special Requirements | 必須要能夠與資料庫連線保持順暢。 |
| Technology and Data Variations List | 1a. 使用網頁瀏覽器進入登入畫面。  2a. 使用鍵盤輸入認證資料。 |
| Frequency of Occurrence | 時常發生（若時常有新的版本需測試） |
| Miscellaneous | 忘記或遺失認證資料時是否只能請求系統管理員協助？  是否有更便利可靠的認證身分方式？ |

|  |  |
| --- | --- |
| Use case ID | UC-02 |
| Use Case Name | 執行測試 |
| Scope | IoT相容性測試系統 |
| Level | User goal |
| Primary Actor | Tester |
| Stakeholders and Interests | Tester：操作此系統的使用者，可自由新增測試項目、選擇想要測試的測試版本並送出測試任務。  Jenkins伺服器：接收測試項目，執行測試，將測試結果回傳至系統。 |
| Preconditions | 使用者已成功進行**登入**（UC-01）。 |
| Success Guarantee | 使用者成功送出測試任務，系統將自動執行測試，並在完畢後更新測試任務。 |
| Main Success Scenario | 1. 使用者勾選所有需要測試之測試版本。  使用者不斷重複1.步驟直到勾選完所有所欲測試之測試版本。  2. 使用者輸入本次測試之辨識名稱。  3. 使用者開始執行測試，網頁介面重新載入。  4. 系統將測試任務寫入資料庫中。  5. 系統通知Jenkins伺服器執行測試。  6. Jenkins伺服器從資料庫中讀取測試任務。  7. Jenkins伺服器執行相對應之測試。  8. Jenkins伺服器將測試結果回傳至系統。  9. 系統接收測試結果。  10. 系統將結果儲存至硬碟中。  11. 系統更新資料庫中之測試資訊。 |
| Extensions | 3a. 未選擇足夠的測試版本（至少包含兩個項目）。  　1. 跳出錯誤視窗，告訴使用者尚未選擇完成。  3b. 辨識名稱與過去之測試重複。   1. 跳出錯誤視窗，告訴使用者名稱重複。   4a. 與資料庫連線發生錯誤。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。  5a. 系統無法連線至Jenkins伺服器。  　1. 系統更新測試任務為「連線失敗」。  6a. Jenkins伺服器與資料庫連線發生錯誤。  　1. Jenkins伺服器回傳測試失敗訊息給系統。  　　1a. Jenkins伺服器無法連線至系統。   1. Jenkins伺服器將錯誤原因寫入log檔案中，儲存至伺服器所在之硬碟中。   　2. 系統跳出錯誤視窗，告訴使用者錯誤原因。  7a. Jenkins伺服器無法找到對應之測試。  　1. Jenkins伺服器回傳測試失敗訊息給系統。  　　1a. Jenkins伺服器無法連線至系統。  　　　1. Jenkins伺服器將錯誤原因寫入log檔案中，儲 　　　　存至伺服器所在之硬碟中。  　2. 系統跳出錯誤視窗，告訴使用者錯誤原因。  8a. Jenkins伺服器無法連線至系統。  　1a. Jenkins伺服器無法連線至系統。  1. Jenkins伺服器將錯誤原因寫入log檔案中，儲存 至伺服器所在之硬碟中。  9a. 測試結果報告之資訊與資料庫資訊不符。  　1. 將有誤部分紀錄為log資訊輸出於伺服器端介面。  　2. 刪除所接收之測試結果報告。  10a. 儲存硬碟空間不足。  　1. 在伺服器端介面輸出硬碟空間不足，需要清出空間 　　後重新執行測試之資訊。  　2. 刪除所有本次已儲存之測試結果報告。  11a. 與資料庫連線發生錯誤。  　1. 將欲寫入資料庫之指令及錯誤原因輸出至伺服器 　　端介面。 |
| Special Requirements | 必須要能夠與資料庫及Jenkins伺服器連線保持順暢。 |
| Technology and Data Variations List | None |
| Frequency of Occurrence | 時常發生（若時常有新的版本需測試） |
| Miscellaneous | 測試版本若有新增或移除情形，需要有人進行管理。 |

|  |  |
| --- | --- |
| Use case ID | UC-03 |
| Use Case Name | 查看測試歷史紀錄 |
| Scope | IoT相容性測試系統 |
| Level | User Goal |
| Primary Actor | Tester |
| Stakeholders and Interests | Tester：想要能夠快速地找到想要查看的歷史紀錄，並獲得正確易懂的資訊。 |
| Preconditions | 使用者已**執行測試**（UC-02）。 |
| Success Guarantee | 使用者可以統一瀏覽所有測試任務。 |
| Main Success Scenario | 1. 使用者選擇查看測試歷史紀錄。 2. 系統讀取資料庫中的所有測試結果，並整理成清單供使用者選取。 3. 使用者選取清單上想要查看的歷史紀錄後，顯示該歷史紀錄的詳細資料。   使用者重複2-3的步驟直到查看完成。 |
| Extensions | 1a. 伺服器斷線，使用者無法進入測試歷史紀錄頁面。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。  2a. 與資料庫連線發生錯誤。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。  3a. 系統找不到資料庫中的歷史紀錄詳細資料。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。 |
| Special Requirements | 必須要能夠與伺服器及資料庫連線保持順暢。 |
| Technology and Data Variations List | 4a. 使用文字敘述或是圖表的方式來呈現容易理解的測試結果資訊。 |
| Frequency of Occurrence | 時常發生（若時常有新的版本需測試） |
| Miscellaneous | 是否需要提供搜尋功能來查找歷史紀錄？  是否有更直觀的方式來呈現歷史測試結果？ |

|  |  |
| --- | --- |
| Use case ID | UC-05 |
| Use Case Name | 更動測試版本 |
| Scope | IoT相容性測試系統 |
| Level | User goal |
| Primary Actor | Tester |
| Stakeholders and Interests | Tester：想要透過簡單的操作方式來新增或刪除測試版本來供**執行測試**（UC-02）使用。  System：希望準確的接受請求來更新資料庫之測試版本。 |
| Preconditions | 使用者已成功進行**登入**（UC-01），並且符合管理權限。 |
| Success Guarantee | 確實更動測試版本至資料庫。 |
| Main Success Scenario | 1. 使用者進入更動測試版本頁面。 2. 顯示現有的測試版本。 3. 使用者更動測試版本。 4. 使用者按下儲存更動。 5. 返回更動測試版本頁面。   使用者重複3-5的步驟直到全部更動完成。 |
| Extensions | \*a. 網路連線失敗   1. 確保網路連線正常。 2. 重新啟動系統。   2a. 與資料庫連線發生錯誤。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。  4a. 資料庫有重複之測試版本。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 返回更動測試版本頁面。  4b. 與資料庫連線發生錯誤。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 重新載入系統。  4c. 使用者輸入之測試版本不符合規格。  　1. 跳出錯誤視窗，告訴使用者錯誤原因。  　2. 返回更動測試版本頁面。 |
| Special Requirements | 必須要能夠與資料庫連線保持順暢。 |
| Technology and Data Variations List | None |
| Frequency of Occurrence | 時常發生（若時常有新的版本需測試） |
| Miscellaneous | 不同的測試版本有不同的資訊內容，如何整合輸入？ |

## 1.7 Non-functional Requirements and Constraints

|  |  |  |
| --- | --- | --- |
| NFR ID | Category | Feature Description |
| NFR-01 | Performance | 當執行測試時需要在合乎預測的時間內執行完畢。 |
| NFR-02 | Reliability | 系統必須能夠列出每筆測試的結果，不可遺漏。 |
| NFR-03 | Design constraints | 由於實際測試須透過Jenkin伺服器上已有之測試程式，因此資料庫中之測試任務需提供符合其程式規格之參數。 |
| NFR-04 | Usability | 系統需列出所有目前可測試之裝置項目以及其所屬之測試版本，並可輕易選擇配對後點選測試。 |

## 1.8 Glossary

|  |  |
| --- | --- |
| Item | Definition or Description |
| IoT | IoT，即物聯網，將現實世界數位化，應用領域主要包括以下方面：運輸和物流領域、健康醫療領域範圍、智慧型環境（家庭、辦公、工廠）領域、個人和社會領域等，具有十分廣闊的市場和應用前景。 |
| Jenkins伺服器 | Jenkins是一個用Java編寫的開源的持續整合工具，可通過各種手段或排程觸發預先建立之Procedure，本系統將透過API進行觸發。 |
| React.js | ReactJS 是 Facebook 推出的 JavaScript 函式庫，若以 MVC 框架來看，React 定位是在 View 的範疇。 |
| Node.js | Node.js是一個能夠在伺服器端運行JavaScript的開放原始碼、跨平台JavaScript執行環境。 |
| SSL | Secure Sockets Layer是為網路通信提供安全及數據完整性的一種安全協議。在傳輸層對網路連接進行加密。 |
| Nginx | Nginx是一個異步框架的 Web伺服器，也可以用作反向代理，負載平衡器 和 HTTP快取。 |
| 測試項目  （TestItem） | 即測試版本的類別，項目包括但不止於瀏覽器、手機、平板、產品韌體，每個測試版本都會有一個項目欄位。 |
| 測試版本  （TestVersion） | 配對測試的基本單位，一個配對測試包含兩個以上不同測試項目的測試版本。 |
| 測試任務  （TestMission） | 包含數個測試版本交互配對而成的測試列表，Jenkins需針對列表上所有配對進行測試。 |

## 1.9 Software Environments

　　本系統為以Nginx為架構之網路伺服器，內含一個以Node.js + Express撰寫而成的網頁系統，一個Jenkins伺服器，以及存取用的mongoDB資料庫。當測試人員透過網頁介面開始執行測試後會將測試任務儲存至資料庫中並通知Jenkins伺服器，Jenkins伺服器則讀取資料庫中的測試任務並執行測試完畢後將產生之結果報表以SSH回送至網頁介面供使用者查看結果報表。

# Domain model

## 2.1 Domain class diagram showing only concepts

由於本系統最初開發之目的為與學長的Jenkins伺服器進行嫁接，由網頁系統傳送必須之資訊給Jenkins伺服器後進行測試後回傳資訊，因此在與學長討論後率先規劃出以下預想之使用者介面：



圖2-1 登入畫面



圖2-2 執行測試頁面



圖2-3 歷史紀錄頁面



圖2-4 詳細測試結果頁面

**※Homework4修正**：我們希望開發人員能夠在網頁上直接手動管理測試版本以及其所屬的類別，因此，我們新增了一個前端頁面。



圖2-5 版本管理頁面

根據想像圖，我們首先討論出以下幾項可能會有之Class名稱：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PhoneOSVersion | FirmwareVersion | ApparatusVersion | Tester | Developer |
| TestResult | TestReport | IoTTestingSystem | TestPair | TestMission |

接著，我們為了讓系統更加自由，我們接連為Class做出調整。首先是作為測試基底的三個測試項目：

|  |  |  |
| --- | --- | --- |
| PhoneOSVersion | FirmwareVersion | ApparatusVersion |

這三者紀錄的資訊包括項目類別（比如作業系統、韌體等等）及版本編號兩項，為了讓系統可以便於新增其他裝置，因此我們讓這些類別繼承自一個抽象類別TestVersion並以此代表所有可能的測試項目所具有之測試版本，簡化Domain Model上的呈現。而使用者方面所擁有之兩種不同的使用者：

|  |  |
| --- | --- |
| Tester | Developer |

我們的作法與測試版本相同，我們直接以User作為抽象類別讓兩個不同的使用者權限去實作。接著是兩項測試結果的類別：

|  |  |
| --- | --- |
| TestResult | TestReport |

經過討論後我們覺得這兩個Class較不必要，其中TestResult可以作為Attribute存在於TestMission（用於儲存所有測試版本之配對）底下，而TestReport則是用於儲存Jenkins伺服器所產生之結果報表存放位置，因此我們選擇與TestPair（儲存複數個TestVersion，代表一項配對測試）結合，每個配對底下都會搭配到一份結果報表Html檔。

最後，由於IoTTestingSystem項目過為龐大，我們修改名稱為TestingService，負責處理儲存測試任務以及Jenkins的溝通。最後結果如下：



圖2-6 Concept Domain Model

**※Homework 4修正**：在設計Sequence Diagram時我們發現TestingService的意義有些不明，決定暫時先取消該設計，直接在TestMission底下直接執行測試。



圖2-7 Concept Domain Model

## 2.2 Add associations

接著，我們為其追加關係。

➃

➂

➁

➀

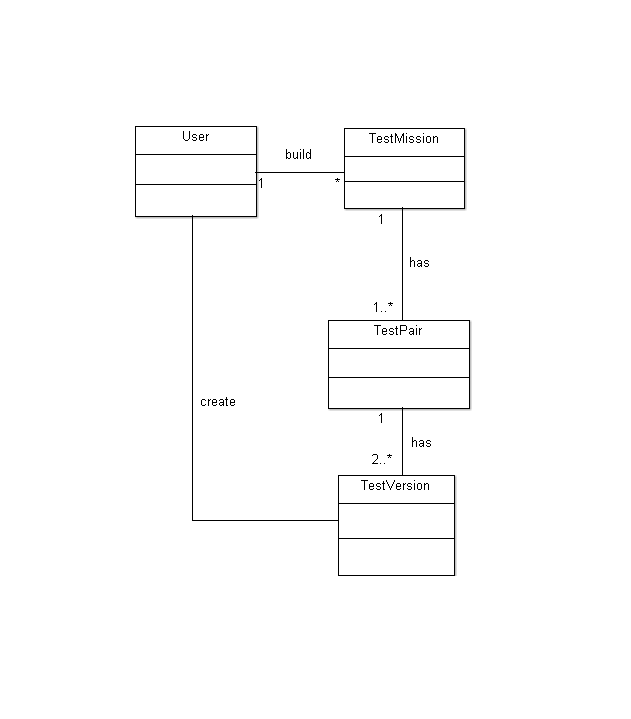


圖2-8 Domain Model with associations

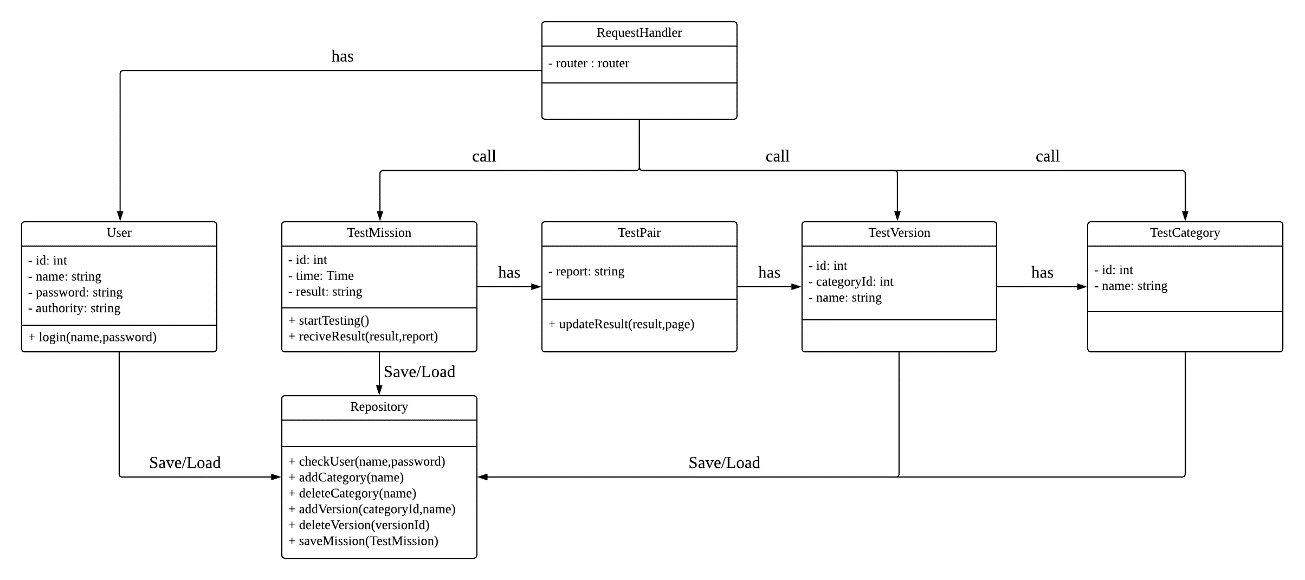
|  |  |
| --- | --- |
| # | Relationship |
| 1 | User建立TestMission來決定要測試的項目。 |
| 2 | TestMission將TestVerision配對成多個TestPair。 |
| 3 | 每個TestPair擁有兩個以上的TestVersion，表示一項配對測試。 |
| 4 | User可創建並定義TestVersion。 |

## 2.3 Add attributes



圖2-9 Domain Model

**※Homework 5修正**：在實際撰寫程式之後，我們稍微變更了Domain Model的設計並添加了RequestHandler及Repository兩個Class，用於存取前端及後端資訊。

****

# Design

## 3.1 Logical Architecture

****

圖3-1 Package Diagram

**說明**：在我們的系統中，前端從後端的Domain中取得相關的資料並顯示於使用者介面上。當前端進行如執行測試、管理測試版本等操作時，則會使後端向外部的Technical Service如Jenkins、Database發出請求，待其處理完畢回傳資訊更新Domain內的資料。

## 3.2 Use-Case Realizations with GRASP Patterns

### 3.2.1 System sequence diagram

* **Login（登入）**

****

圖3-2 Login SSD

* **Execute Testing（執行測試）**

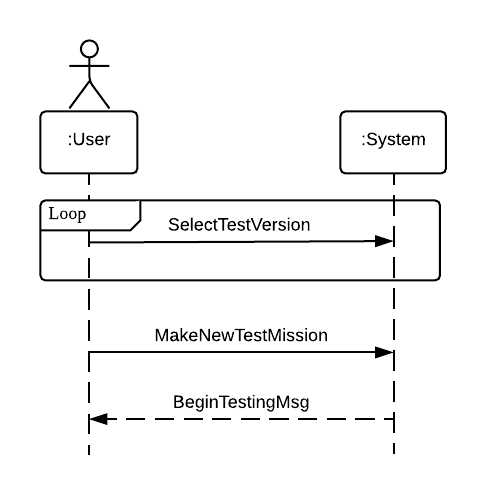
****

圖3-3 Execute Testing SSD

※ SelectTestVersion由於只是網頁表單的Select動作，並不會特別列出額外之Sequence diagram。

* **View Testing History（查看測試歷史紀錄）**

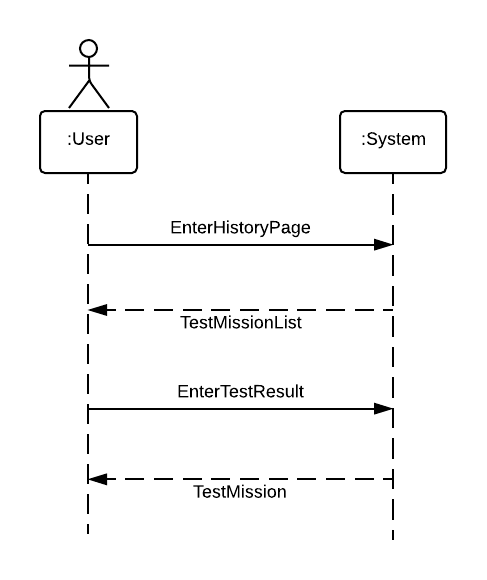
****

圖3-4 View Testing History SSD

* **Manage TestVersion（更動測試版本）**

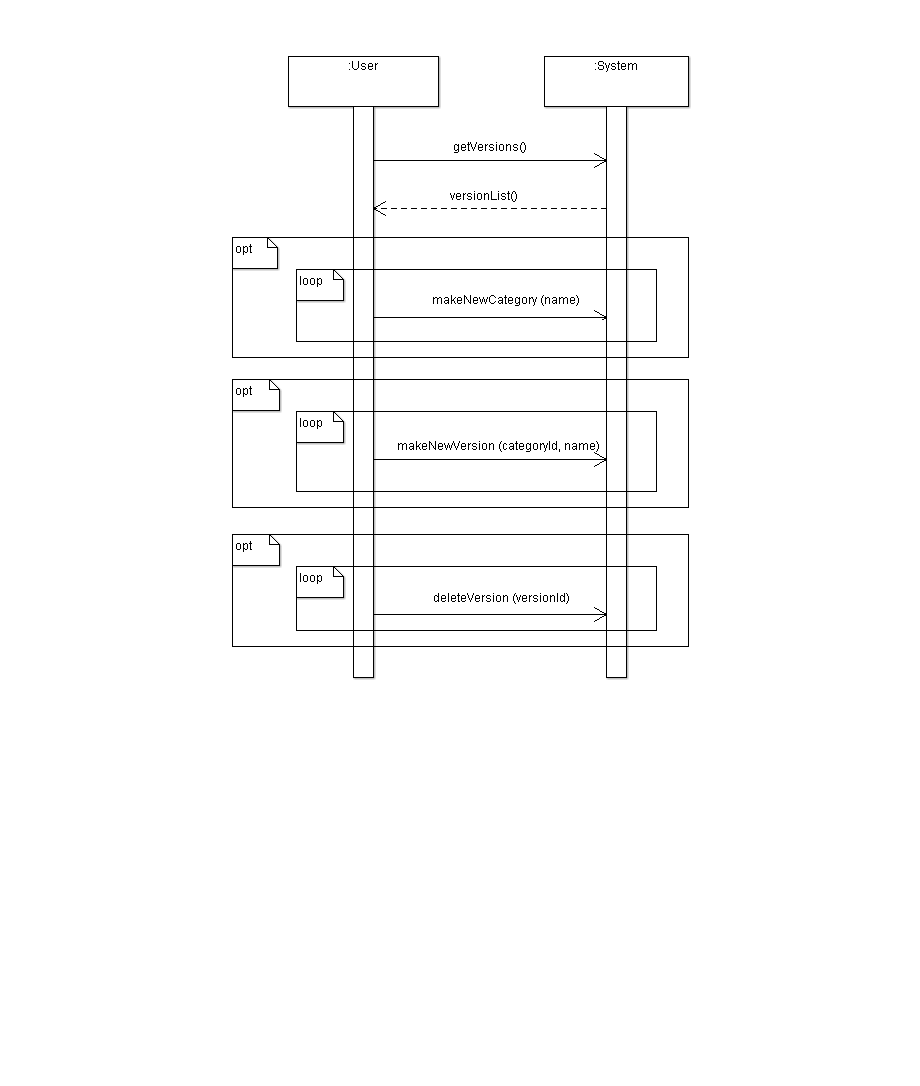


圖3-5 Manage TestVersion SSD

### 3.2.2 Contract

**Contract01 – Login**：

|  |  |
| --- | --- |
| Operation | Login(username: string, password: string) |
| Cross Reference | Use Case：UC-01 |
| Precondition | None. |
| Postconditions | 1. 進入系統主畫面，並依照使用者權限顯示可執行之操作項目。 |

**Contract02 – SelectTestVersion**：

|  |  |
| --- | --- |
| Operation | SelectTestVersion |
| Cross Reference | Use Case：UC-02 |
| Precondition | 使用者已**登入**（UC-01）。 |
| Postconditions | 1. 對所有選擇之TestVersion標示已選擇記號（打勾）。 |

**Contract03 – MakeNewTestMission**：

|  |  |
| --- | --- |
| Operation | SetTestVersionCategory |
| Cross Reference | Use Case：UC-02 |
| Precondition | 1. 使用者已**登入**（UC-01）。  2. 使用者已選擇完TestVersion（至少包含兩種Category，每種Category至少一項測試版本）。 |
| Postconditions | 1. 使用者接收到「已開始執行測試」訊息。  2. 使用者可在歷史紀錄頁面查詢到執行之測試。 |

**Contract04 – EnterHistoryPage**：

|  |  |
| --- | --- |
| Operation | SelectTestVersion |
| Cross Reference | Use Case：UC-03 |
| Precondition | 1. 使用者已**登入**（UC-01）。 |
| Postconditions | 1. 使用者介面呈現所有已執行之測試紀錄，並列出執行結果及成果報表連結。 |

**Contract05 – GetVersions**：

|  |  |
| --- | --- |
| Operation | getVersions() |
| Cross Reference | Use Case：UC-04 |
| Precondition | 使用者已**登入**（UC-01）。 |
| Postconditions | 1. 取得當前所有的Category和Version，並顯示在使用者介面。 |

**Contract06 – MakeNewCategory**：

|  |  |
| --- | --- |
| Operation | makeNewCategory(name: string) |
| Cross Reference | Use Case：UC-04 |
| Precondition | 使用者已**登入**（UC-01）。 |
| Postconditions | 1. 新增Category供**執行測試**（UC-02）時選擇。 |

**Contract07 – MakeNewVersion**：

|  |  |
| --- | --- |
| Operation | makeNewVersion(categoryId: int, name: string) |
| Cross Reference | Use Case：UC-04 |
| Precondition | 1. 使用者已**登入**（UC-01）。 2. 系統至少存在一個Category。 |
| Postconditions | 1. 新增Version至所選擇之Category裡供**執行測試**（UC-02）時選擇。 |

**Contract08 – DeleteVersion**：

|  |  |
| --- | --- |
| Operation | deleteVersion(versionId: ing) |
| Cross Reference | Use Case：UC-04 |
| Precondition | 1. 使用者已**登入**（UC-01）。 2. 系統至少存在一個Version。 |
| Postconditions | 1. 刪除指定的Version。 |

### 3.2.3 Sequence diagram

**Login：**

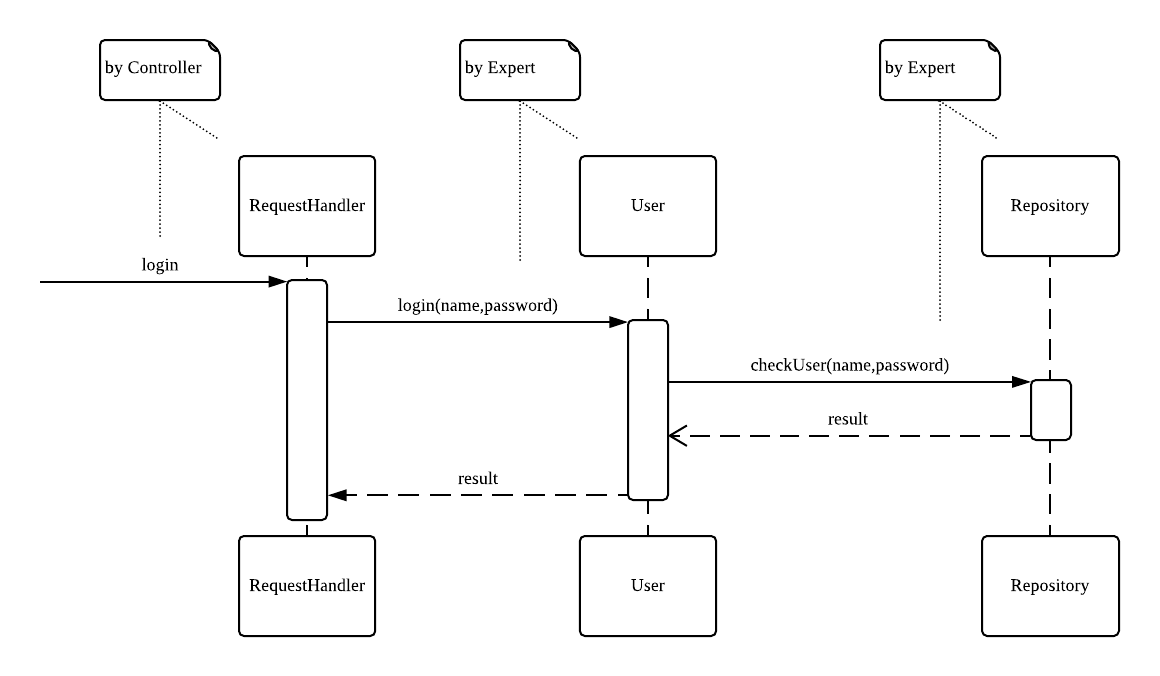


圖3-6 Login Sequence Diagram

**MakeNewTestMission：**

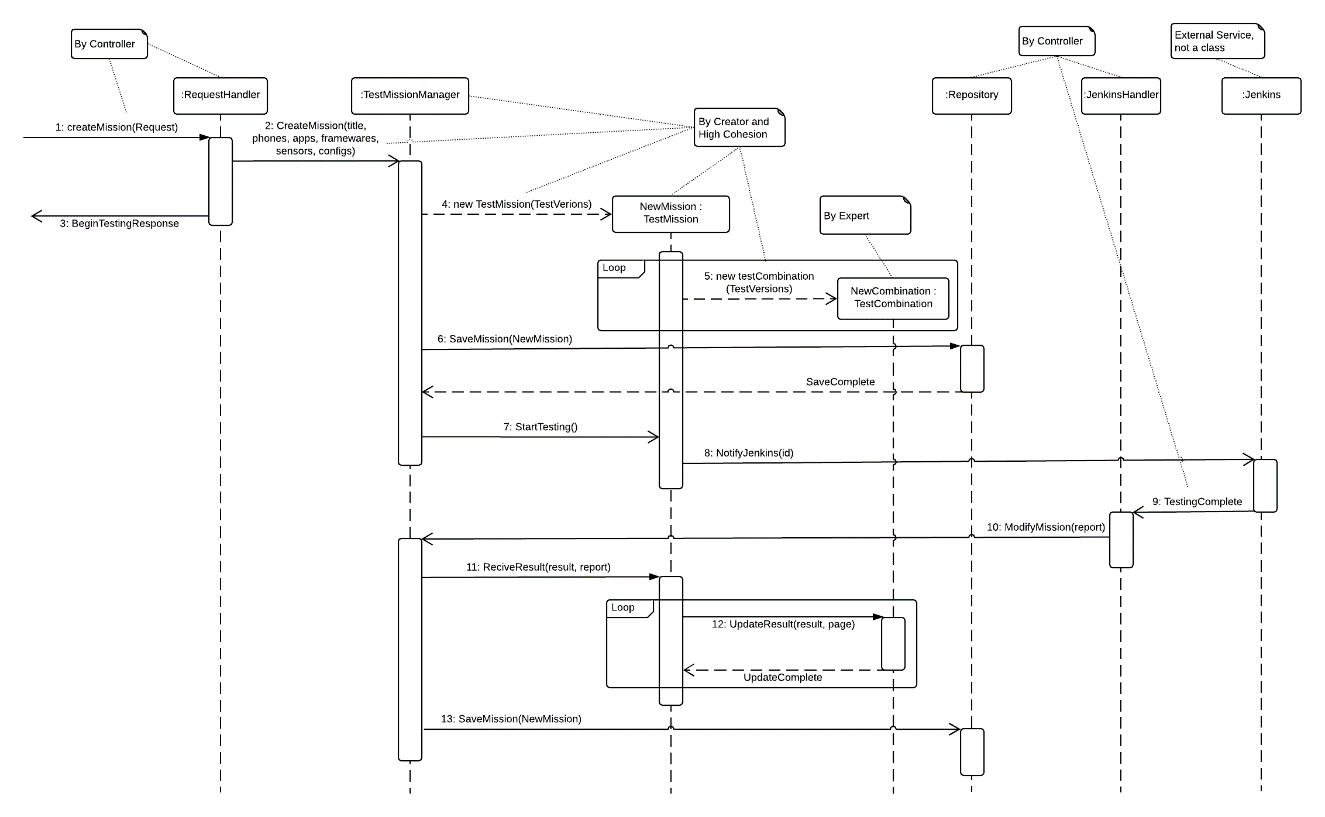


圖3-7 MakeNewTestVersion Sequence Diagram

**EnterHistoryPage：**

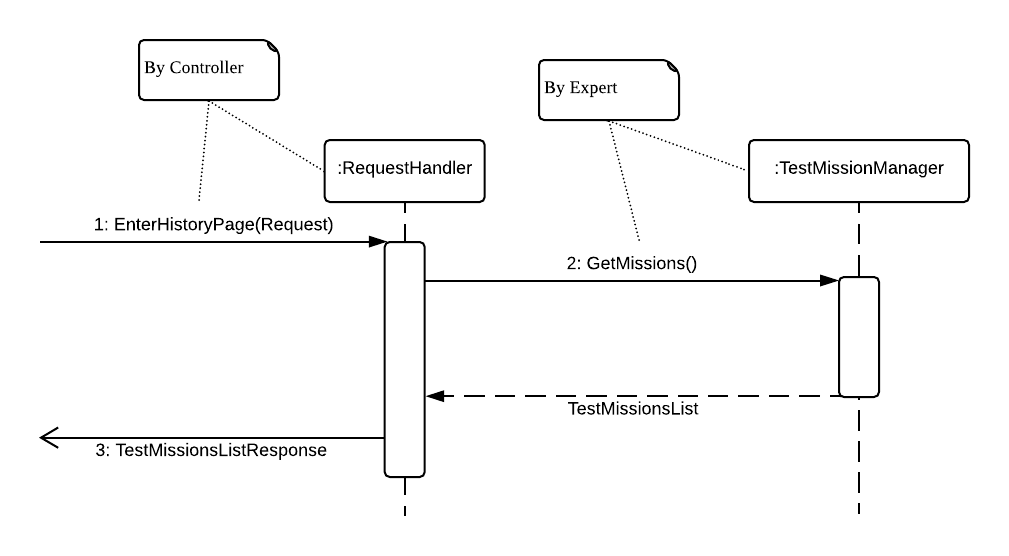


圖3-8 EnterHistoryPage Sequence Diagram

**EnterTestResult：**

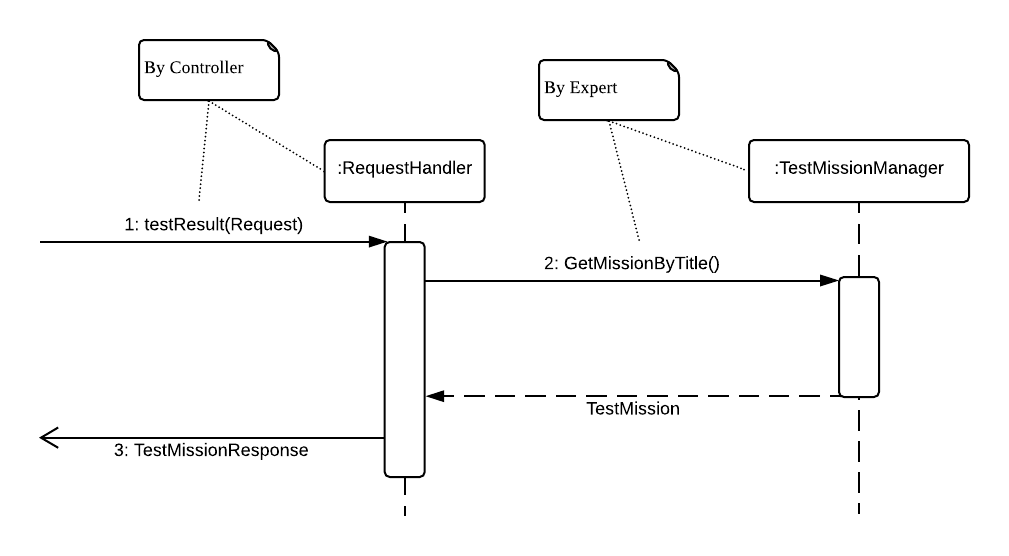


圖3-9 EnterTestResult Sequence Diagram

**GetVersions：**

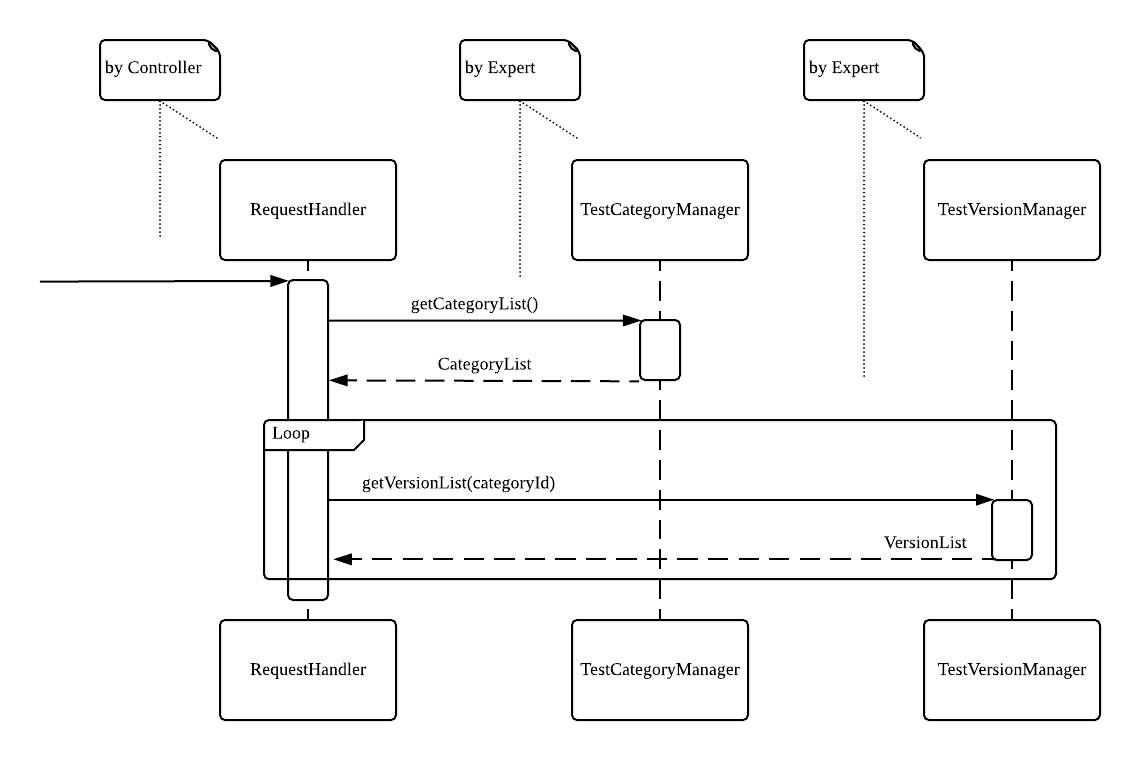


圖3-10 GetVersions Sequence Diagram

**MakeNewCategory：**

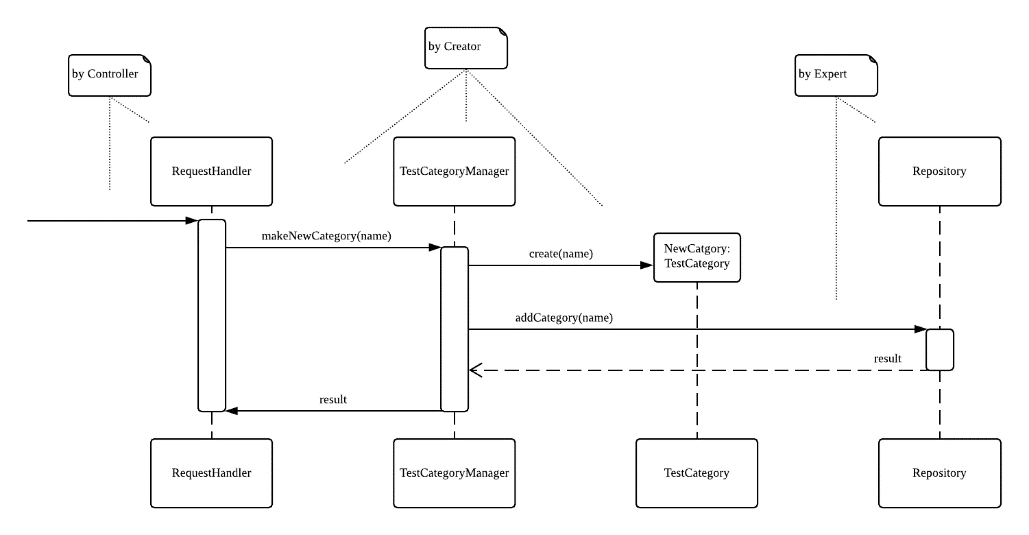


圖3-11 MakeNewCategory Sequence Diagram

**MakeNewVersion：**

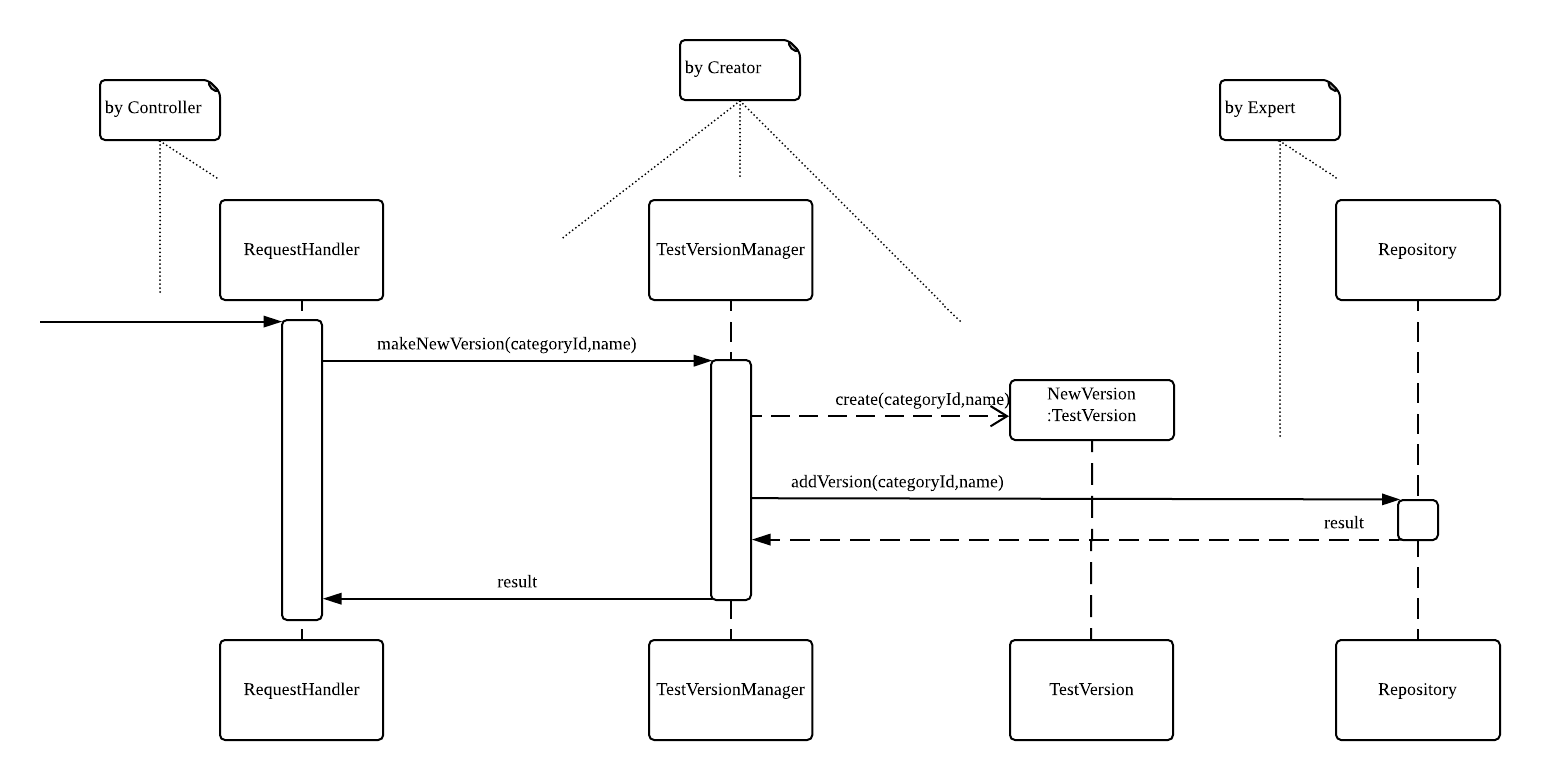


圖3-12 MakeNewVersion Sequence Diagram

**DeleteVersion：**

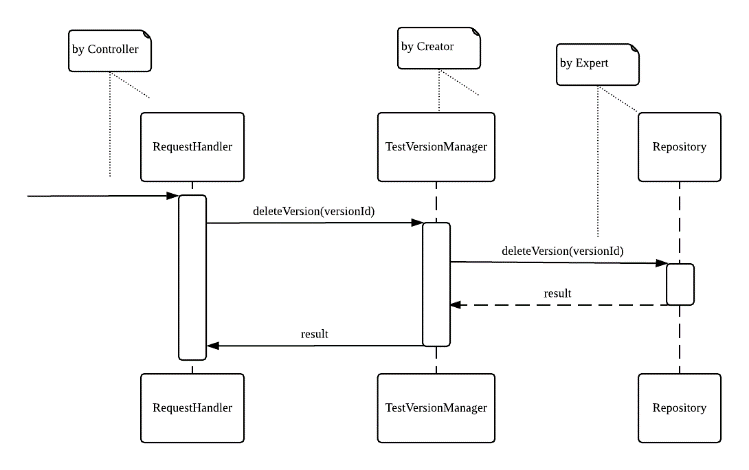
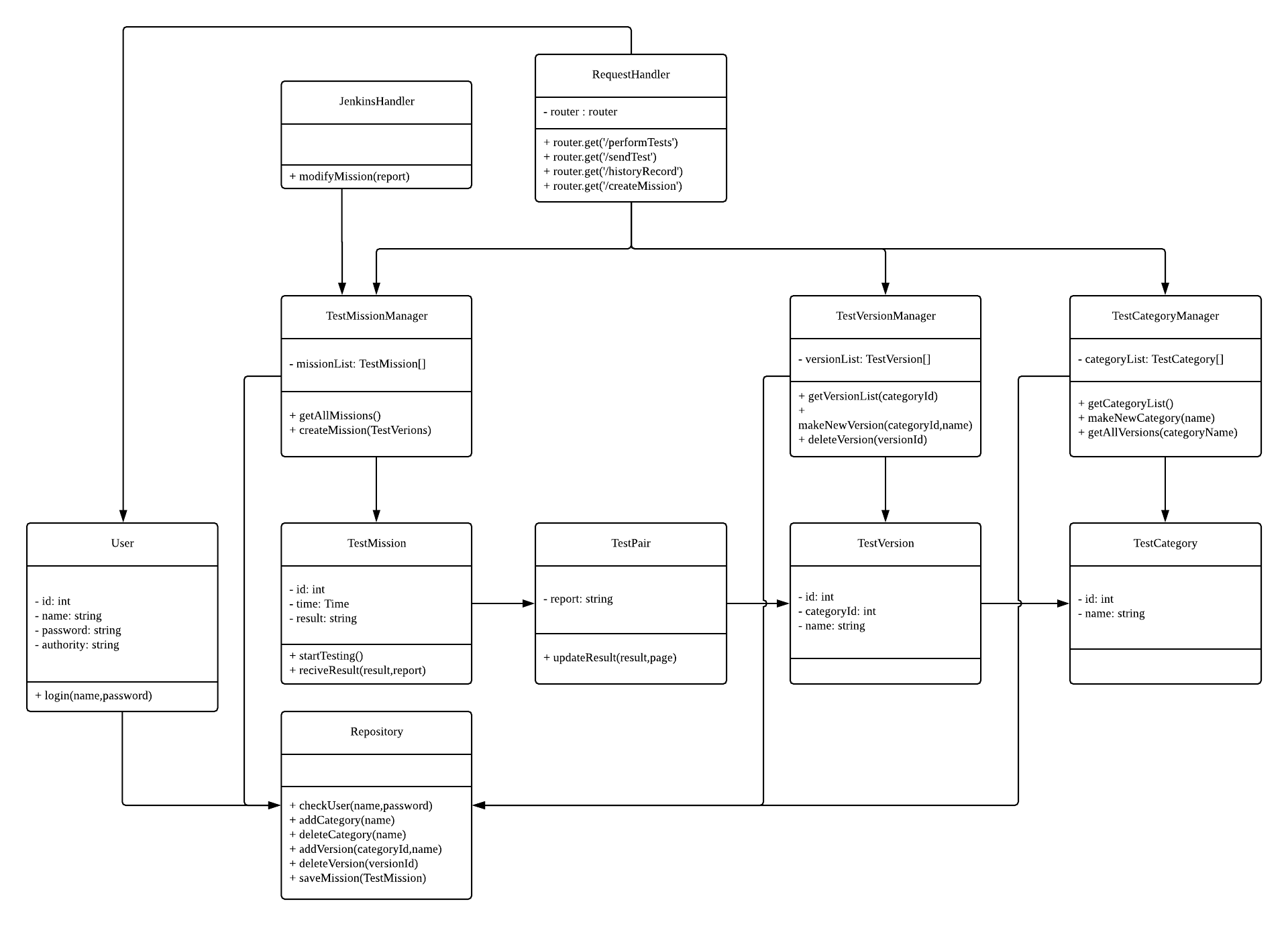


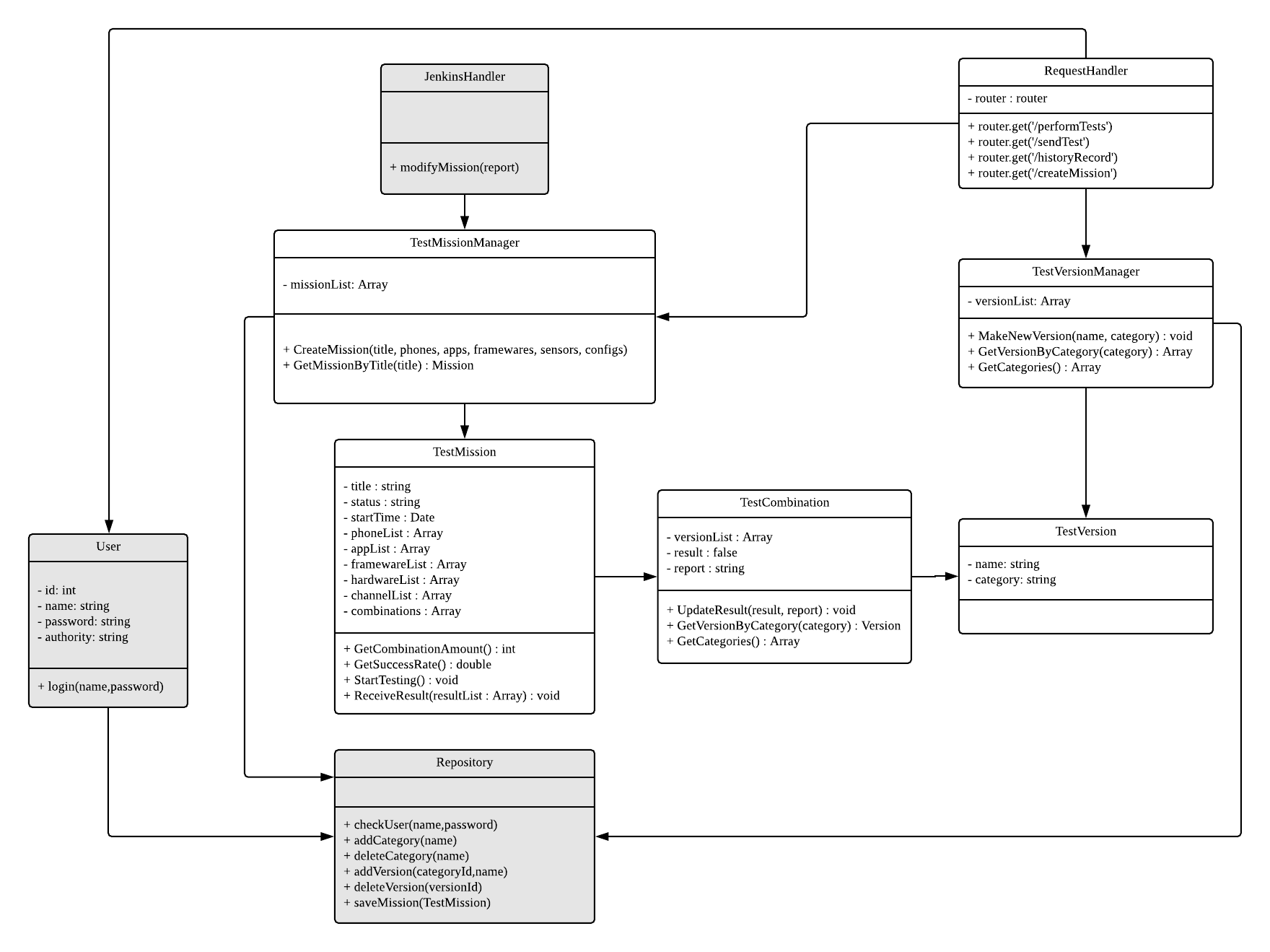
圖3-13 DeleteVersion Sequence Diagram

## 3.3 Design Class Model

****

# Implementation Class Model

## 4.1 Implementation class diagram



## 4.2 Difference between implementation class model and design class model

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Method | Design | Implement |
| TestVersionManager | GetVersionList | Yes | No |
| MakeNewVersion | Yes | Yes |
| DeleteVersion | Yes | No |
| GetVersionByCategory | No | Yes |
| GetCategories | No | Yes |
| TestCategoryManager  (Remove) | GetCategoryList | Yes | No |
| MakeNewCategory | Yes | No |
| GetAllVersions | Yes | No |
| TestCategory(Remove) | None | None | None |
| TestCombination  (Origin: TestPair) | UpdateResult | Yes | Yes |
| GetVersionByCategory | No | Yes |
| GetCategories | No | Yes |
| TestMission | StartTesting | Yes | Yes |
| ReciveResult | Yes | Yes |
| GetCombinationAmount | No | Yes |
| GetSuccessRate | No | Yes |
| TestMissionManager | CreateMission | Yes | Yes |
| GetMissions(mod.) | Yes | Yes |
| GetMissionByTitle | No | Yes |

表4-1 Comparison with design and implementation class

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of added | Number of removed | Number of modified |
| Class | 0 | 2 | 2 |
| Method | 7 | 5 | 0 |

表4-2 Summary of implementation class/method changed

## 4.3 Calculate Line of Code

|  |  |  |  |
| --- | --- | --- | --- |
| No | Class Name | Number of methods | Line of Code in Class |
| 1 | TestVersion | 2 | 15 |
| 2 | TestVersionManager | 6 | 36 |
| 3 | TesrCombination | 6 | 32 |
| 4 | TestMission | 9 | 69 |
| 5 | TestMissionManager | 4 | 26 |
| 6 | RequestHandler | 6 | 80 |
| Sum |  | 33 | 258 |

# Programming

## 5.1 Snapshots of system execution



圖5-1 主頁面

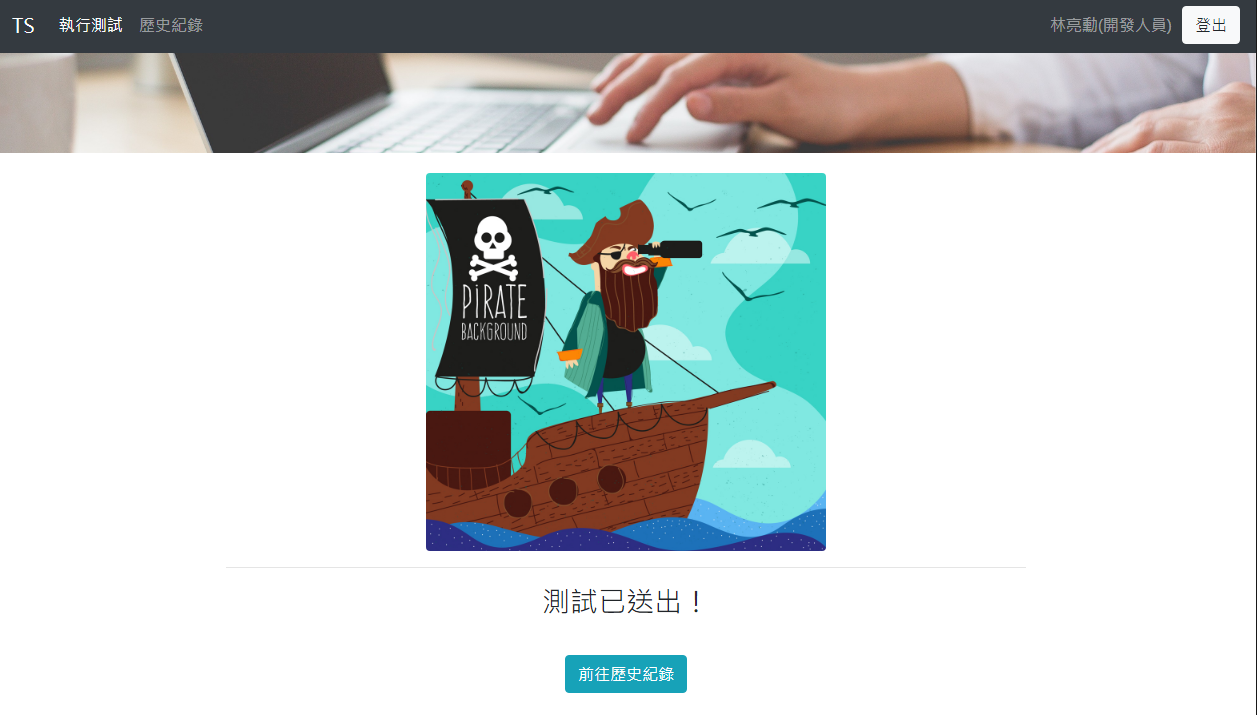


圖5-2 送出測試頁面

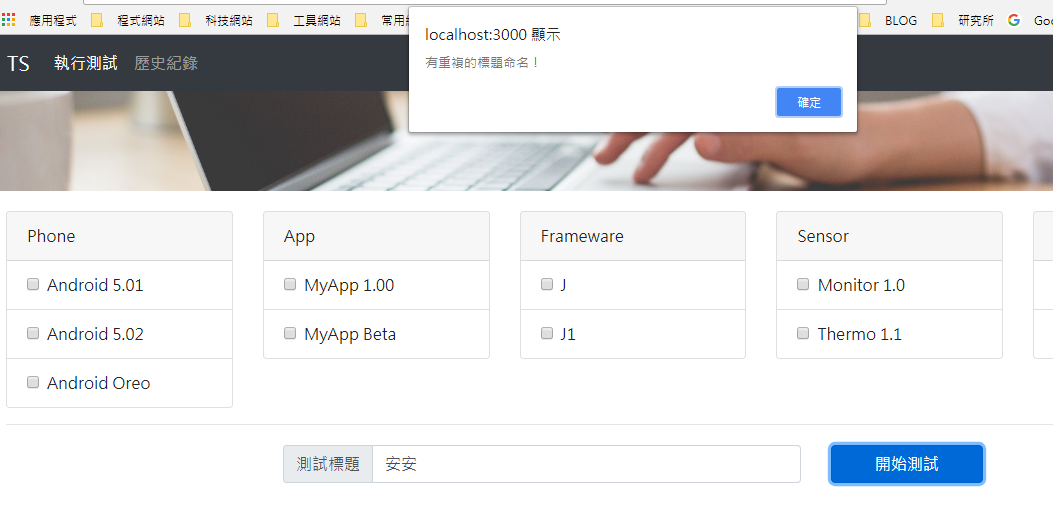


圖5-3 測試名稱重複顯示



圖5-4 歷史紀錄頁面



圖5-5 詳細結果頁面

## 5.2 Source Code Listing

### 5.2.1 RequestHandler.js

var express = require('express');

var router = express.Router();

var testdata = require('../test/testdata');

const TestVersion = require('../lib/TestVersion');

/\* GET home page. \*/

router.get('/', function(req, res, next) {

res.render('index', { title: 'Express' });

});

/\* GET performTests page. \*/

router.get('/performTests', function(req, res, next) {

var categories = testdata.testVersionManager.GetCategories();

var categoryList = new Array();

for (const key in categories) {

var obj = {

category: categories[key],

list: testdata.testVersionManager.GetVersionByCategory(categories[key])

}

categoryList.push(obj);

}

res.render('performTests', { categoryList: categoryList });

});

/\* GET sendTest page. \*/

router.get('/sendTest', function(req, res, next) {

res.render('sendTest', { title: 'Express' });

});

/\* GET historyRecord page. \*/

router.get('/historyRecord', function(req, res, next) {

var missions = testdata.testMissionManager.GetMissions();

res.render('historyRecord', { missions: missions });

});

/\* GET testResults page. \*/

router.get('/testResults', function(req, res, next) {

console.log('missionName = ' + req.query.missionName);

var mission = testdata.testMissionManager.GetMissionByTitle(req.query.missionName);

console.log('mission = ' + mission);

res.render('testResults', { mission: mission, test: JSON.stringify(mission) });

});

/\* GET createMission. \*/

router.get('/createMission', function(req, res, next) {

// console.log('query = ' + JSON.stringify(req.query));

console.log('title = ' + req.query.title);

console.log('phone = ' + req.query.phone);

console.log('app = ' + req.query.app);

console.log('frameware = ' + req.query.frameware);

console.log('sensor = ' + req.query.sensor);

console.log('config = ' + req.query.config);

var phoneList = new Array();

for (const key in req.query.phone) {

phoneList.push(new TestVersion(req.query.phone[key], 'Phone'));

}

var appList = new Array();

for (const key in req.query.app) {

appList.push(new TestVersion(req.query.app[key], 'App'));

}

var framewareList = new Array();

for (const key in req.query.frameware) {

framewareList.push(new TestVersion(req.query.frameware[key], 'Frameware'));

}

var sensorList = new Array();

for (const key in req.query.sensor) {

sensorList.push(new TestVersion(req.query.sensor[key], 'Sensor'));

}

var configList = new Array();

for (const key in req.query.config) {

configList.push(new TestVersion(req.query.config[key], 'Config'));

}

try{

testdata.testMissionManager.CreateMission(req.query.title, phoneList, appList,

framewareList, sensorList, configList);

// 隨機設定幾筆成功以做測試

var thisMission = testdata.testMissionManager.GetMissionByTitle(req.query.title);

for (const key in thisMission.combinations) {

var r = Math.floor(Math.random()\*(2)+0);

if(r==1) thisMission.combinations[key].result = true;

}

}

catch(error){

console.log('error');

}

finally{

res.sendStatus(200);

}

});

module.exports = router;

### 5.2.2 TestVersion.js

class TestVersion{

constructor(name, category){

this.name = name;

this.category = category;

}

GetName(){

return this.name;

}

GetCategory(){

return this.category;

}

}

module.exports = TestVersion;

### 5.2.3 TestVersionManager.js

var Version = require('../lib/testVersion');

class TestVersionManager{

constructor(){

this.versionList = new Array();

}

MakeNewVersion(name, category){

var newVersion = new Version(name, category);

this.versionList.push(newVersion);

}

GetVersionByIndex(index){

return this.versionList[index];

}

GetVersionByNameAndCategory(name, category){

var result = this.versionList.find((item, index, array) => {

return item.name == name && item.category == category;

});

return result;

}

GetVersionByCategory(category){

var result = this.versionList.filter((item, index, array) => {

return item.category == category;

});

return result;

}

GetCategories(){

var categoryList = new Array();

this.versionList.forEach((version, versionIndex, versionArray) => {

if(categoryList.find((category, categoryIndex, categoryArray) => {

return category == version.category; }) === undefined){

categoryList.push(version.category);

};

});

return categoryList;

}

}

module.exports = TestVersionManager;

### 5.2.4 TestCombination.js

class TestCombination{

constructor(versionList){

this.versionList = new Array();

this.versionList = versionList;

this.result = false;

this.report = null;

}

UpdateResult(result, report){

this.result = result;

this.report = report;

}

GetVersionByCategory(category){

var result = this.versionList.find((item, index, array) => {

return item.category == category;

});

return result;

}

GetCategories(){

var categoryList = new Array();

this.versionList.forEach((version, versionIndex, versionArray) => {

categoryList.push(version.GetCategory());

});

return categoryList;

}

GetResult(){

return this.result;

}

GetReport(){

return this.report;

}

}

module.exports = TestCombination;

### 5.2.5 TestMission.js

const TestVersion = require('../lib/testVersion');

const TestCombination = require('../lib/testCombination');

class TestMission{

constructor(title, PhoneVersions, appVersions, FWVersions, ChannelVersions, HWVersions){

this.status = "inital";

this.title = title;

this.startTime = new Date();

this.phoneList = PhoneVersions;

this.appList = appVersions;

this.framewareList = FWVersions;

this.channelList = ChannelVersions;

this.hardwareList = HWVersions;

this.combinations = new Array();

PhoneVersions.forEach((Phone, PhoneIndex, PhoneArray) => {

appVersions.forEach((app, appIndex, appArray) => {

FWVersions.forEach((FW, FWIndex, FWArray) => {

ChannelVersions.forEach((channel, channelIndex, channelArray) => {

HWVersions.forEach((HW, HWIndex, HWArray) => {

var newCombinationList = new Array();

newCombinationList.push(PhoneVersions.length > 0 ? Phone : null);

newCombinationList.push(appVersions.length > 0 ? app : null);

newCombinationList.push(FWVersions.length > 0 ? FW : null);

newCombinationList.push(ChannelVersions.length > 0 ? channel : null);

newCombinationList.push(HWVersions.length > 0 ? HW : null);

var newCombination = new TestCombination(newCombinationList);

this.combinations.push(newCombination);

})

})

});

});

});

}

GetTitle(){

return this.title;

}

GetStartTime(){

return this.startTime;

}

GetCombinationAmount(){

return this.combinations.length;

}

GetStatus(){

return this.status;

}

GetCombinationByIndex(index){

return this.combinations[index];

}

GetSuccessRate(){

var successAmount = 0;

var failAmount = 0;

this.combinations.forEach((combination) => {

if(combination.result)

successAmount++;

else

failAmount++;

});

return Math.round(successAmount / (successAmount + failAmount) \* 100);

}

StartTesting(){

this.status = "testing";

}

ReciveResult(resultList){

resultList.forEach((result, resultIndex, resultArray) => {

});

}

}

module.exports = TestMission;

### 5.2.6 TestMissionManager.js

const TestMission = require("./testMission");

var testdata = require('../test/testdata');

class TestMissionManager{

constructor(){

this.missionList = new Array();

}

CreateMission(title, phones, apps, framewares, sensors, configs){

var result = this.missionList.find((mission, index, array) => {

return mission.title == title;

});

if(result != undefined){

throw new Error("Title repeat.");

}

var newMission = new TestMission(title, phones, apps, framewares, sensors, configs);

this.missionList.push(newMission);

}

GetMissions(){

return this.missionList;

}

GetMissionByTitle(title){

var result = this.missionList.find((mission, index, array) => {

return mission.title == title;

});

return result;

}

}

module.exports = TestMissionManager;

# Unit Testing

## 6.1 Snapshots of testing result

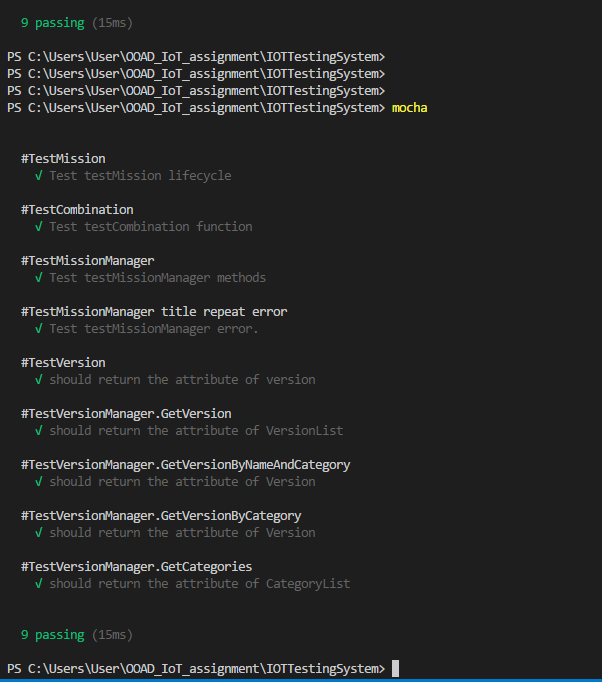


圖6-1 測試畫面

## 6.2 Unit Testing Code Listing

### 6.2.1 test/testVersion.js

const should = require('should');

const TestVersion = require('../lib/testVersion');

var testVersion = new TestVersion("Android 5.01", "SmartPhone");

var testVersionError = new TestVersion();

describe('#TestVersion', () => {

it('should return the attribute of version', done => {

testVersion.GetName().should.equal("Android 5.01");

testVersion.GetCategory().should.equal("SmartPhone");

should.not.exist(testVersionError.GetName());

should.not.exist(testVersionError.GetCategory());

done();

})

});

### 6.2.2 test/testVersionManager.js

const should = require('should');

const TestVersion = require('../lib/testVersion');

const TestVersionManager = require('../lib/testVersionManager');

var testVersionManager = new TestVersionManager();

var testVersionManagerEmpty = new TestVersionManager();

describe('#TestVersionManager.GetVersion', () => {

it('should return the attribute of VersionList', done => {

testVersionManager.MakeNewVersion("Android 5.01", "SmartPhone");

testVersionManager.MakeNewVersion("Android 5.02", "SmartPhone");

testVersionManager.MakeNewVersion("Android Oreo", "SmartPhone");

testVersionManager.MakeNewVersion("J", "Frameware");

testVersionManager.MakeNewVersion("J1", "Frameware");

testVersionManager.MakeNewVersion("Monitor 1.0", "Sensor");

testVersionManager.MakeNewVersion("Thermo 1.1", "Sensor");

testVersionManager.GetVersionByIndex(0).GetName().should.equal("Android 5.01");

testVersionManager.GetVersionByIndex(0).GetCategory().should.equal("SmartPhone");

testVersionManager.GetVersionByIndex(6).GetName().should.equal("Thermo 1.1");

testVersionManager.GetVersionByIndex(6).GetCategory().should.equal("Sensor");

should.not.exist(testVersionManagerEmpty.GetVersionByIndex(0));

done();

})

});

describe('#TestVersionManager.GetVersionByNameAndCategory', () => {

it('should return the attribute of Version', done => {

testVersionManager.GetVersionByNameAndCategory("J", "Frameware").GetName().should.equal("J");

testVersionManager.GetVersionByNameAndCategory("J", "Frameware").GetCategory().should.equal("Frameware");

should.not.exist(testVersionManagerEmpty.GetVersionByNameAndCategory("J", "Frameware"));

done();

})

});

describe('#TestVersionManager.GetVersionByCategory', () => {

it('should return the attribute of Version', done => {

let versionList = testVersionManager.GetVersionByCategory("SmartPhone");

versionList[0].GetCategory().should.equal("SmartPhone");

versionList[0].GetName().should.equal("Android 5.01");

versionList[1].GetName().should.equal("Android 5.02");

versionList[2].GetName().should.equal("Android Oreo");

let versionEmptyList = testVersionManagerEmpty.GetVersionByCategory("SmartPhone");

(versionEmptyList.length > 0).should.not.be.true;

done();

})

});

describe('#TestVersionManager.GetCategories', () => {

it('should return the attribute of CategoryList', done => {

let categoryList = testVersionManager.GetCategories();

categoryList[0].should.equal("SmartPhone");

categoryList[1].should.equal("Frameware");

categoryList[2].should.equal("Sensor");

let categoryEmptyList = testVersionManagerEmpty.GetCategories();

(categoryEmptyList.length > 0).should.not.be.true;

done();

})

});

exports.TestVersionManager = testVersionManager;

exports.TestVersionManagerEmpty = testVersionManagerEmpty;

### 6.2.3 test/testMissionAndCombination.js

const should = require('should');

const TestVersion = require('../lib/testVersion');

const TestMission = require('../lib/testMission');

const TestMissionManager = require('../lib/testMissionManager');

const testdata = require('./testdata');

var testVersionManager = testdata.testVersionManager;

var testMission = new TestMission("myTestMission",

testVersionManager.GetVersionByCategory("Phone"),

testVersionManager.GetVersionByCategory("App"),

testVersionManager.GetVersionByCategory("Frameware"),

testVersionManager.GetVersionByCategory("Sensor"),

testVersionManager.GetVersionByCategory("Config"));

var testMissionManager = testdata.testMissionManager;

describe('#TestMission', () => {

it('Test testMission lifecycle', done => {

testMission.GetTitle().should.equal("myTestMission");

testMission.GetCombinationAmount().should.equal(48);

testMission.GetStatus().should.equal("inital");

testMission.StartTesting();

testMission.GetStatus().should.equal("testing");

testMission.GetSuccessRate().should.equal(0);

done();

})

});

describe('#TestCombination', () => {

it('Test testCombination function', done => {

let categories = testMission.GetCombinationByIndex(0).GetCategories();

categories[0].should.equal("Phone");

categories[1].should.equal("App");

categories[2].should.equal("Frameware");

categories[3].should.equal("Sensor");

categories[4].should.equal("Config");

testMission.GetCombinationByIndex(0).GetVersionByCategory("Phone").GetName().should.equal("Android 5.01");

testMission.GetCombinationByIndex(0).GetVersionByCategory("App").GetName().should.equal("MyApp 1.00");

testMission.GetCombinationByIndex(0).GetVersionByCategory("Frameware").GetName().should.equal("J");

testMission.GetCombinationByIndex(0).GetVersionByCategory("Sensor").GetName().should.equal("Monitor 1.0");

testMission.GetCombinationByIndex(0).GetVersionByCategory("Config").GetName().should.equal("JP");

testMission.GetCombinationByIndex(47).GetVersionByCategory("Phone").GetName().should.equal("Android Oreo");

testMission.GetCombinationByIndex(47).GetVersionByCategory("App").GetName().should.equal("MyApp Beta");

testMission.GetCombinationByIndex(47).GetVersionByCategory("Frameware").GetName().should.equal("J1");

testMission.GetCombinationByIndex(47).GetVersionByCategory("Sensor").GetName().should.equal("Thermo 1.1");

testMission.GetCombinationByIndex(47).GetVersionByCategory("Config").GetName().should.equal("TW");

done();

})

});

describe('#TestMissionManager', () => {

it('Test testMissionManager methods', done => {

var phones = new Array();

var apps = new Array();

var framewares = new Array();

var sensors = new Array();

var configs = new Array();

phones.push(new TestVersion("iOS 10", "Phone"));

apps.push(new TestVersion("app 0.5", "App"));

framewares.push(new TestVersion("alpha", "Frameware"));

sensors.push(new TestVersion("Monitor1", "Sensor"));

configs.push(new TestVersion("CH", "Config"));

testMissionManager.CreateMission("testOtherMission", phones, apps, framewares, sensors, configs);

var missionList = testMissionManager.GetMissions();

missionList.length.should.equal(2);

missionList[1].GetCombinationAmount().should.equal(1);

missionList[1].GetCombinationByIndex(0).GetVersionByCategory("Phone").GetName().should.equal("iOS 10");

testMissionManager.GetMissionByTitle("testOtherMission").should.equal(missionList[1]);

done();

})

});

/\*

describe('#TestMissionManager', () => {

it('Test testMissionManager, and version\'s category less than 5.', done => {

var phones = new Array();

var apps = new Array();

var framewares = new Array();

var sensors = new Array();

var configs = new Array();

apps.push(new TestVersion("app 0.5", "App"));

apps.push(new TestVersion("app 0.7", "App"));

framewares.push(new TestVersion("alpha", "Frameware"));

configs.push(new TestVersion("CH", "Config"));

configs.push(new TestVersion("US", "Config"))

testMissionManager.CreateMission("testSmallMission", phones, apps, framewares, sensors, configs);

var missionList = testMissionManager.GetMissions();

missionList.length.should.equal(3);

missionList[2].GetCombinationAmount().should.equal(4);

missionList[2].GetCombinationByIndex(0).GetVersionByCategory("Phone").GetName().should.equal("app 0.5");

testMissionManager.GetMissionByTitle("testSmallMission").should.equal(missionList[2]);

done();

})

});

\*/

describe('#TestMissionManager title repeat error', () => {

it('Test testMissionManager error.', done => {

try{

testMissionManager.CreateMission("testMission", phones, apps, framewares, sensors, configs);

}

catch(error){

error.message.should.equal("Title repeat.");

}

finally{

done();

}

})

});

### 6.2.4 test/testdata.js

const TestVersionManager = require('../lib/testVersionManager');

const TestMission = require('../lib/testMission');

const TestMissionManager = require('../lib/testMissionManager');

var testVersionManager = new TestVersionManager();

testVersionManager.MakeNewVersion("Android 5.01", "Phone");

testVersionManager.MakeNewVersion("Android 5.02", "Phone");

testVersionManager.MakeNewVersion("Android Oreo", "Phone");

testVersionManager.MakeNewVersion("MyApp 1.00", "App");

testVersionManager.MakeNewVersion("MyApp Beta", "App");

testVersionManager.MakeNewVersion("J", "Frameware");

testVersionManager.MakeNewVersion("J1", "Frameware");

testVersionManager.MakeNewVersion("Monitor 1.0", "Sensor");

testVersionManager.MakeNewVersion("Thermo 1.1", "Sensor");

testVersionManager.MakeNewVersion("JP", "Config");

testVersionManager.MakeNewVersion("TW", "Config");

var testMissionManager = new TestMissionManager();

testMissionManager.CreateMission("針對所有裝置測試",

testVersionManager.GetVersionByCategory("Phone"),

testVersionManager.GetVersionByCategory("App"),

testVersionManager.GetVersionByCategory("Frameware"),

testVersionManager.GetVersionByCategory("Sensor"),

testVersionManager.GetVersionByCategory("Config"));

exports.testVersionManager = testVersionManager;

exports.testMissionManager = testMissionManager;

**Time log #5（Measurement）**：

|  |  |  |  |
| --- | --- | --- | --- |
| 名稱 | 日期 | 時間 | 項目 |
| 林亮勳 | 2018/5/8 | 15:30 ~ 18:30 | Implent program |
| 2018/5/8 | 21:30 ~ 23:30 | Implent program |
| 2018/5/9 | 18:30 ~ 21:30 | Implent program |
| 2018/5/10 | 13:00 ~ 22:00 | Implent program & Doc. |

|  |  |  |  |
| --- | --- | --- | --- |
| 名稱 | 日期 | 時間 | 項目 |
| 林照晟 | 2018/5/8 | 15:30 ~ 18:30 | Implent program |
| 2018/5/8 | 19:00 ~ 22:30 | Implent program |
| 2018/5/9 | 18:30 ~ 22:30 | Implent program |
| 2018/5/10 | 13:00 ~ 22:00 | Implent program & PPT. |