國立台北商業大學

資訊管理系

112．資訊系統專案設計

**系統手冊**

**組別：第112510組**

**題目：iCrawler**

**指導老師：陳信宏**

**組長：10856040鍾皓年**

**組員：10856010郭宗翰 10856030彭鈺程**

**10856031彭鈺達 10856032陳永祥**

**中華民國112年5月10日**

目錄

[第一章 前言 1](#_Toc134191273)

[1-1背景介紹 1](#_Toc134191274)

[1-2動機 1](#_Toc134191275)

[1-3系統目的與目標 1](#_Toc134191276)

[1-4預期成果 1](#_Toc134191277)

[第二章 營運計畫 2](#_Toc134191278)

[2-1 可行性分析 2](#_Toc134191279)

[2-2 商業模式－Business model 3](#_Toc134191280)

[2-3 市場分析－STP 3](#_Toc134191281)

[2-4競爭力分析－SWOT 4](#_Toc134191282)

[第三章 系統規格 5](#_Toc134191283)

[3-1 系統架構 5](#_Toc134191284)

[3-2 系統軟、硬體需求與技術平台 5](#_Toc134191285)

[3-3 開發標準與使用工具 6](#_Toc134191286)

[第四章 專案時程與組織分工 7](#_Toc134191287)

[4-1 專案時程 7](#_Toc134191288)

[4-2 專案組織與分工 8](#_Toc134191289)

[第五章 需求模型 9](#_Toc134191290)

[5-1 功能分解圖（Functional decomposition diagram） 9](#_Toc134191291)

[5-2 需求清單： 10](#_Toc134191292)

[第六章 程序模型 11](#_Toc134191293)

[6-1 資料流程圖(Data flow diagram)。 11](#_Toc134191294)

[6-2 程序規格書(Process specification)。 14](#_Toc134191295)

表目錄

[第二章 營運計畫 2](#_Toc134581901)

[表 3-3-1開發標準與使用工具 4](#_Toc134581902)

[第三章 系統規格 5](#_Toc134581903)

[表 3-2-1系統軟、硬體需求 5](#_Toc134581904)

[表 3-3-1開發標準與使用工具 6](#_Toc134581905)

[第四章 專案時程與組織分工 7](#_Toc134581906)

[表 4-1-1、專案時程 7](#_Toc134581907)

[表4-2-1、專案組織與分工 8](#_Toc134581908)

[第六章 程序模型 11](#_Toc134581910)

[表 6-2-1、程序規格書 委託系統 14](#_Toc134581911)

[表 6-2-2、程序規格書 委託區 14](#_Toc134581912)

[表 6-2-3、程序規格書 分潤 14](#_Toc134581913)

[表 6-2-4、程序規格書 會員系統 15](#_Toc134581914)

[表 6-2-5、程序規格書 客服系統 15](#_Toc134581915)

圖目錄

[第五章 需求模型 9](#_Toc134580817)

[圖 5-1、功能分解圖 9](#_Toc134580818)

[第六章 程序模型 11](#_Toc134580819)

[圖 6-1-1、系統環境圖 11](#_Toc134580820)

[圖 6-1-2、圖0 iCrawler 12](#_Toc134580821)

[圖 6-1-3、圖1委託系統 13](#_Toc134580822)

第一章 前言

1-1背景介紹

在現在這個大數據時代，科技的進步日新月異，不管是5G、大數據等科技都在不斷的成熟。而網路上的資訊量非常龐大，因此需要一種方式來收集、分析和整理這些資料。

1-2動機

隨著互聯網的迅速發展，人們需要大量的資訊來滿足日常生活和工作需求。爬蟲卻可以從網路上獲取大量的資訊，可以有效地提高人們的資訊搜尋效率和準確性。

1-3系統目的與目標

一個爬蟲的軟體，讓一些對資料有需求，且沒有擁有程式語言能力的人能夠使用。

1-4預期成果

方便沒有程式語言能力的人，去做爬蟲功能。

第二章 營運計畫

2-1 可行性分析

市場可行性：提供了一個半自動化的爬蟲工具，讓不懂程式語言的人能夠輕鬆擷取網頁內容。

目標受眾：市場研究專業人士、數據分析師、學生等。

競爭分析：現在的半自動爬蟲的市場無法做重複的爬蟲。

市場增長潛力：隨著更多人需要獲取網絡數據的情況下。大數據和數據分析領域直線成長。

財務可行性：開發這個程式所需的成本，包括軟體開發、維護、伺服器和託管成本，以及可能的市場營銷費用。

預期收入：通過訂閱費、授權費用或廣告方式產生收入。

風險評估：有可能違反智慧財產權。

2-2 商業模式－Business model

基於使用量的模式：計費方式根據客戶的使用量，例如每月爬取的頁面數、爬取的數據量或操作次數。這適用於需要大量爬取的用戶，並根據他們的實際需求來定價。

2-3 市場分析－STP

S： 地域區分：爬蟲可以面向不同國家和地區的市場，全世界都可以用。

統計變數：根據不同收入水平，必須有電腦和網路才能使用。

生活型態：常接觸到電腦、會使用電腦來查詢資料 。

T： 集中行銷：針對特定的利基市場，例如:針對有資料需求的人。

P： 功能利益面：爬蟲網頁的功能利益包括自動化網路數據收集、處理與分析，提高工作效率，減少人力成本，提供即時資訊。

適用情境面：爬蟲網頁只適用於特定行業，例如金融、電子商務、新聞媒體等，可以應用於大量數據的收集、分析、處理等工作。

文化象徵面：爬蟲可以被看作是一個現代化和高效率的工具，這符合現代社會對於科技與效率的追求。

2-4競爭力分析－SWOT

▼表 3-3-1開發標準與使用工具

|  |  |  |
| --- | --- | --- |
| 內部因素  外部因素 | 優勢 (Strengths) | 劣勢 (Weaknesses) |
| 時間節省：協助用戶自動化重複性任務，節省了時間和精力。數據提取：有助於用戶輕鬆從網站上提取數據，進行分析或報告。  可定制性：具有彈性的設置選項，使用戶能夠定制其爬蟲操作以滿足特定需求。 | 網站變化：網站結構的變化可能需要不斷調整和更新爬蟲，使其保持有效。  可能受限於網站規則：部分網站可能設置了反爬蟲機制，導致無法正常收集資料。 |
| 機會 (Opportunities) | SO  (Strengths-Opportunities) | WO  (Weaknesses-Opportunities) |
| 市場需求：非技術人員和企業在數據自動化方面的需求不斷增加，爬蟲工具可以滿足這種需求。數據分析：數據爬蟲可以幫助企業更好地理解市場趨勢、競爭情況和客戶需求。擴大功能：將工具擴展到不同平台，例如移動應用，以滿足更廣泛的用戶需求。 | 基於可定制性和時間節省的優勢，開發更多針對不同行業和用途的爬蟲模板，以滿足市場需求。  利用市場需求增加的機會，擴展銷售，以吸引更多的非技術人員和企業客戶。 | 鑒於網站結構的不斷變化是一個劣勢，建立自動化的網站結構變化檢測功能，以減少調整和更新的需要。  可以與網站所有者的合作，以了解其反爬蟲機制，並提供解決方案以緩解這一劣勢。 |
| 威脅 (Threats)： | ST  (Strengths-Threats) | WT  (Weaknesses-Threats) |
| 競爭：有其他類似的爬蟲工具。  法律風險：由於網絡爬蟲可能觸碰到法律問題，需要嚴謹的規範。 | 基於數據提取和可定制性的優勢，提供教育服務，以幫助用戶更好地應對法律風險和網站規則的挑戰。 | 通過投資於研究和開發，提高軟體的效能和適應性，以應對競爭對手的爬蟲工具。 |

第三章 系統規格

3-1 系統架構



3-2 系統軟、硬體需求與技術平台

▼表 3-2-1系統軟、硬體需求

|  |  |  |
| --- | --- | --- |
| 軟、硬體需求 | | |
| 作業系統版本 | 最低系統需求 | 建議系統需求 |
| Windows 7 以上版本 | Windows 10 以上版本 |
| 處理器磁碟可用空間 | 雙核心以上 | 四核心以上 |
| 1 GB以上可用空間 | 2 GB以上可用空間 |
| RAM、網路 | 4 GB以上可用記憶體 | 8 GB以上可用記憶體 |
| 4G行動網路、Wi-Fi | |

3-3 開發標準與使用工具

▼表 3-3-1開發標準與使用工具

|  |  |
| --- | --- |
| 系統開發環境 | |
| 作業系統 | Win10、Win11 |
| 開發平台 | Python |
| 應用程式 | Visual Studio Code、 |
| 程式開發工具 | |
| 前端 | HTML、CSS、JavaScript |
| 後端 | Python |
| 文件美工工具 | |
| 文件 | Microsoft word |
| 簡報 | Microsoft PowerPoint、Canva |
| 圖樣 |  |
| 專案管理平台 | |
| 專案管理 | GitHub、GitKraken |
| 檔案存放 | MySQL、GitHub |

第四章 專案時程與組織分工

4-1 專案時程

▼表 4-1-1、專案時程

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **年** | **111年** | | | | | | | | | | |
| **月** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 系統發想 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 工具學習 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 前端設計 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 前端開發 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 資料庫設計 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 資料庫建置 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 後端設計 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 後端開發 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 系統整合 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 系統測試 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 手冊製作 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Logo製作 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| PPT製作 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

4-2 專案組織與分工

▼表4-2-1、專案組織與分工

●主要負責人 〇次要負責人

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 項目/組員 | | 10856041  鍾皓年 | 10856010  郭宗翰 | 10856030  彭鈺程 | 10856031  彭鈺達 | 10856032  陳永祥 |
| 後端開發 | selenium |  | ● |  | 〇 |  |
| 資料比對 | 〇 |  | ● |  |  |
| 伺服器架設 | ● |  |  |  |  |
| webengine | ● |  |  | 〇 |  |
| 腳本錄製 |  |  | 〇 | ● |  |
| Xpath | ● |  |  | 〇 |  |
| 登入/註冊系統 | ● |  | 〇 |  |  |
| 資料庫 |  | ● |  |  |  |
| 瀏覽器資料處理 |  |  |  | ● |  |
| 資料處理 | ● |  |  |  |  |
| 資料傳送 | ● |  |  |  |  |
| 資料下載 |  |  | ● |  |  |
| Bug處理 |  | ● |  |  |  |
| 統整 | ● |  |  |  |  |
| 前端開發 | 登入/註冊介面 |  |  | ● |  |  |
| 瀏覽器介面 |  |  |  | ● |  |
| 查詢介面 |  |  |  | ● |  |
| 錄製介面 |  | ● |  |  |  |
| 爬蟲介面 |  |  | ● |  |  |
| 下載介面 |  |  |  | ● |  |
| 美術設計 | UI |  |  |  | ● |  |
| Logo設計 |  |  |  |  | ● |
| 色彩設計 |  |  |  | ● |  |
| 文件撰寫 | 統整 |  |  | ● |  |  |
| 第1章 前言 |  | ● |  |  |  |
| 第2章 營運計畫 |  |  | 〇 | ● |  |
| 第3章 系統規格 |  |  | ● |  |  |
| 第4章專題時程與組織分工 |  | 〇 | ● |  |  |
| 第5章 需求模型 |  |  | ● |  |  |
| 第6章 程序模型 |  | ● |  |  |  |
| 第7章 資料模型 |  |  | ● | 〇 |  |
| 第8章 資料庫設計 |  |  |  | ● |  |
| 第9章 程式 |  |  | ● |  |  |
| 第10章 測試模型 |  |  | ● |  |  |
| 第11章 操作手冊 |  |  | ● |  |  |
| 第12章 使用手冊 |  |  | ● |  |  |
| 項目/組員 | | 10856041  鍾皓年 | 10856010  郭宗翰 | 10856030  彭鈺程 | 10856031  彭鈺達 | 10856032  陳永祥 |
| 報告 | 簡報製作 |  |  | ● |  |  |
| 影片製作 | ● |  |  |  |  |
| 海報製作 |  |  | ● |  |  |

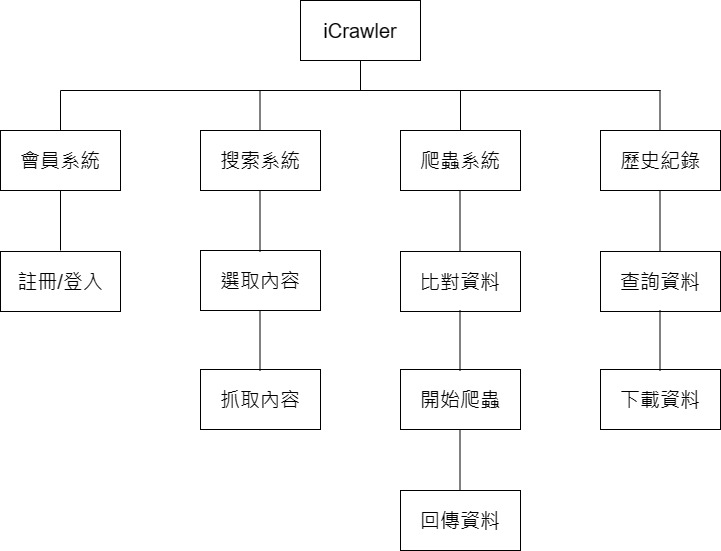
▼表4-2-2、專案成果工作內容與貢獻表

|  |  |  |  |
| --- | --- | --- | --- |
| 序號 | 姓名 | 工作內容<各限100字以內> | 貢獻度 |
| 1 | 組長  鍾皓年 | 伺服器架設、webengine、Xpath、登入/註冊系統、資料處理、資料傳、統整系統  影片製作 | 30 % |
| 2 | 組員  郭宗翰 | selenium、資料庫、Bug處理、錄製介面  文件製作第1章、第6章 | 25 % |
| 3 | 組員  彭鈺程 | 資料比對、資料下載、登入/註冊介面、爬蟲介面  文件製作第3章、第4章、第5章、第7章、第9章、第10章  、第11章、第12章及統整  簡報製作、海報製作 | 20 % |
| 4 | 組員  彭鈺達 | 腳本錄製、瀏覽器資料處理、瀏覽器介面、查詢介面、下載介面、UI、色彩設計  文件製作第2章、第8章 | 24 % |
| 5 | 組員  陳永祥 | Logo設計 | 1 % |
|  | | | 總計:100% |

第五章 需求模型

5-1 功能分解圖（Functional decomposition diagram）

▼圖 5-1、功能分解圖



5-2 需求清單：

**功能需求**

會員系統

1.1 使用者可以註冊和登入帳號及密碼

搜索系統

2.1 選取反白內容進行抓取

爬蟲系統

3.1 讀取抓取的資料，根據標籤進行比對內容

3.2根據比對內容去爬蟲

3.3回傳爬蟲資料

歷史紀錄

4.1 將使用者爬取的資料儲存在資料庫，以供使用者查看和分析

**非功能需求**

合法性

5.1 遵守相關的法規和法律，並提供提示，以防止非法或不當的使用

第六章 程序模型

6-1 資料流程圖(Data flow diagram)。

▼圖 6-1-1、系統環境圖



▼圖 6-1-2、圖0 iCrawler

▼圖 6-1-3、圖1 委託系統



6-2 程序規格書(Process specification)。

▼表 6-2-1、程序規格書 會員帳號系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 1 | **功能名稱** | 創建帳號密碼 |
| **操作說明** | 新增帳號、密碼、使用者名稱 | | |
| **輸入值** | 帳號、密碼、使用者名稱 | | |
| **輸出值** | 帳號、密碼、使用者名稱 | | |

▼表 6-2-2、程序規格書 登入、登出系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 1.1 | **功能名稱** | 會員登入、登出 |
| **操作說明** | 登入、登出帳號 | | |
| **輸入值** | 帳號、密碼 | | |
| **輸出值** | 帳號、密碼 | | |

▼表 6-2-3、程序規格書 搜索系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 2 | **功能名稱** | 搜索系統 |
| **操作說明** | 多次欲爬蟲的內容 | | |
| **輸入值** | 反白內容 | | |
| **輸出值** | 反白內容的xpath | | |

▼表 6-2-4、程序規格書 爬蟲系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 3 | **功能名稱** | 爬蟲系統 |
| **操作說明** | 透過得知反內容的標籤進行比對後，進行爬蟲 | | |
| **輸入值** | 反白內容 | | |
| **輸出值** | 爬蟲結果 | | |

▼表 6-2-5、程序規格書 比對系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 3.1 | **功能名稱** | 標籤比對 |
| **操作說明** | 讀取抓取的資料，根據標籤進行比對 | | |
| **輸入值** | 反白內容的標籤 | | |
| **輸出值** | 反白內容的xpath | | |

▼表 6-2-6、程序規格書 爬蟲系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 3.2 | **功能名稱** | 爬蟲 |
| **操作說明** | 讀取比對結果，進行爬蟲 | | |
| **輸入值** | 反白內容的xpath | | |
| **輸出值** | 爬蟲結果 | | |

▼表 6-2-7、程序規格書 回傳爬蟲系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 3.3 | **功能名稱** | 回傳爬蟲資料 |
| **操作說明** | 把爬好的資料傳入資料庫 | | |
| **輸入值** | 爬蟲資料 | | |
| **輸出值** | 爬蟲資料 | | |

▼表 6-2-8、程序規格書 回傳爬蟲系統

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | 3.3.1 | **功能名稱** | 查看爬蟲資料 |
| **操作說明** | 查看存放在資料庫的爬蟲資料 | | |
| **輸入值** |  | | |
| **輸出值** | 爬蟲資料 | | |

第七章 資料模型

7-1 實體關聯圖(Entity relationship diagram)。



7-2 資料字典(Data dictionary)。

▼表 7-2-1、資料字典

|  |  |  |
| --- | --- | --- |
| **資料表編號** | **資料表名稱** | **資料表名稱(中文)** |
| **T01** | **scraping** | **爬蟲** |
| **T02** | **user** | **使用者** |

第八章 資料庫設計

8-1 資料庫關聯表(Constraints)。

8-2 表格及其Meta data

▼表 8-2-1、T01 Meta data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T01 scraping 爬蟲資料 | | | | |
| 欄位名稱 | 欄位中文名稱 | 資料型態 | 資料長度 | 主索引 |
| number | 資料流水號 | INT |  | Y |
| user\_id | 使用者ID | INT |  |  |
| scrap\_time | 爬取時間 | DATE |  |  |
| scrap\_data | 爬蟲資料 | LONGTEXT |  |  |

▼表 8-2-2、T02 Meta data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T02 user 使用者資料 | | | | |
| 欄位名稱 | 欄位中文名稱 | 資料型態 | 資料長度 | 主索引 |
| user\_id | 使用者ID | INT |  | Y |
| account | 帳號 | VARCHAR | 15 |  |
| password | 密碼 | VARCHAR | 15 |  |
| username | 使用者名稱 | VARCHAR | 25 |  |

第九章 程式

9-1 軟體架構與程式清單。

▼表 9-1-1、前端程式清單

|  |  |  |
| --- | --- | --- |
| **編號** | **檔案名稱** | **功能說明** |
| **1** | **main.py** | **初始登入畫面** |
| **2** | **main.py** | **註冊畫面** |
| **3** | **整合.py** | **主瀏覽器** |
| **4** | **整合.py** | **爬蟲需求畫面** |
| **5** | **table.py** | **爬取資料的資料表** |

▼表 9-1-2、後端程式清單

|  |  |  |
| --- | --- | --- |
| **編號** | **檔案名稱** | **功能說明** |
| **1** | **Backend\_wiring\_login.py** | **登入資料傳送** |
| **2** | **Backend\_wiring\_register.py** | **註冊資料功能** |
| **3** | **Backend\_wiring\_select.py** | **表格資料傳送** |
| **4** | **Backend\_wiring\_upload.py** | **更新表格資料傳送** |
| **5** | **Backend\_wiring\_Xpath.py** | **爬取資料傳送** |
| **6** | **server.py** | **伺服器** |
| **7** | **selectdata.py** | **查詢系統連接資料庫功能** |
| **8** | **upload\_result.py** | **爬蟲系統連接資料庫功能** |
| **9** | **register\_system.py** | **註冊系統連接資料庫功能** |
| **10** | **login\_system.py** | **登入系統連接資料庫功能** |
| **11** | **data\_Comparison.py** | **比對系統連接資料庫功能** |
| **12** | **整合.py** | **爬蟲功能** |
| **13** | **整合.py** | **Xpath功能** |

▼表 9-2-1、前端程式規格描述

|  |  |
| --- | --- |
| 程式名稱 | main.py |
| 目的 | 初始登入畫面 |
| 部分程式碼 | |
| class LoginDialog(QDialog):  def \_\_init\_\_(self, communicator):  super().\_\_init\_\_()  self.communicator = communicator  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_)) # 在這裡定義 script\_dir  self.get\_login\_state = ""  self.initUI()  self.username = ""  def initUI(self):  layout = QVBoxLayout()  # 帳號欄位  account\_layout = QHBoxLayout()  self.account\_label = QLabel('帳號', self)  self.account\_textbox = QLineEdit(self)  account\_layout.addWidget(self.account\_label)  account\_layout.addWidget(self.account\_textbox)  layout.addLayout(account\_layout)  # 密碼欄位  password\_layout = QHBoxLayout()  self.password\_label = QLabel('密碼', self)  self.password\_textbox = QLineEdit(self)  password\_layout.addWidget(self.password\_label)  password\_layout.addWidget(self.password\_textbox)  layout.addLayout(password\_layout)    # 水平布局來放置返回按鈕和確定按鈕  buttons\_layout = QHBoxLayout()  # 登入按鈕  login\_button = QPushButton('登入', self)  login\_button.setFixedSize(80, 30)  login\_button.clicked.connect(self.onLoginButtonClicked) # 使用包裝後的函數  # 註冊按鈕  register\_button = QPushButton('註冊', self)  register\_button.clicked.connect(self.open\_register\_page)  register\_button.setFixedSize(80, 30)  # 將兩個按鈕放入水平佈局中  buttons\_layout.addWidget(login\_button)  buttons\_layout.addWidget(register\_button)  buttons\_layout.addStretch(1)  layout.addLayout(buttons\_layout)  layout.addWidget(register\_button)  self.setLayout(layout)  self.setGeometry(500, 100, 500, 300) # 設視窗大小及位置  self.setWindowTitle('登入')  self.show()    def open\_register\_page(self):  register\_page = RegisterPage(self, self.communicator)  register\_page.exec()  #有可能有亂碼  def onLoginButtonClicked(self):  account = self.account\_textbox.text()  password = self.password\_textbox.text()    if account == "" or password == "":  login\_success\_popup = login\_password(self, self.communicator, self.account\_textbox, self.password\_textbox)  login\_success\_popup.exec()  return    response = subprocess.run(["python", os.path.join(self.script\_dir, "Backend\_wiring\_login.py"), account, password], stdout=subprocess.PIPE)  try:  decoded2\_response = response.stdout.decode('gbk')  except UnicodeDecodeError:  decoded2\_response = response.stdout.decode('utf-8', errors='replace')  evaluated\_data = eval(decoded2\_response)    if evaluated\_data == False:  self.get\_login\_state = "False"  else:  if isinstance(evaluated\_data, tuple) and len(evaluated\_data) == 2:  result = [str(evaluated\_data[0])]  sublist = evaluated\_data[1]  if sublist and len(sublist) > 0:  inner\_tuple = sublist[0]  result.extend(map(str, inner\_tuple))  self.get\_login\_state = result  if self.get\_login\_state[0] == "True":  self.username = self.get\_login\_state[1]  self.user\_id = self.get\_login\_state[2]  login\_success\_popup = login\_yes(self, self.communicator, self.account\_textbox, self.password\_textbox, self.username, self.user\_id)  login\_success\_popup.exec()  else:  login\_success\_popup = login\_no(self, self.communicator, self.account\_textbox, self.password\_textbox)  login\_success\_popup.exec() | |

|  |  |
| --- | --- |
| 程式名稱 | main.py |
| 目的 | 註冊畫面 |
| 部分程式碼 | |
| class RegisterPage(QDialog):  def \_\_init\_\_(self, loginDialog, communicator):  super().\_\_init\_\_()  self.loginDialog = loginDialog  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))  self.communicator = communicator  self.get\_register\_state = ""  self.initUI()  def initUI(self):  layout = QVBoxLayout()  # 帳號欄位  account\_layout = QHBoxLayout()  self.account\_label = QLabel('帳號', self)  self.account\_textbox = QLineEdit(self)  account\_layout.addWidget(self.account\_label)  account\_layout.addWidget(self.account\_textbox)  layout.addLayout(account\_layout)  # 密碼欄位  password\_layout = QHBoxLayout()  self.password\_label = QLabel('密碼', self)  self.password\_textbox = QLineEdit(self)  password\_layout.addWidget(self.password\_label)  password\_layout.addWidget(self.password\_textbox)  layout.addLayout(password\_layout)  # 確認密碼欄位  confirm\_password\_layout = QHBoxLayout()  self.confirm\_password\_label = QLabel('確認密碼', self)  self.confirm\_password\_textbox = QLineEdit(self)  confirm\_password\_layout.addWidget(self.confirm\_password\_label)  confirm\_password\_layout.addWidget(self.confirm\_password\_textbox)  layout.addLayout(confirm\_password\_layout)  # 暱稱欄位  username\_layout = QHBoxLayout()  self.username\_label = QLabel('暱稱', self)  self.username\_textbox = QLineEdit(self)  username\_layout.addWidget(self.username\_label)  username\_layout.addWidget(self.username\_textbox)  layout.addLayout(username\_layout)    # 水平布局來放置返回按鈕和確定按鈕  buttons\_layout = QHBoxLayout()  # 確定按鈕  yes\_button = QPushButton('確定', self)  yes\_button.setFixedSize(80, 30)  yes\_button.clicked.connect(self.onRegisterButtonClicked)  # 將按鈕放入水平佈局中  buttons\_layout.addWidget(yes\_button)  buttons\_layout.addStretch(1)  layout.addLayout(buttons\_layout)  self.setLayout(layout)  self.setGeometry(500, 100, 500, 300) # 設視窗大小及位置  self.setWindowTitle('註冊')  self.show()  def return\_to\_login(self):  self.close()  self.loginDialog.show()  def onRegisterButtonClicked(self):  account = self.account\_textbox.text()  password = self.password\_textbox.text()  confirm\_password = self.confirm\_password\_textbox.text()  username = self.username\_textbox.text()  #驗證密碼是否相同  if (password != confirm\_password):  register\_confirm\_password\_popup = Register\_confirm\_password(self, self.communicator, self.account\_textbox, self.password\_textbox, self.username\_textbox)  register\_confirm\_password\_popup.exec()  else:  response = subprocess.run(["python", os.path.join(self.script\_dir, "Backend\_wiring\_register.py"), account, password, username], stdout=subprocess.PIPE)  decoded1\_response = response.stdout.decode()  self.get\_register\_state = decoded1\_response  if self.get\_register\_state[0:4] == "True":  register\_success\_popup = Register\_yes(self, self.communicator, self.account\_textbox, self.password\_textbox, self.username\_textbox)  register\_success\_popup.exec()  else:  register\_success\_popup = Register\_no(self, self.communicator, self.account\_textbox, self.password\_textbox, self.confirm\_password\_textbox, self.username\_textbox)  register\_success\_popup.exec() | |

|  |  |
| --- | --- |
| 程式名稱 | 整合.py |
| 目的 | 主瀏覽器 |
| 部分程式碼 | |
| class WebBrowserWindow(QMainWindow):  def \_\_init\_\_(self, main\_window, username, user\_id):  super().\_\_init\_\_()  self.main\_window = main\_window  self.drivers = None  self.selected\_xpath = None  self.selected\_xpath = []  self.selected\_button\_xpath = None  self.xpath = None  self.scraping\_in\_progress = False  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))  self.init\_ui()  self.username = username  self.user\_id = user\_id  def init\_ui(self):  # 設置窗口標題和圖示  self.setWindowTitle('網頁瀏覽器')  # self.setWindowIcon(QIcon('icons/penguin.png'))  self.resize(1200, 800)  self.setStyleSheet('''  QPushButton {  background-color: #008CBA;  color: white;  border: none;  padding: 5px 10px;  margin: 2px;  }  ''')  # 創建 Web 瀏覽器視窗  self.browser = QWebEngineView()  self.browser.setUrl(QUrl('https://www.google.com.tw'))  self.setCentralWidget(self.browser)  # 設置 URL 地址欄  self.urlbar = QLineEdit()  self.urlbar.returnPressed.connect(self.navigate\_to\_url)  # 監聽網頁 URL 變化事件  self.browser.urlChanged.connect(self.renew\_urlbar)  # 創建用於顯示選擇的文本的文本編輯框  self.selected\_text\_editor = QTextEdit(self)  self.selected\_text\_editor.setReadOnly(True)  self.selected\_text\_editor.setFixedHeight(100)  # 監聽網頁選擇文本事件  self.browser.selectionChanged.connect(self.handle\_selection\_changed)  # 添加以下代碼以記錄 URL 變化事件  self.browser.urlChanged.connect(self.log\_url\_change)  # 創建記錄反白文本和元素資訊的按鈕  self.record\_highlight\_button = QPushButton("反白文字", self)  self.record\_highlight\_button.clicked.connect(self.show\_element\_info)  self.record\_highlight\_button.setFixedSize(80, 30)  # 創建傳送資料按鈕  self.send\_xpath\_button = QPushButton("反白比對", self)  self.send\_xpath\_button.clicked.connect(self.send\_xpath\_to\_server) # 設定按鈕點擊事件處理函數  self.send\_xpath\_button.setFixedSize(80, 30)  self.send\_xpath\_button.setEnabled(False) # 初始狀態設為不可用  self.send\_xpath\_button.setStyleSheet("background-color: #CCCCCC; color: #555555;") # 灰色樣式  # 創建返回按鈕  self.back\_button = QPushButton("返回上頁", self)  self.back\_button.clicked.connect(self.browser.back)  self.back\_button.setFixedSize(80, 30)  # 創建查詢資料按鈕  self.serch\_button = QPushButton("查詢資料", self)  self.serch\_button.clicked.connect(self.open\_table)  self.serch\_button.setFixedSize(80, 30)  # 創建開始爬蟲按鈕  self.scraping\_button = QPushButton("開始爬蟲", self)  self.scraping\_button.clicked.connect(ScrapingDialog)  self.scraping\_button.setFixedSize(80, 30)  self.scraping\_button.setEnabled(False) # 初始狀態設為不可用  self.scraping\_button.setStyleSheet("background-color: #CCCCCC; color: #555555;") # 灰色樣式  # 創建登出按鈕  self.logout\_button = QPushButton("登出", self)  self.logout\_button.clicked.connect(self.logout)  self.logout\_button.setFixedSize(80, 30)  # 用戶名標籤  self.account\_label = QLabel(f'用戶名:{username}', self)  self.user\_id = QLabel(f'用戶id:{user\_id}', self)  # 創建一個水平佈局並將按鈕添加到其中  button\_layout = QHBoxLayout()  button\_layout.addWidget(self.back\_button)  button\_layout.addWidget(self.record\_highlight\_button)  button\_layout.addWidget(self.send\_xpath\_button)  button\_layout.addWidget(self.scraping\_button)  button\_layout.addStretch(1) #將按鈕推到左邊  button\_layout.addWidget(self.user\_id)  button\_layout.addWidget(self.account\_label)  button\_layout.addWidget(self.serch\_button)  button\_layout.addWidget(self.logout\_button)    # 主佈局包含其他小部件和水平佈局  layout = QVBoxLayout()  layout.addWidget(self.urlbar)  layout.addWidget(self.browser)  layout.addLayout(button\_layout)  central\_widget = QWidget()  central\_widget.setLayout(layout)  self.setCentralWidget(central\_widget) | |

|  |  |
| --- | --- |
| 程式名稱 | 整合.py |
| 目的 | 爬蟲需求畫面 |
| 部分程式碼 | |
| # 爬蟲需求畫面  class ScrapingDialog(QDialog):  scraping\_done\_signal = QtCore.pyqtSignal() # 定義一個信號，用於通知爬蟲操作已完成\  def \_\_init\_\_(self, main\_window, browser\_window = None):  super().\_\_init\_\_()  self.browser\_window = browser\_window  self.main\_window = main\_window  self.scraped\_data = []  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))  self.initUI()  def initUI(self):  layout = QVBoxLayout()  scraping\_layout = QVBoxLayout()  self.scraping\_label = QLabel('需要爬幾頁:', self)  self.scraping\_textbox = QLineEdit(self)  scraping\_layout.addWidget(self.scraping\_label)  scraping\_layout.addWidget(self.scraping\_textbox)  buttons\_layout = QHBoxLayout()  scraping\_button = QPushButton('確認', self)  scraping\_button.setFixedSize(80, 30)  scraping\_button.clicked.connect(self.scrape\_data)  buttons\_layout.addStretch(1)  buttons\_layout.addWidget(scraping\_button)  layout.addLayout(scraping\_layout)  layout.addLayout(buttons\_layout)  layout.addStretch()  self.setLayout(layout)    self.setGeometry(500, 100, 200, 100)  self.setWindowTitle('爬取需求')  # 執行爬蟲操作  def scrape\_data(self):  self.close()  repeat\_count = int(self.scraping\_textbox.text())  if self.browser\_window.scraping\_in\_progress:  return # 如果已經有爬蟲操作在運行，則不執行新的操作  self.browser\_window.scraping\_in\_progress = True # 設定標誌變數，表示爬蟲操作正在進行中  self.browser\_window.scraping\_button.setEnabled(False)  self.browser\_window.scraping\_button.setText("正在爬蟲...")  self.browser\_window.drivers = webdriver.Chrome()  self.browser\_window.drivers.maximize\_window()  self.browser\_window.drivers.get(self.browser\_window.browser.url().toString())  def scrape\_in\_thread():  # 初始化迴圈索引  n = 1  while True:  if self.browser\_window.drivers is None:  break # WebDriver 會話已經關閉，退出迴圈    # 構造標題的 XPATH  xpath = self.browser\_window.xpath + f"[{n}]"  try:  # 將 xpath 字符串轉換為 By.XPATH 對象  xpath\_locator = (By.XPATH, xpath)  # 使用 find\_element\_by\_xpath 找到標題元素  title\_element = self.browser\_window.drivers.find\_element(\*xpath\_locator)  # 獲取標題文本  title\_text = title\_element.text  # 顯示標題  self.scraped\_data.append(f"第{n}筆: {title\_text}")  # 增加迴圈索引  n += 1  except NoSuchElementException:  # 找不到元素時退出迴圈  break  except StaleElementReferenceException:  # 元素過時，重新查找元素  continue  # 開始重複執行爬取  try:  for i in range(1, repeat\_count+1):  # 在單獨的線程中執行爬蟲操作  self.scraped\_data.append(f"第{i}頁: ")  self.scraping\_thread = threading.Thread(target=scrape\_in\_thread)  self.scraping\_thread.start()  try:  #滾動至頁面最底  self.scroll\_to\_bottom()  try:  # 尋找並點擊下一頁按鈕  actions = ActionChains(self.browser\_window.drivers)  actions.move\_by\_offset((int(self.main\_window.xoffset) + 300), (int(self.main\_window.yoffset) - 50)).click().perform()  # 初始滑鼠座標至(0, 0)  actions.move\_by\_offset(-(int(self.main\_window.xoffset) + 300), -(int(self.main\_window.yoffset) - 50)).perform()  except:  pass  i += 1  except:  pass  except Exception as e:  print(f"發生錯誤：{e}")  finally:  # 關閉瀏覽器  with self.browser\_window.drivers as driver:  driver.quit()  # 爬蟲結果上傳DB  self.upload\_result()  # 爬蟲完成後，使用信號更新 UI  self.scraping\_done\_signal.emit()  self.browser\_window.scraping\_button.setText("爬蟲完成")  time.sleep(4)  self.browser\_window.scraping\_button.setText("開始爬蟲") # 恢復按鈕文字  self.browser\_window.scraping\_button.setEnabled(True) # 啟用「爬蟲」按鈕  self.browser\_window.scraping\_in\_progress = False # 重置標誌變數，表示爬蟲操作已完成  def scroll\_to\_bottom(self):  # 使用 JavaScript 模擬滾輪滑動到底部  self.browser\_window.drivers.execute\_script("window.scrollTo(0, document.body.scrollHeight);")  time.sleep(2) # 等待一段時間，讓新資料加載完成  def upload\_result(self):  subprocess.run(['python', os.path.join(self.script\_dir, 'Backend\_wiring\_upload.py'), str(self.browser\_window.user\_id), str(self.scraped\_data)], stdout=subprocess.PIPE)  self.scraped\_data = [] | |

|  |  |
| --- | --- |
| 程式名稱 | table.py |
| 目的 | 爬取資料的資料表 |
| 部分程式碼 | |
| class MainWindow(QMainWindow):  def \_\_init\_\_(self, username, user\_id):  super().\_\_init\_\_()  self.setWindowTitle("資料表")  self.setGeometry(500, 100, 1200, 800)  self.username = username  self.user\_id = user\_id  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))  self.model = TableModel()  table\_view = QTableView()  # 設置水平標題為等寬模式  table\_view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeMode.Stretch)  table\_view.setModel(self.model)  central\_widget = QWidget()  layout = QVBoxLayout(central\_widget)  # 垂直佈局包裹水平佈局和按鈕  v\_layout = QVBoxLayout()  v\_layout.addWidget(table\_view)  self.select\_button = QPushButton("查詢")  self.select\_button.setFixedSize(80, 30)  self.select\_button.setStyleSheet("background-color: #008CBA;")  self.select\_button.clicked.connect(self.select\_button\_clicked)  # 新增下載按鈕  self.download\_button = QPushButton("下載")  self.download\_button.setFixedSize(80, 30)  self.download\_button.setEnabled(False) # 初始狀態設為不可用  self.download\_button.setStyleSheet("background-color: #CCCCCC; color: #555555;")  self.download\_button.clicked.connect(self.download\_button\_clicked)  account\_label = QLabel(f'用戶名 : {username}', self)  user\_id\_label = QLabel(f'用戶id : {user\_id}', self)    # 將按鈕添加到水平佈局  h\_layout = QHBoxLayout()  h\_layout.addWidget(account\_label)  h\_layout.addWidget(user\_id\_label)  h\_layout.addStretch(1) #將按鈕推到右邊  h\_layout.addWidget(self.select\_button)  h\_layout.addWidget(self.download\_button)    # 添加水平佈局到垂直佈局  v\_layout.addLayout(h\_layout)  layout.addLayout(v\_layout)  self.setCentralWidget(central\_widget)  def download\_button\_clicked(self):  global rows  # 詢問使用者輸入檔案名稱  while True:  file\_name, ok\_pressed = QInputDialog.getText(self, "輸入檔案名稱", "請輸入檔案名稱:")  if not ok\_pressed:  return # 使用者按下取消，終止下載  if not file\_name:  print("檔案名稱不能為空")  continue # 檔案名稱為空，重新詢問  # 加上 .csv 副檔名  csv\_filename = f"{file\_name}.csv"  if os.path.exists(csv\_filename):  # 使用 QMessageBox 提醒使用者  msg = QMessageBox()  msg.setIcon(QMessageBox.Icon.Warning)  msg.setText(f"檔案 '{csv\_filename}' 已存在，請選擇其他名稱。")  msg.setWindowTitle("檔案已存在")  msg.exec()  else:  break # 檔案不存在，退出迴圈  try:  with open(csv\_filename, 'w', newline='', encoding='utf-8') as csvfile:  csv\_writer = csv.writer(csvfile)  # 寫入標題  csv\_writer.writerow(headers)  # 寫入資料  csv\_writer.writerows(rows)  print(f"資料已成功下載到 {csv\_filename}")  except Exception as e:  print(f"下載資料時發生錯誤: {e}")  def select\_button\_clicked(self, download\_button):  global rows  try:  response = subprocess.run(  ["python", os.path.join(self.script\_dir, "Backend\_wiring\_select.py"), self.user\_id],  stdout=subprocess.PIPE,  text=True,  timeout=60  )  # 使用 eval 將字符串轉換為真正的列表  rows\_str = response.stdout.strip()  rows = eval(rows\_str)  # 更新模型的資料  self.model.layoutAboutToBeChanged.emit()  self.model.rows = rows  self.model.layoutChanged.emit()  except subprocess.CalledProcessError as e:  print(f"發生錯誤: {e}")  self.download\_button.setEnabled(True) # 啟用下載按鈕  self.download\_button.setStyleSheet("") # 移除樣式，恢復預設外觀 | |

▼表 9-2-2、後端程式規格描述

|  |  |
| --- | --- |
| 程式名稱 | Backend\_wiring\_login.py |
| 目的 | 登入/登出資料傳送 |
| 部分程式碼 | |
| import sys  import argparse  import asyncio  import websockets  async def send\_data(account, password):  uri = "ws://140.131.114.149:80"  async with websockets.connect(uri) as websocket:  message = f"login, {account}, {password}"  await websocket.send(message)  response = await websocket.recv()  return response  async def main\_async(account, password):  response = await send\_data(account, password)  print(f"{response}")  # 在這裡進行其他的操作  async def main():  parser = argparse.ArgumentParser(description="Backend wiring with WebSocket")  parser.add\_argument("account", type=str, help="Account for authentication")  parser.add\_argument("password", type=str, help="Password for authentication")  args = parser.parse\_args()    # 使用 gather 等待 main\_async 完成  await asyncio.gather(main\_async(args.account, args.password))  if \_\_name\_\_ == '\_\_main\_\_':  # 使用 asyncio.run 來運行 main 函數  asyncio.run(main()) | |

|  |  |
| --- | --- |
| 程式名稱 | Backend\_wiring\_register.py |
| 目的 | 註冊資料傳送 |
| 部分程式碼 | |
| import sys  import argparse  import asyncio  import websockets  async def send\_data(account, password, username):  uri = "ws://140.131.114.149:80" # 請替換為虛擬機的 IP 地址和端口  async with websockets.connect(uri) as websocket:  message = f"register, {account}, {password}, {username}"  await websocket.send(message)  response = await websocket.recv()  return response  async def main\_async(account, password, username):  response = await send\_data(account, password, username)  print(f"{response}")  # 在這裡進行其他的操作  def main():  # 處理命令列參數  parser = argparse.ArgumentParser(description="Backend wiring with WebSocket")  parser.add\_argument("account", type=str, help="Account for authentication")  parser.add\_argument("password", type=str, help="Password for authentication")  parser.add\_argument("username", type=str, help="Username for authentication")  args = parser.parse\_args()  # 在這裡使用 args.account 和 args.password 進行相應的操作  loop = asyncio.get\_event\_loop()  loop.run\_until\_complete(main\_async(args.account, args.password, args.username))  if \_\_name\_\_ == '\_\_main\_\_':  main() | |

|  |  |
| --- | --- |
| 程式名稱 | Backend\_wiring\_select.py |
| 目的 | 表格資料傳送 |
| 部分程式碼 | |
| import sys  import argparse  import asyncio  import websockets  from websockets.exceptions import ConnectionClosedError  async def send\_data(user\_id):  uri = "ws://140.131.114.149:80"  try:  async with websockets.connect(uri) as websocket:  message = f"select, {user\_id}"  await websocket.send(message)  response = await websocket.recv()  return response  except ConnectionClosedError as e:  print(f"WebSocket connection closed unexpectedly: {e}")  except Exception as e:  print(f"An error occurred: {e}")  async def main\_async(user\_id):  response = await send\_data(user\_id)  print(f"{response}")  # 在這裡進行其他的操作  async def main():  parser = argparse.ArgumentParser(description="Backend wiring with WebSocket")  parser.add\_argument("user\_id", type=str, help="user\_id for authentication")  args = parser.parse\_args()    # 使用 gather 等待 main\_async 完成  await asyncio.gather(main\_async(args.user\_id))  if \_\_name\_\_ == '\_\_main\_\_':  # 使用 asyncio.run 來運行 main 函數  asyncio.run(main()) | |

|  |  |
| --- | --- |
| 程式名稱 | Backend\_wiring\_Xpath.py |
| 目的 | 爬取資料傳送 |
| 部分程式碼 | |
| import sys  import argparse  import asyncio  import websockets  async def send\_data(xpath):  uri = "ws://140.131.114.149:80" # 請替換為虛擬機的 IP 地址和端口  async with websockets.connect(uri) as websocket:  message = f"xpath, {xpath}"  await websocket.send(message)  response = await websocket.recv()  return response  async def main\_async(xpath):  response = await send\_data(xpath)  print(f"{response}")  # 在這裡進行其他的操作  async def websocket\_handler(websocket, path):  async for message in websocket:  print(f"Received message: {message}")  # 处理消息并发送响应  def main():  # 處理命令列參數  parser = argparse.ArgumentParser(description="Backend wiring with WebSocket")  parser.add\_argument("xpath", type=str, help="Account for authentication")    args = parser.parse\_args()  # 在這裡使用 args.account 和 args.password 進行相應的操作  loop = asyncio.get\_event\_loop()  loop.run\_until\_complete(main\_async(args.xpath))  if \_\_name\_\_ == '\_\_main\_\_':  main() | |

|  |  |
| --- | --- |
| 程式名稱 | server.py |
| 目的 | 伺服器 |
| 部分程式碼 | |
| import asyncio  import websockets  import json  from MemberSystem import login\_system, register\_system  from Comparison import data\_Comparison  from Select\_Data import selectdata  from ScrapSystem import upload\_result  connected\_clients = set()  async def handle\_connection(websocket, path):  # 當有新的 WebSocket 連接建立時，這個函數將被呼叫    while True:  try:  data = await websocket.recv() # 接收來自客戶端的數據  # 在這裡處理收到的數據  data\_list = data.split(", ")  # 初始化 response 變數  response = "Invalid request" # 或者使用其他適當的預設值  user\_id = ""    # 判斷類別  if data\_list[0] == 'login':  response = str(login\_system.login(data\_list[1], data\_list[2]))  # user\_id = login\_system.login\_id(data\_list[1], data\_list[2])  elif data\_list[0] == 'register':  response = str(register\_system.register(data\_list[1], data\_list[2], data\_list[3]))  elif data\_list[0] == 'xpath':  data\_list.pop(0)  dou\_data = []  dou\_data.append(data\_list)  # 呼叫 data\_Comparison.data\_process 並將結果傳給 response  cleaned\_data = [item.replace('"[', "").replace('"]', "").strip() for item in dou\_data[0]]  result = data\_Comparison.data\_process(cleaned\_data)  # print(result)  response = str(result)  elif data\_list[0] == 'select':  response = str(selectdata.select\_user\_id(data\_list[1]))  elif data\_list[0] == 'upload':  scrap\_data = []  for item in data\_list[2:]:  scrap\_data.append(item)  response = upload\_result.upload\_scrape\_data(int(data\_list[1]), str(scrap\_data))  response = str(response)  if user\_id != "":  await websocket.send(response, user\_id)  else:  await websocket.send(response)  except websockets.exceptions.ConnectionClosedError:  break  except websockets.exceptions.ConnectionClosedOK:  break    # 休眠一小段時間，以減少 CPU 使用率  await asyncio.sleep(0.1)  start\_server = websockets.serve(handle\_connection, "140.131.114.149", 80)  asyncio.get\_event\_loop().run\_until\_complete(start\_server)  asyncio.get\_event\_loop().run\_forever() | |

|  |  |
| --- | --- |
| 程式名稱 | selectdata.py |
| 目的 | 查詢系統連接資料庫功能 |
| 部分程式碼 | |
| import mysql.connector  def select\_user\_id(user\_id):  try:  db = mysql.connector.connect(  host="140.131.114.242",  port="3306",  user="admin112510",  password="@aA0937404883",  database="112-webpy"  )  cursor = db.cursor()  # 檢查帳號和密碼的 SQL 語句  sql = "SELECT user\_id,scrap\_time,scrap\_data FROM scraping WHERE user\_id = %s" # 修正 SQL 語句  val = (user\_id,)  cursor.execute(sql, val)  results = cursor.fetchall() # 获取所有符合条件的行  if results:  # 如果结果不为空，返回结果列表  return results  else:  # 如果结果为空，回傳 ["False", None]  return ["False", None]  except Exception as e:  # 處理例外狀況，例如資料庫連接問題  print(f"Error: {e}")  return ["Error", None]  finally:  db.close() | |

|  |  |
| --- | --- |
| 程式名稱 | upload\_result.py |
| 目的 | 爬蟲系統連接資料庫功能 |
| 部分程式碼 | |
| import mysql.connector  from datetime import datetime  def upload\_scrape\_data(user\_id, scraped\_data):  try:  db = mysql.connector.connect(  host="140.131.114.242",  port="3306",  user="admin112510",  password="@aA0937404883",  database="112-webpy"  )  cursor = db.cursor()  scrape\_time = datetime.now().strftime("%Y-%m-%d")  # 插入資料的 SQL 指令  sql = "INSERT INTO scraping (user\_id, scrap\_time, scrap\_data) VALUES (%s, %s, %s)"  val = (user\_id, scrape\_time, scraped\_data)  try:  # 執行插入資料的指令  cursor.execute(sql, val)  # 提交變更  db.commit()  except mysql.connector.Error as err:  # 錯誤處理  print(f"錯誤：{err}")  except Exception as e:  # 處理例外狀況，例如資料庫連接問題  print(f"Error: {e}")  return ["Error", None]  finally:  db.close()  cursor.close() | |

|  |  |
| --- | --- |
| 程式名稱 | register\_system.py |
| 目的 | 註冊系統連接資料庫功能 |
| 部分程式碼 | |
| import mysql.connector  def register(account, password, username):  try:  db = mysql.connector.connect(  host="140.131.114.242",  port="3306",  user="admin112510",  password="@aA0937404883",  database="112-webpy"  )  cursor = db.cursor()  sql = "INSERT INTO user (account, password, username) VALUES (%s, %s, %s)"  val = (account, password, username)  cursor.execute(sql, val)  db.commit()  return "True"  except:  return "False" | |

|  |  |
| --- | --- |
| 程式名稱 | login\_system.py |
| 目的 | 登入系統連接資料庫功能 |
| 部分程式碼 | |
| import mysql.connector  def login(account, password):  try:  db = mysql.connector.connect(  host="140.131.114.242",  port="3306",  user="admin112510",  password="@aA0937404883",  database="112-webpy"  )  cursor = db.cursor()  # 檢查帳號和密碼的 SQL 語句  sql = "SELECT \* FROM user WHERE account = %s AND password = %s"  val = (account, password)  cursor.execute(sql, val)  result = cursor.fetchall()  # 如果帳號和密碼匹配，再從資料庫中獲取使用者名稱  if result:  user\_sql = "SELECT username, user\_id FROM user WHERE account = %s AND password = %s"  cursor.execute(user\_sql, (account, password))  result\_user = cursor.fetchall()  if result\_user:  # 如果使用者名稱存在，回傳 "True" 和使用者名稱  return "True", result\_user  else:  # 如果使用者名稱不存在，回傳 "False"  return "False"  else:  # 如果帳號和密碼不匹配，回傳 "False"  return "False"  finally:  db.close() | |

|  |  |
| --- | --- |
| 程式名稱 | data\_Comparison.py |
| 目的 | 比對系統連接資料庫功能 |
| 部分程式碼 | |
| import ast  def compare\_and\_display(data):  # Step 1: 將取得的資料切割開來  data\_list = ast.literal\_eval(data)  # Step 2: 將每筆資料分組，進行比較並儲存相異部分  differences = []  for i in range(len(data\_list) - 1):  group1 = data\_list[i]  group2 = data\_list[i + 1]  # 比較兩個組的不同部分，顯示具體差異  diff = ''.join(f'+{ord(c2) - ord(c1)}' if c1 != c2 else c1 for c1, c2 in zip(group1, group2))  differences.append(diff)  # Step 3: 將所有相異的資料進行相減，並顯示出來  result = '\n'.join(differences)  print(result)  return result  def data\_process(data):  data\_str = ','.join(data)  re\_data = compare\_and\_display(data\_str)  return re\_data | |

|  |  |
| --- | --- |
| 程式名稱 | 整合.py |
| 目的 | 爬蟲功能 |
| 部分程式碼 | |
| # 爬蟲需求畫面  class ScrapingDialog(QDialog):  scraping\_done\_signal = QtCore.pyqtSignal() # 定義一個信號，用於通知爬蟲操作已完成\  def \_\_init\_\_(self, main\_window, browser\_window = None):  super().\_\_init\_\_()  self.browser\_window = browser\_window  self.main\_window = main\_window  self.scraped\_data = []  self.script\_dir = os.path.dirname(os.path.realpath(\_\_file\_\_))  self.initUI()  def initUI(self):  layout = QVBoxLayout()  scraping\_layout = QVBoxLayout()  self.scraping\_label = QLabel('需要爬幾頁:', self)  self.scraping\_textbox = QLineEdit(self)  scraping\_layout.addWidget(self.scraping\_label)  scraping\_layout.addWidget(self.scraping\_textbox)  buttons\_layout = QHBoxLayout()  scraping\_button = QPushButton('確認', self)  scraping\_button.setFixedSize(80, 30)  scraping\_button.clicked.connect(self.scrape\_data)  buttons\_layout.addStretch(1)  buttons\_layout.addWidget(scraping\_button)  layout.addLayout(scraping\_layout)  layout.addLayout(buttons\_layout)  layout.addStretch()  self.setLayout(layout)    self.setGeometry(500, 100, 200, 100)  self.setWindowTitle('爬取需求')  # 執行爬蟲操作  def scrape\_data(self):  self.close()  repeat\_count = int(self.scraping\_textbox.text())  if self.browser\_window.scraping\_in\_progress:  return # 如果已經有爬蟲操作在運行，則不執行新的操作  self.browser\_window.scraping\_in\_progress = True # 設定標誌變數，表示爬蟲操作正在進行中  self.browser\_window.scraping\_button.setEnabled(False)  self.browser\_window.scraping\_button.setText("正在爬蟲...")  self.browser\_window.drivers = webdriver.Chrome()  self.browser\_window.drivers.maximize\_window()  self.browser\_window.drivers.get(self.browser\_window.browser.url().toString())  def scrape\_in\_thread():  # 初始化迴圈索引  n = 1  while True:  if self.browser\_window.drivers is None:  break # WebDriver 會話已經關閉，退出迴圈    # 構造標題的 XPATH  xpath = self.browser\_window.xpath + f"[{n}]"  try:  # 將 xpath 字符串轉換為 By.XPATH 對象  xpath\_locator = (By.XPATH, xpath)  # 使用 find\_element\_by\_xpath 找到標題元素  title\_element = self.browser\_window.drivers.find\_element(\*xpath\_locator)  # 獲取標題文本  title\_text = title\_element.text  # 顯示標題  self.scraped\_data.append(f"第{n}筆: {title\_text}")  # 增加迴圈索引  n += 1  except NoSuchElementException:  # 找不到元素時退出迴圈  break  except StaleElementReferenceException:  # 元素過時，重新查找元素  continue  # 開始重複執行爬取  try:  for i in range(1, repeat\_count+1):  # 在單獨的線程中執行爬蟲操作  self.scraped\_data.append(f"第{i}頁: ")  self.scraping\_thread = threading.Thread(target=scrape\_in\_thread)  self.scraping\_thread.start()  try:  #滾動至頁面最底  self.scroll\_to\_bottom()  try:  # 尋找並點擊下一頁按鈕  actions = ActionChains(self.browser\_window.drivers)  actions.move\_by\_offset((int(self.main\_window.xoffset) + 300), (int(self.main\_window.yoffset) - 50)).click().perform()  # 初始滑鼠座標至(0, 0)  actions.move\_by\_offset(-(int(self.main\_window.xoffset) + 300), -(int(self.main\_window.yoffset) - 50)).perform()  except:  pass  i += 1  except:  pass  except Exception as e:  print(f"發生錯誤：{e}")  finally:  # 關閉瀏覽器  with self.browser\_window.drivers as driver:  driver.quit()  # 爬蟲結果上傳DB  self.upload\_result()  # 爬蟲完成後，使用信號更新 UI  self.scraping\_done\_signal.emit()  self.browser\_window.scraping\_button.setText("爬蟲完成")  time.sleep(4)  self.browser\_window.scraping\_button.setText("開始爬蟲") # 恢復按鈕文字  self.browser\_window.scraping\_button.setEnabled(True) # 啟用「爬蟲」按鈕  self.browser\_window.scraping\_in\_progress = False # 重置標誌變數，表示爬蟲操作已完成  def scroll\_to\_bottom(self):  # 使用 JavaScript 模擬滾輪滑動到底部  self.browser\_window.drivers.execute\_script("window.scrollTo(0, document.body.scrollHeight);")  time.sleep(2) # 等待一段時間，讓新資料加載完成  def upload\_result(self):  subprocess.run(['python', os.path.join(self.script\_dir, 'Backend\_wiring\_upload.py'), str(self.browser\_window.user\_id), str(self.scraped\_data)], stdout=subprocess.PIPE)  self.scraped\_data = [] | |

|  |  |
| --- | --- |
| 程式名稱 | 整合.py |
| 目的 | xpath功能 |
| 部分程式碼 | |
| def show\_element\_info(self):  selected\_text = self.main\_window.selected\_text\_to\_record  if selected\_text:  js\_code = '''  function getPathTo(element) {  var path = [];  while (element !== null && element.nodeType === 1) {  var name = element.tagName.toLowerCase();  var siblings = element.parentNode.childNodes;  var index = 1;  for (var i = 0; i < siblings.length; i++) {  var sibling = siblings[i];  if (sibling === element) {  path.unshift(name + '[' + index + ']');  break;  }  if (sibling.nodeType === 1 && sibling.tagName.toLowerCase() === name) {  index++;  }  }  element = element.parentNode;  }  return path.join('/');  }  var selectedText = window.getSelection().toString();  var element = window.getSelection().anchorNode.parentElement;  var path = getPathTo(element);  selectedText + ' , {' + path + '}';  '''  self.browser.page().runJavaScript(js\_code, self.handle\_js\_call) | |

第十章 測試模型

10-1 測試計畫：

第十一章 操作手冊

第十二章 使用手冊

第十三章 感想

10856040鍾皓年:這次專題我學到了管理底下組員的方法和管理專案時程的方 法，如何讓組員有效的使用時間與靈活運用知識非常有挑戰性。

10856010郭宗翰:

10856030彭鈺程:在這次專題中讓我了解到團隊合作的重要性，大家都要按照組長安排的去做，不然大家都會像一盤散沙，各做各的。每個人都有擅長的和不擅長的地方，我們要相互扶持，這樣做事的效率才會更高。在專題的過程中，我還學習到了很多新知識，像是爬蟲、js，這些都是學校沒有教的，我們必須自己學習，也非常感謝指導老師和中研院的博士在百忙之中抽出時間來指導我們和我們一起面臨問題。

10856031彭鈺達:這次的專題非常有挑戰性，學習新的知識，配合團隊中其他成員的時間，還要一起花時間討論，確實能增進我的團隊合作，並讓我的程式設計更進一步，更的是絞盡腦汁寫程式的當下感受，debug讓人麻木的感覺，一點也不想再體驗，只能說非常感謝我們的組長，在我寫不出來時去煩他的包容心，也感謝他可以幫我找出bug在哪，最感謝中研院的老師願意花時間幫我們看我們的專題，提供想法。

第十四章 參考資料

https://www.youtube.com/watch?v=KBLQ7GJLlQE&t=453s

https://tomchen.me/2019/09/03/Python/Python/%5BPython%5D%20%E7%88%AC%E8%9F%B2%E7%AD%86%E8%A8%983-%20Selenium/#%E7%8D%B2%E5%BE%97%E5%85%83%E7%B4%A0%E7%9A%84%E7%9B%B8%E9%97%9C%E8%A8%8A%E6%81%AF

https://www.cadch.com/modules/news/article.php?storyid=198

https://stackoverflow.com/questions/59175008/qhboxlayout-size-resize-move

https://badgameshow.com/steven/python/python-string-split/

<https://jimmy-huang.medium.com/python%E4%B9%8Bwebsocket%E4%BB%8B%E7%B4%B9%E8%88%87%E5%AF%A6%E4%BD%9C-8ec2474badaa>

<https://steam.oxxostudio.tw/category/python/pyqt6/pyqt6_pyqt5.html>

<https://zx7978123.medium.com/python-%E8%B7%A8%E6%AA%94%E6%A1%88%E5%85%A8%E5%9F%9F%E6%80%A7%E8%AE%8A%E6%95%B8-9a7740b71cea>

<https://www.delftstack.com/zh-tw/howto/python/python-import-variable-from-another-file/>

<https://www.learncodewithmike.com/2021/10/pandas-compare-values-between-dataframes.html>

https://stackoverflow.com/questions/57813303/how-to-get-html-of-a-page-loaded-in-qwebengineview

https://ithelp.ithome.com.tw/articles/10249159

https://www.programcreek.com/python/example/97321/PyQt5.QtWebEngineWidgets.QWebEngineView

https://qtwebkit.github.io/doc/qtwebkit/qwebview.html

**附錄**

|  |  |
| --- | --- |
| **評審建議事項** | **修正情形** |
| **1.加強相關的競爭分析** | **已在文件修改** |
| **2.加強可行性評估** | **已在文件修改** |
| **3.缺少財務可行性的數據** | **因更改題目以刪除** |
| **4.登入畫面的結果的比例不對** | **已修改程式讓畫面協調** |
| **5.計費方式已決定由委託人決定** | **因更改題目以刪除** |
| **6.費用部分可考慮信用卡預繳款（非實際 支付）** | **因更改題目以刪除** |