國　立　台　北　商　業　大　學

資訊管理系

1　1　2　·　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　1　1　2　年　1　1　月　2　2　日

目錄

表目錄

圖目錄

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。

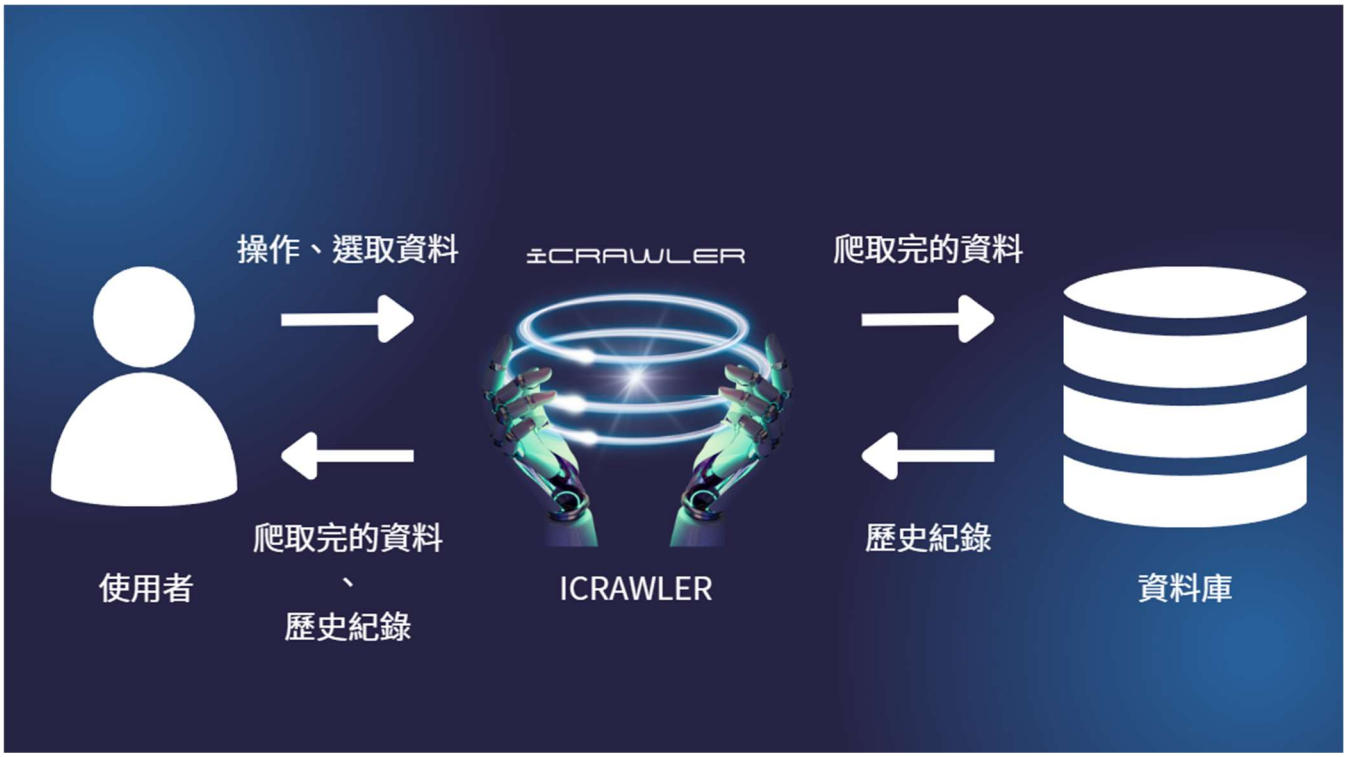
2-4 競爭力分析－SWOT

▼表 2-4-1 SWOT 分析

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



▲圖 3-1-1 系統架構圖

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	Python、QT
後端	Python、JavaScript
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	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　○次要負責人

項目/組員		10856040 鍾皓年	10856010 郭宗翰	10856030 彭鈺程	10856031 彭鈺達	10856032 陳永祥
後端開發	selenium		●		○	
	爬蟲結果上傳資料庫		●			
	資料比對	○		●		
	伺服器架設	●				
	webengine	●			○	
	腳本錄製			○	●	
	Xpath	●			○	
	登入/註冊系統	●		○		
	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			○	●	
	第 3 章 系統規格			●		
	第 4 章 專題時程與組織分工		○	●		
	第 5 章 需求模型			●		

項目/組員		10856041 鍾皓年	10856010 郭宗翰	10856030 彭鈺程	10856031 彭鈺達	10856032 陳永祥
文件撰寫	第 6 章 程序模型		●			
	第 7 章 資料模型			●	○	
	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		●			
報告	簡報製作			●		
	影片製作	●				
	海報製作			●		

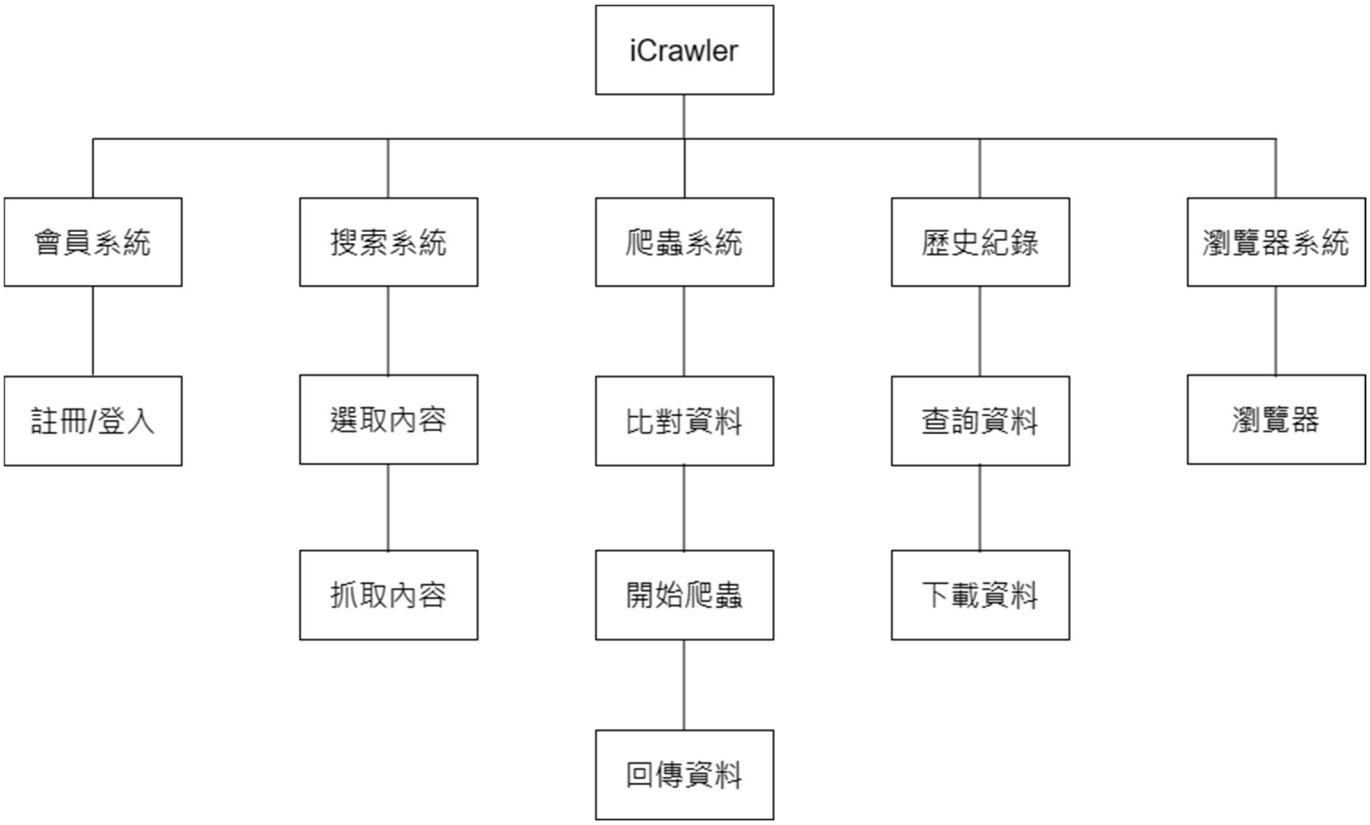
▼表 4-2-2 專案成果工作內容與貢獻表

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、資料傳送、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、第 12 章及文件統整、檢查	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 5-1-1 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

瀏覽器系統

2.1 讓使用者在瀏覽器上進行瀏覽及反白

搜索系統

3.1 選取反白內容進行抓取

爬蟲系統

4.1 讀取抓取的資料，根據標籤進行比對內容

4.2 根據比對內容去爬蟲

4.3 回傳爬蟲資料

歷史紀錄

5.1 將使用者爬取的資料儲存在資料庫，以供使用者查看及下載

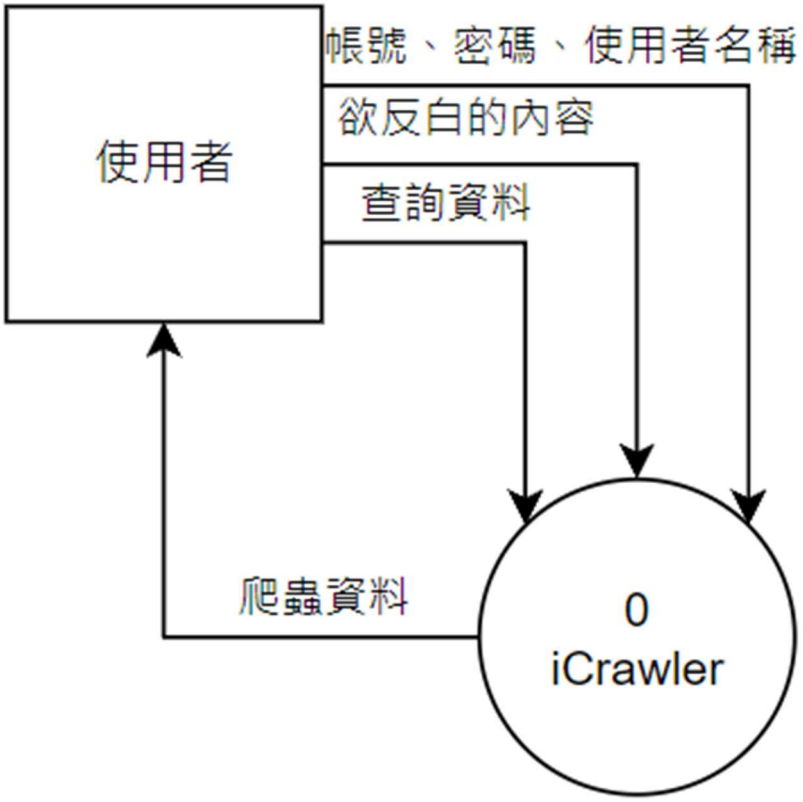
非功能需求

合法性

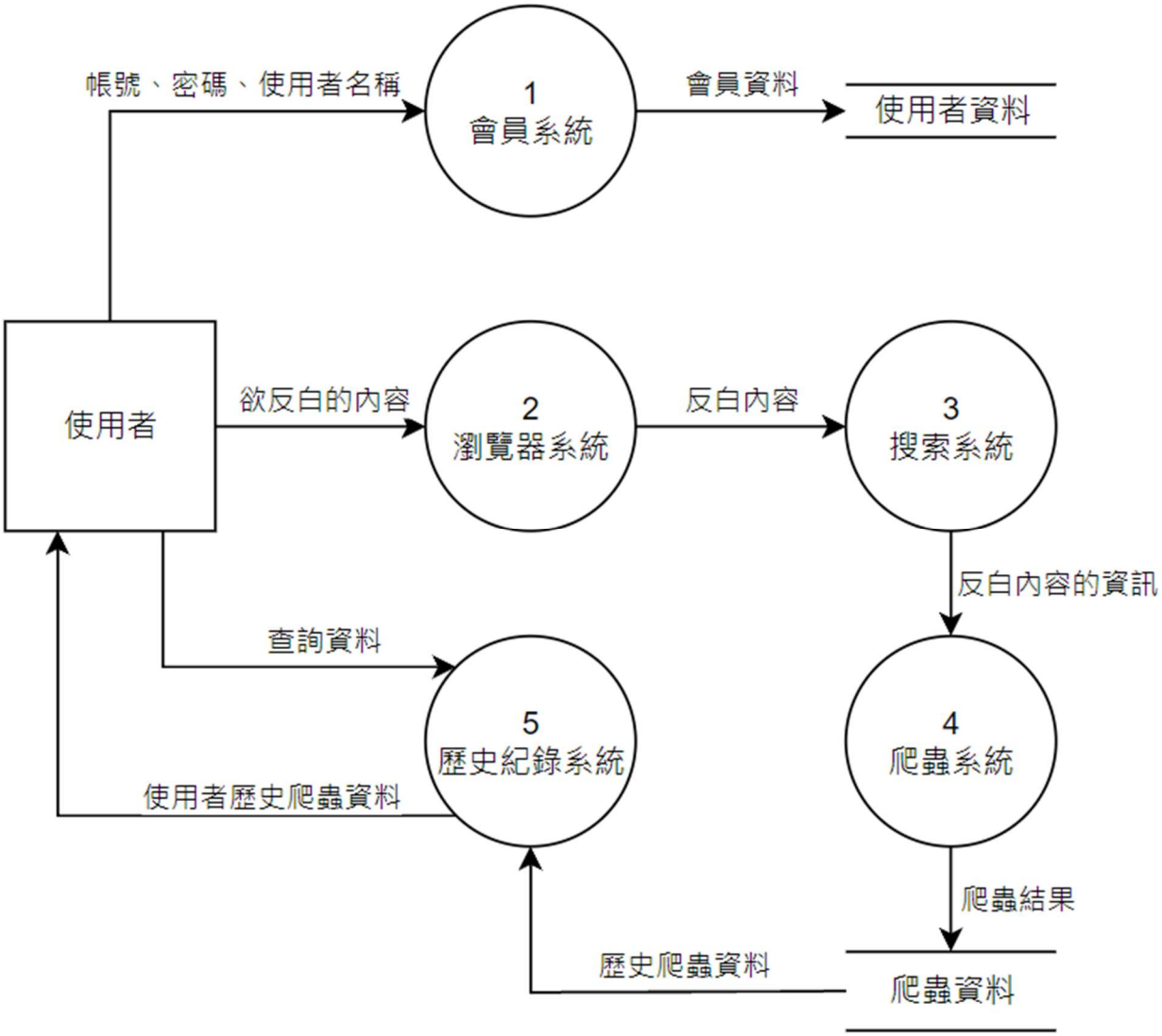
5.1 遵守相關的法規和法律，並提供提示，以防止非法或不當的使用

第六章 程序模型

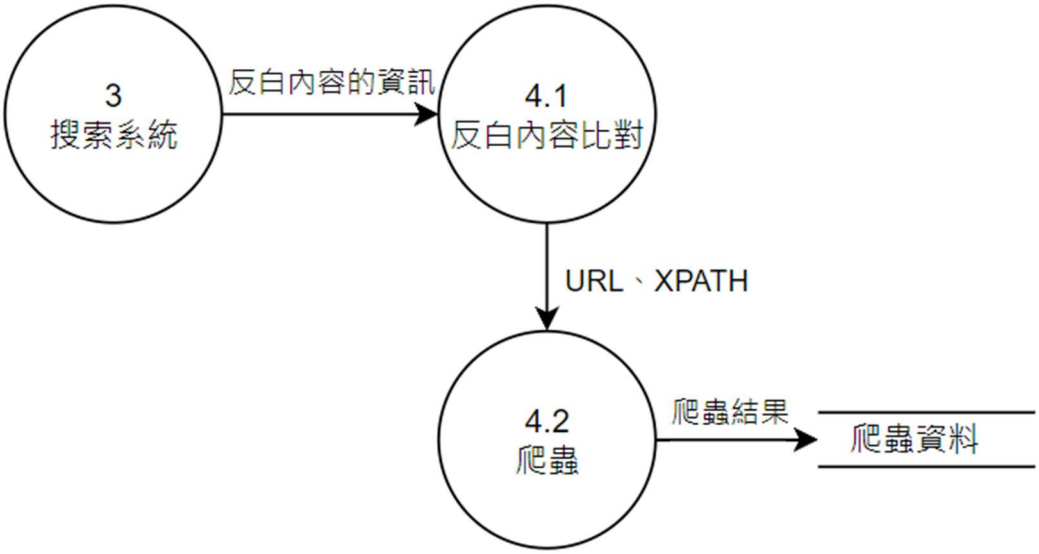
6-1 資料流程圖(Data flow diagram)。



▲圖 6-1-1 系統環境圖



▲圖 6-1-2 圖 0 iCrawler



▲圖 6-1-3 圖 1 爬蟲系統

6-2 程序規格書(Process specification)。

▼表 6-2-1 程序規格書 會員系統

編號	1	功能名稱	會員系統
操作說明	註冊帳號、登入帳號		
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 瀏覽器系統

編號	2	功能名稱	瀏覽器系統
操作說明	操作瀏覽器進行瀏覽及反白		
輸入值	欲反白的內容		
輸出值	反白內容		

▼表 6-2-3 程序規格書 搜索系統

編號	3	功能名稱	搜索系統
操作說明	搜索使用者反白內容的標籤、URL		
輸入值	反白內容		
輸出值	反白內容的資訊		

▼表 6-2-4 程序規格書 爬蟲系統

編號	4	功能名稱	爬蟲系統
操作說明	透過得知反白內容的標籤進行比對後，進行爬蟲		
輸入值	反白內容的資訊		
輸出值	爬蟲結果		

▼表 6-2-5 程序規格書 反白內容比對

編號	4.1	功能名稱	反白內容比對
操作說明	讀取抓取的資料，根據標籤進行比對		
輸入值	反白內容的資訊		
輸出值	URL、XPATH		

▼表 6-2-6 程序規格書 爬蟲

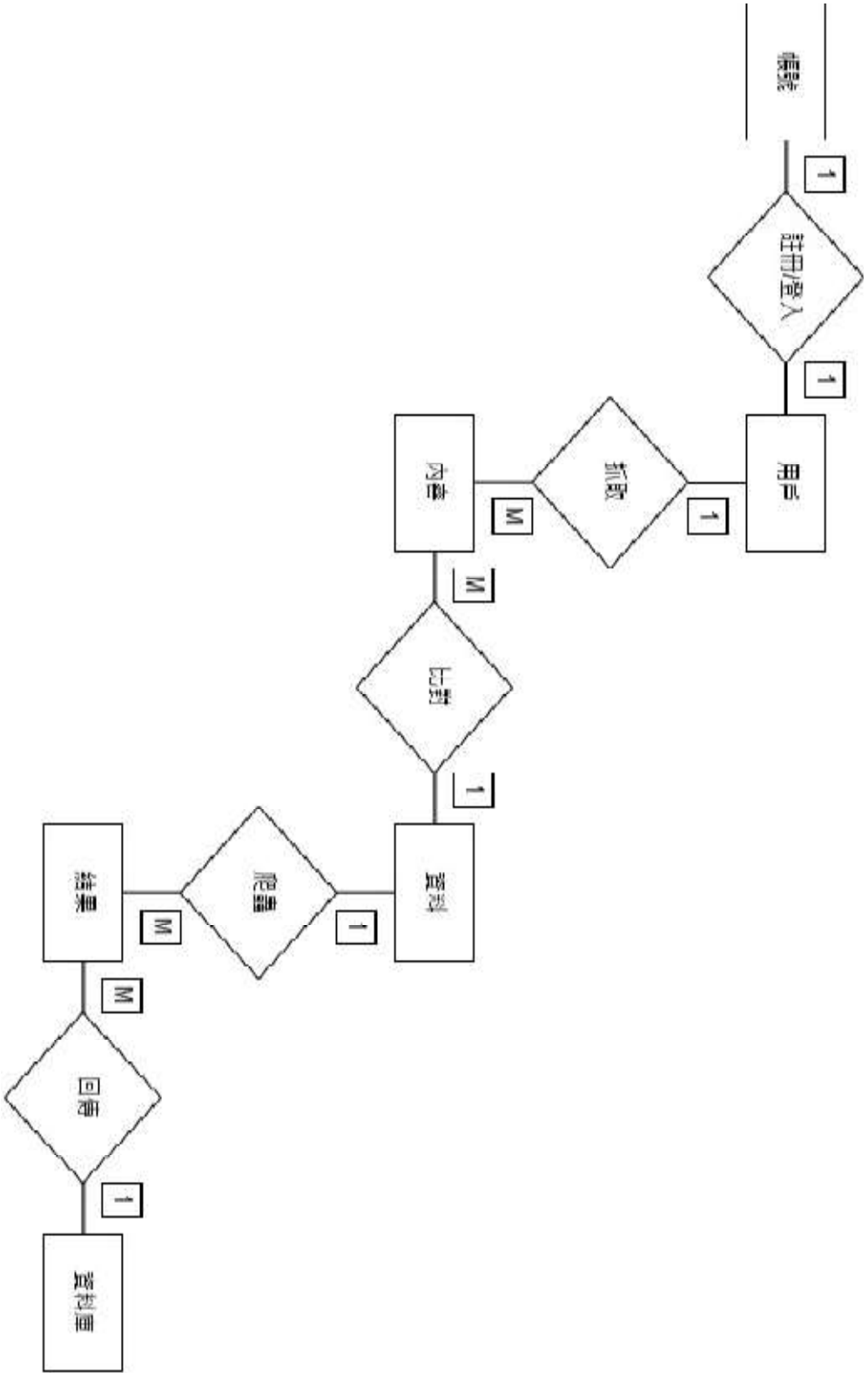
編號	4.2	功能名稱	爬蟲
操作說明	讀取比對結果，進行爬蟲		
輸入值	URL、XPATH		
輸出值	爬蟲結果		

▼表 6-2-7 程序規格書 歷史紀錄系統

編號	5	功能名稱	歷史紀錄系統
操作說明	讓使用者查詢自己的歷史資料，以供使用者查看、下載		
輸入值	查詢資料、歷史爬蟲資料		
輸出值	使用者歷史爬蟲資料		

第七章 資料模型

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



▲圖 7-1-1 實體關聯圖

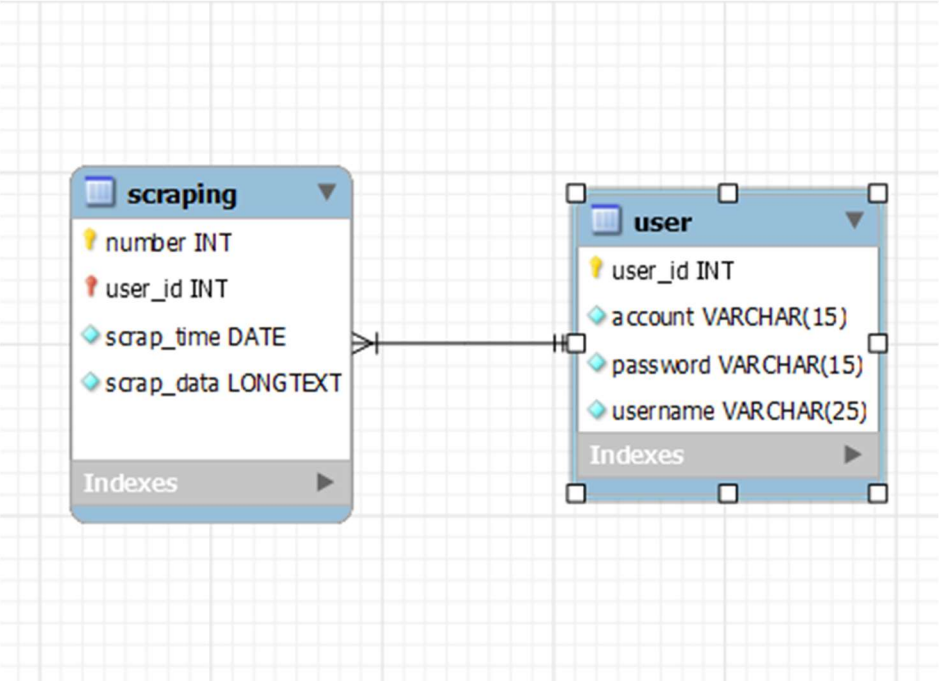
7-2 資料字典(Data dictionary)。

▼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(中文)	欄位數
T01	scraping	爬蟲	4
T02	user	使用者	4

第八章 資料庫設計

8-1 資料庫關聯表(Constraints)。



▲圖 8-1-1 資料庫關聯圖

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		Y
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

9-2 程式規格描述

▼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.St
retch)

        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：說明採用之測試方法及其進行方式

1. 登入/註冊功能：使用者使用我們程式前，需先登入/註冊

2. 瀏覽器功能 ：使用者可以在我們的瀏覽器上上網

3. 搜索功能：使用者能在我們瀏覽器反白他想爬取的資料

4. 爬蟲功能：使用者利用搜索功能得到的反白內容的資訊執行爬蟲

5. 歷史紀錄功能：使用者可以查看及下載他所爬取的內容

10-2 測試個案與測試結果

▼表 10-2-1 測試個案與測試結果

編號	功能名稱	測試流程	測試結果
1	登入/註冊功能	輸入帳號密碼，測試能否註冊與登入	可使用
2	瀏覽器功能	測試瀏覽器能不能連上 web 與網路	可使用
3	搜索功能	測試能否獲取反白內容的資料，並進行比對後回傳	可使用
4	爬蟲功能	測試能否進行爬蟲功能，並將爬取的資料傳到資料庫	可使用
5	歷史紀錄功能	測試能否查看歷史紀錄並下載	可使用

第十一章 操作手冊

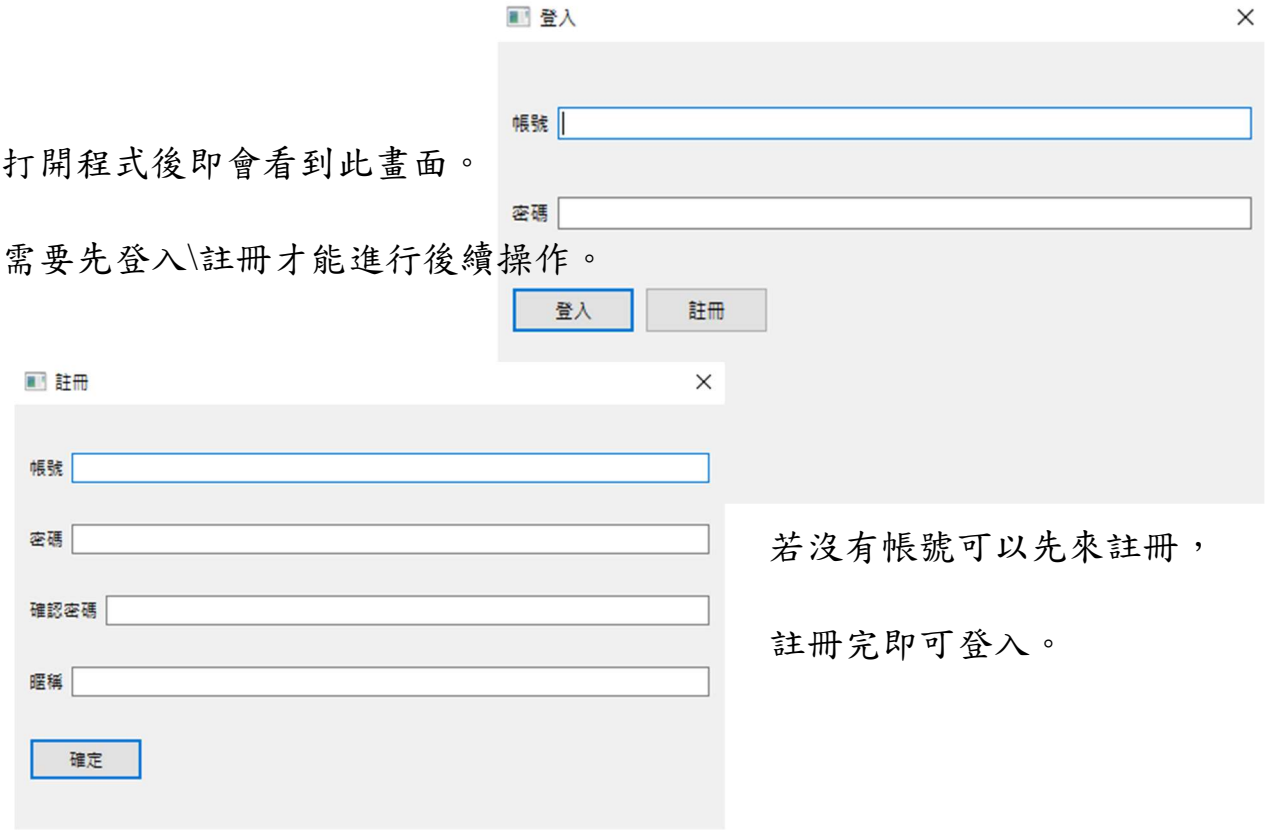
介紹系統支援件極其安裝及系統管理

▼表 11-1-1 測試個案與測試結果

系統安裝元件資訊	
系統名稱	iCrawler
版本	1.0
檔案大小	無
軟體類別	搜尋類 app
語言	繁體中文
版本需求	無
內容分級	普遍級
權限需求	完整網路存取權

第十二章 使用手冊

介紹各畫面、操作之移轉，以類似 State Transition Diagram 之表示之。



打開程式後即會看到此畫面。

需要先登入\註冊才能進行後續操作。

若沒有帳號可以先來註冊，

註冊完即可登入。

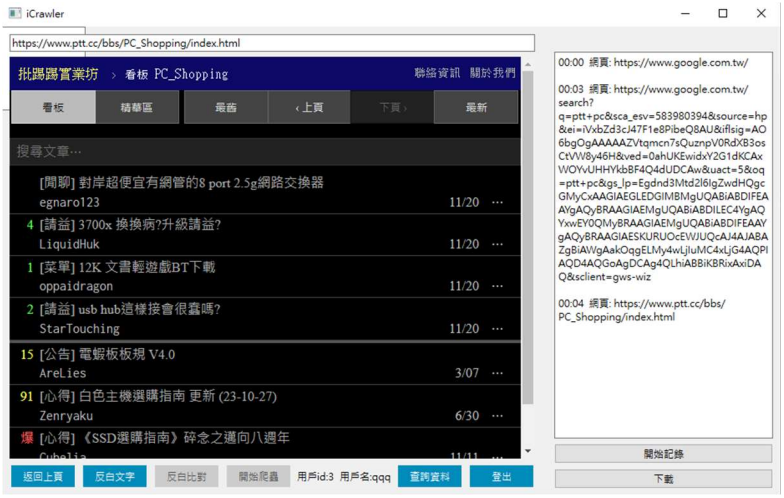
▲圖 12-1-1 登入/註冊功能之介面介紹

執行瀏覽器基本操作

點擊來執行返回瀏覽器上頁、爬蟲（反白文字、反白比對、開始爬蟲）、查看爬蟲資料、會員登出。

讓使用者紀錄上/下頁按鈕及下載使用者動作的紀錄。

▲圖 12-1-2 瀏覽器功能之介面介紹

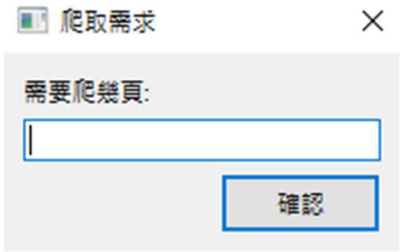


進入至欲爬取資料的網站（以 PTT 論壇為例）。

使用者分別反白欲爬取的資料（以文章標題為例）每反白一次就要按下底下的反白文字按鈕(共兩次)，再按下反白比對。

最後，按下開始記錄按鈕點選網頁中的上/下頁按鈕即可爬蟲。

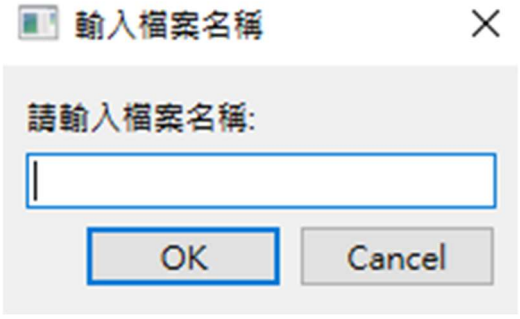
按下開始爬蟲後輸入需要爬取的頁數，即會打開 Selenium 開始執行爬蟲。



▲圖 12-1-3 爬蟲功能之介面介紹

爬蟲完成後按下瀏覽器底下的查詢資料按鈕即會跳轉至底下的畫面。

再按下查詢按鈕後就會展示所有爬取的資訊。



使用者可以將爬取下來的資料下載至本機，按下底下下載按鈕後，再輸入檔案名稱就會載至本機（檔案類型是 CSV）。

▲圖 12-1-4 歷史紀錄功能之介面介紹

第十三章 感想

感想

10856040 鍾皓年: 這次的專題讓我深刻體會到管理的複雜性和挑戰，尤其是在領導底下的團隊。這段時間不僅學到了有效管理底下組員的方法，更深入瞭解了如何妥善安排和控制專案時程。這段經驗中，我學到了如何適時給予肯定和建構性的回饋，以維持團隊的積極性和合作性。同時，在管理專案時程的方面，我深感時間管理的重要性。這不僅包括了確保每個階段的任務都能按計劃進行，還要處理不可預測的變數和風險。學習如何制定合理而靈活的時程，以應對突發情況和確保順利的專案進行，是一項非常寶貴的經驗。

10856010 郭宗翰:這次的專題讓我學習到了許多在學校所沒有教的，像是爬蟲，一開始我是先使用 urllib 來進行爬蟲，後來經過中研院老師的介紹，得知了 Selenium 這套爬蟲軟體，他可以模擬我們使用者一般在瀏覽器上進行操作，這對於我後端要寫的爬蟲有著莫大的進步，在寫程式的過程中也學會如何使用及利用身邊和網路上的資源，遇到報錯的時候，就會去詢問同學或是將錯誤代碼貼到網路上去搜尋有無線索，靠著這些一步一腳印完成佔了我們專題很重要的一部分，最後也感謝身邊願意解惑的同學及專題指導老師及中研院老師。

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

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