資安導論——利用 API 即時查詢課程評價

壹、背景與動機

每學期查詢通識總是花費大量心力與時間，因此我決定整理並分析輔仁 ClassIn 選課資訊交流平台資料，以 Flask + Selenium 架設本地課程搜尋 API，透過 Ngrok 對外開放，再用 Dify 平台串接 Gemini AI 整理資料，最終提供智慧化聊天式課程推薦服務，達到節省精力的目的。

貳、技術與工具介紹

▼表 1 技術與工具介紹簡介

編號	技術分類	重點說明
1	Python 核心	使用 Python 3.x，採標準結構（try/except, dict, list）進行資料處理與例外處理。
2	Flask 輕量級後端框架	建立 /search API 接收查詢參數，並用 @after_request 加自訂 Header 解決 ngrok 警告。
3	自動化瀏覽器操作	使用 Selenium 模擬操作，undetected-chromedriver 避開反爬蟲（啟用 headless 模式等設定）。
4	資料解析與清洗	透過 BeautifulSoup 解析 Selenium 取得的 HTML，精準抽取課程資訊與評價。
5	資料組織與轉換	整理成 Python dict/list 結構，並將英文欄位轉成繁體中文標籤（如 課程名稱、教師姓名）。
6	HTML 動態生成	使用純 Python 字串產生 HTML 表格，UTF-8 編碼支援中文輸出。
7	虛擬環境管理	用 venv 建立環境，pip 安裝套件，pip freeze 鎖定 requirements.txt，並在 .gitignore 排除虛擬環境。
8	版本控制設定	使用 JetBrains IntelliJ Community 版，透過 vcs.xml / workspace.xml 管理開發設定。
9	Ngrok 外部公開串接	建立 HTTPS 隧道將本機 Flask API 外曝，並加上 ngrok-skip-browser-warning Header。

10	Dify AI 平台串接	使用 Dify 串接 ngrok API，經 Gemini 1.5 Pro 整理資料，自動生成表格並回覆使用者。

一、Python 核心語言

（一）使用 Python 3.x 撰寫整個後端應用程式。

（二）使用標準結構（try/except, dict, list）進行資料處理與例外處理。

二、輕量級、好部署的後端開發 Web 框架：Flask

（一）建立 /search API Endpoint，接受查詢參數 (q) 搜尋課程資料。

（二）使用 @after_request 加入自訂 Header（例如解決 ngrok 警告）。

三、自動化瀏覽器操作

（一）Selenium

1. 自動控制 Chrome 瀏覽器模擬人類操作（搜尋、點擊分類、抓資料）。
2. 使用 By, ActionChains, NoSuchElementException 進行元素操作與例外處理。

（二）undetected-chromedriver (uc)

1. 加強版的 Selenium driver，專門避開網站的反自動化偵測。
2. 使用 --headless=new, --disable-blink-features=AutomationControlled 等設定，使爬蟲更隱形。

四、資料解析與清洗：BeautifulSoup (bs4)

（一）從 Selenium 取得的 HTML 中解析出需要的資料（課程名稱、教師、星等評價、留言等）。

（二）透過 CSS Selector 精準選取 DOM 元素。

五、資料組織與轉換

（一）使用 Python 字典（dict）與列表（list）組織資料。

（二）將英文欄位轉換成繁體中文標籤，例如：

1. title ➔ 課程名稱
2. teacher ➔ 教師姓名
3. clarity_star_count ➔ 清晰度星數

六、HTML 頁面動態生成

（一）使用純 Python 字串組合方式，生成 HTML <table> 表格。

（二）自動輸出排版良好、適合瀏覽的網頁頁面。

（三）確保 HTML 使用 UTF-8 編碼以支援中文。

搜尋關鍵字：「自然」的結果

卡片索引	課程名稱	教師姓名	要求量星數	作業量星數	考試量星數	收穫度星數	趣味性星數	清晰度星數	推薦度星數	留言時間	留言內容	標籤
0	醫學發展對倫理的影響-英	司馬忠	1	1	0	4	4	5	5	2016/9/8 上午 1:57:45	每次的第二堂課，老師會發紙讓同學們跟附近的人一起寫老師問題的生物倫理的回答，上面的寫的名字就是代替點名。 沒有小考、上課都是看老師的PPT，期中期末都是交報告，有幾次缺席跟很多次第二節才到，最後期末73分，老師佛心啊~	[]
1	數學與邏輯	李泰明	1	1	0	5	5	5	5	2016/1/28 下午 2:53:32	老師上課會發紙張，要你上課作筆記，這個就當作是成績+點名 沒有期中期末考	[]
2	電腦應用	揚志田	2	2	0	5	3	5	5	2016/1/28 下午 1:56:53	老師不會點名，上課內容很豐富，教學很認真，稍微有一點催眠而已，如果想學到word、excel等等的內容，真的可以學到很多~ 對以後使用也很有幫助。	['個人作業2次']
3	臨床醫學檢查與疾病診斷	莊志光	1	1	0	5	4	5	5	2016/1/27 下午 9:05:35	志光老師是馬偕的罕見疾病科醫生喲！ 老師每堂課都是用簽到方式點名， 很推薦給想吸收滿滿醫學知識的人（當然也推薦給想輕鬆過的人啦） 考前一週老師會放重點整理到ICAN上， 記得好好看過然後去考試就OK了~ 考試非常簡單，就是是非題與選擇題。 分數滿甜的！	[]
4	普通化學	管克新	2	1	0	4	5	5	5	2016/1/26 下午 11:45:12	老師不定期點名但點名是加分用(點5次全到總分+10) 不需要請假 上課要買老師自編的講義(原文書影印本)一本$180 老師講話很有趣，常常笑到肚子痛 他常常分享之前在國外念書的事 計分：期中50%期末50% 期中期末都是考試，開書考，會調分(最高分的人+到100，其他人+一樣的分數) 我期中58加分後80，期末不知道，總成績83 推薦給想學東西而且想拿分數的同學^^	[]
5	健康與疾病	呂旭*	1	1	0	5	4	5	5	2016/1/25 上午 12:11:49	非常推薦!!! 只有期中、期末考，考前一周會發考試講義，只能拿自己的，所以要到！考試都是申論題，把講義背起來就好了！ 一學期大概會點6-7次名(考前幾周必點) 點名都有到、考試都有寫，老師給分很高！	[]

▲圖 1 部分 HTML 頁面：上圖顯示搜尋「自然」類別的課程結果

七、虛擬環境管理

（一）venv

建立虛擬環境：python3 -m venv dcard_proxy_env。

（二）pip

安裝必要套件：Flask、Selenium、undetected-chromedriver、beautifulsoup4。

（三）requirements.txt

使用 pip freeze > requirements.txt 鎖定套件版本。

（四）.gitignore

排除 dcard_proxy_env/ 等不應進入版本控制的資料夾。

八、版本控制設定：vcs.xml / workspace.xml

使用 JetBrains IDE Intellij community edition 開發，自動管理版本控制（VCS）設定與個人偏好（如主題、字型）。

九、外部公開與即時串接：Ngrok

（一）建立從本機端（localhost）到外網的 HTTPS 隧道。

（二）把 Flask 的 /search API 變成外部可存取的公開網址（例如 https://xxx.ngrok.io）。

（三）特別加上 header ngrok-skip-browser-warning: true 解決 ngrok 免費版瀏覽器警告。

十、AI 應用平台串接：Dify

（一）介紹

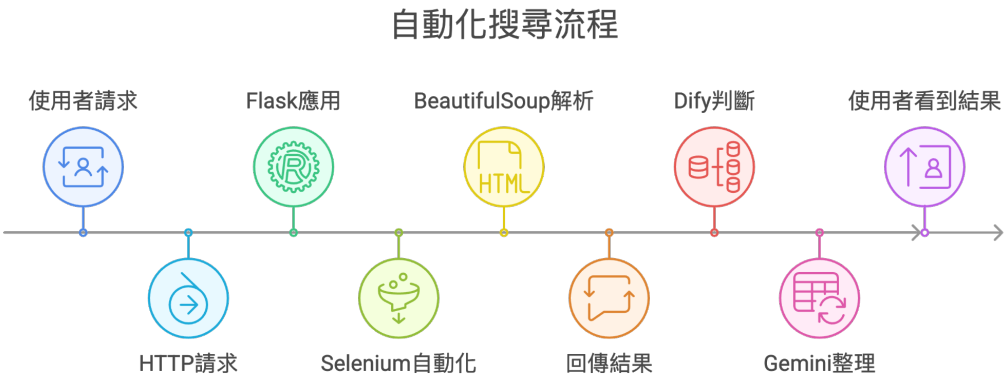
1. AI 流程開發平台，支援模型（如 GPT-4）+ 外部 API 查詢。
2. 將 /search API 當作 Data Plugin，讓 AI 可以動態呼叫，取得 ClassIn 資料。
3. 實現「使用者輸入 ➔ AI 啟動 API ➔ 搜尋結果 ➔ 整理成智慧回答」的應用流程。

（二）主要流程（聊天模型架構圖）

1. 使用者輸入關鍵字

➔ 觸發聊天機器人流程。

2. Dify 顯示暫時回覆
   ➔ 顯示「搜尋中，請稍候」的暫時訊息，提升使用體驗。
3. Dify 觸發 HTTP Request 呼叫 ngrok 公開的 Flask /search API
   ➔ 後端搜尋資料並回傳結果。
4. 系統收到資料後進行條件判斷（If-Else）
   ➔ 判斷是否有資料、是否出錯。
5. 若成功取得資料，傳送給 Gemini 1.5 Pro 處理
   ➔ 進行留言摘要、表格格式化、統計整理。
6. 最終回覆使用者整理後的課程搜尋結果
   ➔ 以表格+簡化留言方式呈現。



▲圖 2 搜尋流程簡介：展示從使用者到 AI 整理的完整步驟

（三）使用模型

Gemini-1.5-Pro（Google）透過 LangGenius Connector 串接，負責資料摘要與表格格式化。

（四）特色設定

1. 支援建議問題（人文通識、自然通識、社會通識、歷史通識、體育）。
2. 條件分支處理系統錯誤、無資料狀況。
3. 將 API 結果變成易讀的表格並摘要顯示。

參、問題說明

在專題開發初期，我原本規劃使用 Dcard 作為主要資料來源，透過搜尋「輔仁通識推薦」等關鍵字進行內容收集。然而，實作時遇到 Dcard 強化了反爬蟲機制，即使使用 Selenium 搭配 undetected_chromedriver，仍常因風控封鎖而失敗。考量到穩定性與時間成本，我決定轉而改為以 ClassIn 作為資料蒐集來源。

在爬取 ClassIn 資料的過程中，我希望補充每篇課程評論的星星評分數據，因此進行了 SVG 分析。由於 ClassIn 網頁上的星星評分是以 SVG 呈現，且不同填滿狀態的星星使用了不同的 class（例如 text-secondary 表示填滿、text-gray-200 表示未填滿），因此我在程式設計上需要精確判斷 SVG 標籤的 class 屬性來正確計算實際評分。在這部分的開發中，遇到了許多需要細緻解析 HTML 結構與動態元素載入的技術挑戰。

此外，專案原先考慮將服務部署在 Dify 平台上，但由於 Dify 當前未內建完整的 Python 執行環境（缺少 Flask、Selenium、undetected_chromedriver 等必要功能），我開始尋找其他免費可用的伺服器。嘗試過 Google Colab，但因 Colab 不支援 ngrok 或直接公開 Flask API，無法滿足部署需求；另外，Fly.io 和 Google Cloud Platform 雖免費提供雲端伺服器，但強制要求綁定信用卡，不符合個人意願。

因此，最終我決定僅在本機端部署開發完成的系統，保證系統功能完整，同時避免額外的註冊與金流風險。

肆、影片演示使用流程與產出（HTTP 請求需等候至多30秒）

一、點選自然通識
二、搜尋腦科學
三、搜尋數學與數學文化（模擬低評價不列出）
三、搜尋動1（模擬使用者輸入錯誤）
四、關閉 API 進行搜尋（模擬系統錯誤）
五、關閉 ngrok 進行搜尋（模擬系統錯誤）

伍、附錄：完整程式碼與註解

```
from flask import Flask, request, Response
import undetected_chromedriver as uc
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
from bs4 import BeautifulSoup
import time
import json

# 初始化 Flask 應用
app = Flask(__name__)

# 加入 ngrok 跳過警告標頭
@app.after_request
def add_ngrok_skip_header(response):
    response.headers["ngrok-skip-browser-warning"] = "skip"
    return response

# 定義 /search 路由
@app.route('/search')
def search():
    try:
        # 取得查詢參數
        query = request.args.get('q', '').strip()
        if not query:
            return Response(
                "<h3>請提供搜尋關鍵字，例如 ?q=網球 或 ?q=人文</h3>",
                content_type='text/html; charset=utf-8'
```

```
        )

    # 預設可點擊的分類標籤
    valid_categories = ['人文通識', '自然通識', '社會通識', '歷史通識', '體育']
    matched_category = next((cat for cat in valid_categories if query == cat), None)

    # 設定 Chrome 瀏覽器選項（無頭模式）
    options = uc.ChromeOptions()
    options.add_argument('--headless=new')
    options.add_argument('--disable-gpu')
    options.add_argument('--no-sandbox')
    options.add_argument('--window-size=1920,1080')
    options.add_argument('--start-maximized')
    options.add_argument('--disable-blink-features=AutomationControlled')

    # 啟動 undetected_chromedriver
    driver = uc.Chrome(options=options)
    driver.get("https://classin.info/view")
    time.sleep(3)

    # 根據是否是分類 or 自訂搜尋進行點擊或輸入
    if matched_category:
        try:
            category_button = driver.find_element(By.XPATH, f'//button[contains(text(), "{matched_category}")]')
            ActionChains(driver).move_to_element(category_button).click(category_button).perform()
            print(f"✅ 成功點擊分類：{matched_category}")
        except NoSuchElementException:
            print(f"❌ 找不到分類按鈕：{matched_category}")
        time.sleep(3)
    else:
        try:
            search_input = driver.find_element(By.CSS_SELECTOR, 'input[placeholder*="課程名稱"]
')
            search_input.clear()
            search_input.send_keys(query)
            search_button = driver.find_element(By.CSS_SELECTOR, 'button > svg')
            ActionChains(driver).move_to_element(search_button).click().perform()
            print(f"🔍 搜尋關鍵字：{query}")
        except Exception as e:
            print(f"❌ 搜尋欄錯誤：{e}")
        time.sleep(3)

    # 排序：點擊"推薦高至低"
    try:
        sort_button = driver.find_element(By.XPATH, '//button[contains(text(), "推薦高至低")]')
        ActionChains(driver).move_to_element(sort_button).click(sort_button).perform()
        print("✅ 成功點擊推薦高至低")
    except NoSuchElementException:
        print("❌ 找不到推薦高至低按鈕")

    time.sleep(3)
```

```
    # 解析當前頁面 HTML
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    driver.quit()

    # 擷取每個課程卡片資料
    cards = soup.select('div[class*="Card__CardWrapper"]')
    results = []

    for idx, card in enumerate(cards):
        result = {
            'card_index': idx,
            'title': None,
            'teacher': None,
            'teacher_demand_star_count': 0,
            'homework_star_count': 0,
            'exam_star_count': 0,
            'gain_star_count': 0,
            'fun_star_count': 0,
            'clarity_star_count': 0,
            'comment_star_count': 0,
            'comment_time': None,
            'comment_text': None,
            'tags': []
        }

        # 解析課程標題與教師
        titles = card.select('div[class^="Typography__Title"]')
        if len(titles) >= 3:
            result['title'] = titles[1].get_text(strip=True)
            result['teacher'] = titles[2].get_text(strip=True)

        # 解析各項星等指標
        teacher_divs = card.select('div.Typography__SubTitle-sc-qms70n-1')
        for t_div in teacher_divs:
            text = t_div.get_text(strip=True)
            parent_flex = t_div.find_parent('div', class_='md:flex')
            if not parent_flex:
                continue
            if text == '教師要求':
                result['teacher_demand_star_count'] = len(parent_flex.select('svg.text-secondary'))
            elif text == '作業量':
                result['homework_star_count'] = len(parent_flex.select('svg.text-secondary'))
            elif text == '考試量':
                result['exam_star_count'] = len(parent_flex.select('svg.text-secondary'))
            elif text == '收穫多少':
                result['gain_star_count'] = len(parent_flex.select('svg.text-primary'))
            elif text == '課程有趣':
                result['fun_star_count'] = len(parent_flex.select('svg.text-primary'))
            elif text == '講課清晰':
                result['clarity_star_count'] = len(parent_flex.select('svg.text-primary'))

        # 解析課程標籤
        tag_divs = card.select('div.bg-secondary.text-white')
```

```
        for tag in tag_divs:
            result['tags'].append(tag.get_text(strip=True))

        # 解析推薦評論
        rating_block = card.select_one('div.lg\:w-4\/12')
        if rating_block:
            result['comment_star_count'] = len(rating_block.select('svg.text-primary'))
            time_tag = rating_block.select_one('div.ml-4')
            if time_tag:
                result['comment_time'] = time_tag.get_text(strip=True)
            comment_tag = rating_block.select_one('pre.whitespace-pre-wrap')
            if comment_tag:
                result['comment_text'] = comment_tag.get_text(strip=True)

        results.append(result)

    # 中英欄位名稱對應表
    column_mapping = {
        "card_index": "卡片索引",
        "title": "課程名稱",
        "teacher": "教師姓名",
        "teacher_demand_star_count": "要求量星數",
        "homework_star_count": "作業量星數",
        "exam_star_count": "考試量星數",
        "gain_star_count": "收穫度星數",
        "fun_star_count": "趣味性星數",
        "clarity_star_count": "清晰度星數",
        "comment_star_count": "推薦度星數",
        "comment_time": "留言時間",
        "comment_text": "留言內容",
        "tags": "標籤"
    }

    # 轉換成繁體中文欄位
    converted_data = []
    for record in results:
        new_record = {}
        for key, value in record.items():
            new_key = column_mapping.get(key, key)
            new_record[new_key] = value
        converted_data.append(new_record)

    # 產生 HTML 表格內容
    def generate_html_table(data_list):
        if not data_list:
            return "<p>無資料</p>"

        headers = list(data_list[0].keys())
        html = "<table border='1' cellpadding='5' cellspacing='0' style='border-collapse: collapse; font-size:16px;'
>"
        html += "<thead><tr>" + "".join([f"<th>{header}</th>" for header in headers]) + "</tr></thead><tbody>"

        for item in data_list:
```

```
            html += "<tr>" + "".join([f"<td>{item.get(header, '')}</td>" for header in headers]) + "</tr>"
        html += "</tbody></table>"
        return html

    table_html = generate_html_table(converted_data)

    # 最後組合完整 HTML 頁面
    final_html = f"""
    <html>
    <head>
        <meta charset=\"utf-8\">
        <title>課程搜尋結果</title>
    </head>
    <body>
        <h2>搜尋關鍵字：「{query}」的結果</h2>
        {table_html}
    </body>
    </html>
    """

    return Response(final_html, content_type='text/html; charset=utf-8')

  except Exception as e:
    # 例外處理：若發生錯誤回傳錯誤訊息頁面
    return Response(
        f"<h3>發生錯誤：{str(e)}</h3>",
        content_type='text/html; charset=utf-8'
    )

# 啟動伺服器
if __name__ == '__main__':
  app.run(debug=True)
```