

國立臺北商業大學

資訊管理系

111' 資訊系統專案設計

系統手冊



組別：第 111202 組

題目：知音

指導老師：楊進雄老師

組長：11036037 林詩蓉

組員：11036008 陳珮蓉 11036015 陳韋辰

11036016 白皓云 11136006 廖暄毓

中華民國 111 年 11 月 23 日

目錄

第一章 背景與動機	8
1-1 簡介	8
1-2 問題與機會	9
1-3 相關系統探討	10
第二章 系統目標與預期成果	12
2-1 系統目標	12
2-2 預期成果	13
第三章 系統規格	14
3-1 系統架構	14
3-2 系統軟、硬體需求與技術平台	15
3-3 使用標準與工具	16
第四章 專案時程與組織分工	17
4-1 專案時程	17
4-2 專案組織分工	18
第五章 需求模型	19
5-1 使用者需求	19
5-2 使用個案圖 (Use case diagram)	23

5-3	使用個案描述	24
5-4	分析類別圖 (Analysis class diagram)	27
第六章 設計模型		28
6-1	循序圖 (Sequence diagram)	28
6-2	設計類別圖 (Design class diagram)	33
第七章 實作模型		34
7-1	部署圖 (Deployment diagram)	34
7-2	套件圖 (Package diagram)	34
7-3	元件圖 (Component diagram)	35
7-4	狀態機 (State machine)	35
第八章 資料庫設計		36
8-1	資料庫關聯表	36
8-2	表格及其 Meta data	37
第九章 程式		39
9-1	元件清單及其規格描述	39
9-2	其他附屬之各種元件	41
第十章 測試模型		66
10-1	測試計劃	66

10-2 測試個案與測試結果資料	66
第十一章 操作手冊	67
第十二章 使用手冊	68
第十三章 感想	73
第十四章 參考資料	78

圖目錄

圖 1-3-1 SWOT 分析	11
圖 3-1-1 系統架構圖.....	14
圖 4-1-1 專案時程圖.....	17
圖 5-2-1 使用個案圖（使用者）.....	23
圖 5-2-2 使用個案圖（管理者）.....	23
圖 5-3-1 註冊會員活動圖.....	24
圖 5-3-2 登入系統活動圖.....	24
圖 5-3-3 登出系統活動圖.....	24
圖 5-3-4 修改密碼活動圖.....	24
圖 5-3-5 輸入連結活動圖.....	25
圖 5-3-6 送出文章活動圖.....	25
圖 5-3-7 分析文章活動圖.....	25
圖 5-3-8 查閱歷史資料活動圖.....	25
圖 5-3-9 查閱文章情緒分佈活動圖.....	26
圖 5-3-10 帳戶管理活動圖.....	26
圖 5-3-11 文章管理活動圖.....	26
圖 5-4-1 分析類別圖.....	27
圖 6-1-1 註冊會員循序圖.....	28
圖 6-1-2 登入系統循序圖.....	28
圖 6-1-3 登出系統循序圖.....	28

圖 6-1-4 修改密碼循序圖.....	29
圖 6-1-5 輸入連結循序圖.....	29
圖 6-1-6 送出文章循序圖.....	30
圖 6-1-7 分析文章循序圖.....	30
圖 6-1-8 查閱歷史資料循序圖.....	31
圖 6-1-9 查閱文章情緒分布循序圖.....	31
圖 6-1-10 帳戶管理循序圖.....	32
圖 6-1-11 文章管理循序圖.....	32
圖 6-2-1 設計類別圖.....	33
圖 7-1-1 部署圖.....	34
圖 7-2-1 套件圖.....	34
圖 7-3-1 元件圖.....	35
圖 7-4-1 狀態機.....	35
圖 8-1-1 資料庫關聯表.....	36

表目錄

表 1-3-1 相關系統比較.....	10
表 3-2-1 測試環境與軟硬體需求表.....	15
表 3-3-1 使用標準與工具表.....	16
表 4-2-1 分工表.....	18
表 5-1-1 使用者功能需求表.....	19
表 5-1-2 非功能需求表.....	22
表 8-2-1 資料表描述：帳號.....	37
表 8-2-2 資料表描述：文章.....	38
表 9-1-1 元件清單及其規格描述.....	39
表 10-1-1 測試流程.....	66
表 10-2-1 測試結果.....	66
表 11-1-1 操作手冊表.....	67
表 12-1-1 使用手冊表.....	68

第一章 背景與動機

1-1 簡介

音樂已經被證實存在於每個已知的文明中，不同時代亦有不同代表性的音樂出現，且音樂的力量早在 17 世紀中葉就已經被證實：巴洛克時期，被稱為最後的一位文藝復興人物的德國學者 Kircher 提出人格特質和情緒組成與一定的音樂類型具有關聯性。例如憂鬱的人對憂傷的音樂更有反應；高興的人更偏好輕快活潑的音樂，因為它會加速你血液的流動（朱浚溢 2015）。一戰和二戰之後，音樂作為一種輔助治療的方式得到了廣泛的應用。例如老兵的復健和精神創傷士兵的心理治療；20 世紀中葉，音樂治療開始被全面的普及，特別是在美洲地區，各大學音樂學院都開始開設相關課程，目前音樂治療已經廣泛地被應用到神經康復、心理治療、疼痛管理等領域。隨著神經科學的發展，音樂的療癒力量也得到了科學研究的實證。

有鑑於此，我們認為音樂已經不僅僅是人們茶餘飯後的消遣娛樂，它更代表著我們的感受、我們的情緒、甚至是我們的心，因此要如何挑選符合你當下情緒狀態的音樂，就成了至關重要的問題。目前市面上各大音樂串流平台雖有提供歌曲分類供使用者從中挑選，但這些歌曲卻未必符合聆聽者當下的心情與感受，要發揮音樂療癒的力量，必須選擇符合人格特質與情緒狀態的歌曲，故本組欲開發能透過辨識使用者文字中的情緒，來推薦相應歌曲的系統。

1-2 問題與機會

問題：

現代人生活步調快速，繁忙的生活也讓大家生活中累積不少壓力，隨著可攜式播放設備，諸如耳機、隨身音響等裝置的普及，透過聽音樂來舒壓與打發時間也已然成為趨勢，然而人們如同在海量的歌曲海中漂流，卻不知道如何選擇最適合自己當下心境的歌曲，如果不選擇符合自己當下情緒狀態的歌曲，透過聽歌抒壓的效果可能會大打折扣。

機會：

「知音」可以從使用者的其他社群平臺爬取文章，進而去判斷使用者寫下文章時的情緒狀態，並依該情緒狀態推薦相對應情緒分類的歌曲，讓使用者能簡單又快速的找到符合自己當下情緒狀態的歌，達到抒發情緒與壓力最好的效果。

1-3 相關系統探討

本組透過網路資料與組員的使用經驗，分析目前主流的音樂串流平台，包括 Apple Music、KKBOX、Spotify 與 YouTube Music 等，並進一步探討其與「知音」的差異，最後透過 SWOT 分析來說明「知音」的優劣勢以及市場定位。

表 1-3-1 相關系統比較

◎：具該功能

平台 項目	知音	Apple Music	KKBOX	Spotify	YouTube Music
平台	網頁	App	App	App/網頁	App/網頁
免費版本	◎		◎		◎
內建歌詞		◎	◎	◎	
查詢歌曲	◎	◎	◎	◎	◎
圖表記錄	◎				
外部輸入	◎				

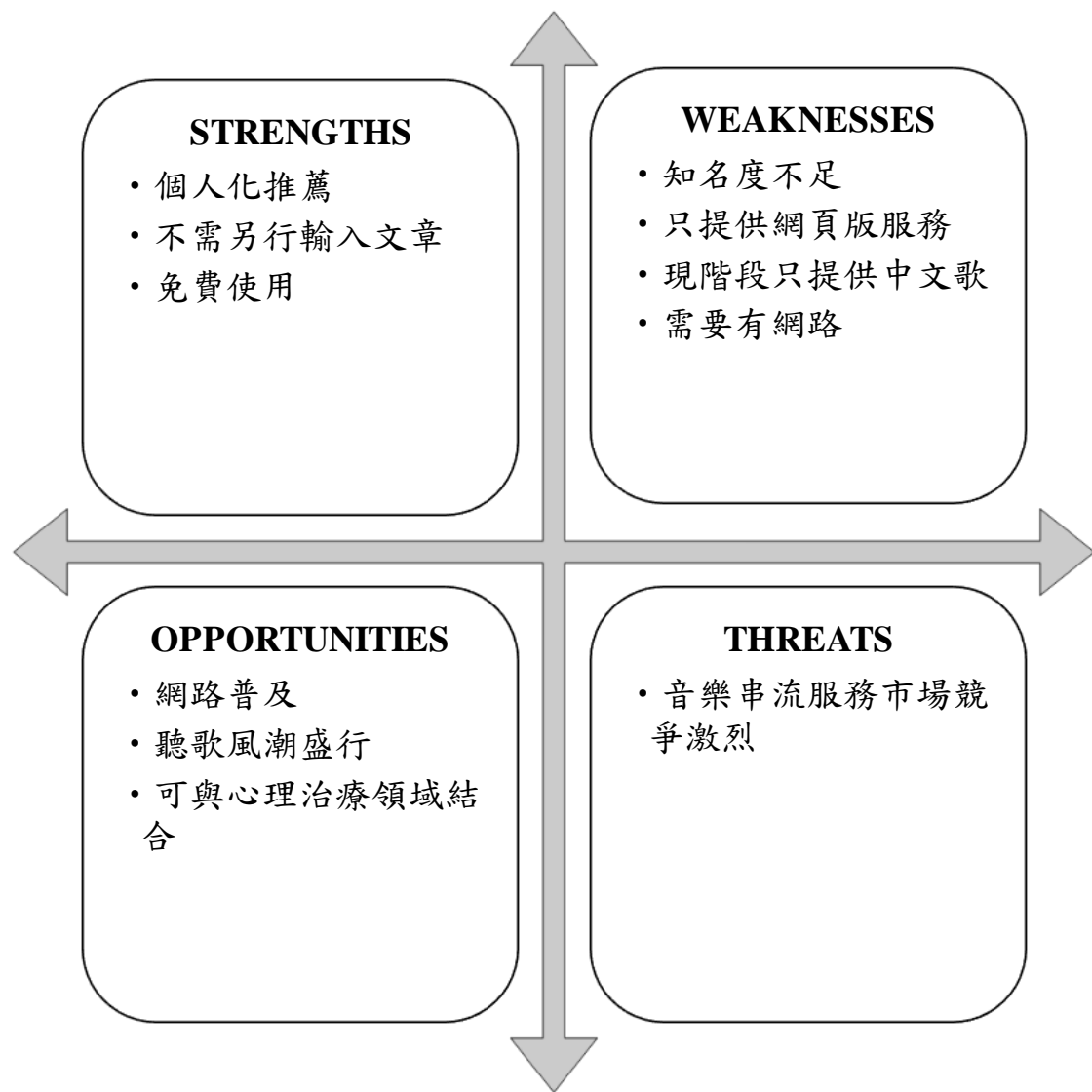


圖 1-3-1 SWOT 分析

第二章 系統目標與預期成果

2-1 系統目標

「知音」之系統目標可分為以下項目：

一、 推薦適合使用者當下情緒狀態的歌曲

透過使用者輸入的文章，判讀其當前情緒並分成喜、怒、哀、懼、愛、恨 6 種，以此推薦相應情緒分類的歌曲。

二、 豐富使用者體驗

1. 個人化推薦：推薦的歌曲能切合使用者的情緒，符合其當前的聽歌需求，提升使用者體驗。
2. 文章情緒分析與分類：可分析使用者最近的情緒狀態，同時呈現其他使用者情緒在 6 種情緒類別的分佈情形，期盼激發使用者心中的共鳴。

2-2 預期成果

1. 針對使用者需求推薦個人化的歌曲，提升滿意度，讓用戶願意持續使用「知音」。
2. 使用者能透過搜尋功能找的特定文章或歌曲，提高使用便利性。
3. 透過個人化推薦歌曲區隔其他音樂串流平台，增加用戶對「知音」的黏著度。
4. 從即時圖表用戶可得知其他使用者的各情緒狀況的比例，增加網站趣味性，豐富使用者體驗。

第三章 系統規格

3-1 系統架構

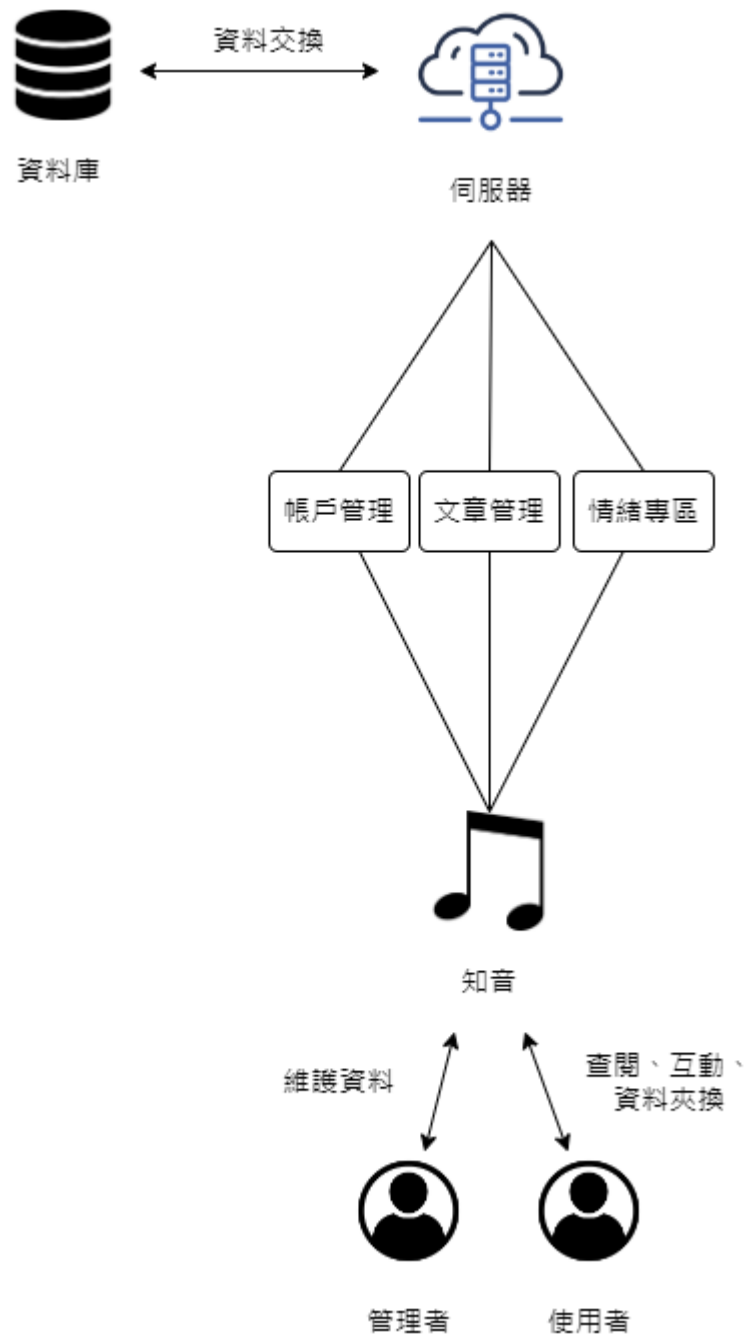


圖 3-1-1 系統架構圖

3-2 系統軟、硬體需求與技術平台

表 3-2-1 測試環境與軟硬體需求表

測試環境	
處理器	Intel Core i5 CPU
瀏覽器	Google Chrome、Microsoft Edge、Firefox
網路	有線網路、無線網路 WI-FI/4G 或 5G 網路
軟硬體需求	
作業系統	Windows
處理器	建議雙核心以上
瀏覽器需求	Google Chrome、Microsoft Edge、Firefox
網路需求	有線網路、無線網路 WI-FI/4G 或 5G 網路
記憶體	建議 300MB 以上可用空間

3-3 使用標準與工具

表 3-3-1 使用標準與工具表

開發環境	Visual Studio Code
網站前端	HTML, Javascript, Next.js, CSS
網站後端	Django, Python
資料庫	MySQL
管理工具	Navicat
美工	Adobe Illustrator, Figma
文件	Microsoft Words
簡報	Microsoft PowerPoint
專案管理	GitHub
版本控制	GitHub
系統架構	Draw.io
系統分析與設計	Draw.io

第四章 專案時程與組織分工

4-1 專案時程

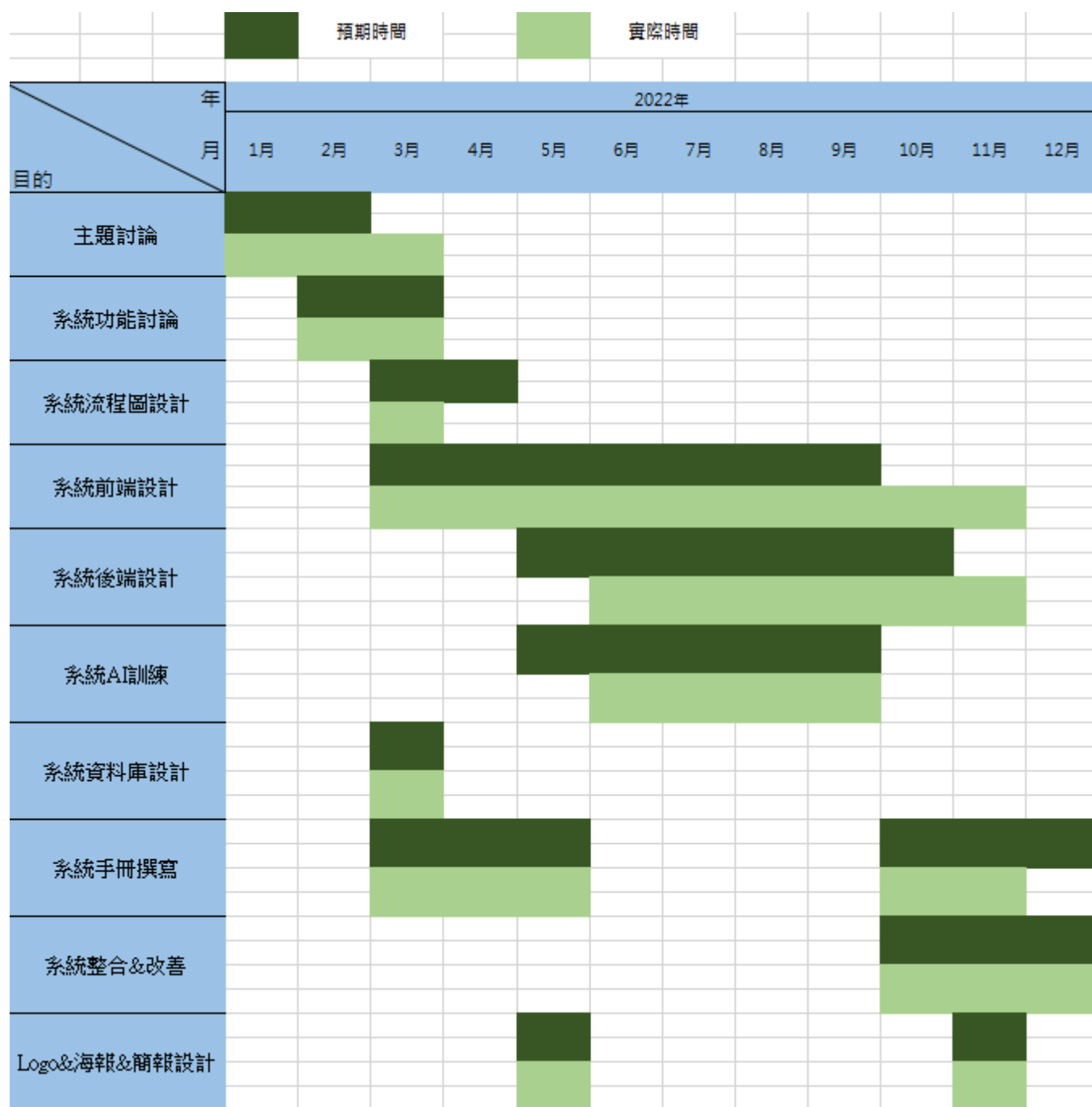


圖 4-1-1 專案時程圖

4-2 專案組織分工

表 4-2-1 分工表

●：主要負責 ○：協助負責

負責人 工作分配	林詩蓉	陳珮蓉	陳韋辰	白皓云	廖暄毓
技術學習	●	●	●	●	●
題目構思	●	●	●	●	●
系統功能分析	●	●	●	●	●
UI/UX 設計					●
資料庫建置		●	○		
AI 模型		●			
訓練資料搜集	●		●	●	
前端				○	●
後端	●	●	○		
文件製作	○	○	○	●	○
簡報製作	●	○	○	○	○
影片製作	●	●	●	●	●
海報製作			●		●

第五章 需求模型

5-1 使用者需求

表 5-1-1 使用者功能需求表

事件	觸發器	來源	活動/使用案例	回應	目的地
註冊會員	輸入使用者帳密 及電子郵件	使用者	註冊會員	註冊成功 或 註冊失敗	使用者
登入系統	輸入使用者帳密	使用者	登入系統	登入成功 或 登出失敗	使用者
登出系統	按下登出鍵	使用者	登出系統	登出成功 或 登出失敗	使用者
修改密碼	點選修改密碼	使用者	修改密碼	修改成功 或 修改失敗	使用者
輸入連結	輸入文章連結	使用者	輸入連結	輸入成功 或 輸入失敗	使用者

事件	觸發器	來源	活動/使用案例	回應	目的地
送出文章	點選送出文章	使用者	送出文章	送出成功 或 送出失敗	使用者
查閱歷史 資料	使點選信件紀錄	使用者	查閱歷史資料	查詢成功 或 查詢失敗	使用者
查閱文章 情緒分佈	點選我的情緒圖	使用者	查閱文章情緒分 佈	查詢成功 或 查詢失敗	使用者
修改帳戶	維護帳戶資料	管理者	帳戶管理	儲存成功 或 儲存失敗	管理者
查詢帳戶	維護帳戶資料	管理者	帳戶管理	儲存成功 或 儲存失敗	管理者
刪除帳戶	維護帳戶資料	管理者	帳戶管理	儲存成功 或 儲存失敗	管理者

事件	觸發器	來源	活動/使用案例	回應	目的地
查詢文章	維護帳戶資料	管理者	文章管理	儲存成功 或 儲存失敗	管理者
刪除文章	維護帳戶資料	管理者	文章管理	儲存成功 或 儲存失敗	管理者

表 5-1-2 非功能需求表

項目	說明
維護性	收到使用者的修改建議，若經評估為普通修正項目，會在 1-2 個工作天內完成；若為重大需求或設計的修改項目，則會在一週內完成修改。
相容性	支援各種螢幕尺寸，且支援任何瀏覽器。
易使用性	操作介面簡單且容易使用。
安全性	會員密碼在註冊時會透過 SHA256 加密
兼容性	支持多種作業系統。

5-2 使用個案圖 (Use case diagram)

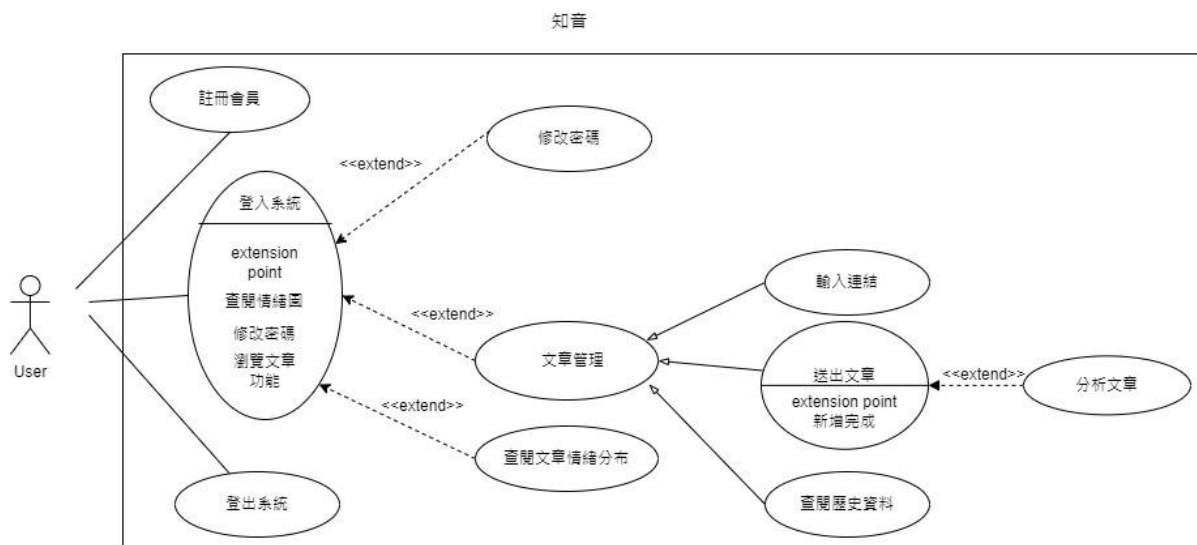


圖 5-2-1 使用個案圖 (使用者)

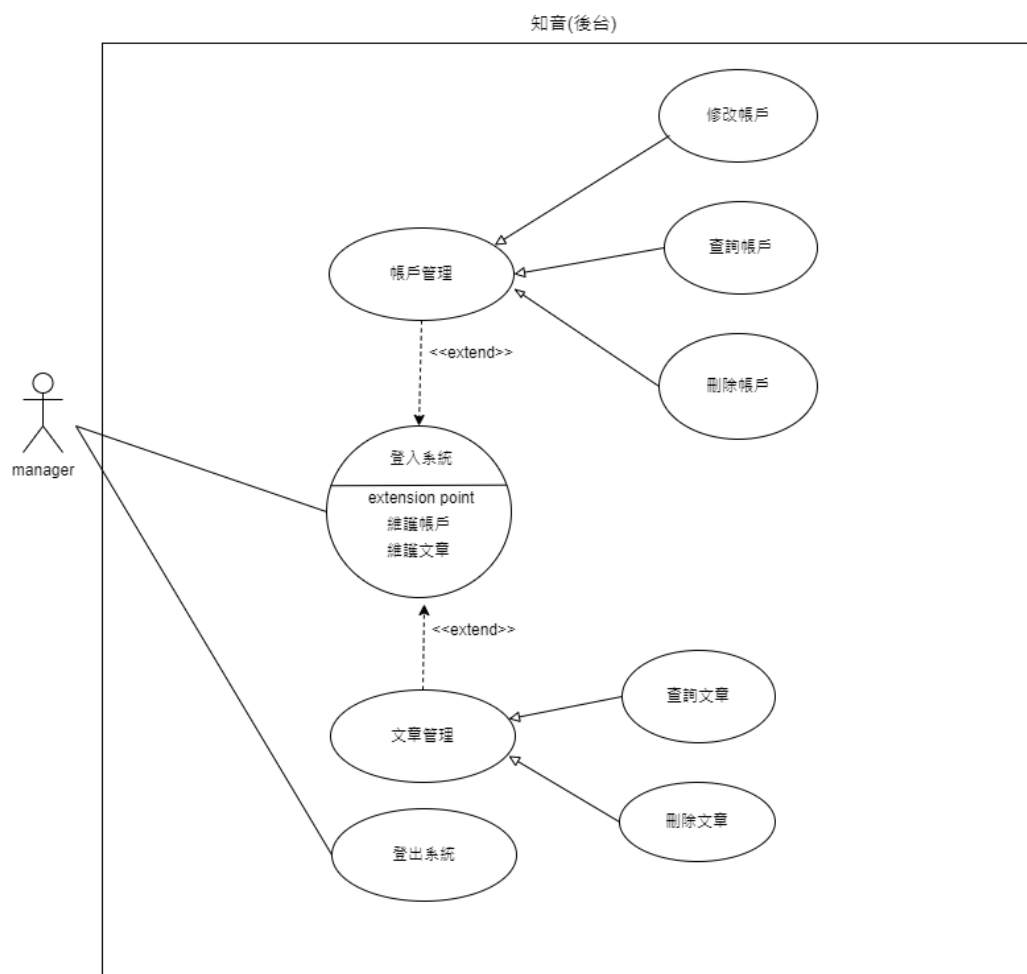


圖 5-2-2 使用個案圖 (管理者)

5-3 使用個案描述

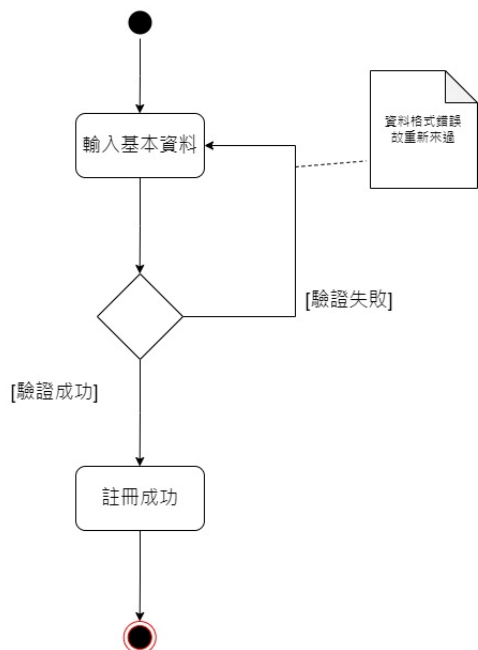


圖 5-3-1 註冊會員活動圖

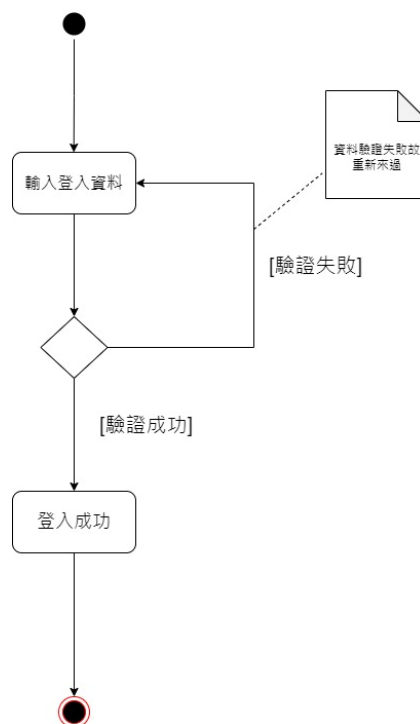


圖 5-3-2 登入系統活動圖



圖 5-3-3 登出系統活動圖

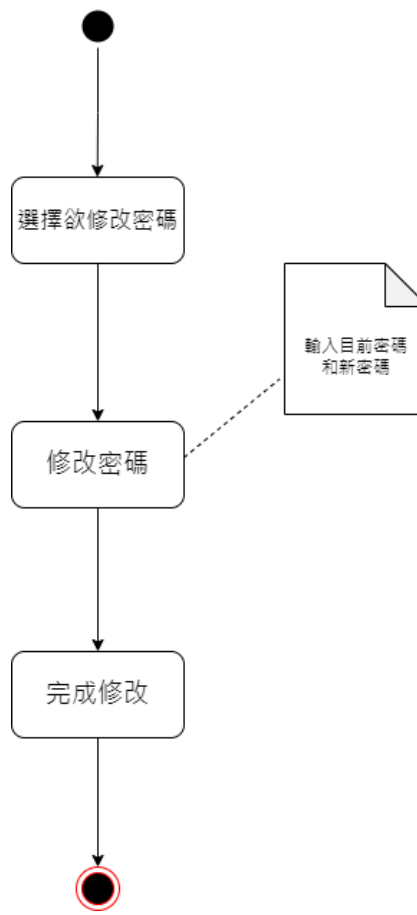


圖 5-3-4 修改密碼活動圖

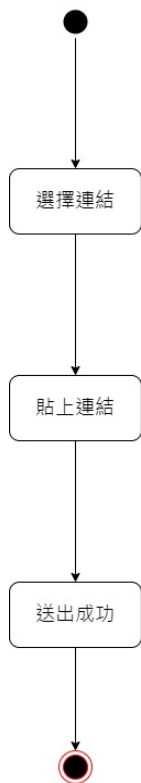


圖 5-3-5 輸入連結活動圖



圖 5-3-6 送出文章活動圖

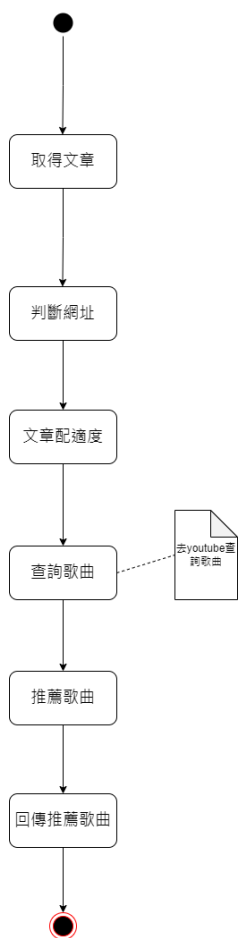


圖 5-3-7 分析文章活動圖



圖 5-3-8 查閱歷史資料活動圖

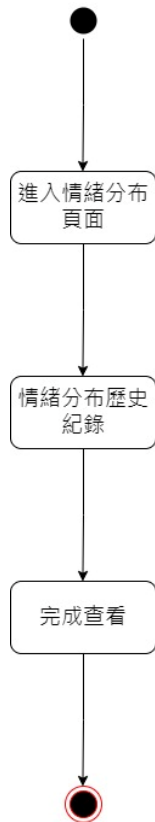


圖 5-3-9 查閱文章情緒分佈活動圖

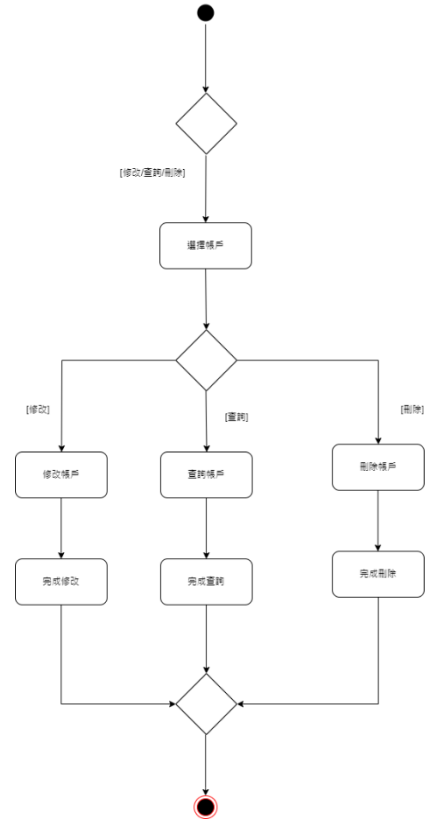


圖 5-3-10 帳戶管理活動圖

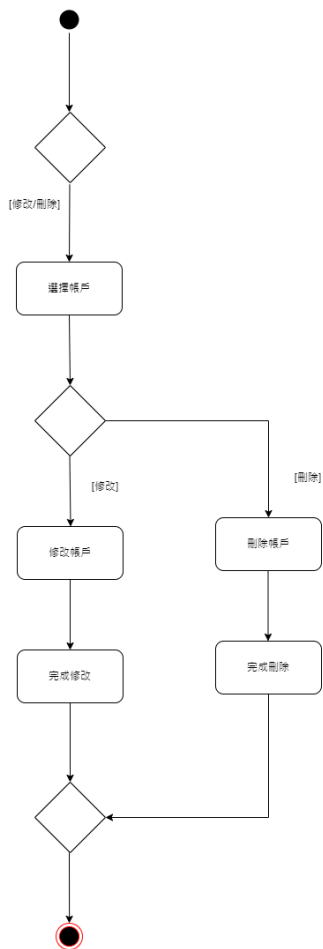


圖 5-3-11 文章管理活動圖

5-4 分析類別圖 (Analysis class diagram)

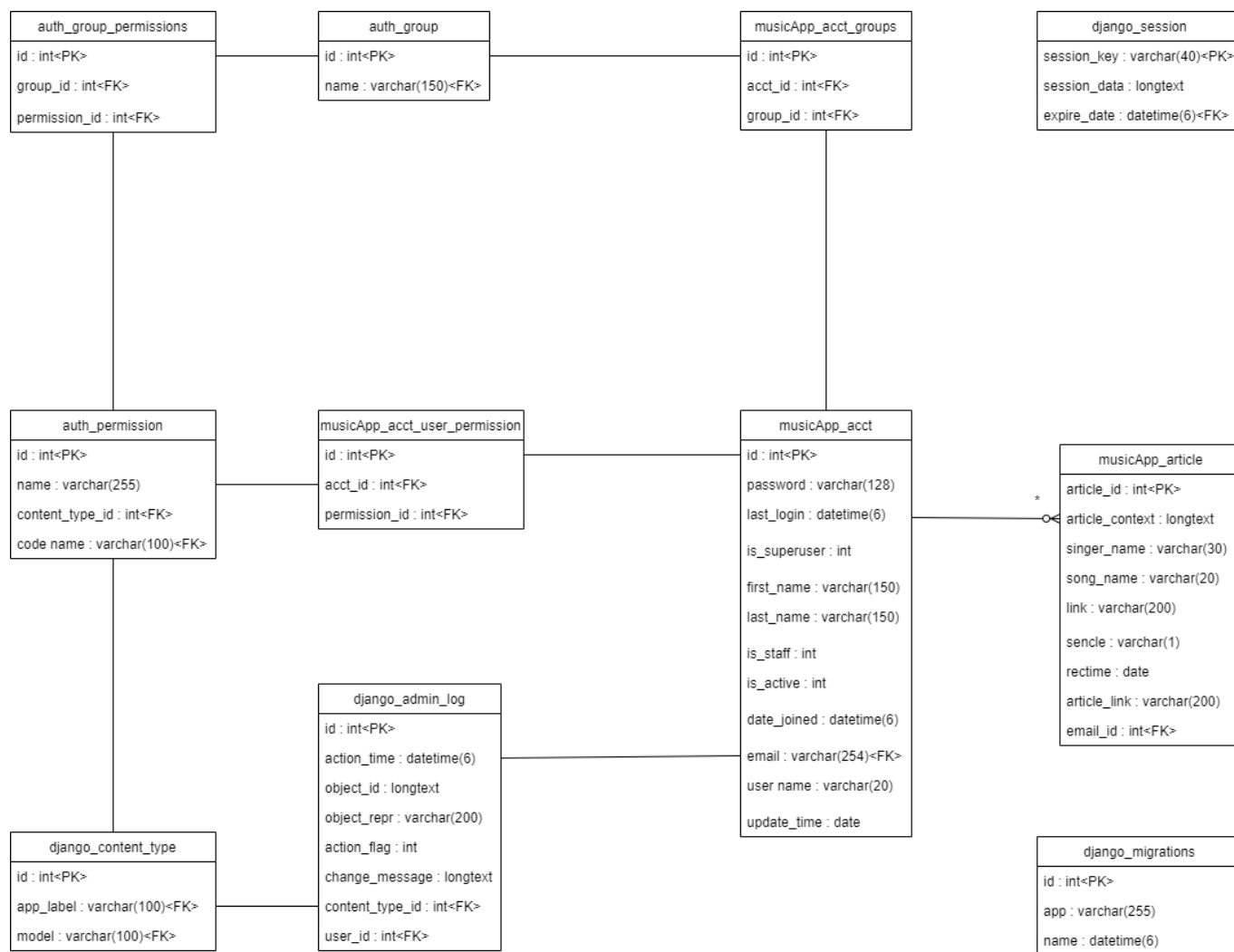


圖 5-4-1 分析類別圖

第六章 設計模型

6-1 循序圖 (Sequence diagram)

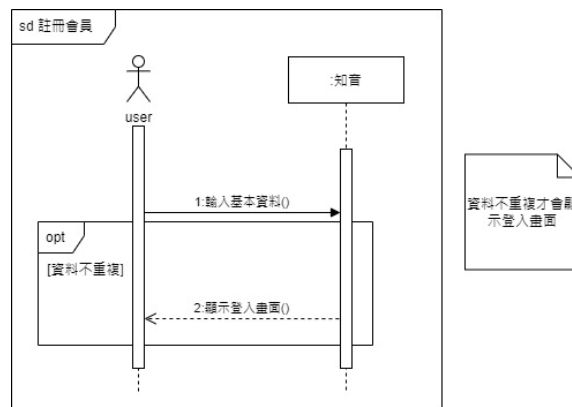


圖 6-1-1 註冊會員循序圖

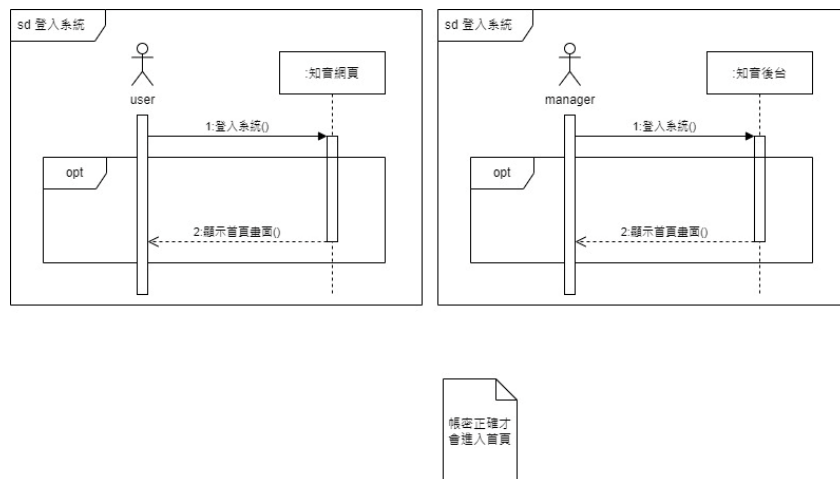


圖 6-1-2 登入系統循序圖

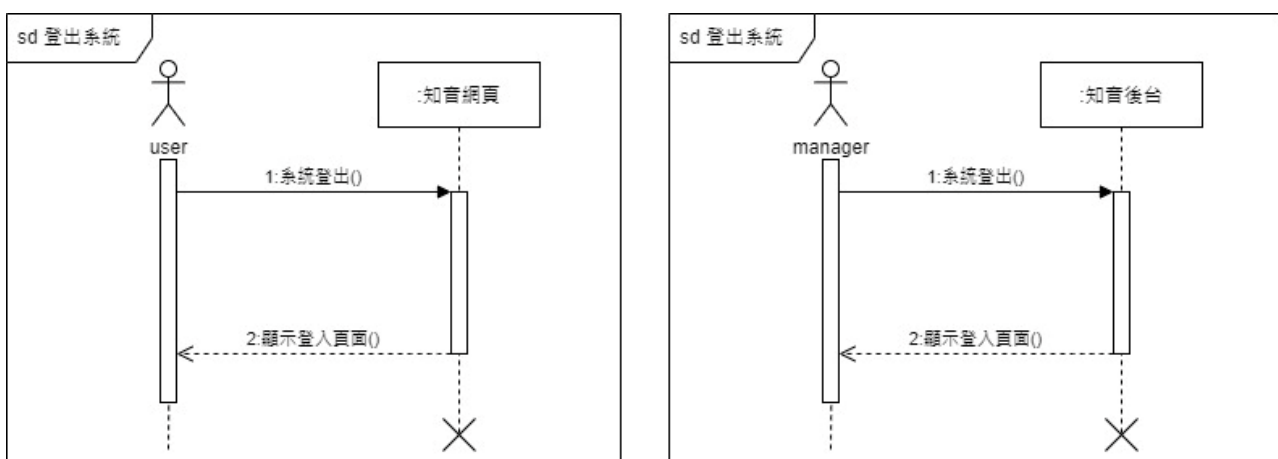


圖 6-1-3 登出系統循序圖

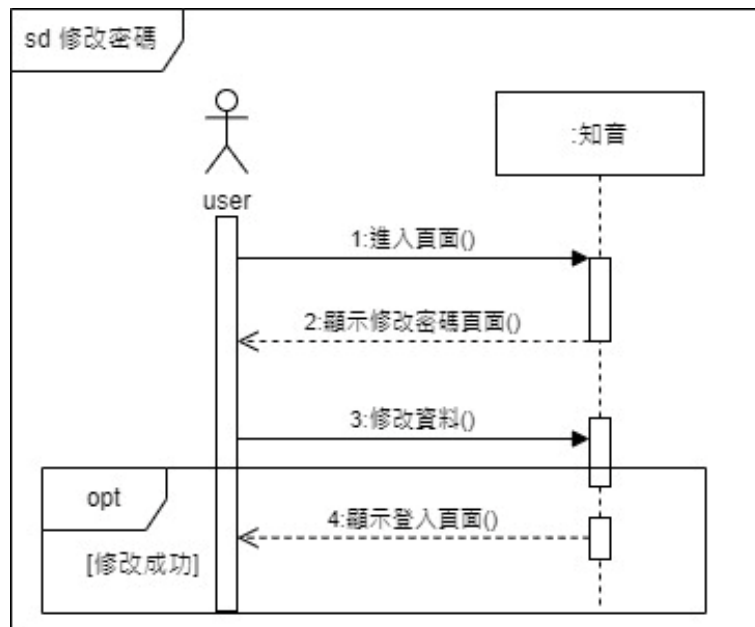


圖 6-1-4 修改密碼循序圖

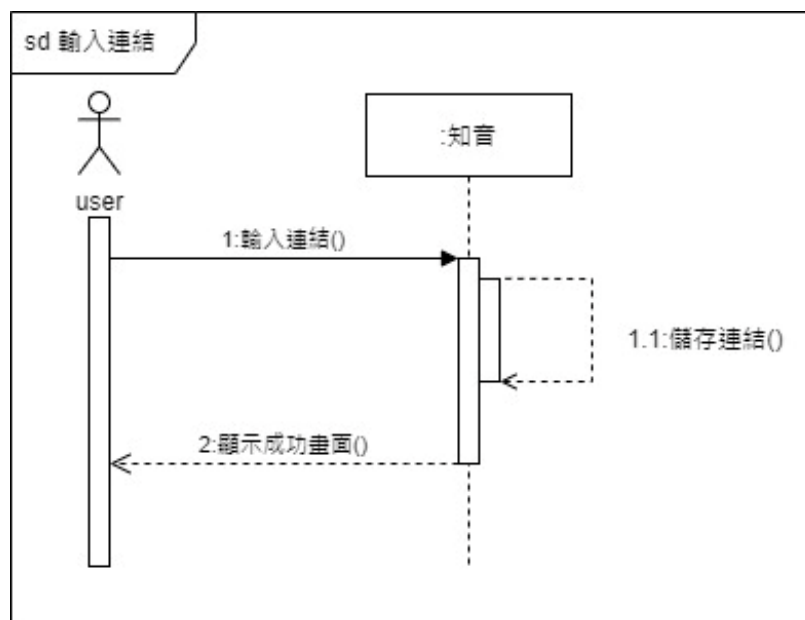


圖 6-1-5 輸入連結循序圖

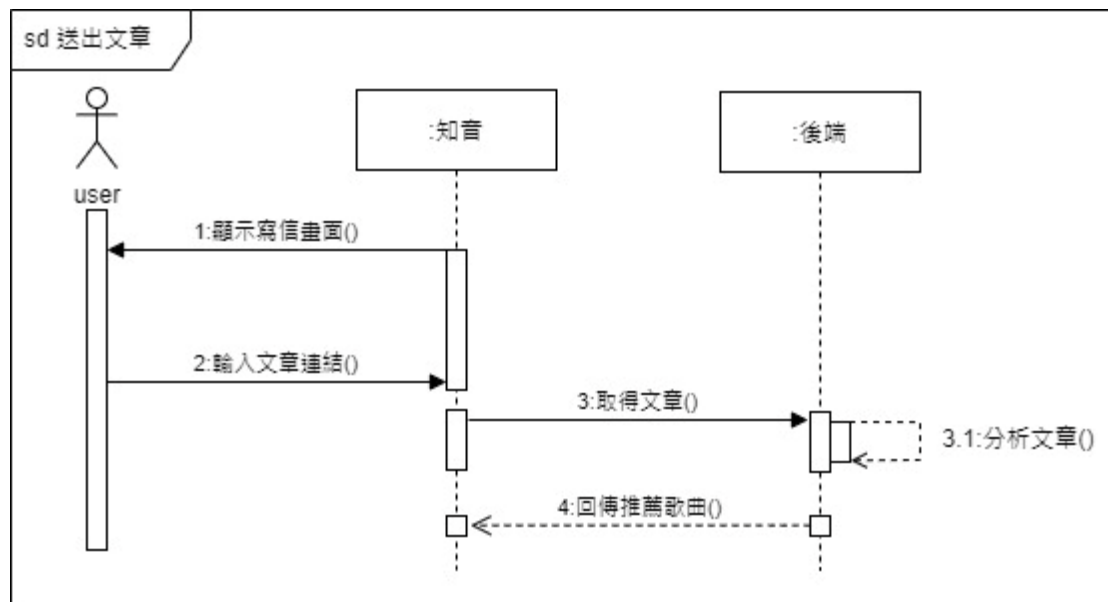


圖 6-1-6 送出文章循序圖

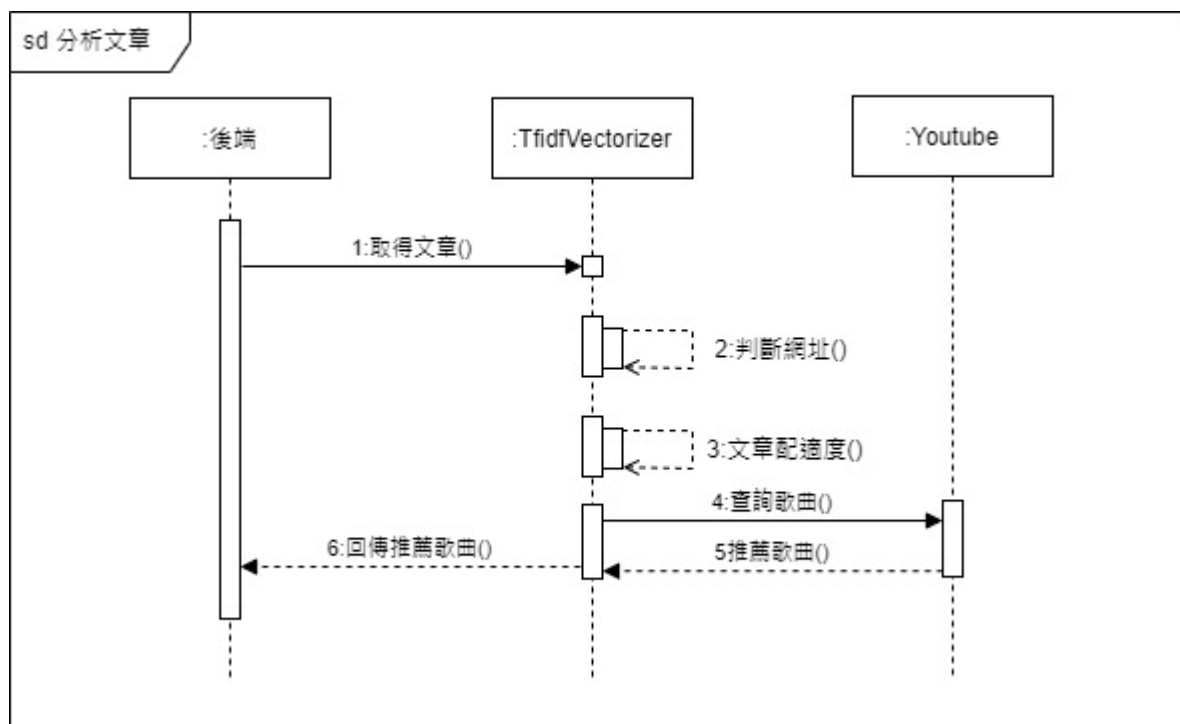


圖 6-1-7 分析文章循序圖

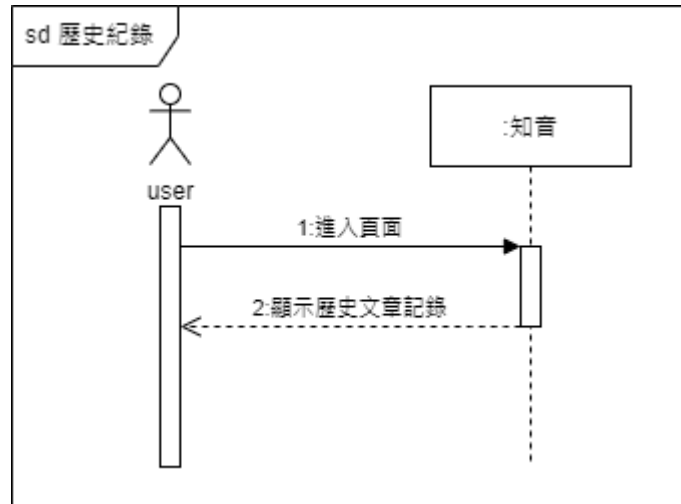


圖 6-1-8 查閱歷史資料循序圖

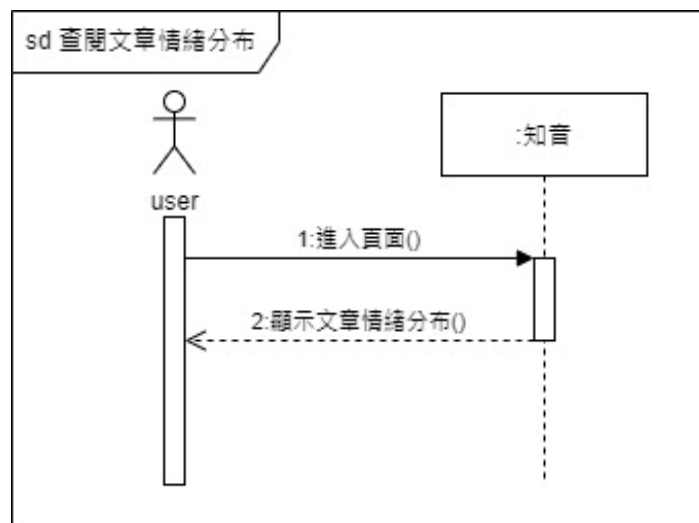


圖 6-1-9 查閱文章情緒分布循序圖

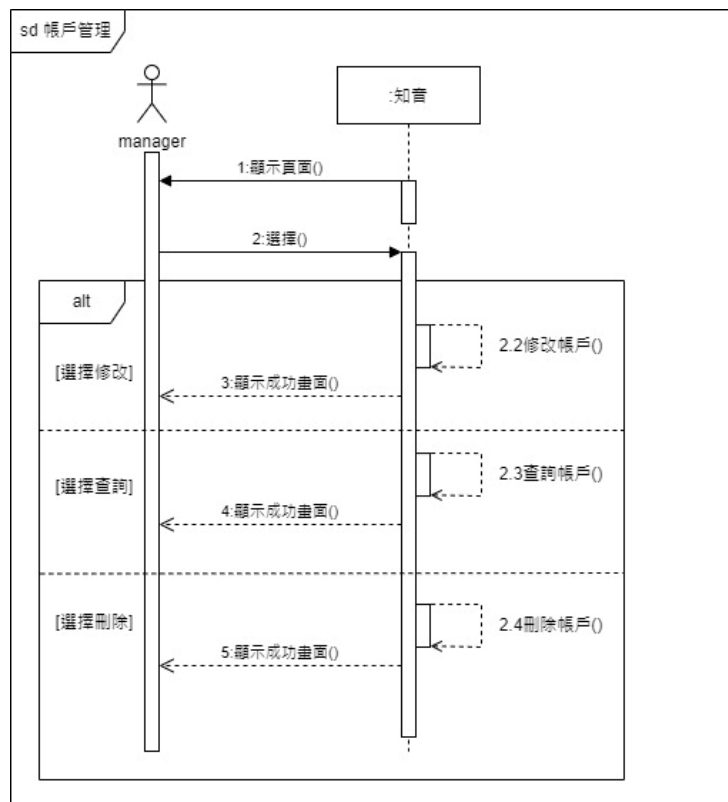


圖 6-1-10 帳戶管理循序圖

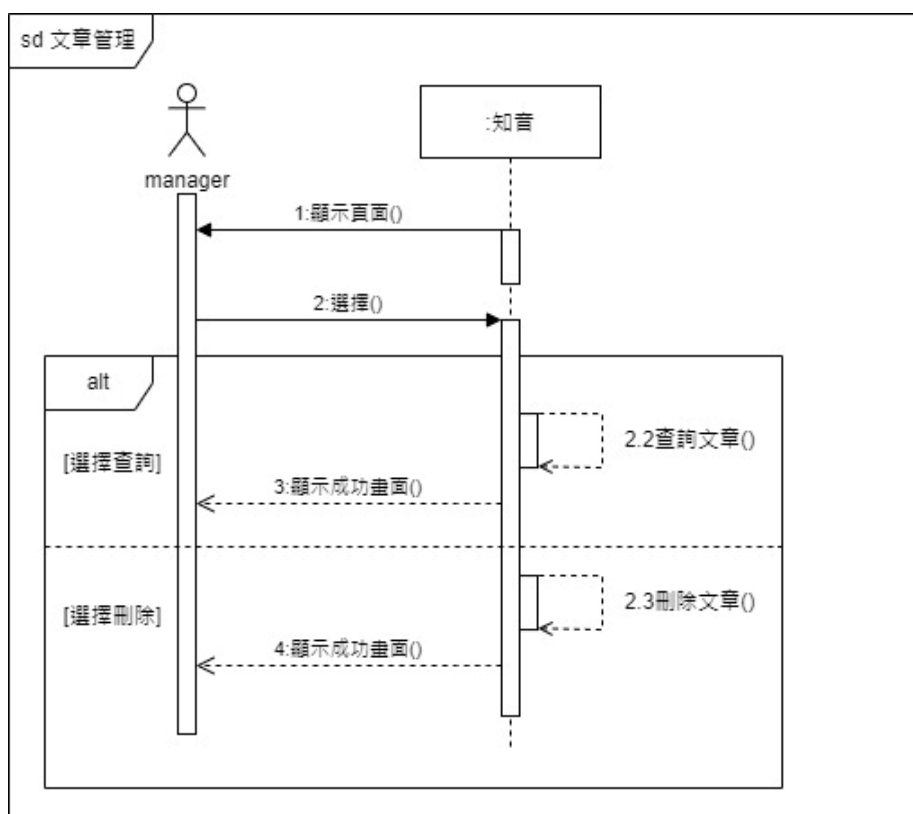


圖 6-1-11 文章管理循序圖

6-2 設計類別圖 (Design class diagram)

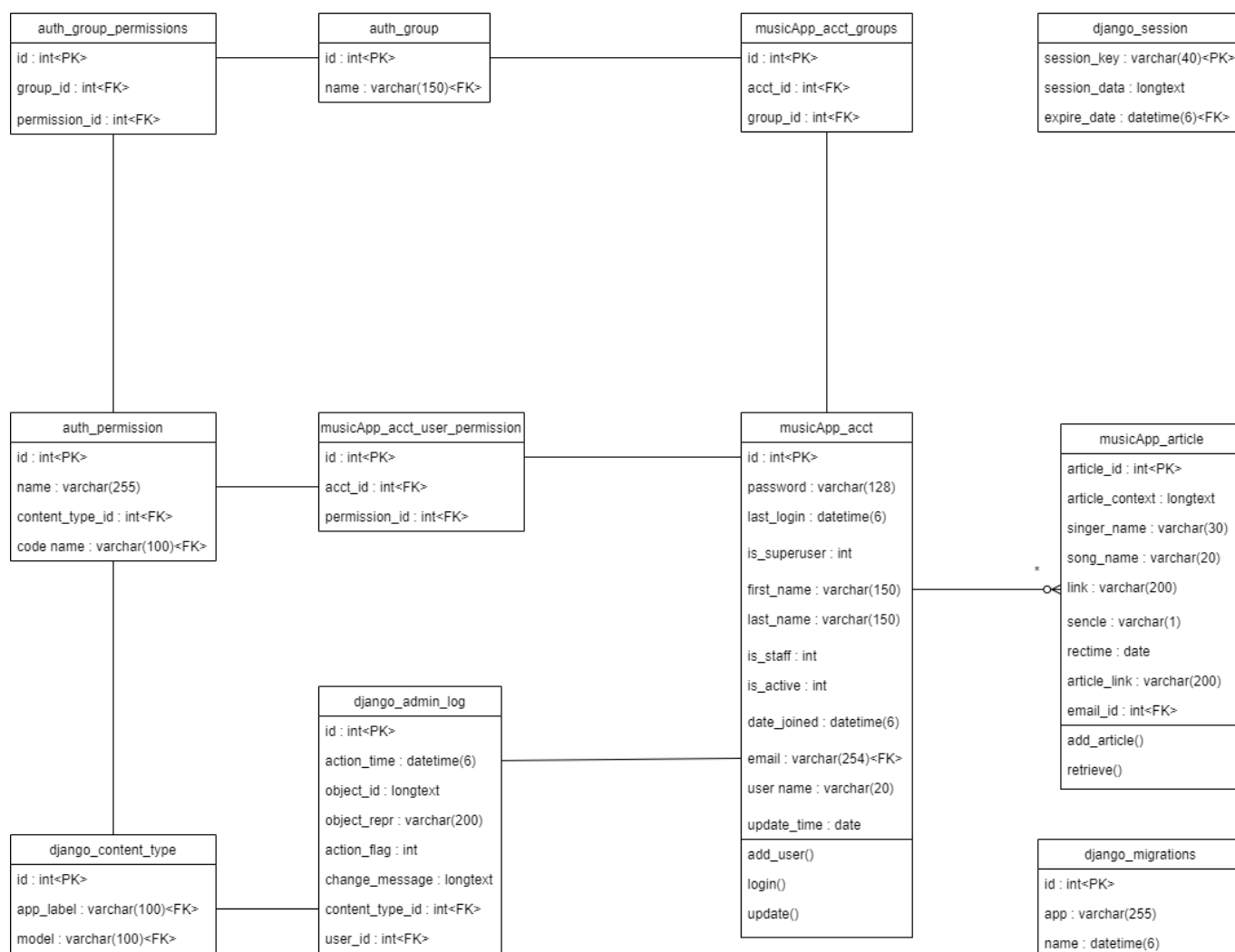


圖 6-2-1 設計類別圖

第七章 實作模型

7-1 部署圖 (Deployment diagram)

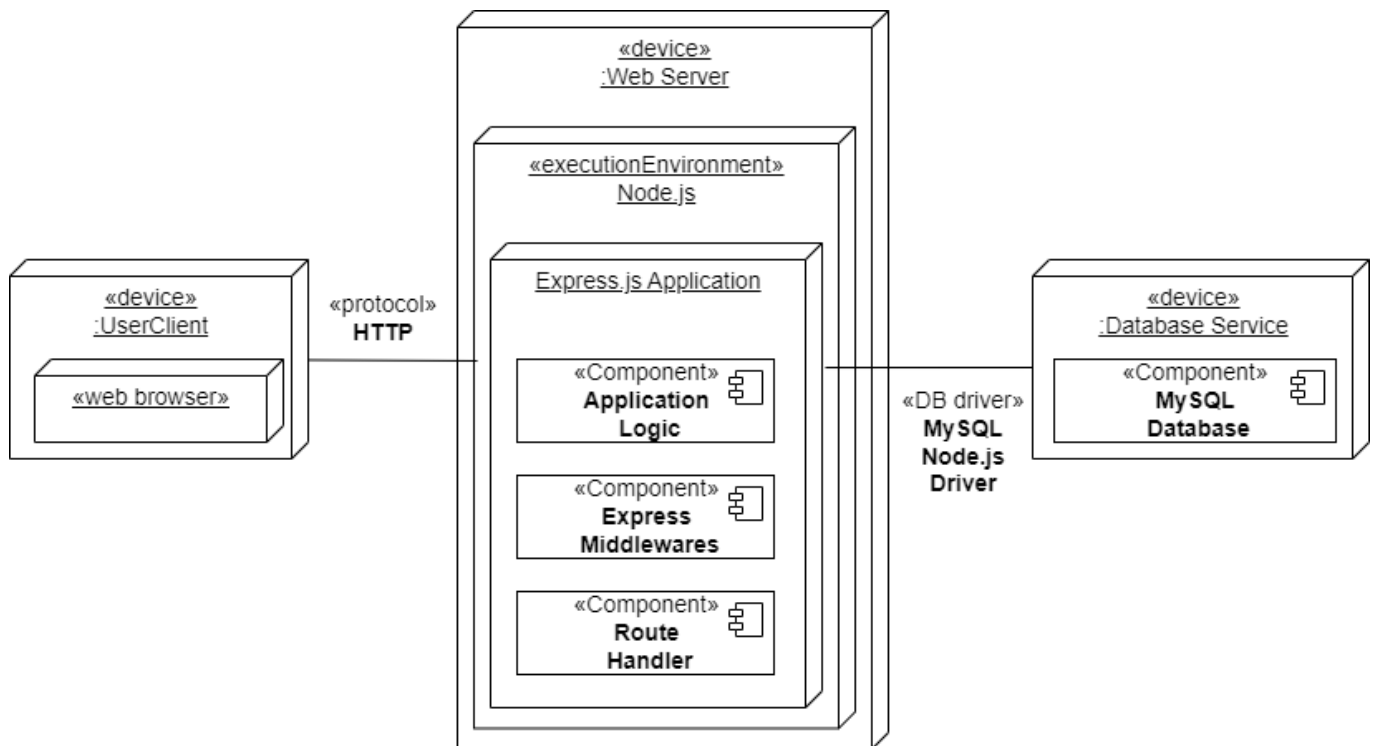


圖 7-1-1 部署圖

7-2 套件圖 (Package diagram)

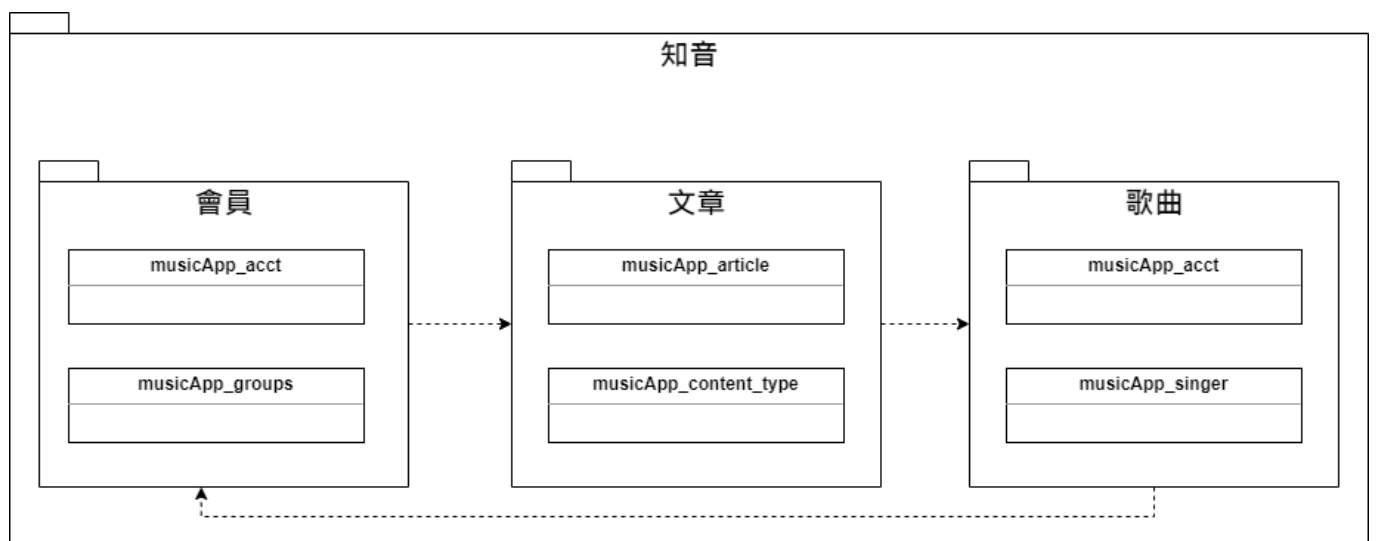


圖 7-2-1 套件圖

7-3 元件圖 (Component diagram)

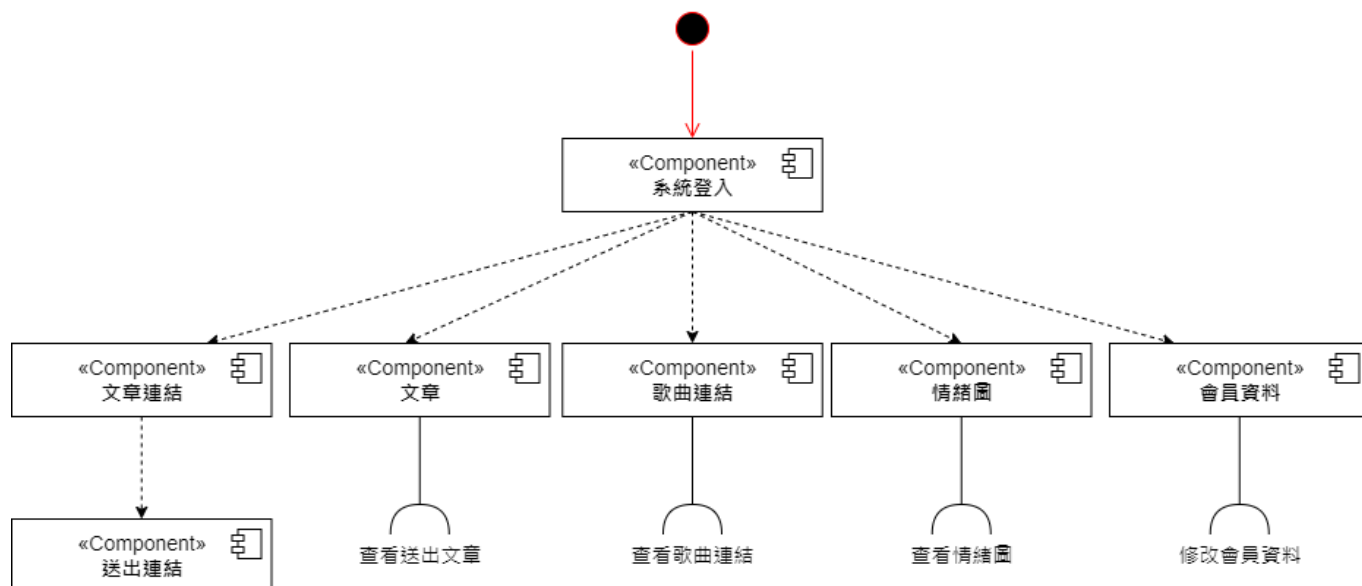


圖 7-3-1 元件圖

7-4 狀態機 (State machine)

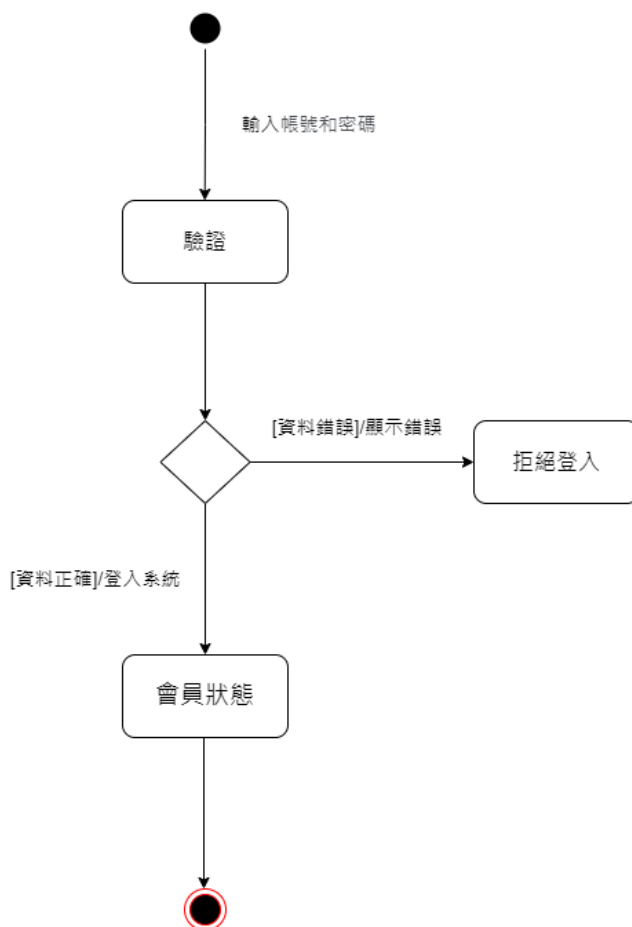


圖 7-4-1 狀態機

第八章 資料庫設計

8-1 資料庫關聯表

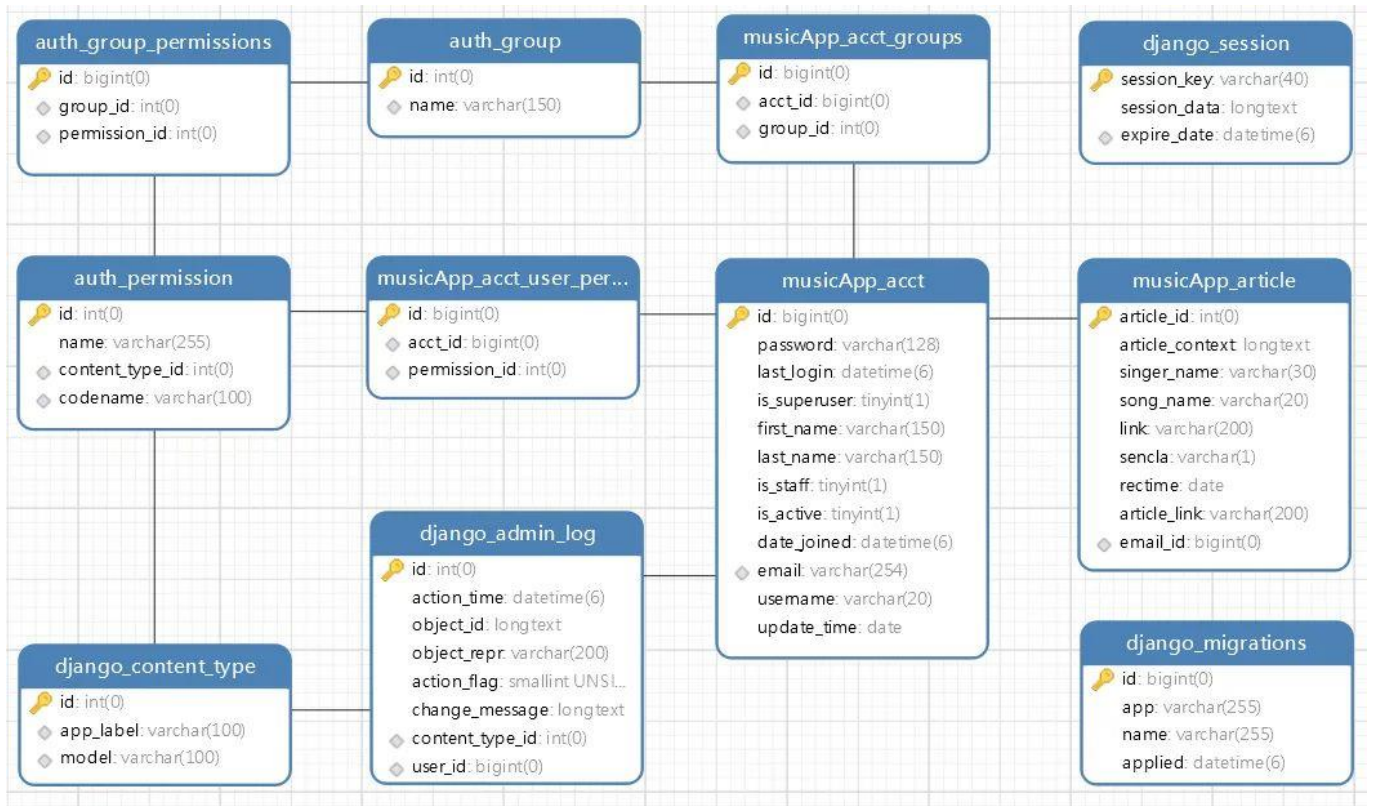


圖 8-1-1 資料庫關聯表

8-2 表格及其 Meta data

表 8-2-1 資料表描述：帳號

資料表名稱：musicApp_acct					
欄位名稱	中文名稱	資料類型	長度	允許空值	備註
id	帳號	int	–	否	PK
password	密碼	varchar	128	否	
last_login	上次登入	datetime	6	否	
is_superuser	管理員	int	–	否	
first_name	名	varchar	150	否	
last_name	姓	varchar	150	否	
is_staff	員工	int	–	否	
is_active	會員	int	–	否	
date_joined	加入時間	datetime	6	否	
email	電子郵件	varchar	254	否	FK
user name	使用者名稱	varchar	20	否	
update_time	更新日期	date	–	否	

表 8-2-2 資料表描述：文章

資料表名稱：musicApp_article					
欄位名稱	中文名稱	資料類型	長度	允許空值	備註
article_id	文章編號	int	-	否	PK
article_context	文章內容	longtext	-	是	
singer_name	歌手名稱	varchar	30	否	
song_name	歌曲名稱	varchar	20	否	
link	連結	varchar	200	否	
sencle	情緒分類	varchar	1	否	
rectime	創立時間	date	-	否	
article_link	文章連結	varchar	200	否	
email_id	電郵編號	int	-	否	FK

第九章 程式

9-1 元件清單及其規格描述

表 9-1-1 元件清單及其規格描述

資料夾名稱：model_compar		
編號	檔案名稱	功能說明
1-1	song_compar.py	推薦歌曲
1-2	dcard_continuous.py	Dcard 文章爬蟲
1-3	ptt_crawler.py	PTT 文章爬蟲
1-4	dcard_if_ptt.ipynb	判斷是 Dcard 或 PTT 連結
1-5	youtube_crawler.ipynb	YouTube 爬蟲
資料夾名稱：django_web/djangoweb_code_1022/code/MusicMain/musicApp		
編號	檔案名稱	功能說明
2-1	manage.py	在 runserver 時去抓程式的設定檔
2-2	admin.py	權限管理
2-3	apps.py	設定名稱
2-4	models.py	後端資料庫建製
資料夾名稱：django_web/djangoweb_code_1022/code/MusicMain/musicApp/api		
編號	檔案名稱	功能說明
3-1	serializers.py	各功能的基礎模型
3-2	views.py	主程式

資料夾名稱：django_web/djangoweb_code_1022/code/MusicMain/MusicMain		
編號	檔案名稱	功能說明
4-1	settings.py	系統各項設定
4-2	urls.py	設定網址的連結字串
資料夾名稱：project-front/code1/components/meetups		
編號	檔案名稱	功能說明
5-1	IndexForm.js	顯示首頁頁面、API 發送
5-2	RegisterForm.js	顯示註冊頁面、API 發送
5-3	PersonalForm.js	顯示個人天地頁面、API 接收
5-4	SendArticleForm.js	顯示寫信頁面、API 接收發送
5-5	MailRecordForm.js	顯示信件記錄頁面、API 接收
5-6	ChangePasswordForm.js	顯示修改密碼頁面、API 發送
5-7	MyChartForm.js	顯示我的情緒圖頁面、API 接收

9-2 其他附屬之各種元件

程式名稱	song_compar.py
部分程式碼	
<pre>from sklearn.feature_extraction.text import TfidfVectorizer import urllib.request as req import numpy as np from scipy.linalg import norm import pandas as pd import bs4, requests # 載入 Selenium 相關模組 from selenium import webdriver from selenium.webdriver.common.by import By from selenium.webdriver.chrome.options import Options def tfidf_similarity(s1, s2): def add_space(s): return ''.join(list(s)) # 將字中間加入空格 s1, s2 = add_space(s1), add_space(s2) # 轉化為 TF 矩陣 cv = TfidfVectorizer(tokenizer=lambda s: s.split()) corpus = [s1, s2] vectors = cv.fit_transform(corpus).toarray() # 計算 TF 係數 return np.dot(vectors[0], vectors[1]) / (norm(vectors[0]) * norm(vectors[1])) def stopwordslist(filepath): stopwords = [line.strip() for line in open(filepath, 'r', encoding='utf-8').readlines()] return stopwords def movestopwords(sentence): stopwords = stopwordslist('stopword.txt') # 這裏加載停用詞的路徑 outstr = "" for word in sentence: if word not in stopwords: if word != "\t" and "\n": outstr += word return outstr</pre>	

ppt 和 dcard 的判斷

def dcardCraw(url):

設定 Chrome Driver 的執行檔路徑

options = Options()

options.add_argument("--incognito") # 啟動進入無痕模式

options.add_argument("--window-size=1,1") # 頁面長度寬度調整

chrome_options.add_argument('--headless') # 啟動 Headless 無頭(隱藏瀏覽器)

隱藏"Chrome 正在受到自動軟體的控制"

options.add_experimental_option("excludeSwitches", ["enable-automation"])

options.add_experimental_option('useAutomationExtension', False)

options.add_argument('--disable-gpu') #關閉 GPU 避免某些系統或是網頁出錯

options.add_argument('--hide-scrollbars') # 隱藏滾動條, 應對一些特殊頁面

options.chromedriver_executable_path =

"C:\\Users\\student\\Desktop\\model_compar\\chromedriver.exe"

#建立 Driver 物件實體, 用程式操作瀏覽器運作

driver = webdriver.Chrome(options = options)

driver.minimize_window() #視窗縮小化

driver.get(url)

data = driver.page_source #取得網頁的原始碼

讓 BeautifulSoup 協助我們解析 HTML 格式文件

root = bs4.BeautifulSoup(data, "html.parser")

dcard 標籤會不定時更換須注意, 用列表顯示全部爬蟲下來的標題

titles = root.find("div", class_ = "sc-ba53eaa8-0 hKkUKs")

for title in titles:

result = title.text.strip().replace("\n", "").replace(' ', "")

print(result) #印出內文

driver.close()

return result

def pttCraw(url):

#建立一個 Request 物件, 附加 Request Headers 的資訊

request = req.Request(url, headers={

"cookie": "over18=1",

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"

```

}))
with req.urlopen(request) as response:
    data = response.read().decode("utf-8")

# print(data)
# 「解析」原始碼，取得每篇文章的問題
# utf-8(比較省空間)有部分的漢字不能轉換所以要用 GB18030 編碼

root = bs4.BeautifulSoup(data, "html.parser") # 讓 BeautifulSoup 協助我們解析 HTML 格式文件
titles = root.find("div", class_ = "bbs-screen bbs-content").text # 用爬蟲抓內文

# 去除掉 target_content
target_content = '※ 發信站: 批踢踢實業坊(ptt.cc),'
content = titles.split(target_content)

# 去除掉 作者 看板 標題 時間
results = root.select('span.article-meta-value')

if len(results)>3:
    # 作者 看板 標題 時間
    firstLine = "作者" + results[0].text + "看板" + results[1].text + "標題" + results[2].text + "時間" + results[3].text

    content = content[0].split(firstLine)

# 去除掉文末 --
main_content = content[1].replace('--', "")

# 去除掉換行
main_content = main_content.replace('\n', "")

# 印出內文
print(main_content)

return main_content

class YoutubeSpider():
    def __init__(self, api_key):
        self.base_url =
        "https://www.googleapis.com/youtube/v3/search?type=video&part=snippet&maxResults=1"

```

```

self.api_key = api_key

def get_html_to_json(self, path):
    """組合 URL 後 GET 網頁並轉換成 JSON"""
    api_url = f'{self.base_url}&key={self.api_key}{path}'
    r = requests.get(api_url)
    if r.status_code == requests.codes.ok:
        data = r.json()
    else:
        data = None
    return data

def get_ytSearch(self, theKey):
    path = f'&q={theKey}'
    data = self.get_html_to_json(path)

    try:
        uploads_id = data['items'][0]['id']['videoId']
        # uploads_id = data #輸出是 dict
    except KeyError:
        uploads_id = None
    return uploads_id

YOUTUBE_API_KEY = "AIzaSyBKdJO0Q7tS8jQyuZUx0kNmFD2L73Bn1E"
youtube_spider = YoutubeSpider(YOUTUBE_API_KEY)

def find_song(url):
    train=pd.read_csv('done_2021-08to12.csv') # 歌曲資料

    if url[12:15] == "dca":
        article = dcardCraw(url)
    else:
        article = pttCraw(url)

    lyrics=train['lyrics']
    i=0
    num=0
    highpri=0
    for text in lyrics:
        text=movestopwords(text)
        text=text.replace(' ','')

```

```

text=text.replace(',', ' ')
if tfidf_similarity(text, article)>highpri:
    highpri=tfidf_similarity(text, article)
    num=i
    i+=1

author = train.singer.iloc[num]
songName = train.name.iloc[num]
youtube_theKey = author + ' ' + songName
uploads_id = youtube_spider.get_ytSearch(youtube_theKey)

print('配適度:',highpri,'歌手:',train.singer.iloc[num],'歌名:',train.name.iloc[num], '情緒:',train.moodCat.iloc[num])
print("連結:https://www.youtube.com/watch?v="+uploads_id)

```

```

find_song('https://www.dcard.tw/f/relationship/p/238632575')
# find_song("https://www.ptt.cc/bbs/Gossiping/M.1664530650.A.4E3.html")

```

```

# 需 pip install
# pip install requests
# pip install BeautifulSoup4
# pip install selenium

```

程式名稱	models.py
------	-----------

部分程式碼

```

from django.db import models
from django.contrib.auth.models import AbstractUser
from django.urls import reverse

# Create your models here.
# 使用者
class Acct(AbstractUser):
    email = models.EmailField(unique=True) #主鍵
    username = models.CharField('姓名', max_length=20) #暱稱
    update_time = models.DateField('最後修改日期', auto_now=True) # 修改時間

    USERNAME_FIELD = 'email' #唯一值
    REQUIRED_FIELDS = ['username']

    def __str__(self):

```

```
return self.email
```

```
def get_absolute_url(self):  
    return reverse("index", kwargs={"pk": self.pk})
```

使用者札記

```
class Article(models.Model):  
    article_id = models.AutoField(primary_key=True) #文章編號  
    article_context = models.TextField('文章內容', max_length=500, null= True, blank=True) #文章  
    內容  
    singer_name = models.CharField('歌手名稱', max_length=30) #歌手名稱  
    song_name = models.CharField('歌曲名稱', max_length=20) # 歌曲名稱  
    link = models.URLField('歌曲連結',max_length=200) #歌曲連結  
    sencla = models.CharField('情感分類',max_length=1) #情感分類  
    rectime = models.DateField('新增時間', auto_now_add=True) #創建文章的時間  
    email = models.ForeignKey('Acct', related_name='acct_email', on_delete=models.RESTRICT) #  
    使用者編號外鍵，auth 是 django 資料庫的內建資料表名稱(關於 user 的資料)
```

歌手

```
class Singer(models.Model):  
    singer_id = models.AutoField(primary_key=True) #歌手編號  
    singer_name = models.CharField('歌手名稱', max_length=30) #歌手名稱
```

程式名稱

serializers.py

部分程式碼

```
from dataclasses import field  
from rest_framework import serializers  
from musicApp.models import Article, Singer, Acct  
from django.contrib.auth.models import User
```

```
class SingerSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Singer  
        fields = ['singer_id', 'singer_name']  
        # fields = '__all__' #選擇全部欄位
```

```
class UserSerializer(serializers.Serializer):  
    email = serializers.EmailField()  
    username = serializers.CharField()  
    password = serializers.CharField()
```

```

class LoginSerializer(serializers.Serializer):
    email = serializers.EmailField()
    password = serializers.CharField()

class CrawlerSerializer(serializers.Serializer): #新增文章的類資料表(爬蟲)
    art_craw = serializers.URLField()

class ArticleSerializer(serializers.ModelSerializer):
    class Meta:
        model = Article
        fields = '__all__' #選擇全部欄位

class ChangePassSerializer(serializers.Serializer): # 修改密碼
    model = User
    # password = serializers.CharField()
    old_password = serializers.CharField(required=True)
    new_password = serializers.CharField(required=True)

#google login token
class TokenSerializer(serializers.Serializer):
    token = serializers.CharField()

```

程式名稱

views.py

部分程式碼

```

from musicApp.api.serializers import LoginSerializer, SingerSerializer, UserSerializer,
ArticleSerializer,CrawlerSerializer, ChangePassSerializer,TokenSerializer
from musicApp.models import Acct, Article, Singer
from rest_framework import viewsets, status, generics
from django.contrib.auth.models import User
from rest_framework.decorators import action
from rest_framework.response import Response
from django.contrib.auth import authenticate
from rest_framework_simplejwt.tokens import RefreshToken
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from musicApp.song_crawler import song_compar
import requests
from django.core.exceptions import ObjectDoesNotExist

def get_tokens_for_user(user):
    refresh = RefreshToken.for_user(user)

    return {

```

```

    'refresh': str(refresh),
    'access': str(refresh.access_token),
}

```

```

class SingerViewSet(viewsets.ModelViewSet):
    # ModelViewSet 已包含增刪改查四種功能
    queryset = Singer.objects.all()
    serializer_class = SingerSerializer

```

```

class UserViewSet(viewsets.GenericViewSet):
    queryset = Acct.objects.all()
    serializer_class = UserSerializer

```

```

def get_serializer_class(self):
    if self.action == "google_login":
        return TokenSerializer
    return super().get_serializer_class()

```

```

@api_action( # 新增使用者
    methods=["POST"], detail=False, url_path="add-user"
) # detail:是否列出 POST 參數，url_path 是寫在路徑中

```

```

def add_user(self, request):
    serailzer = self.get_serializer(data = request.data)
    serailzer.is_valid(raise_exception = True)

    email = serailzer.data["email"]
    username = serailzer.data["username"]
    password = serailzer.data["password"]
    Acct.objects.create_user(username = username, email = email, password = password)

    return Response(data={"message":"add success"})
    # return Response(data={"username":Acct.username, "email":Acct.email})
    # 很彈性看要回傳啥都可

```

```

@api_action( # 登入功能
    methods=["POST"], detail=False, url_path="login"
)

```

```

def login(self, request):
    serailzer = LoginSerializer(data = request.data)
    serailzer.is_valid(raise_exception = True)

```



```

email = serailzer.data["email"]
password = serailzer.data["password"]

acct = authenticate(request, email = email, password = password)
if acct:
    token = get_tokens_for_user(acct)
    return Response(data={"result": "login success", "token":token["access"]})
else:
    return Response(data={"result": "login fail"})

```

```

@action( # 登入功能
    methods=["POST"], detail=False, url_path="google_login"
)
def google_login(self, request):
    s = self.get_serializer(data=request.data)
    s.is_valid(raise_exception = True)

    email = check_token(s.validated_data['token'])

    try:
        acct = Acct.objects.get(email = email)
        token = get_tokens_for_user(acct)
        return Response(data={"result": "login success", "token":token["access"]})
    except ObjectDoesNotExist:
        return Response(data={"result":"login fail"}, status=status.HTTP_404_NOT_FOUND)

```

測試用，用 token 登入(JWT 密鑰)，postman 的 authorization 選 Bearer token 放 token

```

@action(
    methods=["GET"], detail=False, url_path="test", permission_classes = [IsAuthenticated]
)
def test(self, request):
    return Response(data={"result":"test"})

```

```

def check_token(token: str):

```

```

    res = requests.get(
        "https://www.googleapis.com/oauth2/v1/userinfo",
        {
            'alt':'json',

```

```

        'access_token':token
    }
)

user_info = res.json()

if not user_info.get('email'):
    return Response(data={"result":"google get email error"},status =
status.HTTP_401_UNAUTHORIZED)

return user_info['email']

class ChangePasswordView(generics.UpdateAPIView): # PATCH

    serializer_class = ChangePassSerializer
    model = User
    permission_classes = (IsAuthenticated,) # 需帶此 token

    def get_object(self, queryset=None):
        obj = self.request.user
        return obj

    def update(self, request, *args, **kwargs):
        self.object = self.get_object()
        serializer = self.get_serializer(data=request.data)

        if serializer.is_valid():
            # Check old password
            if not self.object.check_password(serializer.data.get("old_password")):
                return Response({"old_password": ["Wrong password."]},
status=status.HTTP_400_BAD_REQUEST)
            # set_password also hashes the password that the user will get
            self.object.set_password(serializer.data.get("new_password"))
            self.object.save()
            response = {
                'status': 'success',
                'code': status.HTTP_200_OK,
                'message': 'Password updated successfully',
                'data': []
            }

```

```

        return Response(response)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class CrawlerViewSet(viewsets.GenericViewSet): #新增文章
    queryset = Article.objects.all()
    serializer_class = CrawlerSerializer
    # model = Acct

    @action( # 新增使用者
        methods = ["POST"], detail = False, url_path = "add-article", permission_classes =
[IsAuthenticated]
    )

    def add_article(self, request):
        # self.object = self.get_object()
        serailzer = self.get_serializer(data = request.data)
        serailzer.is_valid(raise_exception = True)

        artCraw = serailzer.data["art_craw"]
        resCraw = song_compar.find_song(artCraw)
        # print(type(resCraw))
        usr_email = Acct.objects.get(email=request.user.email)
        print(usr_email)
        Article.objects.create(article_context = resCraw['article'], singer_name = resCraw['singer'],
song_name = resCraw['song'], link = resCraw['songURL'], sencla = resCraw['art_mood'], email =
usr_email)
        # rectime 不用寫日期去存，django 會自動幫我們以最新的時間存進資料庫中

        return Response(data={"result": resCraw})

class ArticleViewSet(viewsets.ModelViewSet):
    # ModelViewSet 已包含增刪改查四種功能
    queryset = Article.objects.all()
    serializer_class = ArticleSerializer

```

程式名稱	IndexForm.js
部分程式碼	
<pre> return (<Fragment> { /* link */} </pre>	

```

    { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap"
rel="stylesheet"/> */}

    <Head>
      <title>登入</title>
      <meta
        name="description"
        content="Browse a huge list of active React meetups!"
      />
    </Head>

    <div class="grid grid-cols-2 gap-x-10">
      { /* 歡迎文字區 */}
      <div class="grid grid-rows-4 gap-y-4 mt-12">
        <div class="text-6xl">邀請您</div>
        <div class="text-6xl">一起向未來</div>
        <div class="text-6xl">寄封信。</div>
        <div>
          <Link href="register" passHref>
            <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
            hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
            100 focus:outline-none">
              開始寫信...
            </button>
          </Link>
        </div>
      </div>

      { /* 登入表單 */}
      <div class="grid gap-y-3 mx-auto w-full max-w-[550px]">
        <label for="name" class="mb-3 block text-center text-3xl font-bold">
          登入
        </label>

        { /* 帳號 */}
        <div>
          <input
            type="email"
            name="email"
            id="email"
            placeholder="帳號（電子郵件）"
            ref={emailInputRef}

```

```

        class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base font-
medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
    />
</div>

{/* 密碼 */}
<div>
    <input
        type="password"
        name="password"
        id="password"
        placeholder="密碼"
        ref={passwordInputRef}
        class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base font-
medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
    />
</div>

{/* 登入 btn */}
<div>
    {/ <Link href="personal_space" passHref>
        <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
100 focus:outline-none">
            登入
        </button>
    </Link> */}
    <button
        onClick={submitHandler}
        class="w-full rounded-md bg-white transition duration-150 ease-in-out hover:border-
gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-100
focus:outline-none"
    >
        登入
    </button>
</div>

{/* Google 登入 btn */}
<div>

```

```

    <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
100 focus:outline-none">
        以 Google 帳號登入
    </button>
    <label
        for="name"
        class="mb-3 block text-center text-base font-bold"
    >
        還沒註冊?
    </label>
</div>

{ /* 註冊 btn */}
<div>
    <Link href="register" passHref>
        <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
100 focus:outline-none">
            註冊
        </button>
    </Link>
</div>
</div>
</div>
</Fragment>
);

```

程式名稱

Register.js

部分程式碼

```

return (
    <Fragment>
        { /* link */}
        { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap"
rel="stylesheet"/> */}
    <Head>
        <title>註冊</title>
        <meta
            name="description"
            content="Browse a huge list of active React meetups!"
        />
    </Head>

```

```

    { /* 註冊表單 */ }
    <div class="flex items-center justify-center p-12 pt-0">
      <div class="mx-auto w-full max-w-[550px]">
        <form action="https://formbold.com/s/FORM_ID" method="POST">
          <label
            for="name"
            class="mb-3 block text-center text-3xl font-bold"
          >
            註冊
          </label>
          <div class="w-full rounded-md py-3 px-6 text-base outline-none focus:border-gray-800
focus:shadow-md">
            { /* 使用者名稱 */ }
            <div class="mb-5">
              <input
                type="text"
                name="name"
                id="name"
                placeholder="使用者名稱"
                class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
              />
            </div>

            { /* 帳號 */ }
            <div class="mb-5">
              <input
                type="email"
                name="email"
                id="email"
                placeholder="帳號（請輸入電子郵件）"
                class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
              />
            </div>

            { /* 密碼 */ }
            <div class="mb-5">
              <input
                type="password"

```

```

        name="password"
        id="password"
        placeholder="密碼"
        class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
    />
</div>

    { /* 確認密碼 */ }
    <div class="mb-5">
        <input
            type="password"
            name="password"
            id="password"
            placeholder="確認密碼"
            class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
        />
    </div>

    { /* 註冊 btn */ }
    <div class="mb-5">
        <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
100 focus:outline-none">
            註冊
        </button>
    </div>
</div>
</form>
</div>
</div>
</Fragment>
);

```

程式名稱

PersonalForm.js

部分程式碼

```

return (
    <div class="w-full h-full">
        { /* link */ }
        { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap"
rel="stylesheet"/> */ }
    </div>
);

```



```

<div class="flex flex-row place-content-center">

  {/ * 我的書桌 */}
  <div class="basis-1/2 border-solid border-2 border-black mr-10">
    <div class="flex items-center justify-center p-12 pt-0">
      <div class="mx-auto w-full max-w-[550px]">
        <form action="https://formbold.com/s/FORM_ID" method="POST">
          <label
            for="name"
            class="my-10 block text-center text-3xl font-bold"
          >
            <p class="underline underline-offset-4 decoration-2">我的書桌</p>
          </label>

          {/ * 寫信 btn */}
          <div class="mb-5">

            <Link href="personal_space/SendArticle" passHref>

              <button class="w-full rounded-md bg-white transition duration-150 ease-in-
out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">
                寫信
              </button>
            </Link>
          </div>

          {/ * 信件紀錄 btn */}
          <div class="mb-5">
            <Link href="personal_space/MailRecord">
              <button class="w-full rounded-md bg-white transition duration-150 ease-in-
out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">
                信件紀錄
              </button>
            </Link>
          </div>

        </form>
      </div>
    </div>
  </div>

```

```

</div>

{/* 個人檔案 */}
<div class="basis-1/2 border-solid border-2 border-black ml-10">
  <div class="flex items-center justify-center p-12 pt-0">
    <div class="mx-auto w-full max-w-[550px]">
      <form action="https://formbold.com/s/FORM_ID" method="POST">
        <label
          for="name"
          class="my-10 block text-center text-3xl font-bold;"
        >
          <p class="underline underline-offset-4 decoration-2">個人檔案</p>
        </label>

        {/* 修改密碼 btn */}
        <div class="mb-5">

          <Link href="personal_space/ChangePassword" passHref>

            <button class="w-full rounded-md bg-white transition duration-150 ease-in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-100 focus:outline-none">
              修改密碼
            </button>
          </Link>
        </div>

        {/* 我的情緒圖 btn */}
        <div class="mb-5">
          <Link href="personal_space/MyChart" passHref>
            <button class="w-full rounded-md bg-white transition duration-150 ease-in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-100 focus:outline-none">
              我的情緒圖
            </button>
          </Link>
        </div>

      </form>
    </div>
  </div>
</div>

```

<div> <div> <div>); </div> </div> </div>	
程式名稱	SendArticleForm.js
部分程式碼	
<pre> return (<Fragment> { /* link */ } { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap" rel="stylesheet"/> */ } <Head> <title>送出文章</title> <meta name="description" content="Browse a huge list of active React meetups!" /> </Head> { /* 文章送出表單 */ } <div class="flex items-center justify-center p-12 pt-0"> <div class="mx-auto w-full max-w-[550px]"> <form action="https://formbold.com/s/FORM_ID" method="POST"> <label for="name" class="mb-3 block text-left text-1xl font-bold" > 個人天地 - 寫信 </label> <div class="w-full rounded-md border-[#000000] border-[3px] py-3 px-6 text- base outline-none focus:border-gray-800 focus:shadow-md"> { /* 文章連結 */ } <div class="mb-5"> <input type="text" name="url" id="url" placeholder="文章連結" </pre>	

```

        class="w-full rounded-md border border-[#e0e0e0] bg-white py-3
px-6 text-base font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
    />
</div>

    { /* dcard 文章送出 btn */ }
    <div class="mb-5">
        <button class="w-full rounded-md bg-white transition duration-150 ease-
in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">
            Dcard 連結送出
        </button>
    </div>
    { /* ptt 文章送出 btn */ }
    <div class="mb-5">
        <button class="w-full rounded-md bg-white transition duration-150 ease-
in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">
            PTT 連結送出
        </button>
    </div>

    <label
        for="name"
        class="mb-3 block text-center text-base font-bold">
    </label>

    <label
        for="name"
        class="mb-3 block text-center text-base font-bold">
    </label>
    </div>
</form>

</div>
</div>

</Fragment>
);

```

程式名稱

MailRecordForm.js

部分程式碼

```

return (
  <Fragment>
    { /* link */}
    { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap"
rel="stylesheet"/> */}
    <Head>
      <title>信件紀錄</title>
      <meta
        name="description"
        content="Browse a huge list of active React meetups!"
      />
    </Head>

    { /* 文章送出表單 */}

    <div class="flex items-center justify-center p-12 pt-0">
      <div class="mx-auto w-full max-w-[550px]">
        <form action="https://formbold.com/s/FORM_ID" method="POST">
          <label
            for="name"
            class="mb-3 block text-left text-1xl font-bold"
          >
            個人天地 - 信件紀錄
          </label>
          <div class="w-full rounded-md border-[#000000] border-[3px] py-3 px-6 text-
base outline-none focus:border-gray-800 focus:shadow-md">

            { /* 歌曲連結 */}
            <div class="mb-5">
              <button class="w-full rounded-md bg-white transition duration-150 ease-
in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">
                歌曲連結
              </button>
            </div>

            { /* 文章連結 */}
            <div class="mb-5">
              <button class="w-full rounded-md bg-white transition duration-150 ease-
in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-
gray-100 focus:outline-none">

```

文章連結

</button>

</div>

<div class="flex flex-row place-content-center">

{/* 時間 */}

<div class="mb-5">

<button class="w-full rounded-md bg-white transition duration-150 ease-in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-100 focus:outline-none">

時間

</button>

</div>

{/* 心情 */}

<div class="mb-5">

<button class="w-full rounded-md bg-white transition duration-150 ease-in-out hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-100 focus:outline-none">

心情

</button>

</div>

</div>

</div>

</form>

</div>

</div>

</Fragment>

);

程式名稱

ChangePasswordForm.js

部分程式碼

return (

<Fragment>

{/* link */}

{/* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap" rel="stylesheet"/> */}

```

<Head>
  <title>修改密碼</title>
  <meta
    name="description"
    content="Browse a huge list of active React meetups!"
  />
</Head>

{/* 修改密碼表單 */}
<div class="flex items-center justify-center p-12 pt-0">
  <div class="mx-auto w-full max-w-[550px]">
    <form action="https://formbold.com/s/FORM_ID" method="POST">
      <label
        for="name"
        class="mb-3 block text-center text-3xl font-bold"
      >
        修改密碼
      </label>

      {/* 帳號 */}
      <div class="mb-5">
        <input
          type="email"
          name="email"
          id="email"
          placeholder="帳號（請輸入電子郵件）"
          class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
        />
      </div> */}

      {/* 密碼 */}
      <div class="mb-5">
        <input
          type="password"
          name="password"
          id="password"
          placeholder="密碼"
          class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
        />
      </div>
    </form>
  </div>
</div>

```

```

    </div>

    { /* 確認密碼 */ }
    <div class="mb-5">
      <input
        type="confirmPassword"
        name="confirmpassword"
        id="confirmpassword"
        placeholder="確認密碼"
        class="w-full rounded-md border border-[#e0e0e0] bg-white py-3 px-6 text-base
font-medium text-[#6B7280] outline-none focus:border-gray-800 focus:shadow-md"
      />
    </div>

    { /* 儲存修改 button */ }
    <div class="mb-5">
      <button class="w-full rounded-md bg-white transition duration-150 ease-in-out
hover:border-gray-900 hover:text-gray-900 border text-gray-800 px-6 py-2 text-base hover:bg-gray-
100 focus:outline-none">
        儲存修改
      </button>
    </div>

  </form>
</div>
</div>
</Fragment>
);

```

程式名稱

MyChartForm.js

部分程式碼

```

return (
  <Fragment>
    { /* link */ }
    { /* <link href="https://fonts.googleapis.com/css?family=Noto+Serif+TC&display=swap"
rel="stylesheet"/> */ }
    <Head>
      <title>我的情緒圖</title>
      <meta
        name="description"
        content="Browse a huge list of active React meetups!"
      />

```



```
</Head>
<div>
  <Pie data={pie_data} />;
</div>
</Fragment>
);
```

第十章 測試模型

10-1 測試計劃

表 10-1-1 測試流程

編號	功能名稱	測試流程
1	註冊	至登入頁面點選註冊按鈕並輸入所需之資料
2	登入	至登入頁面輸入帳號、密碼
3	寫信	至個人天地點選寫信按鈕後貼上文章連結
4	信件記錄	至個人天地點選信件記錄中確認過往文章存在
5	個人情緒圖	至個人天地點選個人情緒圖按鈕查詢情緒分佈
6	修改帳密	至個人天地點選修改帳密後輸入所需之資料

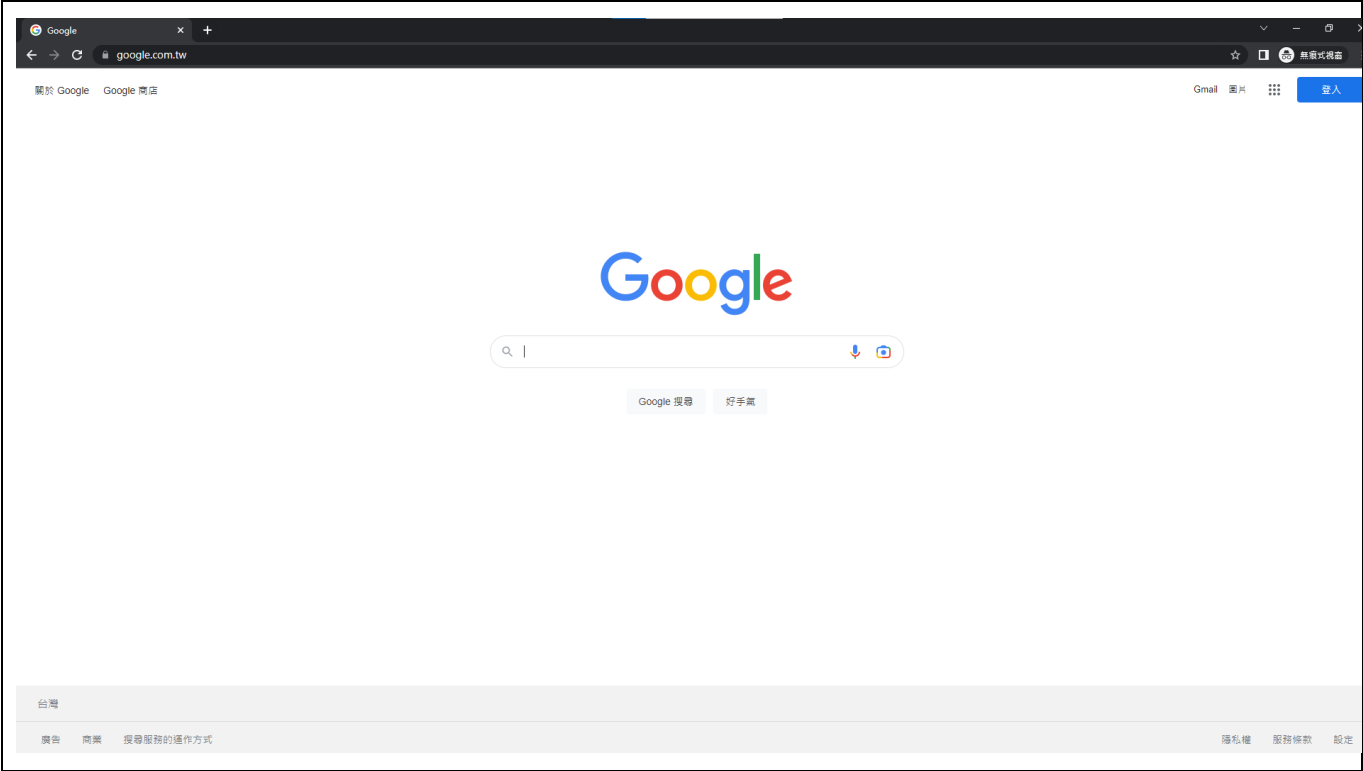

10-2 測試個案與測試結果資料

表 10-2-1 測試結果

編號	功能名稱	測試流程	測試結果
1	註冊	至登入頁面點選註冊按鈕並輸入所需之資料	可使用
2	登入	至登入頁面輸入帳號、密碼	可使用
3	寫信	至個人天地點選寫信按鈕後貼上文章連結	可使用
4	信件記錄	至個人天地點選信件記錄中確認過往文章存在	可使用
5	個人情緒圖	至個人天地點選個人情緒圖按鈕查詢情緒分佈	可使用
6	修改帳密	至個人天地點選修改帳密後輸入所需之資料	可使用

第十一章 操作手冊

表 11-1-1 操作手冊表

步驟一：開啓任一瀏覽器

步驟二：輸入網址 localhost:3000


第十二章 使用手冊

表 12-1-1 操作手冊表

註冊：輸入輸入帳號、密碼	
知音	首頁 個人天地 登出
<div>註冊</div> <div><input type="text" value="roger"/></div> <div><input type="text" value="1234@gmail.com"/></div> <div><input type="password" value="...."/></div> <div><input type="password" value="...."/></div> <div>註冊</div>	

登入：輸入輸入帳號、密碼	
知音	首頁 個人天地 登出
<div>邀請您 一起向未來 寄封信。</div> <div>開始寫信...</div>	<div>登入</div> <div><input type="text" value="roger"/></div> <div><input type="password" value="....."/> </div> <div>登入</div> <div>以Google帳號登入</div> <div>還沒註冊?</div> <div>註冊</div>

登入後進入個人天地

知音

首頁

個人天地

登出

我的書桌

寫信

信件紀錄

個人檔案

修改密碼

我的情緒圖

點選寫信

知音

首頁

個人天地

登出

個人天地 - 寫信

文章連結

Dcard連結送出

PTT連結送出

你的文章內容
知音為你推薦的歌曲

貼上文章連結並送出

知音

首頁

個人天地

登出

個人天地 - 寫信

<https://www.dcard.tw/f/relationship/p/238632575>

Dcard連結送出

PTT連結送出

你的文章內容

知音為你推薦的歌曲

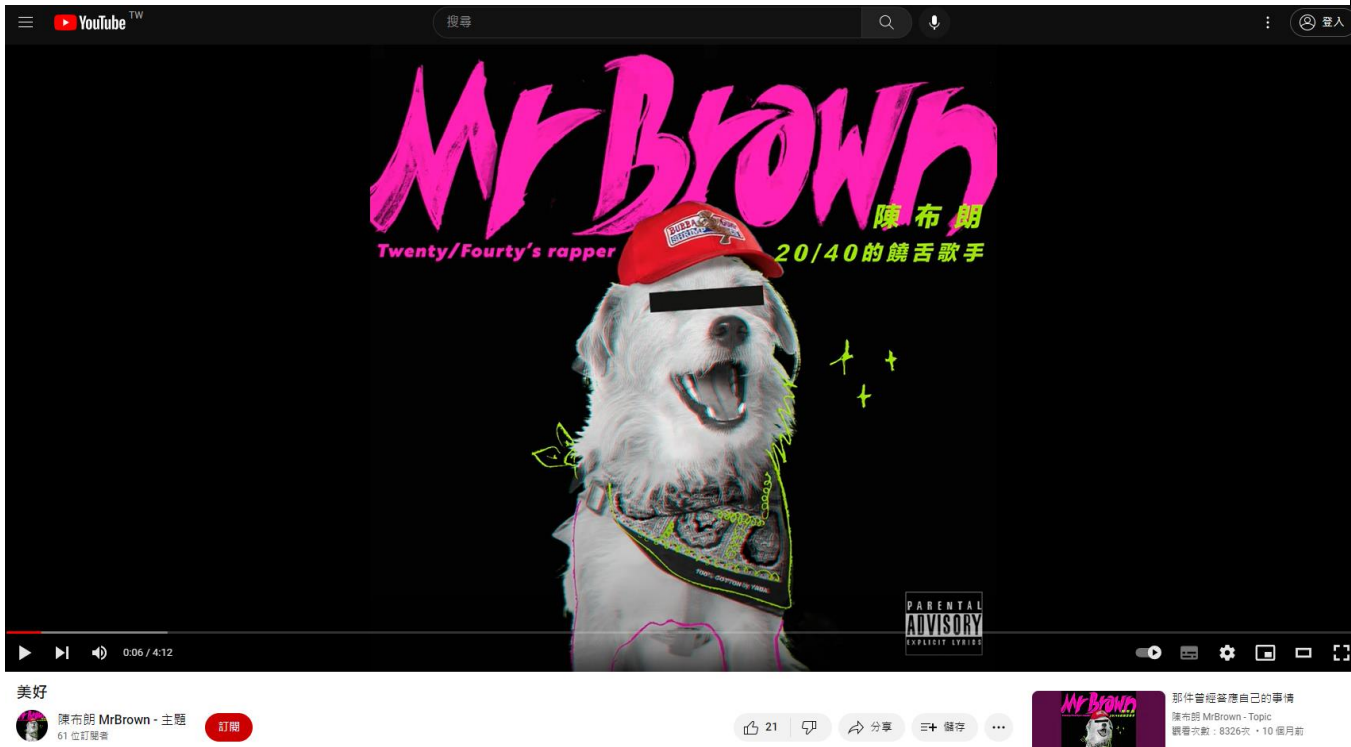
取得推薦的歌曲 YouTube 連結

正文22/04/16前幾天原本還好好的我們，你突然對我說：「我想了很久，我喜歡你，但我…其實沒有那麼喜歡你。」我們確實沒有交往很久，但從我喜歡你以來我一直都真心付出，而我好喜歡你。在別人的眼中，有些人覺得你的條件不夠好，可我從不這麼認為，也從沒因此減少我對你的愛意……我不知道該回答什麼。「那…你是要我們分手嗎？」你沈默了。「我只是想告訴你，我發覺我好像還沒準備好，這樣對你來說不公平。但你現在知道了，你可以選擇分手，或者我們繼續交往，等待我。」我知道你被前任傷害過，可如今的你卻是傷害我的那個。「你還喜歡我嗎？」「喜歡，只是沒有妳那麼喜歡我。」「你想分手嗎？」「我…不知道。我真的不知道，每當你對我好，我會覺得對不起你，因為我還沒辦法給你我的全心。如果你不能接受，那我們就分手，如果你願意等我，我們就像往常一樣」我哭了，而你也哭了。我們討論著我們的關係好幾天。直到最近，我問你，你想分手嗎？你說：「我想跟以前一樣…只是…我現在真的還沒辦法給我全部的愛意，但我會努力。你呢？」我也不知道。如果現在沒那麼喜歡，那以後會嗎？我們有辦法像往常一樣嗎？如果你狠下心告訴我不喜歡我，我願意直接離開。可你的選擇卻是希望我留下，我猶豫了。

美好 - 陳布朗

<https://www.youtube.com/watch?v=e0FXJGt7ZY4>

點選連結進入 YouTube 享受知音為您推薦的歌曲



進入信件記錄查詢過去知音為您推薦的歌曲

知音

首頁

個人天地

登出

個人天地 - 信件紀錄

歌曲連結

文章連結

時間

心情

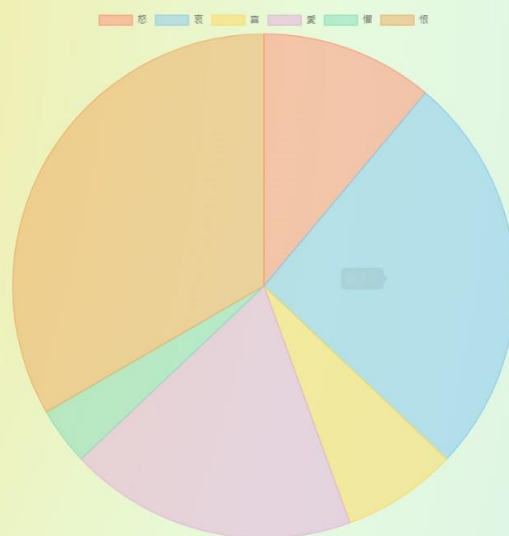
進入我的情緒圖查詢情緒分佈狀態

知音

首頁

個人天地

登出



進入修改密碼可更新舊的密碼

知音

首頁

個人天地

登出

修改密碼

儲存修改

第十三章 感想

11036008 陳姵蓉

在這個專題裡我負責的是網頁後端和 AI 的部分，從建立網頁框架到撰寫 AI 的歌曲推薦，了解到很多之前沒有碰過的技術，像是 django-rest-framework、JWT、youtubeAPI 等等，為了要讓功能可以順利執行，就開始去詢問老師、學長姐問題，他們不厭其煩的告訴我怎麼撰寫，我也上網查了很多的做法，慢慢的把功能拼湊出來，並且能順利執行，完成的當下真的是成就感滿滿，也覺得之前的努力很值得。在撰寫 AI 的程式時是一個極大的挑戰，因為我在 AI 的方面涉略不深，一開始就是到處碰壁，有如無頭蒼蠅一般不知道要從哪開始撰寫，我要謝謝在暑假時認識的一位學長，手把手的告訴我要怎麼寫這些程式碼，再加上皓云去網路上爬蟲下來的很多大量歌曲，我再加 youtubeAPI 讓歌曲連結回傳到系統，才讓這個功能變得更加完整。

經過這一年下來的過程，從一開始的擬定主題到現在的系統結果，我一直都覺得非常有挑戰性。謝謝在這一年幫助我的學長姊、楊進雄老師、林俊杰老師，耐心的教導我們將這些知識應用在系統上。讓我在面對不熟悉的技術時，不會這麼的手足無措，逐步的帶領我們完成。

最後也感謝組員們的努力讓這個系統從無到有的建立起來，雖然沒有這麼完美，但是我深深體會到團隊合作、工作安排、溝通的重要性。

專題對我們來說是二技這兩年來很重要的一個學習機會，一切從無到有，從題目訂定一直到開始執行，經過一次又一次的討論，開始著手，在過程中或許有意見不合的時候，這時候也是考驗我們團隊的合作之一，經過一些磨合後，將不同的意見做整合，能夠把正確且合適的方案應用在專題上，讓此專題有更好的成績展現出來，也能在發表時更團結；我覺得我們的組長非常的優秀，因為每個人所擅長的東西不同，她依據個人能力做出適當的分工及工作分配，使得整個專題能夠順利下去。在這期間，我們學到更多不同的工具使用，指導老師也額外撥出暑假的時間一步一步的帶我們去學習新的工具，讓這些工具能夠應用在我們所需要的地方，一直到現在即將做最後一次的專題發表會，心中得一塊大石也即將落下，希望在最後這一次的專題發表會上能夠有不錯的成果，不錯的名次。

很開心能完成本次的專題，首先我要先謝謝我的組員，在我遇到困難時教導我，給予我學習的資源，讓我在開發與執行專案上學到許多技術上的知識，但一個專案的成功，只有技術是不足夠的，良好的團隊溝通與協同合作的氛圍，更是成功的關鍵，我們從一開始的題目制定、顏色的選擇、框線的設計上都花了許多時間討論協調，在開發的過程中也是遇到問題就先回報，然後尋找如何解決，再一起面對下一個挑戰。

我負責前端開發時，一直很擔心自己會捅出簍子拖累大家，很謝謝和我一起開發的暄毓的細心教導與支持，讓第一次接觸前端開發的我能很快的就步上正軌，也接觸到很多諸如 Tailwind 等前端開發的資源，這些經驗都會成為未來我參與其他專案時的珍貴資產。

最後要感謝楊進雄老師和實驗室學長們，您們就像夜晚矗立於海岸邊的燈塔，在我們陷入迷惘不知如何是好時，替我們點亮前行的明燈，儘管路途波濤洶湧，但最後還是走到了終點。

一轉眼專題就要結束了，在過去一年的專題路上我們得到了很多，有成長也有挫折。

製作專題中很多東西是過去沒有學過的，一開始只能像瞎子摸象，一點一點地去摸索和測試。儘管我們認為做得不錯，卻在一評時慘遇滑鐵盧。唉聲嘆氣並不能解決問題，於是我們再次來過。在老師和學長們的指點下，我們的專題慢慢地步上正軌，在參與其中的同時我也發現了自己很多不足之處，不論是在寫程式的速度或是分配讀書與進行專題的時間上，都有很大的不足。這對將來在職場上或許會有很大的影響，希望在這次專題結束後我在此有所成長。

最後想跟每位組員說辛苦了，有你們的幫忙才能完成這次的專題，希望我們一起努力的過程，能成為未來的寶貴經驗。

因為申請了六年一貫學程，所以這次二技的專題緊接著我五專的結束後馬上就開始了。雖然在課業與升學的壓力下繼續做專題十分有挑戰，但一年過去後，也漸漸要結束了。回想起來一路上的經驗，透過這次專題的機會，學習到了以前沒學好的部分，讓我更精進了程式的撰寫能力。很感謝組長和組員們的包容與支持，讓我在專題、課業、升學下都能維持。也很謝謝老師的指導與從旁協助，在我們需要幫助時，都能提供很好的協助，讓我們在做專題時遇到了瓶頸也不會卡關太久。做專題是一件很辛苦的事情，但也確實能從中學習到許多。這一年來雖然因為專題，讓生活上產生了不少壓力，但到了最後要結束時，我卻是充滿了感激。

第十四章 參考資料

1. 自然語言相似度計算

<https://cloud.tencent.com/developer/article/1145941>

2. Python 網絡爬蟲

<https://www.youtube.com/watch?v=9Z9xKWfNo7k&t=360s>

3. Stackoverflow

<https://stackoverflow.com/>

4. Next.js Tutorial

<https://www.youtube.com/watch?v=A63UxsQsEbU&list=PL4cUxeGkcC9g9gP2onazU5-2M-AzA8eBw>

5. Build a Django REST API with the Django Rest Framework.

<https://www.youtube.com/watch?v=c708Nf0cHrs&t=8881s>

6. SEAN 工作版

<https://seanacnet.com/website-related/cors-error/>

7. Tailwind css

<https://tailwindcss.com/>

附錄-一次審查評審意見之修正情形

評審建議事項	修正情形
問題與驅動力沒有說明清楚	經過與老師和組員的討論過後，我們決定以創造現有社群軟體附加價值的形式呈現，即既不是串流平臺亦不是社群軟體，使用者可以透過其在其他平臺撰寫的文章，作為歌曲推薦的依據。
使用者為什麼要在這個平台互動	已針對個人化的歌曲推薦這點進行強調
使用者黏著的驅動力為何	取得個人化的歌曲推薦，且判斷的文章可直接以使用者在其他社群平臺發表的文章為依據
使用爬蟲連結使用者常用的社交網站是否更實際	已進行修正，以爬蟲爬取 Dcard 和 PTT 文章的方式進行歌曲推薦
使用者發送心情應該會在 Facebook、IG、推特等社交網站上，專題規劃的網站有何特點與它們競爭	本組討論過後，決定採用與其他社交網站共存的方式進行，透過文章爬蟲，以提升附加價值的形式呈現
標記資料如何取得	組員去魔鏡歌詞網爬取歌詞，並由人工的方式標注每首歌的情緒
針對系統功能需詳細了解，特別是在訓練模型部份	這個部分我們是使用別人已經訓練好的套件 sklearn 的 TfidfVectorizer
簡報內可多介紹語意分析機制以及資料採集方式，著重說明本專題的技術亮點	已於二評簡報中增加此部分的說明
社群互動部分與現有社群軟體功能多有重疊，建議可與現有社交平台結合做為擴充功能整合使用	已針對此點進行修正，以爬蟲的方式爬取使用者於社群軟體發表的文章，進行歌曲的推薦
無法清楚陳述「文本情感分析」的開發脈絡及做法	做法為使用組員爬下來的歌詞，先清理資料之後，進行情緒標注。使用者輸入的文章，我們運用 selenium 和 beautifulsoup 套件進行爬蟲，再利用 TfidfVectorizer，讓使用者的文章和每首歌曲的歌詞進行比對，匹配度越高的歌曲就會推薦給使用者 故綜上所述，脈絡為：歌詞爬蟲 -> 標註歌詞情緒 -> 使用者文章鏈結爬蟲 -> 用 TfidfVectorizer 套件把歌詞和文章內容進行比對 -> 最後推薦歌曲給使用者