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 陳姵蓉 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 6](#_Toc103082156)

[1-1 簡介 6](#_Toc103082157)

[1-2 問題與機會 7](#_Toc103082158)

[1-3 相關系統探討 8](#_Toc103082159)

[第二章 系統目標與預期成果 10](#_Toc103082160)

[2-1 系統目標 10](#_Toc103082161)

[2-2 預期成果 11](#_Toc103082162)

[第三章 系統規格 12](#_Toc103082163)

[3-1 系統架構 12](#_Toc103082164)

[3-2 系統軟、硬體需求與技術平台 13](#_Toc103082165)

[3-3 使用標準與工具 14](#_Toc103082166)

[第四章 專案時程與組織分工 15](#_Toc103082167)

[4-1 專案時程 15](#_Toc103082168)

[4-2 專案組織分工 16](#_Toc103082169)

[第五章 需求模型 17](#_Toc103082170)

[5-1 使用者需求 17](#_Toc103082171)

[5-2 使用個案圖（Use case diagram） 20](#_Toc103082172)

[5-3 使用個案描述 21](#_Toc103082173)

[5-4 分析類別圖（Analysis class diagram） 24](#_Toc103082174)

[第六章 設計模型 25](#_Toc103082175)

[6-1 循序圖（Sequence diagram） 25](#_Toc103082176)

[6-2 設計類別圖（Design class diagram） 29](#_Toc103082177)

[第七章 實作模型 25](#_Toc103082175)

[7-1 部署圖（Deployment diagram） 25](#_Toc103082176)

[7-2 套件圖（Package diagram） 25](#_Toc103082176)

[7-3 元件圖（Component diagram） 25](#_Toc103082176)

[7-4 狀態機（State machine） 25](#_Toc103082176)

[第八章 資料庫設計 25](#_Toc103082175)

[8-1 資料庫關聯表 25](#_Toc103082176)

[8-2 表格及其Meta data 25](#_Toc103082176)

[第九章 程式 25](#_Toc103082175)

[9-1 元件清單及其規格描述 25](#_Toc103082176)

[9-2 其他附屬之各種元件 25](#_Toc103082176)

[第十章 測試模型 25](#_Toc103082175)

[10-1 測試計劃 25](#_Toc103082176)

[10-2 測試個案與測試結果資料 25](#_Toc103082176)

[第十一章 操作手冊 25](#_Toc103082175)

[第十二章 使用手冊 25](#_Toc103082175)

[第十三章 感想 25](#_Toc103082175)

[第十四章 參考資料 25](#_Toc103082175)

[附錄 25](#_Toc103082175)

圖目錄

[圖 1-3‑1 SWOT分析 9](#_Toc103087429)

[圖 3‑1‑1 系統架構圖 12](#_Toc103087430)

[圖 4‑1‑1 專案時程圖 15](#_Toc103087431)

[圖 5-2‑1 使用個案圖 20](#_Toc103087432)

[圖 5-3‑1 使用者註冊會員活動圖 21](#_Toc103087433)

[圖 5-3‑2 使用者輸入連結活動圖 22](#_Toc103087434)

[圖 5-3‑3 使用者送出文章活動圖 22](#_Toc103087435)

[圖 5-3‑4 使用者查閱歷史資料活動圖 23](#_Toc103087436)

[圖 5-3‑5 使用者修改資料活動圖 23](#_Toc103087437)

[圖 5-4‑1 分析類別圖 24](#_Toc103087438)

[圖 6-1‑1 登入循序圖 25](#_Toc103087439)

[圖 6-1‑2 登出循序圖 25](#_Toc103087440)

[圖 6-1‑3 註冊循序圖 26](#_Toc103087441)

[圖 6-1‑4 輸入連結循序圖 26](#_Toc103087442)

[圖 6-1‑5 送出文章循序圖 27](#_Toc103087443)

[圖 6-1‑6 修改資料循序圖 27](#_Toc103087444)

[圖 6-1‑7 查詢歷史紀錄循序圖 28](#_Toc103087445)

[圖 6-2‑1 設計類別圖 29](#_Toc103087446)

[圖 7-1‑1 部署圖 29](#_Toc103087446)

[圖 7-2‑1 套件圖 29](#_Toc103087446)

[圖 7-3‑1 元件圖 29](#_Toc103087446)

[圖 7-4‑1 狀態機 29](#_Toc103087446)

[圖 8-1‑1 資料庫關聯表 29](#_Toc103087446)

表目錄

[表 1-3‑1 相關系統比較 8](#_Toc103087375)

[表 3‑2‑1 測試環境與軟硬體需求表 13](#_Toc103087376)

[表 3‑3‑1 使用標準與工具表 14](#_Toc103087377)

[表 4‑2‑1 分工表 16](#_Toc103087378)

[表 5-1‑1 功能需求表 17](#_Toc103087379)

[表 5-1‑2 非功能需求表 19](#_Toc103087380)

[表 8-2‑1 資料表描述：帳號 19](#_Toc103087380)

[表 8-2‑2 資料表描述：文章 19](#_Toc103087380)

[表 9-1‑1 元件清單及其規格描述 19](#_Toc103087380)

[表 10-1‑1 測試流程 19](#_Toc103087380)

[表 10-2‑1 測試結果 19](#_Toc103087380)

[表 11-1‑1 操作手冊表 19](#_Toc103087380)

[表 12-1‑1 使用手冊表 19](#_Toc103087380)

# 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。

## 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。

## 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：

1. 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、恨和欲七種，以此推薦相應情緒分類的歌曲。

1. 豐富使用者體驗
2. 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。
3. 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種7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

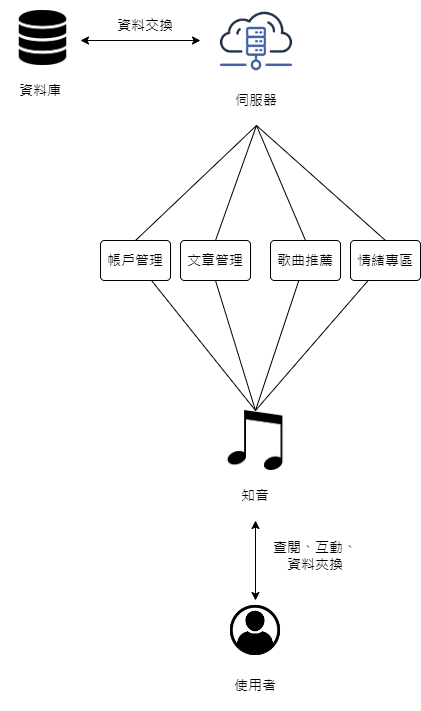


圖 3‑1‑1 系統架構圖

## 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、無限網路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、無限網路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, Google Colaboratory |
| 資料庫 | 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

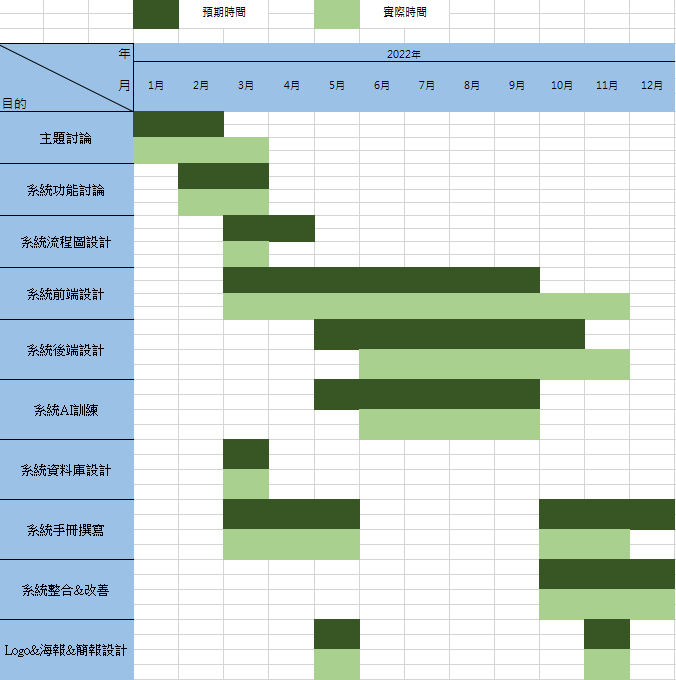


圖 4‑1‑1 專案時程圖

## 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 | 陳姵蓉 | 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

## 使用者需求

表 5-1‑1 功能需求表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 事件 | 觸發器 | 來源 | 活動/使用案例 | 回應 | 目的地 |
| 註冊系統 | 輸入使用者帳密 | 使用者 | 註冊系統 | 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 |
| 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 |
| 送出文章 | 使用者送出文章 | 使用者 | 送出文章 | 送出成功  或  送出失敗 | 使用者 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 事件 | 觸發器 | 來源 | 活動/使用案例 | 回應 | 目的地 |
| 修改資料 | 使用者點選修改帳號、密碼 | 使用者 | 修改資料 | 修改成功  或  修改失敗 | 使用者 |
| 查閱歷史資料 | 使用者點選信件紀錄 | 使用者 | 查閱歷史資料 | 查詢成功  或  查詢失敗 | 使用者 |

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，且登入時需要輸入驗證碼。 |
| 兼容性 | 支持多種作業系統。 |

## 使用個案圖（Use case diagram）

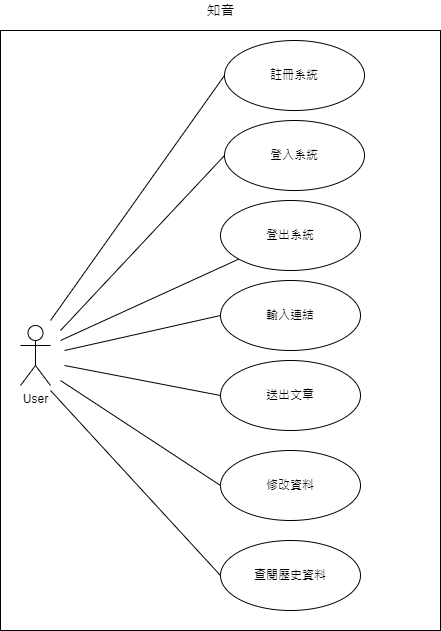


圖 5-2‑1 使用個案圖

## 使用個案描述

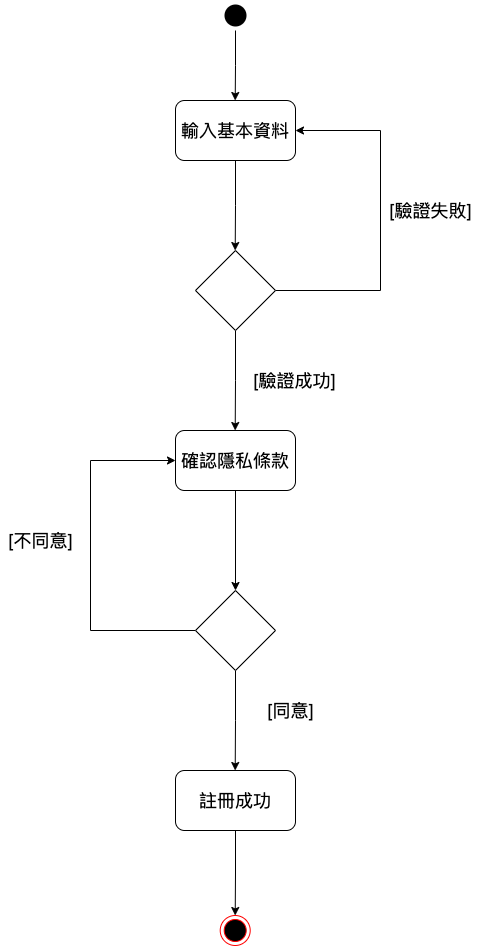


圖 5-3‑1 使用者註冊會員活動圖



圖 5-3‑2 使用者輸入連結活動圖

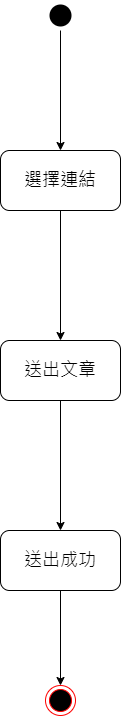


圖 5-3‑3 使用者送出文章活動圖

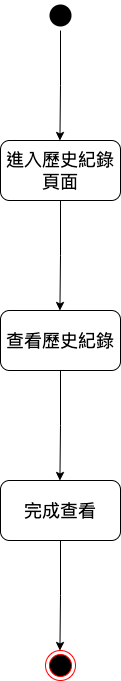


圖 5-3‑4 使用者查閱歷史資料活動圖

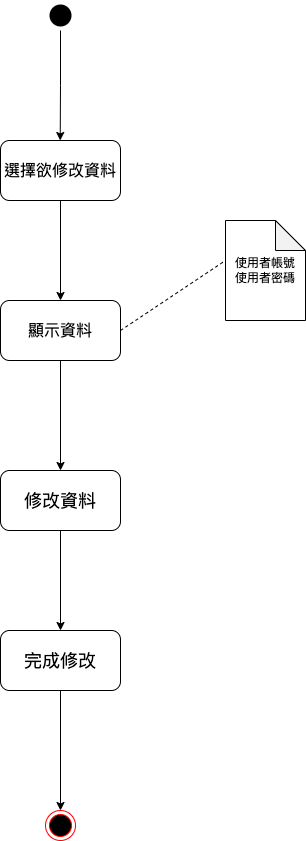


圖 5-3‑5 使用者修改資料活動圖

## 分析類別圖（Analysis class diagram）

一張含有 桌 的圖片

自動產生的描述

圖 5-4‑1 分析類別圖

# 第六章 設計模型

## 循序圖（Sequence diagram）

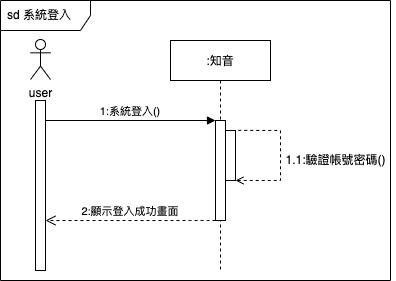


圖 6-1‑1 登入循序圖



圖 6-1‑2 登出循序圖

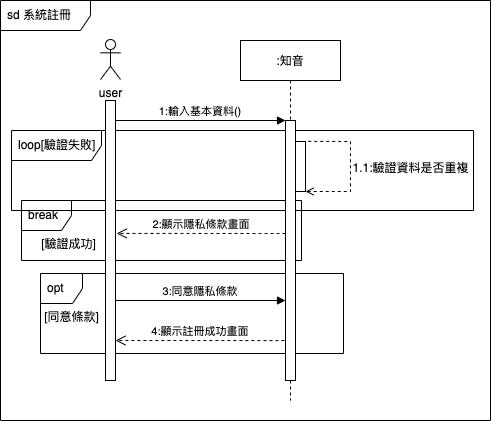


圖 6-1‑3 註冊循序圖

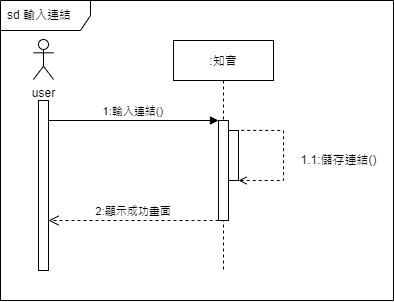


圖 6-1‑4 輸入連結

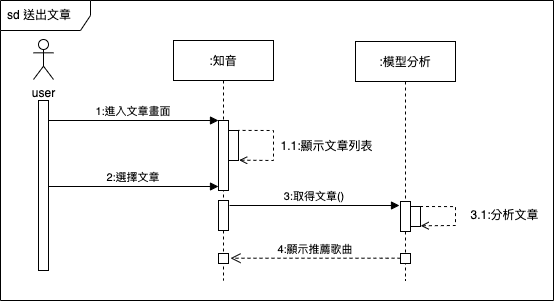


圖 6-1‑5 送出文章循序圖

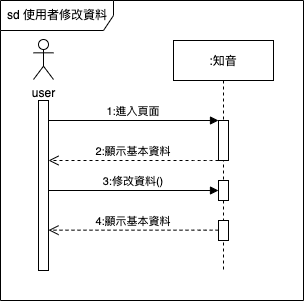


圖 6-1‑6 修改資料循序圖

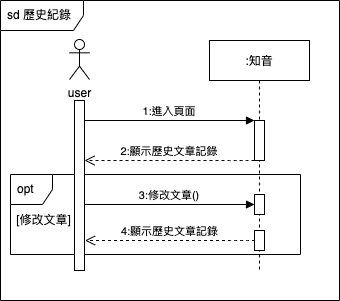


圖 6-1‑7 查詢歷史紀錄循序圖

## 設計類別圖（Design class diagram）

一張含有 桌 的圖片

自動產生的描述

圖 6-2‑1 設計類別圖

# 第七章 實作模型

## 部署圖（Deployment diagram）

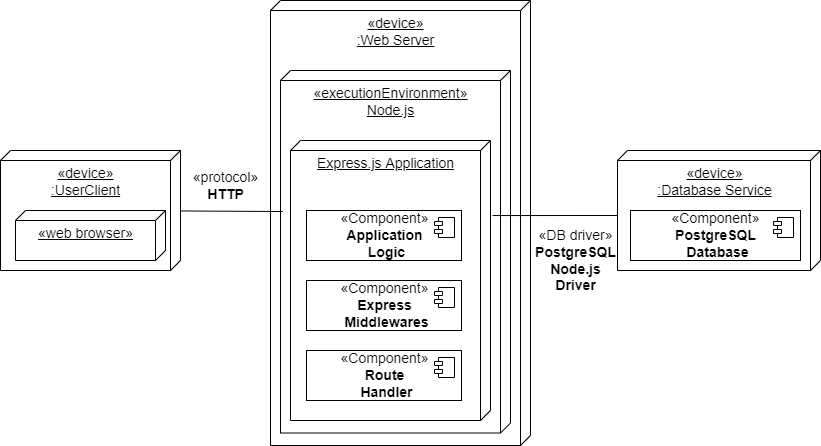


圖 7-1‑1 部署圖

## 套件圖（Package diagram）

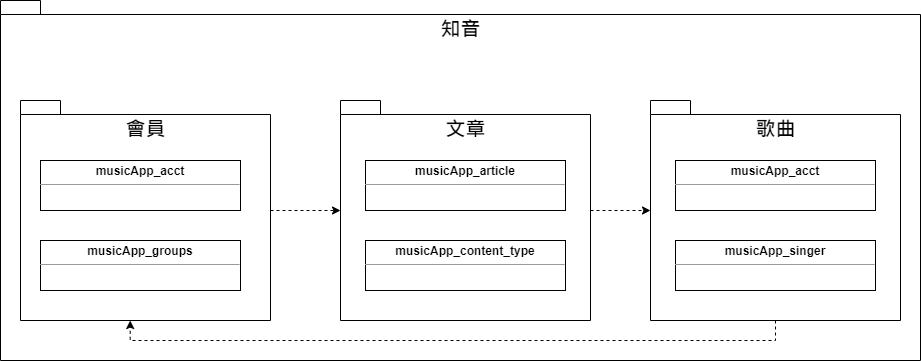


圖 7-2‑1 套件圖

## 元件圖（Component diagram）

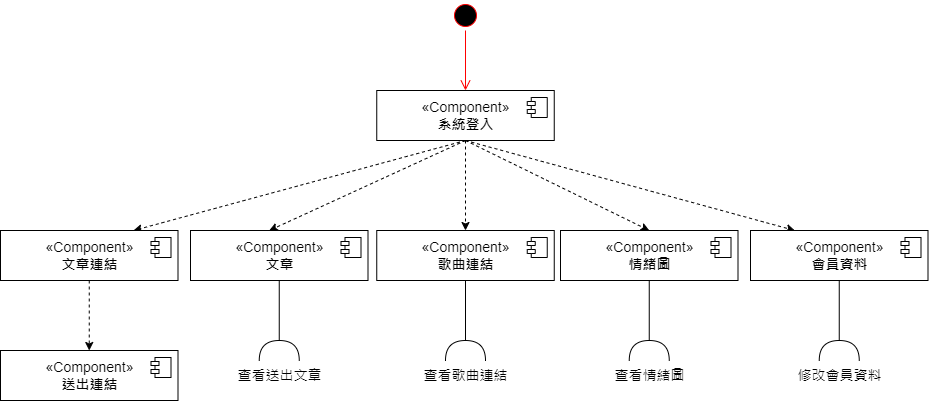


圖 7-3‑1 元件圖

## 狀態機（State machine）

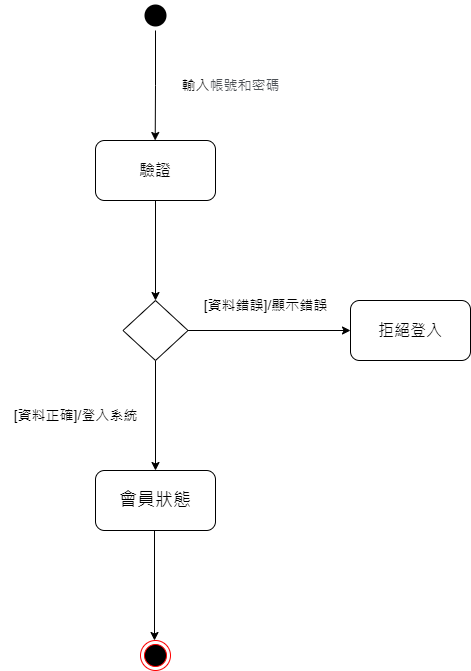


圖 7-4‑1 狀態機

# 第八章 資料庫設計

## 資料庫關聯表

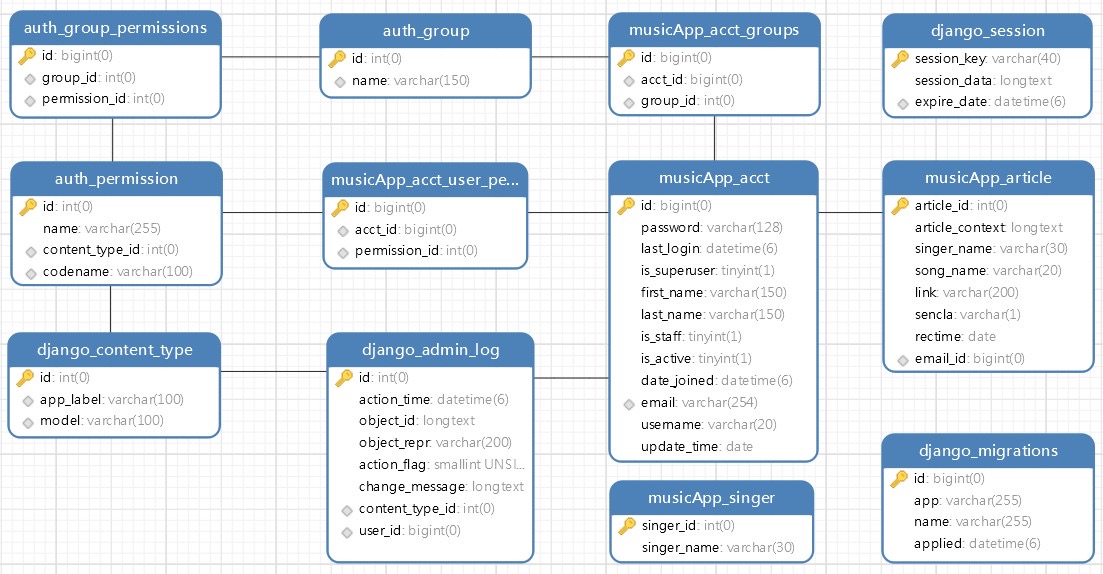


圖 8-1‑1 資料庫關聯表

## 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 | 文章内容 | 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 |  |
| email\_id | 電郵編號 | int | - | 否 | FK |

# 第九章 程式

## 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 | 功能説明 |
| 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 | 功能説明 |
| 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 | 功能説明 |
| 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 | 功能説明 |
| 4-1 | settings.py | 系統各項設定 |
| 4-2 | urls.py | 設定網址的連結字串 |

## 其他附屬之各種元件

|  |  |
| --- | --- |
| 程式名稱 | song\_compar.py |
| 部分程式碼 | |
| 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  # 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.chrome\_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 = "AIzaSyBKdJO0Q7tS8jQyuZUx0kNmgFD2L73Bn1E"  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()    @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    @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\_UNAUTHORIED)  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 | |

# 第十章 測試模型

## 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 |

## 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 |
|  |
| 登入：輸入輸入帳號、密碼 |
|  |

|  |
| --- |
| 登入後進入個人天地 |
|  |
| 點選寫信 |
|  |

|  |
| --- |
| 貼上文章連結並送出 |
|  |
| 取得推薦的歌曲YouTube連結 |
|  |

|  |
| --- |
| 點選連結進入YouTube享受知音爲您推薦的歌曲 |
|  |
| 進入信件記錄查詢過去知音爲您推薦的歌曲 |
|  |

|  |
| --- |
| 進入我的情緒圖查詢情緒分佈狀態 |
|  |
| 進入修改密碼可更新舊的密碼 |
|  |

# 第十三章 感想

11036008 陳姵蓉

11036015 陳韋辰

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。

11036016 白皓云

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、顔色的選擇、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，這些經驗都會成爲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。

11036037 林詩蓉

11136006 廖暄毓

# 第十四章 參考資料