資料庫系統期末專題

跨平台音樂播放系統 (CPMP)

108703015　資科四　吳武峰
109703015　資科三　張予蓉
109703018　資科三　周家儀
109703021　資科三　林佩萱
109703040　資科三　洪晙宸

1、 每位成員負責之任務分工、貢獻百分比

吳武峰:後端(20%)
張予蓉:資料庫schema建置、create、check功能(20%)
周家儀:資料庫get、update功能、spotify api調用整理(20%)
林佩萱:美術設計、UI/UX設計、discord API 串接(20%)
洪睦宸:前端(20%)

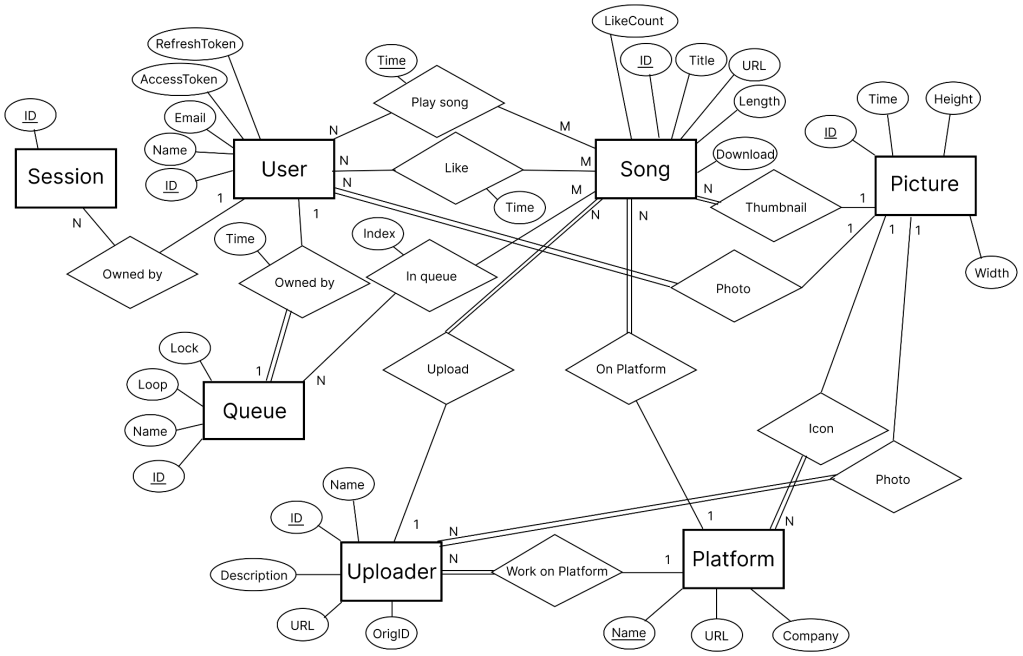
2、 需求分析

1、 歌曲播放紀錄:使用者ID、歌曲ID、播放時間
2、 收藏/我的最愛:使用者ID、歌曲ID、收藏時間
3、 圖片:圖片ID、下載時間戳、圖片寬、圖片高
4、 音樂平台來源:平台名稱、平台網址、平台所屬公司、平台Icon
5、 歌曲在 Queue 中的順序:QueueID、序號、歌曲ID、歌曲進入時間
6、 Queue:QueueID、Queue 名稱、循環狀態、上鎖狀態(決定該 Queue 是否能被進行操作)
7、 使用者對 Queue 的身分:QueueID、使用者ID、身分
8、 Session:SessionID、使用者ID
9、 歌曲:歌曲ID、歌曲來源網址、歌曲來源平台、歌曲名稱、歌曲上傳者ID、歌曲縮圖、歌曲按讚次數、歌曲長度、是否被下載過
10、上傳者:上傳者ID、上傳者頁面網址、上傳者原ID(在原平台的)、上傳者名稱、上傳者所屬平台、上傳者簡介(源自上傳者所屬名台)、上傳者頭像
11、使用者:使用者ID、QueueID、SessionID、信箱、圖片ID

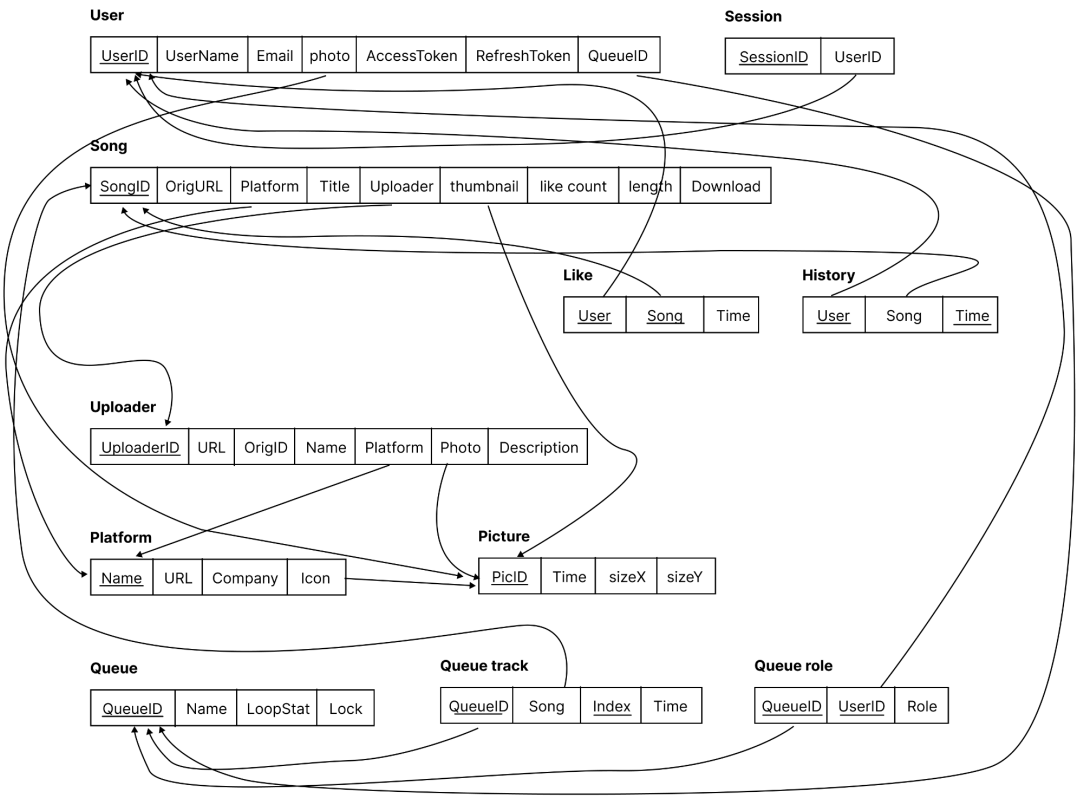
3、 系統功能

1. 公開服務:任何人都可以自由使用，只要擁有 Discord 帳號就能馬上開始使用。
2. 多平台支援:電腦及手機皆可使用瀏覽器瀏覽我們的網頁。
3. 登入:使用 Discord 帳號登入。
4. 更改頭像:在 Discord 帳戶更改頭像，CPMP的頭像也會被更改。
5. 切換背景主題:使用者可自由切換深色/淺色主題。
6. 搜尋:使用者可以搜尋想聽的歌曲，我們會依相關程度列出搜尋結果。
7. 將歌曲加入佇列:使用者可以將指定歌曲加入佇列。
8. 將歌曲自佇列刪除:使用者可以將指定歌曲自佇列中刪除。
9. 加入最愛:使用者可將歌曲加入最愛列表。
10. 按讚列表:按讚列表會顯示使用者按讚的所有歌曲。
11. 播歌:系統會顯示正在播放的歌曲，使用者可以暫停、快轉、切換循環、切歌、打亂順序。
12. 循環:使用者可以設定單曲循環，或讓列表裡的歌曲循環播放。
13. 調整音量大小:使用者可以根據需求調整歌曲播放音量，音量調整是非線性的，較符合人類生理構造的可承受程度。

4、 ER Model



5、 Relational Schema



6、 系統架構

前端

框架：vuejs
首頁提供使用者播放紀錄及系統播放前十名，可以依時間進行篩選。
整體頁面與 spotify 類似，下方有大的播放控制器，配備播放器的所有功能。
使用者可以透過搜尋方框利用不同平台搜尋歌曲，並加至佇列。

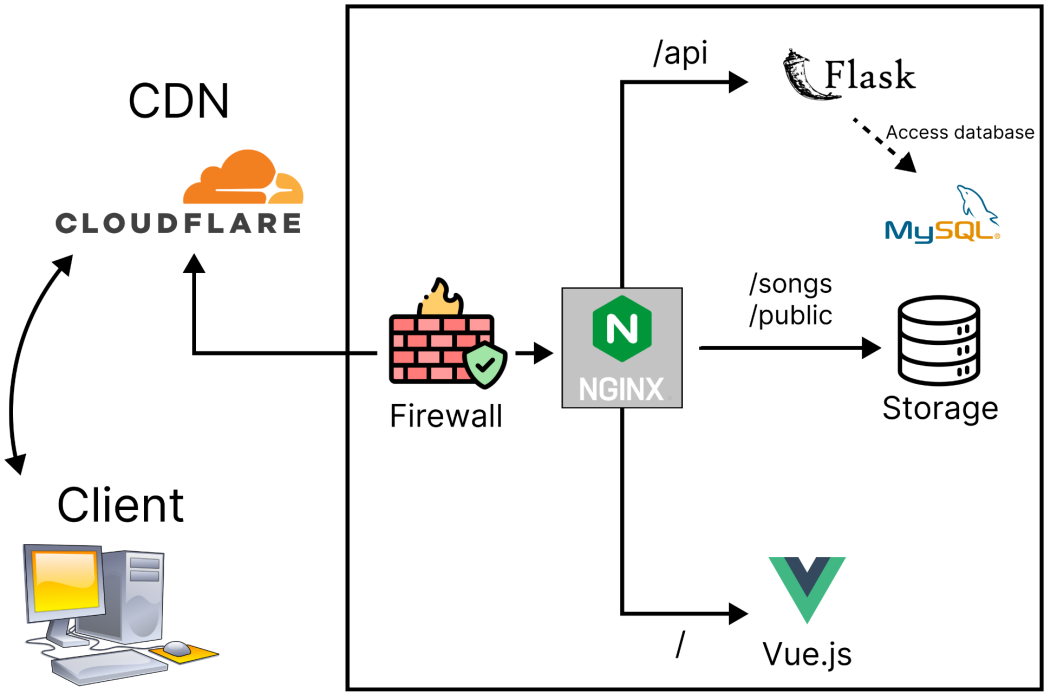
後端

框架：flask
使用 python 語言與 flask 套件實作 RESTful API
使用 Discord API 辨識使用者身分，並取得使用者基本資料(email、暱稱等)
與 SQL 溝通，檢查並防止 SQL Injection。

資料庫

系統：mysql
使用 docker compose，簡化日後系統搬遷的手續。
設置對外防火牆，僅允許特定 IP 段存取資料庫，進一步確保安全性。

概念圖



7、 心得、收穫與建議

從一開始的討論，我們的目標就很明確——要實作出一個能真正派上用場的服務，並規劃在未來公開上線。在經過多次的討論之後，我們才敲定了這個主題。

一開始的發想是源自於 Discord (下稱DC) 這個聊天軟體的播歌機器人。DC 是近年來在玩家社群十分火紅的聊天軟體，其開放的開發者社群與強大的功能很快的成為全球最熱門的實時通訊軟體。該平台允許使用者使用其公開的 API 製作機器人，進而幫助豐富使用者體驗，其中最常見的功能之一，就是提供成員點播串流媒體的音樂。然而我們發現，傳統的 DC 機器人需要使用者在聊天頻道向機器人下達指令，這對非資訊專業的使用者有一定的上手難度，尤其是進行插歌、刪歌等較為複雜的操作，若沒有 array 與 index 的概念，要理解是有一定難度的。於是我們決定實作一個網頁版的歌單編輯器，並在未來串接自行開發的 DC bot，讓一般人也可以直觀的操作點歌系統。

在開發的過程，最困難的莫過於與組員之間的溝通。本來以為正常的開發流程可以把不同 component 發包給不同組員實作，後來發現其實合作開發專案是需要高頻率的溝通的，一週固定一次的週會遠遠不足。而每個人回訊息的時間也不盡相同，往往一個待調整的小設定，溝通的等待時間遠大於開發的時長。身為組長，我在這次專案學到一件事——component之間的分工要再更明確一些，降低溝通成本才能提升開發效率。

本專案的未來預計會實作預存歌單、串接 DC bot、更詳細的權限管理功能、透過 websocket 實時更新 queue、多人共聽、推薦系統.....等等更多功能，以及處理公開服務之後 scale up 需要面對的挑戰。感謝我的組員們，幫我完成這個夢想的雛形，希望在不久的將來，也能透過此專案為 DC 社群做出貢獻。