音樂辨識與段落搜尋
專題成果報告書

指導教授:羅峻旗
資工四A: 陳永義
資工四A: 魏宇龍
資工四A: 林儒靖
資工四A: 胡家瑞

目錄

零、　前言

在資訊爆炸的時代，雖然有很多不同的資料可以參考，但爆量的資訊常常使人們眼花撩亂，不知從何找出自己想要的訊息。進而有了各種搜尋法，透過輸入關鍵字，跑出想要的結果。是否有過一種經驗，腦中有段耳熟能詳的旋律，卻因不知道歌曲資訊而找不到，唯一的線索就是一段旋律。為此，我們希望透過聲音來當作搜尋媒介。

本篇專題的目標之一是我們將利用自己的歌聲來當作查找歌曲的另一種方式。目標二是要去實驗是否能在不利用 AI 技術的前提下做出聲音搜尋的功能。在 AI 技術盛行的時代，為了要與市場產生差異和提高開發者易意願[參閱 3.3 節]，因此我們在技術上捨棄了 AI 技術，並利用 Sphinx-4 套件和我們的演算法做結合來同時滿足我們的兩個目標。

一、　簡介

1.1 系統名稱

音樂辨識與段落搜尋(Music Identifier and Paragraph Search)。

1.2 系統特色

解決使用者想要尋找及聆聽歌曲的需求，以及幫助使用者能進行音樂歌曲的段落搜尋。

1.3 畫面展示

圖 1.應用程式主畫面

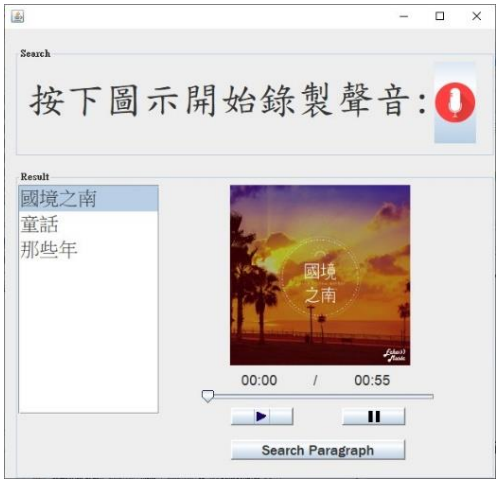


圖 2.搜尋結果及撥放圖

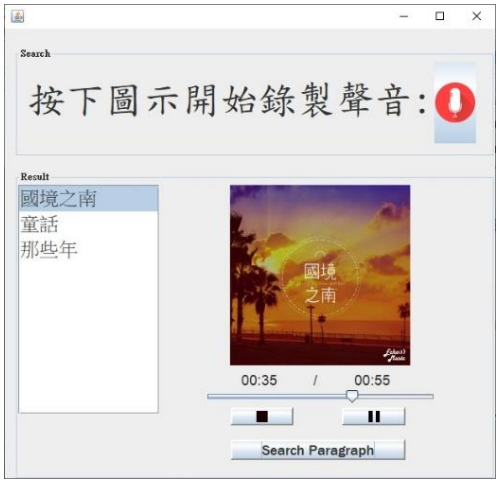


圖 3.播放中圖示

1.4 使用說明及畫面功能標示

為了要讓使用者能快速了解如何使用此系統，以下使用說明會搭配 1.3 的圖片去做使用介紹:

圖 4.當系統開啟時

圖 5.顯示搜尋結果

當系統第一次開啟時會進入主畫面(圖 4)。當按下搜尋按鈕時，頻譜圖繪偵測聲音輸入的狀況，使用者可藉由觀察頻譜的變化來判斷系統是否有接收到使用者的聲音。當按下第二遍搜尋按鈕就會進入比對，並將搜尋結果以圖 5 顯示給使用者看。

在圖 5 的部分，系統會將比對相似度在 45%以上的曲目顯示在左邊的搜尋結果，如果在播放功能想確認自己唱出的段落是否存在或者想在歌曲裡面搜尋不同段落，可點選 Search Paragraph，歌曲會中斷並讓使用者輸入。輸入完畢再次點選按鈕，系統會直接跳到該段落。當使用者想重複搜尋時，只須點下搜尋按鈕就可以再次輸入。

二、 系統整體描述

2.1 使用對象

提供給想要找到歌曲但不知道相關資訊或想聆聽歌曲的使用者。

2.2 系統功能

讓使用者能搜尋到有印象或只知道旋律的歌曲，並提供撥放以及段落搜尋功能。

2.3 使用環境

此系統以視窗程式方式呈現。

三、 本專題與現有技術比較

3.1 與”智慧助理”差異

不管是在任何聲音處理上，現今的智慧助理雖然已有先進的聲音辨認功能了。但是在播放的功能上，依然是沿用快轉，播放，暫停這類舊有的播放功能。因此我們在系統上改變了舊的音樂播放器。加入了段落搜尋的功能，不必再使用快轉到幾分幾秒這種精確的數字來控制智慧助理的播放功能。且與 AI 方式有所不同，我們不必去搜集大量模型去做改良，只要客制化使用者的資料庫內容就可以達到相同的效果。

3.2 與”伴唱系統”差異

以往的伴唱系統是必須搭配厚厚的一本歌譜，須操作複雜的遙控器設備才能進行使用。且如果在做使用前，還得去學習如何操作遙控器。但對老年人來說其實太過繁瑣了。假設今天使用者是一位上了年紀的長輩，在透過這麼複雜的操作後就會放棄使用系統了。且這是目前普及的伴唱系統都有存在這些缺點。

因此我們基於這個舊有的缺點，我們將目前的功能改進成透過唱段落的方式去搜尋。這在搜尋上大幅度的去提升使用者體驗，降低使用複雜度。

3.3 與”AI 智慧辨識” 差異

AI 智慧辨識雖然很強悍，透過技術搭配可以有效去做使用，但 AI 技術需要大量的模型去進行深度學習，整體在製作成本上已經偏高了，且演算法複雜度高還需要花大量時間去訓練模型。另外如果是 AI 技術的使用前提，硬體也需要去進行強化，所以也導致硬體需求高，以目前的來說就會很強調硬體需求。但並不是每個公司都負擔的起這個開發成本。本系統則只需依靠聲音辨識演算法進行聲音的辨認所以硬體需求低，一般民眾或是小公司較能負擔的起開發成本。

此系統可以即時去比對資料，並且在當下就可以給使用者作即時性的反饋，不必經過像 AI 技術的繁瑣要求。且開發的任何要求都相對都比 AI 低，實作上更不用去花大量的金額去提升硬體品質。

四、 本專題的創意點

4.1 開發方面

我們為了要實驗不使用人工智慧等套件也能做到音樂搜尋。因此我們在開發期間捨棄了 python 和人工智慧的套件，也因為我們不使用人工智慧等套件，我們在開發的花費成本不需要太高，只需要一台開發用的電腦就可以做到系統開發。在演算法複雜度上，我們的系統演算法也會比透過人工智慧套件開發的系統演算複雜度還要來的低。我們基於上述的優點去製作我們的系統，加上不需要預先做訓練，在開發硬體的需求和開發成本上會遠低於其他利用人工智慧套件的系統，同時在比對方面，也可以比人工智慧的系統來的更具有即時性。

4.2 功能性

市場目前最有名的音樂搜尋器功能是透過利用音樂原聲帶或的原音才能進行搜尋，但這也就必須搜尋時都是要在有原音的地方，舉例像是電台或著唱片行會播放原聲帶或著原音的地方才能使用，並不具備使用者聲音輸入去做比對。
此系統除了能利用播放段落原聲帶得到目標之外，我們也具備透過使用者個人的聲音輸入去找到使用者想要的目標。此外我們也透過介面上的列表來顯示搜尋後的結果，將超過一定比例相似度的歌曲列出，提高使用者找到目標歌曲的機率。在雜訊抑制上，我們也有別於市面上常用的音樂搜尋軟體，能在有環境音的干擾下做搜尋，且搜尋成功率也不會受到太多影響。

4.3 額外功能

除了原有的目標歌曲搜尋外，我們也能針對一首歌，單獨去做段落搜尋。大致介紹就是能在播放時搜尋歌曲的段落，並將進度條跳轉，可直接利用介面上的按鈕去做搜尋，並且和搜尋歌曲使用方式一樣，也是利用使用者輸入去做使用。目前大眾化手機軟體是極少有此項功能的。且目前測試階段，也能成功跳轉至目標段落。

4.4 統整

統整第二和第三點，此系統的比對時間都是即時性的，這讓我們省去很多找資料集訓練的時間，且在目前市場中智慧娛樂助理中也極少是沒透過人工智慧套件協助所達成，此系統不需使用人工智慧套件，也因如此我們同時也降低了開發所需的硬體成本。這個系統開發後，我們可以更加確認聲音搜尋這項技術是能以演算法和數學理論實現的。

五、 需求詳述說明

5.1 使用者劇情與介面需求

故事情境或要解決的問題描述：

某天小明陪著長輩一起使用伴唱軟體時，發現長輩時常需要翻閱歌譜單和加上遙控器去將編號輸入進系統。在小明觀察長輩的使用方式後發現， 當人工輸入錯誤時，就須要重新輸入一遍，且如果是年紀更長的長輩在操作時，時常需要看著歌單，配著遙控器或帶著其他輸入裝置的說明書，這會導致系統產生使用者體驗不佳，且操作複雜的介面或遙控器會導致有些長輩會因此排斥使用。

基於上述問題，小明希望如果有一個操作簡單，可以省略人工查找時間的軟體出現，這樣就可以取代掉這個操作複雜的傳統伴唱系統。每當看著長輩戴著老花眼鏡在查編號，還要一邊學習如何使用操作複雜得的裝置，難免讓他覺得既然是休閒娛樂，就應該減少長輩在操作上所遇到的各種問題，讓他們可以順利的去使用。

遭遇問題：

在目前所遭遇問題中可以發現兩點:

1. 過繁瑣的操作準備

具上述故事可以發現，目前的伴唱系統需要藉由人工搜尋和搭配遙控器或其他輸入裝置，且輸入裝置是需要去看著說明書去做操作。如果當今天是老年人在做操作，也許還要花時間去做教學。

2. 使用者體驗

繼第一點，在複雜的操作準備下，可能會降低人們在使用的操作意願，且在一個服務範圍非常廣的系統中，如果複雜的操作方式和沒有一個良好的引導，使用者會對於系統操作會直接歸零。

目前做法：

在遭遇問題中可以發現兩點其實是一個相輔相成的，因此我們會在操作準備去著手。透過標示明確的按鈕，和語音控制這兩個元素去製作，當操作方是簡化之後，使用者體驗就會有明顯的改善了。

圖解說明：

以下會有圖片說明

5.2 系統非功能需求

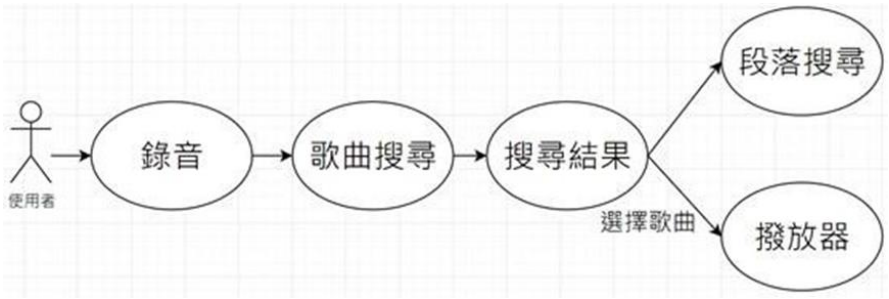
功能需求編號	非功能需求描述
MIPS-NF-001 減少使用者負荷	系統透過圖形化的介面，來幫助使用者能快速以及清楚的能操作本系統。不必透過長時間的記憶與學習
MIPS-NF-002 操作簡單	透過聲音控制和簡單的按鈕，讓使用者可以輕易上手。
MIPS-NF-003 系統可攜性	透過程式包裝，可以在不同的裝置進行使用。只要作業系統可相容，都可以使用此系統
MIPS-NF-004 系統穩定性	在操作簡單的狀況下，使用者可以減少在使用產生的任何錯誤，減少系統崩潰。

5.3 系統功能需求

功能需求編號	功能需求描述
MIPS-F-001 可視化介面	系統以視窗化介面呈現，會將所有功能及結果在視窗介面供使用者閱讀及操作
MIPS-F-002 基本操作介面	在視窗畫面中可以透過系統提供的按鈕搭配個人的聲音進行程式的基本操作。
MIPS-F-003 音樂搜尋	使用者輸入完歌曲旋律後，系統將其輸入的內容來比對資料庫的資料。
MIPS-F-004 歌曲段落搜尋	使用者輸入該歌曲的段落旋律後，系統將其歌曲進度條移至該段落旋律的進度。
MIPS-F-005 搜尋結果列表	在搜尋結果的畫面中，可以得到由高到低的搜尋結果。
MIPS-F-006 音樂播放	系統會提供除了快轉以外的服務，可在視窗進行音樂播放器的基本操作。

5.4 使用案例(Use case, UC)

使用案例之劇本(Scenario)描述下圖為主要使用流程:



使用案例名稱	搜尋目標音樂	
主要參與者	使用者	
利害關係人與目標	使用者:需要搜尋目標歌曲	
描述	使用者開啟此應用程式或需要重複搜尋，必須按下麥克風按鈕並輸入聲音搜尋。	
主要流程	參與者	系統
	1. 點選應用程式圖示，進入系統。 2. 按下中間麥克風的按鈕。 3. 使用者利用音源，樂器，或是人聲的方式輸入進麥克風供系統接收進行紀錄。 4. 輸入完畢，按下按鈕進行搜尋。	1. 系統會顯示主要畫面(使用者介面圖 1)。 2. 系統對使用者提出的要求進行反饋。 3. 系統顯示頻譜，偵測使用者的輸入狀態。 4. 收到按鈕指令，系統提示使用者搜尋狀態並給予反饋。
例外情節	無	
其他需求	無	

使用案例名稱	搜尋結果挑選	
主要參與者	使用者	
利害關係人與目標	使用者:挑選自己的目標歌曲	
描述	系統會將使用者的搜尋結果顯示在列表供使用者挑選。	
主要流程	參與者	系統
	1. 使用者在搜尋列表挑選自己想要的項目。 2. 使用者點擊項目，系統會將使用者選擇的項目加入到播放功能中。	1. 系統搜尋完必會將搜尋結果依照相似度顯示，當使用者按下選擇項目時，列表中的項目會反白項目(使用者介面圖 2)。 2. 當項目接收到點擊時，會將使用者點擊的項目匯入至播放器進行播放。
例外情節	無	
其他需求	無	

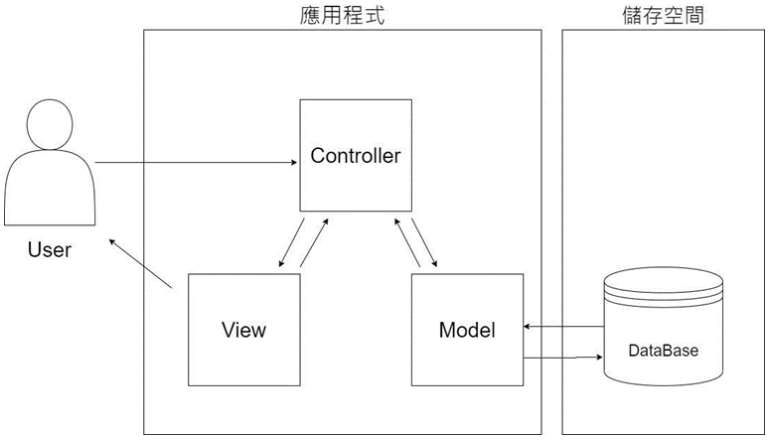
使用案例名稱	樂曲播放器使用	
主要參與者	使用者	
利害關係人與目標	使用者:進行樂曲播放或段落搜尋的基本操作	
描述	使用者開啟此應用程式或需要重複搜尋，必須按下麥克風按鈕並輸入聲音搜尋。	
主要流程	參與者	系統
	1. 使用者進入播放器的功能時， 可以做播放器的基本操作。 2. 當要使用段落搜尋，使用者會按下段落搜尋的按鈕。 3. 使用者看到系統可以開始錄音，可將一小段音樂段落以將音源，樂器，或是人聲的方式輸入進麥克風供系統紀錄。 4. 輸入完畢，按下錄音圖示。	1. 系統接收到相對應的動作例如播放，暫停，下一首，上一首(使用者介面圖 3)。 2. 收到按鈕指令，系統會將按鈕變成錄音模式。 3. 錄音按鈕會依照使用者輸入的狀態去做變化。 4. 收到按鈕指令，系統提示使用者搜尋狀態並給予反饋。
例外情節	無	
其他需求	無	

六、 架構說明

6.1 系統架構

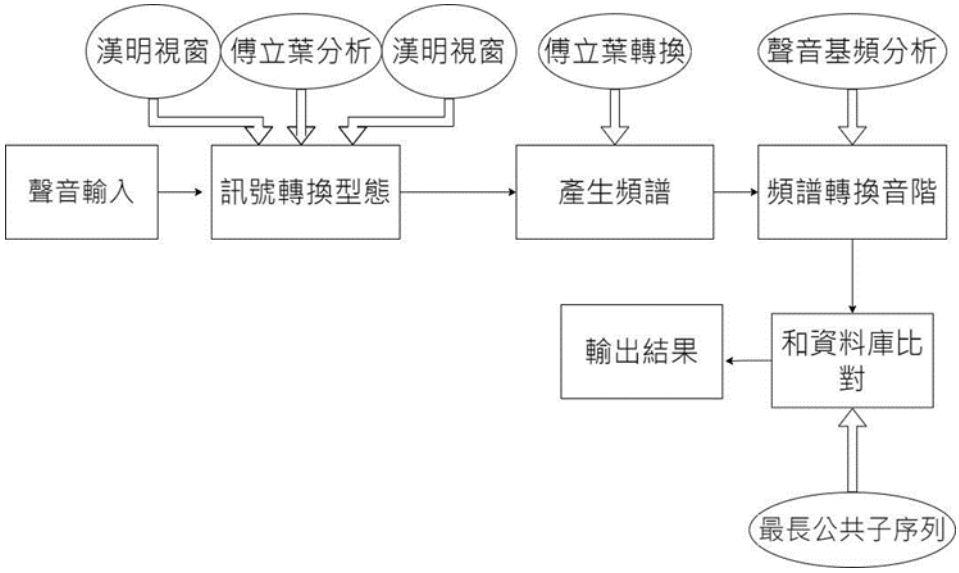
在架構上，藉由參考現在大部分軟體製作方式，此系統是基於 MVC 架構去做建立。用意是為了在之後在實際應用中，可以做不同硬體裝置的變化。且 MVC 在實作方便去偵錯，且可以直接的減少裝置在使用上的效能問題。

以下為架構圖：



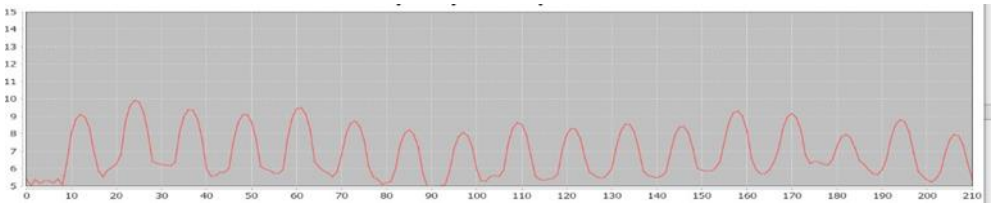
6.2 主程式架構

在資料處裡的方面，是基於基礎理論去實作，以下為資料處理的順序:



在聲音開始輸入時，會先進行聲音的初步處理，透過傅立葉分析(Fourier transform,FT)和漢明窗(Hamming Window)將聲音做初步的取樣和量化，並將聲音訊號透過上述步驟將聲音轉換成雙精準浮點數型態的數字陣列，轉換速度為每 1/100 秒產生一筆 1024 大小之陣列。

接著我們利用轉換後得到的數字陣列來繪製出頻譜圖，頻譜圖會產生數個波峰。當輸入的音訊符合某個音階時，基於泛音的定義，一個音符的八度音會跟前一個聲音差兩倍，那麼頻譜圖上將會產生該音階的第一個波峰，並在相差倍數值的位置也產生波峰。以下為頻譜示意圖：



利用此的特性，我們將會運用這些波峰來作為我們判斷音階的基準：

在判斷輸入聲音的音階之前，我們勢必得先判斷是否有聲音輸入，因此我們將周遭的聲音來當作是否有聲音輸入的判斷基準，我們會將所有的頻譜利用分貝公式算出分貝值，並將新輸入的音訊來與總分貝的平均值做差值運算，若差值達到一定的大小，則判斷為有聲音輸入，反之，則無，利用此方法判斷有無聲音數入，也達到一定的噪音抑制能力；在樂譜中，我們也會利用此方法來去確認歌曲的空白段落。

1. 初版判斷音階方法：

在初版音階判斷中，我們直接將陣列中的所有波峰取出，並直接與樂理上的音階表去做比對判斷，若有 80%以上的波峰相符合，我們就認定這 1/100 秒的聲音為這個音階。以下為音階表：

音名	-1	0	1	2	3	4	5	6	7	8	9	頻率比
C	8.18	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	1
C♯/D♭	8.66	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92	8869.84	1.059463
D	9.18	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64	9397.27	1.122462
D♯/E♭	9.72	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03	9956.06	1.189207
E	10.30	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04	10548.08	1.259921
F	10.91	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65	11175.30	1.334840
F♯/G♭	11.56	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91	11839.82	1.414214
G	12.25	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93	12543.85	1.498307
G♯/A♭	12.98	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88	13289.75	1.587401
A	13.75	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00	14080.00	1.681793
A♯/B♭	14.57	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62	14917.24	1.781797
B	15.43	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13	15804.27	1.887749
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	16744.04	2

但因為音階表需要一次比對多個波峰值，每個波峰都要與音階表上數值相近，且比對判斷需要十分精細，誤差值 1 以上都會導致判斷失誤，若要允許誤差值，也需要另外判斷波峰的前後，因越後面的波峰需要越大的誤差值，種種原因導致我們放棄此辦法，另尋其他更準確且能允許誤差值的方法。

2. 第二版判斷方法：

經過多次實驗後，我們發現音階的判斷不需要將波峰逐一對照音階表，可以利用泛音的特性將波峰之間的間隔來當作判斷音階的依據。

因此我們將所有波峰間的間隔數值儲存，設立計數器去計算，並將間隔數值的前一位、本身和後一位的出現次數都增加，利用數學統計來算出我們需要的數值，此外，我們還將此 1/100 秒的音階做平滑化處理，將近期的五筆資料來一起做音階的判斷，出現最多的值則是當前 1/100 秒的音階。利用這兩種方法，來去降低可能存在的誤差值，這樣一來就能解決先前的判斷失誤的問題。

音階判斷完成後，我們會將時間內的音階整合為樂譜，並將其與資料庫內容透過最常公共子序列來進行比對，並將有一定相似度的結果(45%以上)由相似度高到低顯示給使用者來進行選擇撥放。

6.3 資料庫架構

以下為資料庫架構圖：

music_data			
PK	song_name	varchar(100)	NOT NULL
	scale_feature	longtext	NOT NULL
	path	varchar(300)	NOT NULL
	image_path	varchar(300)	NOT NULL

6.4 人機介面設計

1. 介紹：

以下是我們 UI/UX 的互動方式:

當前的設計是參考大部分得瀏覽器等大眾使用的應用程式。目的是當使用者在做使用時，可以直覺去做操作，不需要透過額外學習。也可以減少使用者在操作上產生錯誤。

2. UI 設計:

目前系統畫面設計採取圖形化介面，透過聲音和按鈕搭配去控制 UI 給使用者的結果及適合的歌曲，在播放部分除了提供基礎的音樂播放的功能。我們也將過去的快轉按鈕置換統一成段落搜尋的按鈕。用意就是要強調此系統的獨有功能。

3. UX 設計:

以下是我們在 UX 上的設計方向:

1. 精簡化:

我們在設計畫面呈現時將畫面視覺簡約化，目的是為了在使用上減少使用者的觀感不適，且不會有多次的跳轉畫面，避免使用複雜化。

2. 大眾化：

由於我們的取向會是希望服務的客群可以很廣，因此基於精簡化的前提下，我們的系統能讓操作能獲得良好的使用體驗。讓大部分客群可以正常使用，減少操作造成的系統錯誤。

七、 開發工具

前端	Java
後端	Java
溝通方式	有線或無線網路溝通
資料庫	Mysql Database
開發環境	NetBeans

八、 測試報告

8.1 測試內容敘述

利用麥克風聲音來做為搜尋輸入，哼唱歌曲為童話的歌曲旋律，預期找到哼唱的歌曲與段落。

8.2 測試環境

	名稱	規格或版本
硬體環境	Windows 環境	一般桌上型電腦
軟體環境	Java JDK	Java JDK15
資料庫端	Windows 環境	phpMyAdmin:XAMPP3.3.0

8.3 測試結果

結果如預期相同，能在資料庫中能找出相似資料，並能依照相似程度進行排序，童話也是系統認定最相似的歌曲，找出的歌曲段落與哼唱的段落也相同。

九、 成本分析

品名	數量	價格
伺服器主機	1	24000~30000 元
麥克風	1	500~1000 元
喇叭	1	500~1000 元

十、 結論及未來發展

10.1 結論

目前的程式雖然可以將想搜尋的音樂顯示在列表中了，且準確率相當高。也能實現雜訊抑制，在有外物聲音干擾的狀況下準確率也不太會引響到判斷的準確率。因此可以證明此系統是具有辨識能力的，且此系統也證實在未透過深度學習的套件的幫助下，也能做到聲音辨識。

10.2 未來發展

1. 在搜尋準確率上，我們會將資料庫內部的音樂長度加長，並同時測是在資料比數和資料長度變長的狀況下，比對的成功率是否能達到預期成效。且在資料庫內部資料比數加大下，是否也能達到預期成果。
2. 目前在搜尋必須要和原唱的音調大致相似才能搜尋到目標歌曲，未來我們會容許使用者在哼唱時可運用不同音調哼唱，只要沒有嚴重的走音，都是可允許並且可提供給系統搜尋的。
3. 同第一點，我們後續會將資料庫內容持續增加，並且將資料庫內容音樂長度以原始長度儲存，用來測試如果資料筆數會不會影響在搜尋時找到目標的時間。在段落搜尋的部分，我們會測試在原始音樂長度做段落搜尋會不會影響到搜尋效率，以及如何將搜尋效率做優化。