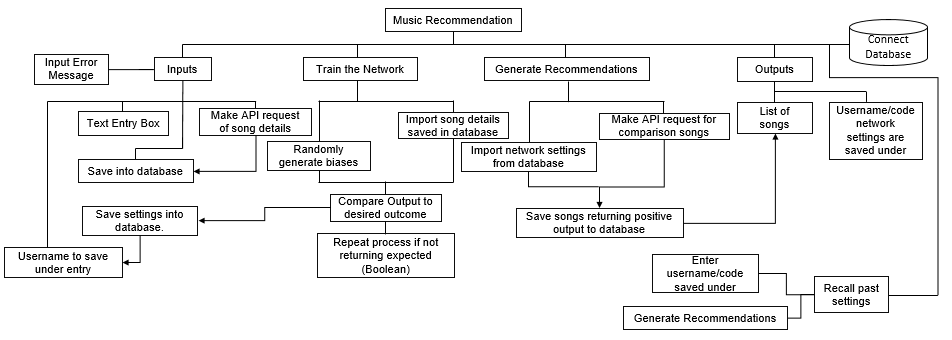
Design of the solution

Decomposing the problem

*“3.2.1 Decompose the problem*

1. *Break down the problem into smaller parts suitable for computational solutions justifying any decisions made”*

In order to break down how my program will run, I have created a hierarchical diagram for the processing behind the recommendation. This breaks down processes that each element of the program has to do to be able to its basic levels. This does not detail much of the interface and user interaction but it allows me to see each of the aspects I will need to develop and how data and processes interact between different branches of the program.

The main parts of my program are the training and application of the neural network, as such the above diagram tends to focus on those and how the interlink with each other, and the inputs and outputs involved. Everything that takes place will be saved into a database connected when the program loads.

The main idea behind this project is that the features of songs can be enumerated to a base level and that by doing this it would be possible to determine what about a song is important to a given person. This unlocks the potential to automate a recommendation process to be done without human interaction after the user enters a set of songs and clicks a ‘go’ button. This automation is where using a neural network presents its advantages, by having a pre-determined set of inputs and (knowing that the user already likes these songs) outputs you can set up the program to configure itself randomly until it reaches the expected output. This process is effectively a crude mimicking of a learning brain and allows the network to potentially simulate a human’s preference in music. By taking a new set of input songs (and data sets) without a known output, by passing them through the same set of conditions that are configured in a way that was determined correct in the ‘training’ stage of the process, it should be able to return a decision that states whether it believes the user will like a song. These outputs should be displayed in an easy to understand list, the value returned by the network should be converted into a Boolean result giving a yes or no decision.

The perfect simulation of a human’s though process would be mimicking the full complexity of connections, but considering that would require over 100 billion virtual ‘neuron like’ interlinked connections it would be impractical and take a huge amount of processing time and power. Therefore, I should take an abstraction of the idea taking it down to a few layers of inputs, biases and outputs, this should allow it to output reasonably accurate values while not taking a huge amount of time to work through its processes. Data such as the users name, and other personal information does not need to be taken into account when making recommendations as the system is basing it entirely off of the chosen songs and the numerical data associated with them.

Describing The Solution

“3.2.2 Describe the solution

(a) Explain and justify the structure of the solution.”

My success criteria states that

“The program should be able to train neural network using the data fetched from the songs the user inputs”,

as part of my problem decomposition, I have shown how the main section of the program is orientated around the training and use of the neural network and how those general points break down and the processes that have to take place at a base level to them out.

“The program can be run from download without any changes made to set it up allowing anyone to use the application without any maintenance at a later date by a ‘administrator’ user, therefore all users have access to the same information.”

This should be very achievable with the structure I have described as it is all very self-reliant, ideally with the database links, the user should have no problems if the program and database are saved and kept in the same local folder as this should allow me to address the database locally without needing to use the full address, if this did end up being a problem the inclusion of a method to search and locate the database’s location via a file explorer window built into the program.

“(b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.”



Input to database process (1)

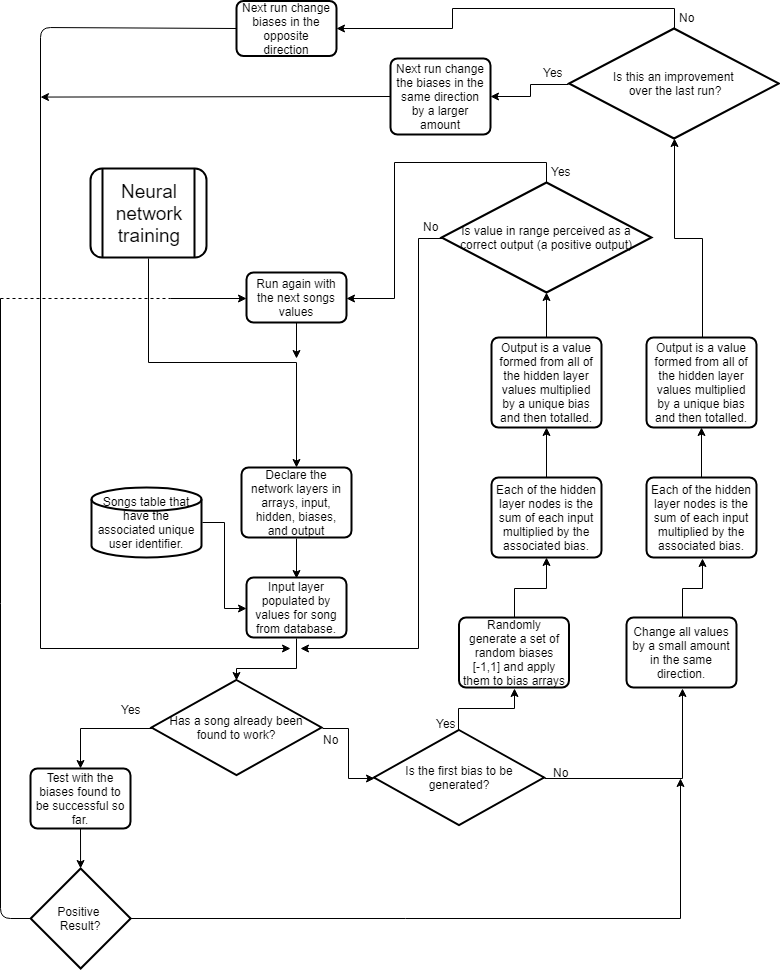
API request flowchart to be added.

These flow charts depict the main processes that make up the bulk of this project, above shows the process that takes the users inputs and makes an API request and then saves the data required into a table that will be used later. As I have been mentioning throughout the design process, the user’s inputs will not make up a huge section of the processing underdone by the program, the largest and fundamental part of the program is outlined in flowchart 3, it depicts the process from after the songs have been chosen and saved into the database. In order for code to ‘learn’ the training function requires a lot of repetitive testing to try and have the right values associated with the key characteristics with a song.

Training the neural network process.

(3)

“(c) Describe usability features to be included in the solution”



I stated in my success criteria that I should make the program simple and easy to use quickly by minimising the number of different menus and pages that must be navigated. In order to do this, I aim to create a program that can work around a single main menu from which all functionality can be access, with a page for receiving results and saving recommendations under a code.

This simple design was influenced by the simple web interface of the Gnoosic recommendation system which I found to be intuitive and when showing it to others it was clear what it did and did not allow for much error in use.

There are clear text entry boxes that will allow the user to enter song names on the same page as it being loaded so it can be quickly accessed without a long window navigation process.

The menu has concise instructions that allow a first-time user to understand what to do while not being overwhelmed by a huge amount of text.

“(d) Identify key variables / data structures / classes justifying choices and any necessary validation.”

My program will use a simple database to store the values associated with each of the songs fetched, a set of songs to compare against and make suggestions from and the songs chosen by each run of the program and the code assigned. All other data will be stored temporarily in the variables declared in the program itself.

tblChosenSongs

tblSongs

tblRecommendations

To make my storing of data efficient the database is designed to meet 3rd normal form, by removing any repeated data.

|  |  |  |  |
| --- | --- | --- | --- |
| tblChosenSongs | | | |
| Field Name | Field Type | Validation? | Key Field? |
| User Code | AutoNumber | Required | Primary |
| Song1ID | Text | Required and exists in tblSongs | Foreign |
| Song2ID | Text | Required and exists in tblSongs | Foreign |
| Song3ID | Text | Required and exists in tblSongs | Foreign |

This table hold the songs that the user entered and is linked to the overall songs table, when the songs are processed after entry the names are checked against the songs table and then the ID is saved into this table.

|  |  |  |  |
| --- | --- | --- | --- |
| tblChosonSongs | | | |
| Field Name | Field Type | Validation? | Key Field? |
| UserCode | AutoNumber | Required | Primary |
| Song1ID | Text | Required and exists in tblSongs | Foreign |
| Song2ID | Text | Required and exists in tblSongs | Foreign |
| Song3ID | Text | Required and exists in tblSongs | Foreign |

Like the way to other table works, this table holds the ID of the songs that have been recommended by the algorithm which can then be fetched from the songs table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Field Name | Field Type | Validation? | Key Field? |
| SongID | Text | Required | Primary |
| Song Name | Text | Required | Indexed |
| Danceability | Number | Required |  |
| Energy | Number | Required |  |
| Speechiness | Number | Required |  |
| Acousticness | Number | Required |  |
| Valence | Number | Required |  |
| Tempo | Number | Required |  |
| Time Signature | Number | Required |  |

This table holds all of the numerical data stored on the songs by Spotify which will be fetched via the API, the ID is an autonumber as it is a unique text field generated for each song by Spotify. Each of the fields is a piece of data that is returned from Spotify and saved into the database from a .json returned file format.

Approach to Testing

“(a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data”

In order to make sure my system is working throughout development and in order to minimise the amount of time spending debugging, I will be iteratively testing each section of code and checking the logic with white box testing to find reasons why loops and comparisons may not be working as intended. When I have completed a section of my program I will use black box testing to make sure that each procedure gives a reasonable output given a set of different inputs. The project does not have many inputs by the user which means that in order to black box test I will have to produce data to put through each function that will test the functions ability to operate under different conditions.

When the project is complete I will run a full system test to make sure that all the procedures/functions run together and the full program produces recommendations that make sense, the best way to test this would be to get a user to see if they agree with the systems recommendations.