Evaluation

Testing and Evaluating the Success of Solution

*“3.4.1 Testing to inform evaluation*

*(a) Provide annotated evidence of testing the solution of robustness at the end of the development process.*

*(b) Provide annotated evidence of usability testing (user feedback).”*

*“3.4.2 Success of the solution*

*(a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.”*

In order to best test my solution I will be comparing my final solution to the success criteria I decided upon in my design section, this will allow me to evaluate how close to my original goal I have been in creating a prototype product.

Some of these success criteria cannot be tested with a distinct pass or fail in any other way than getting a user to test the product and give feedback on how they believe it fairs in regard to success criteria. The criteria that requires a user test and feedback is:

*“1. The graphical user interface is not daunting and easy to understand and use, there is no need for instructions and the program is presented to be intuitive for all users no matter ability/prior experience.”*

To test this I let my main stakeholder user Tom who I identified in my analysis use the program and decide if he found it easy to use from the point of view of not having seen it during development. To measure how he feels my success has been I asked him to score the ease of use, the appearance of the program, and the amount of help required for first time use out of 10. I decided before giving him the program to test that I would regard above a 7 in each category as a pass.

The feedback he gave was that ‘*The interface is simple and easy to use while still looking somewhat nice. At first the set-up can be slightly confusing when retrieving the API key, but after a while you can get used to it.’*

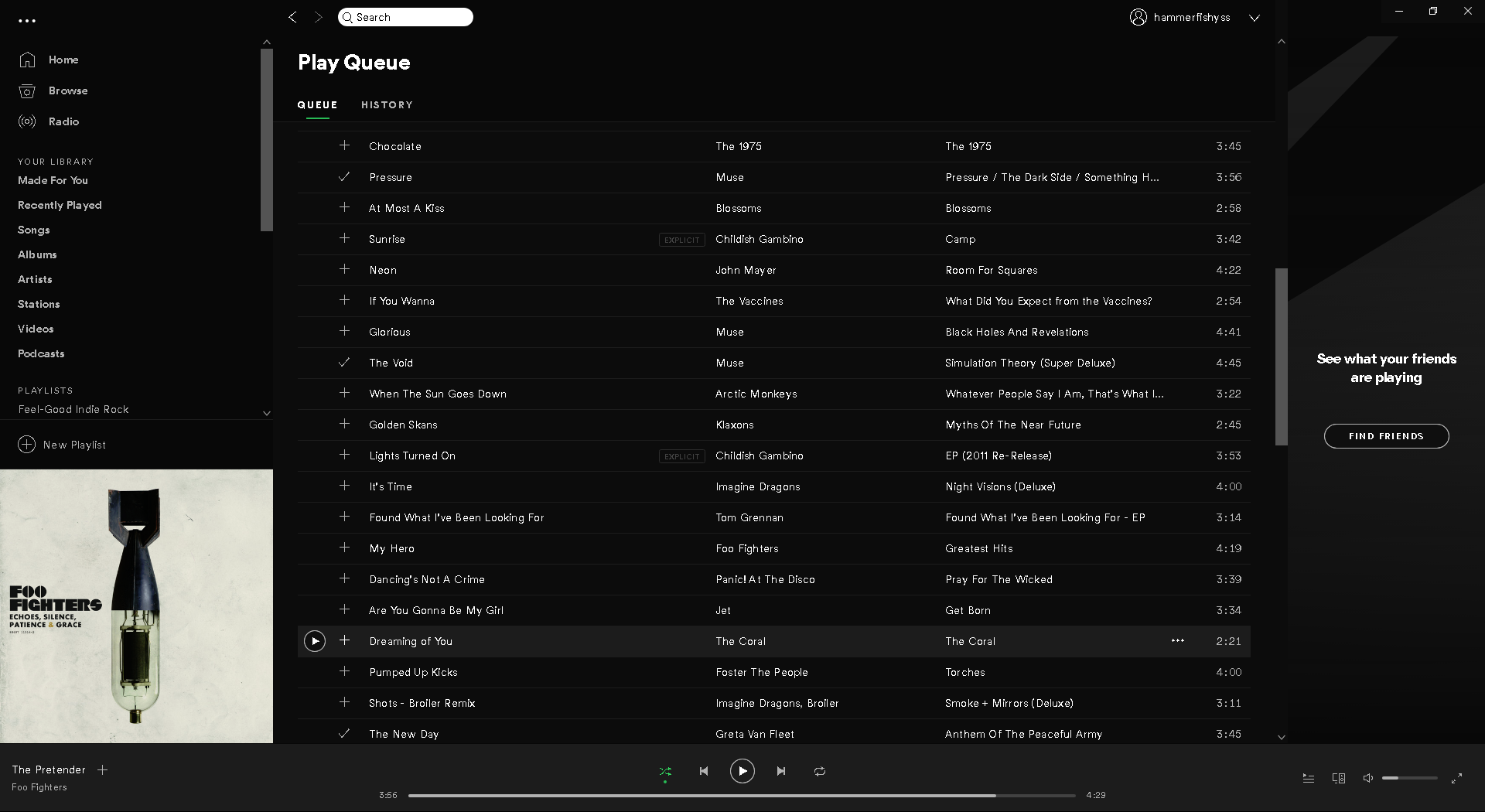
Based on this he gave an **8** and **9** for ease of use and appearance but a **5**for a first time user experience. I believe that with a few changes to the set-up process the first time user experience can be improved but the main section I was prototyping in this work was the algorithm. This is a partial success as in a later iteration this program could be improved to make this a full success and overall this criteria does not impact all that much of the overall functionality.

*“2. The program can take a user’s input of song names and accurately find the songs and pull the data associated with the song from the Spotify API.”*

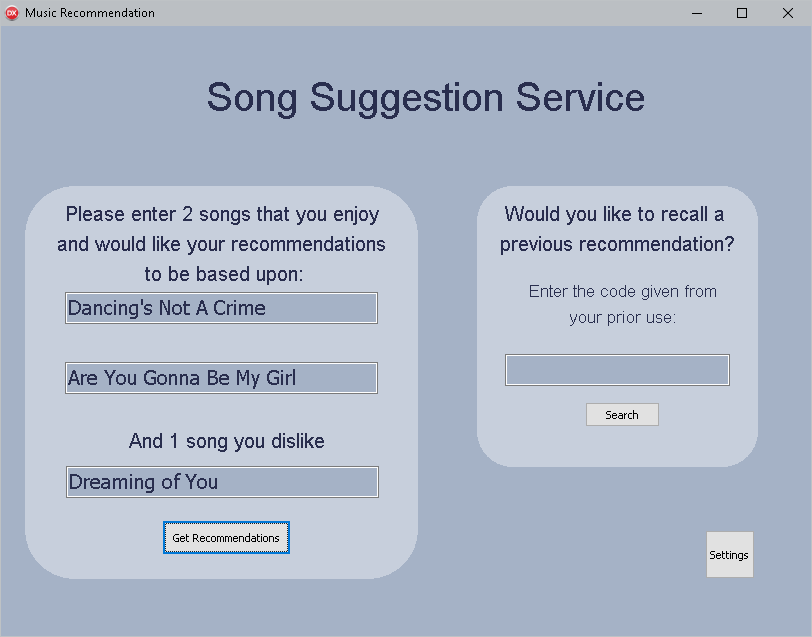
To test this I can run the program and test that to see what new data is put in the data base:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Data** | **Expected Outcome** | **Outcome** |
| 16. | Normal Data (A song name spelt correctly that I know is available on Spotify) | The correct song ID is placed in the database and when loading that ID the correct song is found. | The correct song IDs are placed within the database.  **Pass** |

Evidence of songs being on Spotify:

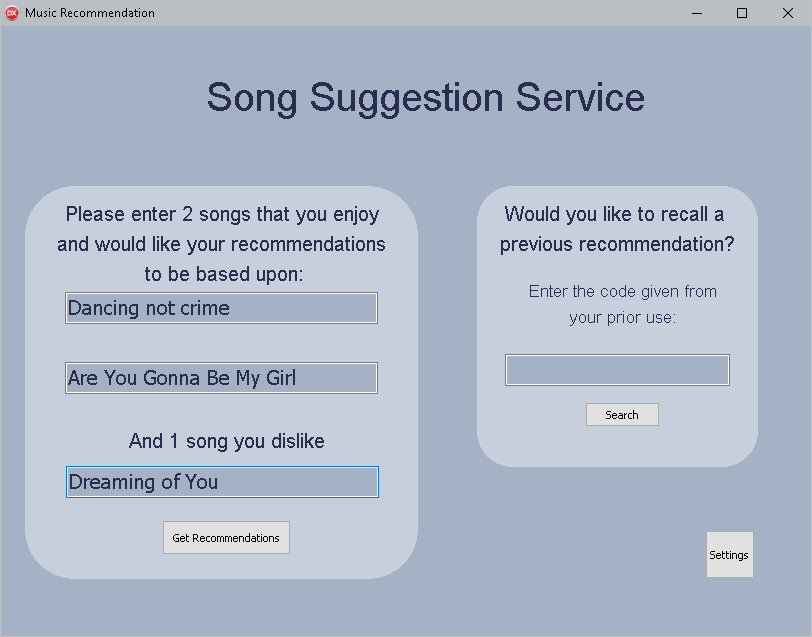


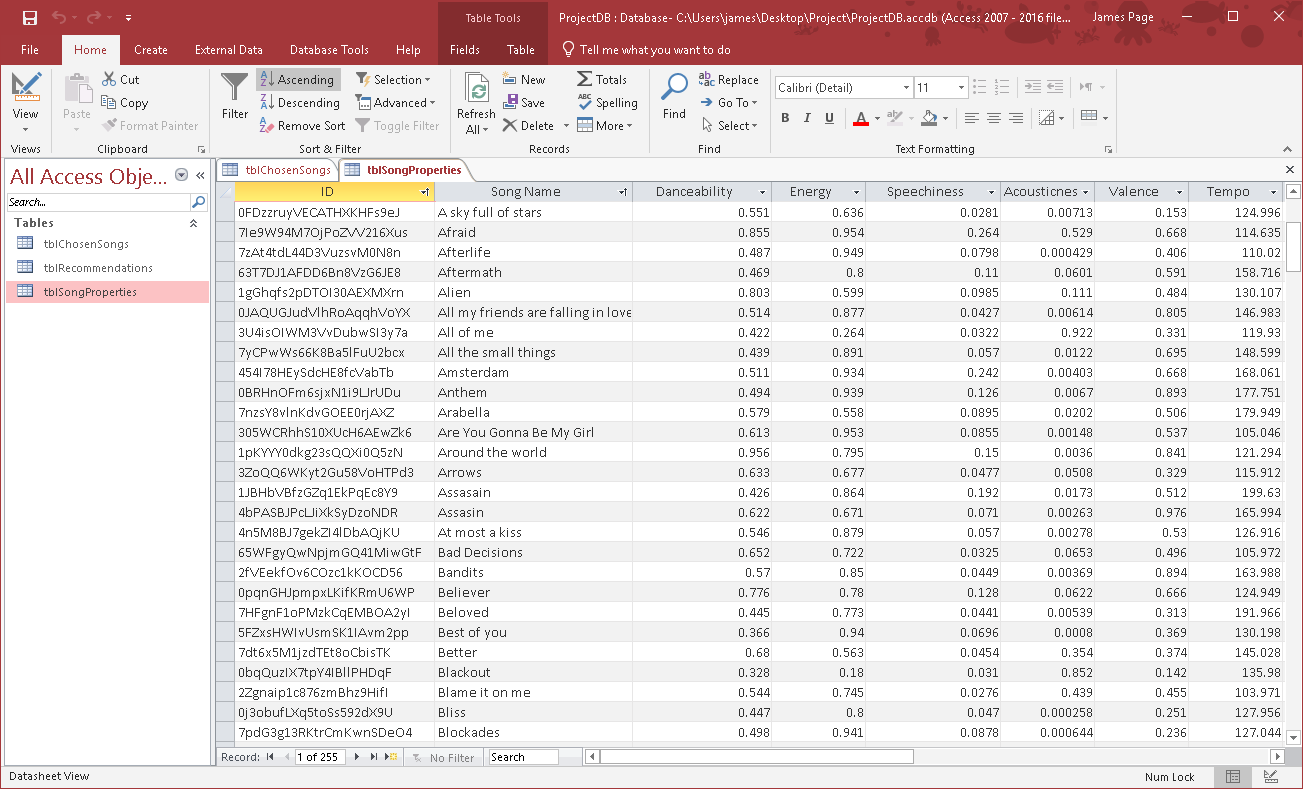
Song names correctly entered:



|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Data** | **Expected Outcome** | **Outcome** |
| 17. | Abnormal Data (The song searched for exists but has been entered incorrectly | The search still returns the correct song unless something it accidentally means a new song name is found. | The second time running it (486) has the same ID even with the misspelt name.  **Pass** |

The name of the first song is slightly incorrectly typed:





This criteria has been fully met as the correct song IDs are found and stored along with the whole collection of data about each song as shown above, this first section of the main process working is the first step to the overall processing being successful.

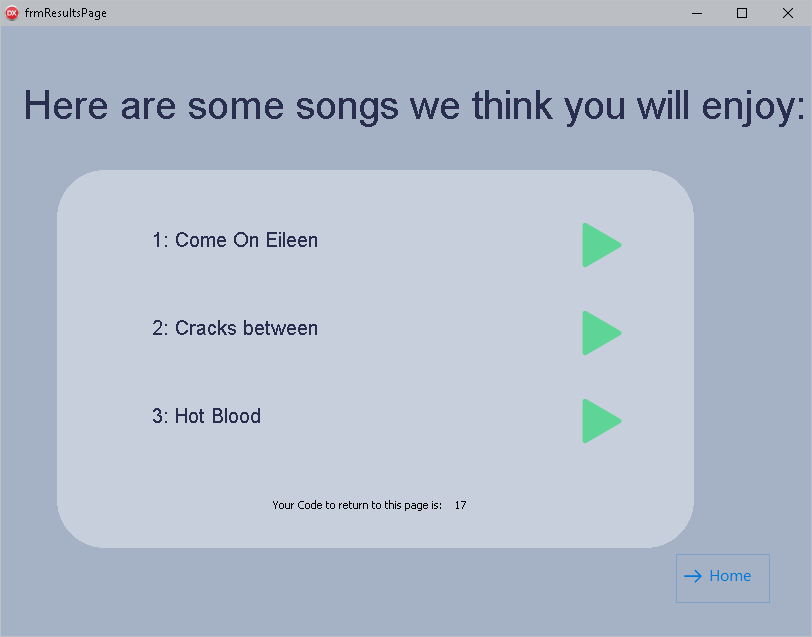
*“3. The system should be able to pass the values through a neural network algorithm and return a Boolean value, predicting like or dislike.”*

*“7. The user is presented a clear list of songs after the process has been run which give details of each song allowing them to easily be found and then listened to.”*

These two criteria come as a pair in the final product as the systems decision as to what the user is presented by the algorithm is tied to the output the user sees.

As I developed the program I realised that the criteria of number 3 would not apply that much as rather than using a Boolean decision on every song, it would be better to find the songs with the best match rather than a huge list of songs that just about match. This means that in my final product I did not meet the 3rd criteria at all, but instead ended with a better more useful system.

The recommendation of songs and outputs can be seen in tests 5 and 6 of my development which show how the program makes a choice of the 3 most likely songs for you to enjoy, and outputs them in a clear easy to read interface.



The 7th criteria is met by allowing the user to very easily listen to the song and find it on Spotify by the implementation of the green ‘play’ buttons that will load up a page for the user to just click play and hear the songs they have been told they might like. This criteria has been met and within the development section tests 10-12 show it working correctly.

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

This criteria also ties in with the results screen due to the neural network processing not being visible to the user except in the occasional wait time between clicking the button and the page showing up. The algorithm works correctly for a large amount of the time and when it does finish running this criteria is fully met. Unfortunately there are some combinations of songs that do not find any configurations that work even after the rework of the processing that I described within the development. This means that in its current state I cannot describe this as a complete success as while it works correctly it is not all of the time.

*“9. The program should be robust and should not run into errors in the code, any potential errors should be caught and return a message to the user informing them of the problem with their inputs.”*

Under the correct use case the program is will work correctly and where possible I have put in place error catching systems to prevent errors and crashes. The program does not include many places for user input, and thus there is not much room for user error, but in the case there is a problem it can cause problems with the overall functionality of the program.

To test the robustness I will test the possible inputs:

The first input the user is presented with is the song entry, but this input’s functionality depends on the state of the access token to connect to the API, which is also a user input.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Test Data** | **Expected Outcome** | **Outcome** |
| 18. | Normal data (the names of the songs are entered correctly with the access token entered correctly too) | Loads the results page without running an error. | **Pass** |
| 19. | Abnormal data (the songs are entered correctly but the access token is not correct) | Shows a prompt to the user letting them know what the problem is and doesn’t run the code that would lead to an error. | **Fail** |

In test 19 I found that when running a test without entering a access token the program returned an error to do with the database indicating the error handling I had implemented had not come into effect and I had missed this during development, from this I cannot say that criteria 9 has been completed successfully.

This was not completely failed though because as seen in test number 13 and 15 of my development, the try and except catches implemented for this text entry point have worked correctly, this shows that the problem here was caused by a rushed test in the development stage that I had assumed to be fixed.

This in part may have been due to trying to meet success criteria 8:

*“8. The program is finished and able to be distributed by the end of February 2019.”*

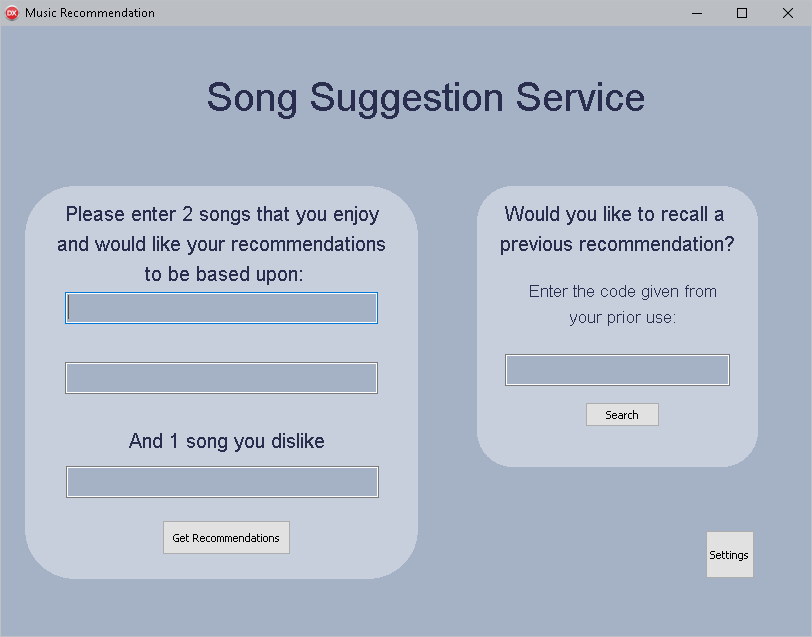
Trying to have a prototype that demonstrated the functionality of the program by that deadline meant that I was more focused on testing to see if the main algorithm worked correctly and could produce results, rather than ironing out issues that I was not running into when testing due to knowing how to correctly use the program.

Final Product

3.4.3 Describe the final product

(a) Provided evidence of the usability features justifying their success, partial success or failure as effective usability features.

(b) Provided comments on how any issues with partially or unmet usability features could be addressed in further development.



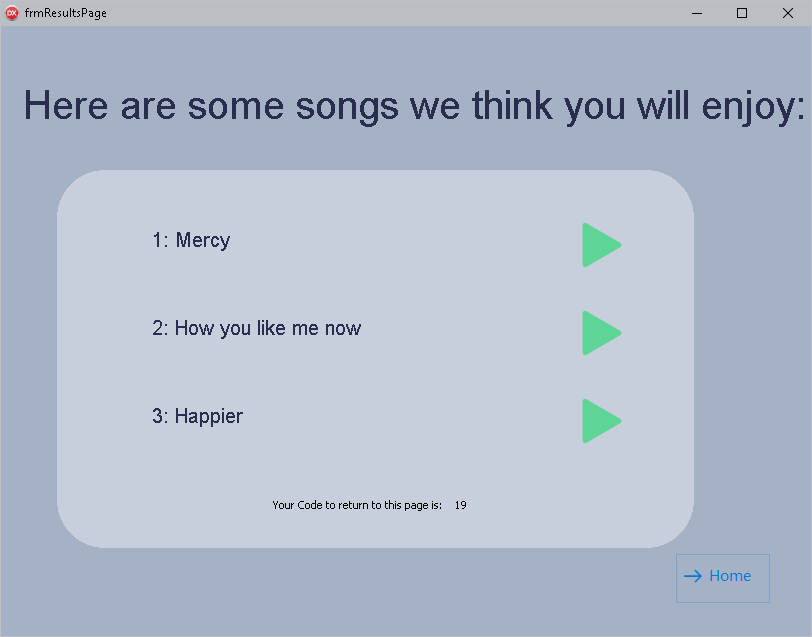
Bold clear and concise instructions, to allow for all abilities to use the program.

Simple labelled buttons so users know what they are clicking

When putting together the program, I tried to copy the interface I had mocked up for the design section. This meant keeping the interface simple and straight forward with large plan text and concise easy to follow instructions.

As seen above there are not many options for the user, I believe that I have managed to create a main menu that anyone could use without outside assistance. From user feedback from my named stakeholder Tom I have said that I was successful in making this main menu simple and usable for everyone.

Clear title so the user knows what this page actual means when they are redirected.



Names of the songs clear, bold and well-spaced, to make sure the user knows which button corresponds to which name

A feature not in the original design of this page – simple instructions on what the number is in reference to the instructions on the main menu.

Universal ‘play’ symbol used to reduce the clutter that would be created in simply providing the user with a link to copy and paste.

Again when designing this page I tried to copy my designed features as they encompassed how to make this project most successful in the design department. Some things have changed from the design such as the inclusion of the recall code that can be used to return to the page another time. These are features that I had not considered when designing the interface so instead I tried to keep to the idea of all parts of the interface should be easy to understand without explanation.

This has translated very well into a final program with the buttons working correctly directing the user quickly and easily. Overall design and feature wise I believe this project has overall been a success with the main improvements made being to brighten up the colour scheme to be slightly more interesting to look at when using the program.

Further Development

3.4.4 Maintenance and development

(a) Discuss the maintainability of the solution and any limitations of the system.

(b) Described how the program could be developed to deal with limitations of potential improvements / changes.

(c) Discuss potential further development of the solution.

One of the main issues with the current design of the project is how it will scale with future use, in order to most effectively recommend songs to the user I should have a very complete and all-inclusive database of songs. This can be compiled quite easily using the Spotify API, unfortunately it is quite a manual task. By having the program working from a local database it means that if it was updated manually regularly to include a new set of songs, it would require constant downloads and updates which is very inconvenient for the user especially as the database becomes even bigger.

To get around this problem, if this project was ever scaled up to a larger project you could make use of connecting it to a central server that can be updated without the users having to update their personal copy of the file. Overall this would improve the usability of the system but for a small prototype without a committed userbase it is not worth the time to set it up.

Another future development with this program is that the core neural network training algorithm has a lot of room for improvement. Currently it works in a fairly rudimentary fashion but it still works correctly, by later improving this you open the door for more efficient processing time and less areas where the user might be found in a long wait time not knowing if they will ever get a result from the program. Some of this is due to limitations in the development environment, in future a different language could be used with more support for parallel processing and general neural network processing features. There exist many very powerful libraries for neural network training for languages such as python and C++, but as part of this project I tried to create my own version of an algorithm to work in exactly the way I designed it.

Overall, as a prototype stage this final product is as much of a success as I could have hoped, the processing works correctly and has provided results that I and my stakeholders have deemed to be evidence of a successful algorithm. All of the future developments I have mentioned here are things that could be well implemented in a future project but would require it to be committed to a larger project that would then be developed again from scratch, restarting the development cycle from the ground up with this project as a research point in its analysis.