Emotion-Based Music Recommendation System

Joshua Todd

ID: 1921479

BSc Computer Science

40 Credit Final Year Project Dissertation

Word Count: 12190

Supervised by

Dr Wendy Yanez Pazmino

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2023-24

Abstract

There is a deep connection between a person's emotions and what music they like listening to. A common theme amongst systems that attempt to harness this strong connection is the lack of live emotion analysis and personalisation. A user's emotions will naturally fluctuate overtime and subsequently so may their music taste, but systems often neglect this. It is also important to allow users to personalise what music is played for each emotion as everyone has a unique response to music, a feature which is also often overlooked. The goal of this project is to produce a music system that can provide live emotion analysis in conjunction with personalised music choices to provide a more meaningful listening experience.

To achieve this goal, a machine learning model was created that evaluates the key facial landmarks of a user's face and makes an emotion prediction. This was done using computer vision techniques and a neural network to spot and identify patterns amongst emotions. We were able to integrate this model within a web application that recommended music based on the current detected mood and a users music preference to provide suitable music. By conducting user testing it was determined that the application was able to provide users with a unique and adaptable listening experience which elevated their listening experience.

Contents

CHAPTER 1

Introduction

The link between music and emotions has long been established for much of human history, and whilst the question to why music can affect our emotions is one that is still debated to this day, the correlation between music and emotions are clear to see [15], [13]. Music has a unique ability to affect the way we feel, it can change, compliment or heighten a current mood. This powerful connection is unique to every person and this project will aim to provide an application which everyone can use to improve their individual listening experiences.

Another important aspect of emotion to consider is how easily it can change over time, whether it is due to music or other factors. This change can result in a person's music preferences changing to align with the new mood. The ability to detect these changes and adjust accordingly is an important goal for this project. The user should have the choice to decide how they wish to deal with a particular emotion, for example a user may wish to listen to music which makes them happy when they are sad to try and lift them out of their sad state. Every user will have unique responses to emotions, and this is something the application will aim to facilitate.

There are solutions discussed in the literature review that try and recommend music based off a user's emotions, however a few pitfalls were identified. A lot of the systems lacked the ability to give live emotion detection, instead they would recommend music based off one image from the user, without taking into consideration any emotion changes. They also suffered from poor personalisation due to pre-fixed song choices for certain emotions, this breaks all connections between the music and the current emotion if the user did not like the current music being played.

This project aims to fill these pitfalls, by creating a platform for users that provides personalised music recommendations through live emotion detection within a fully functional music platform. This should result in a more meaningful listening experience for users to enjoy music in a new way to what they are used to.

To achieve this goal, the first step was to create a machine learning model that can classify between emotions. It was identified via the literature review that the use of a neural network was the best fit for this task, particularly a convolutional neural network, due to its strong ability to distinguish patterns amongst visual data. Key facial landmarks were extracted in the form of coordinates to extract the key information and patterns from a face, which a convoluted neural network will aim to learn and classify. It was not clear which dataset was the most appropriate for this task so two models were created by different datasets and an evaluation step was used to determine which model would best fit the application. Alongside the model, a web application

was developed by evaluating currently available software and a user study to gather requirements. The application can evaluate frames from a live web camera feed to return live emotion analysis. The web application will use a combination of a user's music preferences for each emotion and the currently detected emotion to recommend music. The use of a music API allowed the application to offer additional features and access to a wide range of music and artists.

The evaluation of this project evaluated aspects such as the machine learning model, meeting requirements and a user study. The results of this evaluation are generally positive, and overall, the goals of this project were met.

CHAPTER 2

Literature Review

2.1 Introduction

There have been many ways to try and create music recommendation systems based off users' emotions and this literature review aims to provide an overview of the state-of-the-art systems and where they can be improved. We will split the problem up into two sections, the first being the emotion detection and the second will look at how systems recommend music based on emotions.

2.2 Emotion Detection

This section will evaluate emotion detection techniques and provide insight to how they perform with special consideration to how they perform in the real world as that is the domain in which we will be using them in this project.

When assessing how to capture a person's emotions, according to [17] there are six ways to measure emotions. These methods are cognitive appraisal, subjective feeling, physiological response, expression, action tendency and regulation. For this project a user's expression will be evaluated, in particular their facial emotion. This was chosen since it is one of the most achievable options using computer vision and classification techniques. For this reason, this section will explore how to detect and classify facial emotions.

A common approach to the problem of facial emotion detection is the use a convolutional neural network (CNN). A CNN learns emotions by scanning photos of people's faces for hierarchical features, this pattern recognition is then used to classify the faces into emotion categories. Studies [8] and [14] creates two similar CNN architectures with a mixture of 2D convolutional layers, pooling layers, normalisation layers and dense output layers. [8] uses the FER-2013 dataset [4] which is made up of 28709 photos which are 48x48 and grey scaled and split into seven emotional categories. In [14], the authors create their own dataset of 3000 photos which is split into 5 categories. Architecture [8] achieved a test accuracy of 70% whilst [14] achieved an accuracy of 78%, this higher accuracy is expected due to lower number of emotion classes. The benefit of using CNNs are their high accuracy and there is no need for any feature extraction before making predictions as this process is included the CNN. To make predictions you just need to pre-process the image and feed it straight into the model. However, problems can arise when CNNs are used

in real world examples, as outside factors such a lighting and camera resolution play a factor to the input and subsequently the accuracy of the model can suffer.

In [10] they employ a CNN with pre-processing techniques such as rotations of the images, intensity normalisation and image cropping to try and remove as much meaningless information as possible so the CNN can more accurately extract features. This was successful when tested using the CK+ database [11] as it achieved an accuracy of 96% as the CNN was able to more successfully pick out meaningful features for classification. This database is 981 high-quality photos taken in a controlled environment split into 7 categories, so whilst this accuracy may seem a lot higher than the other CNN approach, this would be expected for this dataset. However, it still proves the importance of the pre-processing steps taken for the photos.

In [5] they use a more classical approach of using feature extraction then using a classifier to classify emotions based off the features. The first step is to use the Viola-Jones method [19] to detect faces within a frame. The Haar cascade is then used to detect facial features in the detected faces using the sequential application of single rectangular filters which analyse the pixel densities of different areas of the face. These different objects are then passed through a support vector machine (SVM) which finds a hyperplane that maximises the margin between the different classes in high dimensional spaces. This method worked well as it was able to achieve a 94% accuracy when trained on the CK+ database. The method also suffers in poor lighting and resolution environments due to the Haar cascade steps reliance on pixel information.

There are multiple ways to extract features from a face and [2] use an Active Shape Model (ASM). The ASM works by extracting points that fit the contour of the face and they then used a SVM to classify the key features. This approach made use of the CK+ database and achieved a successful 95% validation accuracy average over 7 emotions. Whilst this is a strong accuracy, it may struggle in real world scenarios due its dependence on the data set which may not capture the full range of face shape variations.

Instead of just analysing a singular frame, the work done in [7] instead uses a video made up of multiple frames as the input and uses a hybrid neural network with long short-term memory (LSTM) layers and CNN layers. LSTM are a type of recurrent neural network (RNN) and are good for capturing patterns in sequential data by remembering information over time. They use a CNN to extract features and patterns from the faces and a LSTM to track how these features change over time depending on the corresponding emotion. This was very accurate when tracking emotions over time in the form of sequential frames but is difficult to supply sequential frames that show coherent emotions that the neural network can classify.

Based off this research, the approach of key point extraction in the form of three-dimensional coordinates on the face would be interesting and then use that data to train a CNN. This combines the classical approach of feature extraction to optimise the amount of useful information extracted, and the high-power CNNs to spot patterns amongst the coordinates in a spatial environment. The utilisation of coordinate-based approaches in facial recognition may hold an advantage over the Haar cascade techniques. Unlike the latter, which consolidates features into patterns, coordinates could offer the advantage of capturing more intricate facial data. The use of CNNs can examine spatial relationships between the facial points and extract complex patterns.

2.3 Music Recommendation Software Review

This section will discuss mood-based music recommendation systems with respect to how they recommend music to users, it will act as a software review of current solutions to the problem.

In [1] they use a CNN model to classify the different emotions. They then use this emotion to pick songs from a predefined database of songs to give to the user. This leads to song choices that don't have much personalisation due to everybody having different music tastes so users might have different preferences for different emotions. It also does not provide any real time analysis as it makes a one-off judgement of the user's emotion, but as we know this emotion can change

meaning the music preference of the user might also change.

The application created in [3] uses CNNs in a similar method to [8] and achieve a higher test accuracy of 0.80 by only classifying between happy and sad. Whilst this higher accuracy might help with the users experience for those emotions, people have a far wider range of emotions, so it is very limiting to use just two. It also suffered from the same personalisation problems as [1] due to a predefined database and poor results in poor lighting or poor image resolution.

The system created in [6] also uses a CNN architecture in the context of a mobile device and set out to classify between 10 facial emotions. The user would upload a picture of themselves to the mobile app and an emotion would be recognised by the CNN and a song would play. The system made use of the Spotify API and users entering what music they want to listen to for a corresponding mood. This approach is successful in providing users good personalisation for their music choices. However, like the other examples, music suggestions are just one-off pictures and there is no real time emotion analysis to recommend music. The accuracy suffered at 58% due to the high number of emotion classes to distinguish between. Lowering the number of emotions may be beneficial in creating a more accurate model and people may not need music preferences for such a wide range of emotions.

When evaluating more mainstream music platforms such as Spotify [16], they have a much better user experience than the systems discussed in this literature review with features such as a vast music library and playlist creation amongst others. However, Spotify lacks the depth of emotion analysis that the other systems have. It does have the ability to automatically create mood-based playlist based off music you listen to which does add an element of a personalised listening experience, but this has a few shortcomings. These playlists are something you need to manually search for and start listening to with no live emotion detection to change the playlist according to the current mood. It also categorises the music on what the perceived emotion of that song is, which may not align with the actual music you like for an emotion. For example, it would put a perceived happy song in a happy playlist however, a user may like this type of music for another emotion.

There is a need for a system which can provide accurate, real time facial emotion analysis to change music recommendation dynamically instead of the other techniques that don't take into consideration users emotional changes. This will provide the users with greater personalisation of music choices and hence a greater listening experience that lines up with their current mood more accurately. The use of a music api will also help broaden the range of available music and features to users and removes the need to create limiting music databases, increasing the users experience.

CHAPTER 3

Product Specification

3.1 Gathering Requirements

3.1.1 Survey Design

The goal of running a user study was to help gather requirements for the web application. Whilst weaknesses of current systems have been identified through the literature review, it is important to get user input to justify the research and build upon the findings.

The method used to gather this information was using a simple online survey created using Google Forms. The survey was introduced as a final year project requirements gathering survey with the goal of creating a platform that can recommend music for a given emotion. The survey is comprised of three questions:

1. A multi choice question asking what emotions influence what music you listen to.
2. For all the emotions they selected in the first question, participants were asked what music they liked to listen to for that corresponding emotion.
3. What features would the participants want from a music platform.

The first question has the choices, happiness, sadness, fear, disgust, neutral, surprise and anger. These emotions were chosen as they are core emotions and when doing research on datasets, these emotions are the most common used to classify emotions. This question yields important information that will be used to create the emotion detection model by showing how many emotions the model needs to classify. The second question will help gauge what people want to hear for every emotion and will help build the music recommendation system. The third question will help build the additional features around the music recommendation system to help achieve the goal of creating a more complete music platform than those analysed in the literature review. The survey is short in nature to try and mitigate survey fatigue and we felt it covered all the necessary aspects of the project. In total there were 30 responses to the survey with a range of University of Birmingham (UOB) and non-UOB students between the ages of 18-25.

3.1.2 Survey Results

This section will discuss the results of the survey and how they can be used to create the overall application.

Question 1: Please Select all the emotions which influences what music you listen to. For this question there are four emotions that were consistently selected. Happiness (86.7%), sadness (86.7%), neutral (80%) and anger (80%) all scored highly whilst fear (20%), disgust (6.7%) and surprise (20%) all scored a lot lower. This very clear divide backs up the findings of the literature review that users don't want too many emotions to choose between and we need more than just happy and sad. A positive side effect of reducing the number of emotion categories to classify between to just four, is this will help increase the accuracy of the model we will create.

Question 2: For all the emotions selected above, please enter what music you like listening to for that emotion. For this question there were a very wide range of music selected for every emotion with no clear patterns. This justifies the belief created in the literature review that it is very important to allow users to control what music is played to for each emotion. This will provide users a much more personalised listening experience.

Question 3: What features would you want from a music platform? The most popular features chosen in this question are those that people have grown accustomed to having on mainstream platforms such as Spotify. This included the ability to save songs into a playlist to listen to later, to be able to search for music and artists, to play any song they like. This will be more clearly reflected in the technical specification section.

3.2 Technical Specification

3.2.1 Functional Requirements

Now we have formed the clear goals of this project through the literature review and user study, we can break the project down into requirements to achieve this goal. As part of developing the application and forming requirements it can be unclear what is achievable in the given time frame so to help combat this, we will use the MoSCoW method. This method splits up the requirements into categories *must-have*, *should-have*, *could-have* and *won't-have* to help identify the most important features to be completed. If we accomplish all the *must-have* requirements we will have an application that satisfies the constraints of being a minimal viable product. This will be something that satisfies the goals of this project but nothing extra. Building upon this minimal viable product and including as many *should-have* and *could-have* requirements will help elevate the application into something that offers the users a greater user experience and will boost user satisfaction. *Won't-have* requirements will not be used in this case as this methodology is usually used in practice with multiple development cycles and these *won't-have* requirements offer goals for future development cycles, however I am limited to just one development cycle. Instead, I will discuss future work and features to be included that is beyond the scope and capabilities of the project at the end of the dissertation and these could then be converted into *won't-have* requirements. When deciding how to categorise the requirements, I considered how essential the requirements was in achieving the goals of this project and the potential time frame of implementation.

Music Recommendation Feature:

(R.1) **Must** provide real time emotion analysis through the use of the users laptop camera.

(R.2) **Must** classify between the four emotions, happy, sad, neutral and angry.

(R.3) **Must** allow users to save up to five total genres/artists/songs for every emotion into a database - this will act as a basis for the music recommendation.

(R.4) **Must** recommend suitable music based off the users saved music preferences and the currently detected emotion.

(R.5) **Must** be able to save songs into a playlist.

(R.6) **Should** be able to switch between live emotion detection and manually choose an emotion to recommend music for.

(R.7) **Could** allow a user to choose between music recommendations and assigning playlists to an emotion to control exactly what music plays.

Music Controller Feature:

(R.8) **Must** show the current song and artist playing.

(R.9) **Must** be able to pause/play the current song.

(R.10) **Must** be able to skip the current song.

(R.11) **Must** be able to go back a song.

(R.12) **Must** have an option to add the current playing song to a playlist.

(R.13) **Should** be able to seek to a position in the song.

(R.14) **Should** be able to change volume in the application.

(R.15) **Could** be able to enter a shuffle mode.

Playlist Feature:

(R.16) **Must** be able to play any playlist.

(R.17) **Must** be able to create a playlist.

(R.18) **Must** be able to delete a playlist.

(R.19) **Must** be able to display all the songs in the playlist

(R.20) **Must** be able to remove songs from the playlist.

(R.21) **Should** be able to add an optional description for every playlist.

(R.22) **Should** be able to go between shuffle and non-shuffle mode when listening to playlists.

(R.23) **Should** be able to add songs in the playlist to other playlists.

(R.24) **Should** highlight the current playing song in the song list

(R.25) **Should** be able to manually play any song in the playlist

(R.26) **Could** be able to search for items in the playlist

(R.27) **Could** be able to generate a playlist given a user's music preferences.

Search Feature

(R.28) **Must** allow the search of artists and songs.

(R.29) **Must** allow users to play a song they have searched for.

(R.30) **Must** be able to play music and albums displayed on an artist's page.

(R.31) **Must** provide accurate results to the search term.

(R.32) **Should** be able to search other people playlists and be able to play them.

(R.33) **Could** be able to add other people's playlists to their own playlist library.

3.2.2 Non-functional Requirements

Here are the non-functional requirements which discuss more how the application should behave without going into specific functionality.

(R.34) **The application should be easy to use and aesthetically pleasing**. This can only be measured through user testing, but the goal is for the application to be placed in front of the user with very limited prior knowledge and they should easily navigate and understand what everything does. This can be achieved through clear sign posting and descriptions without compromising the look of the application.

(R.35) **The application should be fully responsive to change**. There are several live elements to the application such as current song playing, emotion detected, playlists being added/removed etc, and all these changes should be reflected throughout the app as the changes happen.

(R.36) **Application should be quick to load and be responsive**. This will help provide a better user experience if load times are kept to a minimum. This can be achieved by efficiently implementing all the features to limit the number of resources needed.

(R.37) **The application should be accessible on all modern web browsers**. This includes but is not limited to Chrome, Firefox, Safari, Internet Explorer. This will allow a wider range of users to use the application.

(R.38) **Application should have accurate emotion analysis**. The more accurately the model can make predictions, the users will have an overall better listening experience. This can be evaluated by looking at metrics of the model versus similar systems and user testing.

(R.39) **The system should be easy to maintain and grow**. The application should be implemented such that future enhancements and features should be easy to implement. This will be achievable by writing well-structured code and a modular implementation.

CHAPTER 4

Product Development

4.1 Facial Emotion Detection Model

To create the neural network that will be used to detect emotions we will follow the classical machine learning pipeline. First step is to choose a dataset, this will most likely be a collection of images from which we can extract important information from. The next step is to pre-process the images, if required, and extract the information from the images, this will take the form of coordinates of key landmarks on the user's face. This information will be used to train a convolutional neural network, during this step we will experiment with different layer types and hyper-parameters to try and maximise the accuracy of the model. Finally, we will evaluate the model by looking at metrics such as accuracy and looking how the models perform in action. This process has been split up into sections and further explained.

4.1.1 Choosing a Dataset

The first step in creating a neural network model is finding a suitable dataset from which we can extract useful information from to train the model. In this section we will look at some of the available options conclude which datasets should be used to create a model to suit our needs. When evaluating the suitability of a dataset for our task there are several variables to consider for each dataset. This includes factors such as the environment the photos are taken in, the diversity of the subjects, the size of the datasets and the emotion range of the dataset. When conducting the literature review there are two datasets that were predominantly used more than others, the ck+ [11] and FER-2013 [4], these are the datasets that will be considered for this project.

Starting with the ck+ dataset [11], this is a dataset that was released in 2010 and is an extension of the already existing ck dataset [9]. The ck+ dataset includes 593 video sequences from 123 different subjects ranging over seven emotions. In each video sequence there are three frames which capture the subject going from a neutral expression to the targeted peak expression. Each sequence is labelled with a corresponding emotion. The benefit of having this range of emotions for every subject and emotion is a model could theoretically learn the more subtle differences in facial emotions leading to greater accuracy. A potential drawback to this dataset is the environment in which the photos were taken. Each photo is in a very controlled environment with good lighting, resolution and every subject are centred in the image making an emotion directly into the camera.

This may cause poor generalisation issues as these peak conditions may not simulate real world environments which is the domain in which we will be working in.

The next dataset to discuss is the FER-2013 dataset [4] which was created in 2013 as part of a competition to find the best emotion recognition algorithm. This dataset is far larger than ck+ and is comprised of 28709 pictures of faces ranging over seven emotions. This was created using the Google search API which was used to search for face images for each emotion using 184 key words such as "happy", "angry" and others. The benefit of using that more randomised method of collecting images is that is leads to a greater diversity of subjects and environments. This could lead to greater generalisation properties when used on users in real world scenarios as the training images have come from a real-world setting. A draw back to this dataset is the number of images for every emotion label does have some variations, for example there are 8989 happy images, 6077 sad images, 4953 angry images and 6198 images. This may lead to bias towards to the over-represented categories whilst the model might struggle to recognise the more under-represented classes leading to poorer generalisation properties.

After conducting the research, it still wasn't clear which dataset was the appropriate choice based off their strengths and weaknesses. When evaluating how users will use the application, they will most likely be using the facial detection emotion when they are sat in front of their laptop face on in a more controlled setting. It is therefore unclear whether the randomness of the FER-2013 dataset will benefit the model in this situation. Moving forwards, we will make models for both datasets and compare them on several metrics including evaluation accuracy, precision of each emotion detected and physically testing the model in the domain it will be used in. After completing all this evaluation, we will then pick a model to be used in the application for the emotion detection.

4.1.2 Data extraction

Multiple methods of feature extraction have been discussed in the literature review but for this project we will use key facial landmark extraction. For this task we will make use of the MediaPipe framework [12] to extract these key points for every picture in the dataset. MediaPipe is a framework created by Google and it offers a way to efficiently extract the coordinates of key facial landmarks. For every image, MediaPipe will extract 478 facial landmarks all with x, y, z values which are normalised between zero and one.

This extraction was done using a Python program which reads in all the pictures one by one and extracts the landmark coordinates for each. The output of this was written to a CSV with the corresponding emotion label for each image. For the FER-2013 dataset, this was already split into training and testing data, so a CSV was created for both. This is not the case for the ck+ dataset for which we will have to do a manual split of 70% training data and 30% testing data.

4.1.3 Model Creation

To create the neural network model, we will use the TensorFlow framework [18], made by Google, in Python. TensorFlow is a widely used library used to create neural networks as it abstracts away a lot of the difficulty associated with creating high performing neural networks. It allows us to control variables such as layer types, number of nodes and activation functions without having to implement these ourselves.

The first step of the model creation is to pre-process the data, so it is ready to be used by the neural network. First, we read the coordinate data from the CSV file and reshape this by grouping the all the coordinates together. We end up with every photo having a shape of 478, 3 corresponding to the 478 coordinates for every photo, each having a x, y and z value. Next was to use one-hot encoding to encode the labels into binary vectors for every class of emotion. Instead of the labels being words such as "happy", "sad", "neutral" and "angry", it will encode these to be [1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]. This is now in the form the neural network will

be able to understand. There was no need to do any data normalisation because this has already been done for us by MediaPipe during the data extraction.

When creating a model, it is difficult to know hyper-parameters such as number of layers, number of nodes etc before training, this leads to a lot of trial and error to see what works. This process was made easier by TensorBoard, this is a tool provided by TensorFlow that tracks metrics such as loss and accuracy during the training process and visualises them using graphs. This provides an easy way to assess a model against other models to see which combinations of parameters correlate to high evaluation accuracy and low amount of loss. I also made use of Google Collab, this is a tool provided by Google that allows us to run and execute Python code in the browser and make use of their free to use GPU run times. Training neural networks is computationally intensive, and GPUs massively speed up the training process compared to a laptop which cannot handle the load. To create the architecture, I created a Python program that iteratively trained models with different combinations of node numbers, layer types and number of layers and let this run in Google Collab. Whilst training all the different models, the logs were saved and once all the models had finished running, I could use TensorBoard to see which model performed the best. This whole process was repeated for both datasets.

When creating the model I used the following:

- **1D convolution layers** - By using convolutional layers, the neural network can capture patterns and spatial relationships among the different emotion labels. The reason it is a one-dimensional convolution layer is due to the input type being a sequential stream of coordinates meaning the convolution is applies across a single spatial dimension. The Rectified Linear Unit (ReLU) activation function had the best results when used with the 1D convolution layers. ReLU is an activation function which only outputs positive values, if the output from a node is below zero, the output is set to zero. This is beneficial as it results in nodes which are sparsely activated which can help regularisation and prevent over-fitting. ReLU is also very efficient since the only computation that takes place is comparison, addition and multiplication. Sigmoid and tanh activation functions were considered but they did not perform as well as ReLU.
- **1D max pooling layers** - This layer usually follows after a convolution layer and are used to reduce dimensionality. Max pooling reduces the spatial dimensions of the input and aims to conserve only the useful information by retaining only the maximum values. To determine the size of the window to apply this max function over we can change the pool size hyper parameter.
- **Flattening layer** - Transitioning from convolutional layers to dense layers a flattening layer is required because it is these dense layers that will provide the final classification. Dense layers cannot take output from a convolutional layer as input due to the shape of the data. The flattening layer reshapes the multi-dimensional data coming from the convolutional layer into a one-dimensional vector which can be used by the dense layers.
- **Dense layers** - To finish off the model architecture there are some fully connected layers in which every neuron in a layer in connected to every neuron in the next layer. Every connection between the neurons has a weight parameter which the model will adjust to learn complex patterns and classify data. In this context the input of the dense layer is the patterns learnt from the convolutional layers and the dense layers need to learn the connection between these patterns and their emotion labels. For these layers the ReLU activation function was also used except for the final dense layer for the same reasons it was used in the convolution layers. The final layer uses a softmax activation function, as it converts the output into a probability distribution over the four emotion classes. The final layer has four neurons each corresponding to a different emotion class and each having a probability of that emotion being detected, which has been determined by the softmax activation function.
- **Dropout layer** - These layers are used in conjunction with the dense layer as a method of avoiding over fitting. Over fitting is what occurs when the model begins to learn the noise in the training data instead of the underlying patters. This results in a model that learns the input data well but does not generalise well to new examples which needs to be avoided.

A method used to avoid this over fitting is to use dropout layers which will randomly set a fraction of the input to zero thus introducing noise to the data. This will encourage the neurons to learn the more important features instead of features specific to the data set which could lead to poor generalisation.

- **Adam optimiser** - When compiling the model, TensorFlow has several optimisers available to use which all aim to iteratively adjust the model's parameters during training to provide the best performance. Whilst trying different optimisers, the Adam optimiser performed the best.

Figures 4.2 and 4.1 show the architectures of the two models created, both of which have been fully evaluated in the evaluation section 5.1.

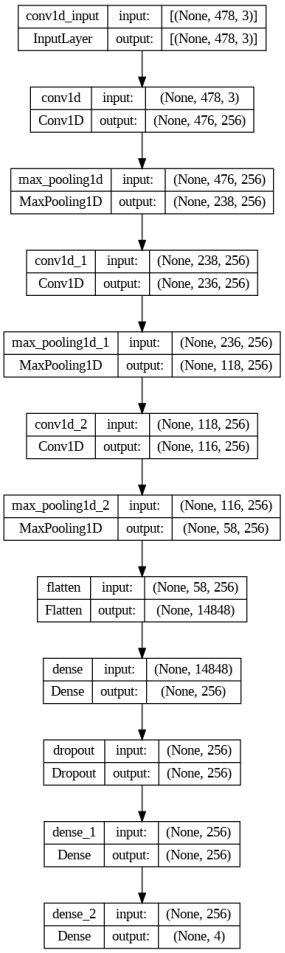


Figure 4.1: Model created using FER-2013 dataset

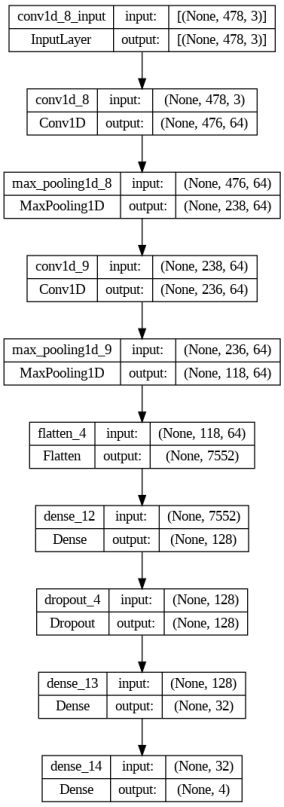


Figure 4.2: Model created using ck+ dataset

4.2 Web Application Development

4.2.1 Architecture

The architecture used for this project was a client-server, three tier architecture. A reason for choosing the three-tier architecture is its clear separation of layers. The client layer, server layer and database layer all have defined roles, and it is easier to break down features across these three layers making implementation and meeting requirements an easier task. This architecture also allows for better scalability and testing over other architecture types like monolithic. These other architecture types tend to have more tightly coupled components which makes it harder to add or change current components without impacting the entire system. Below is a high-level view of the architecture. It must be noted, this diagram does not show the exact implementation of features and layers but instead offers an insight to the structure and the flow of the entire system.

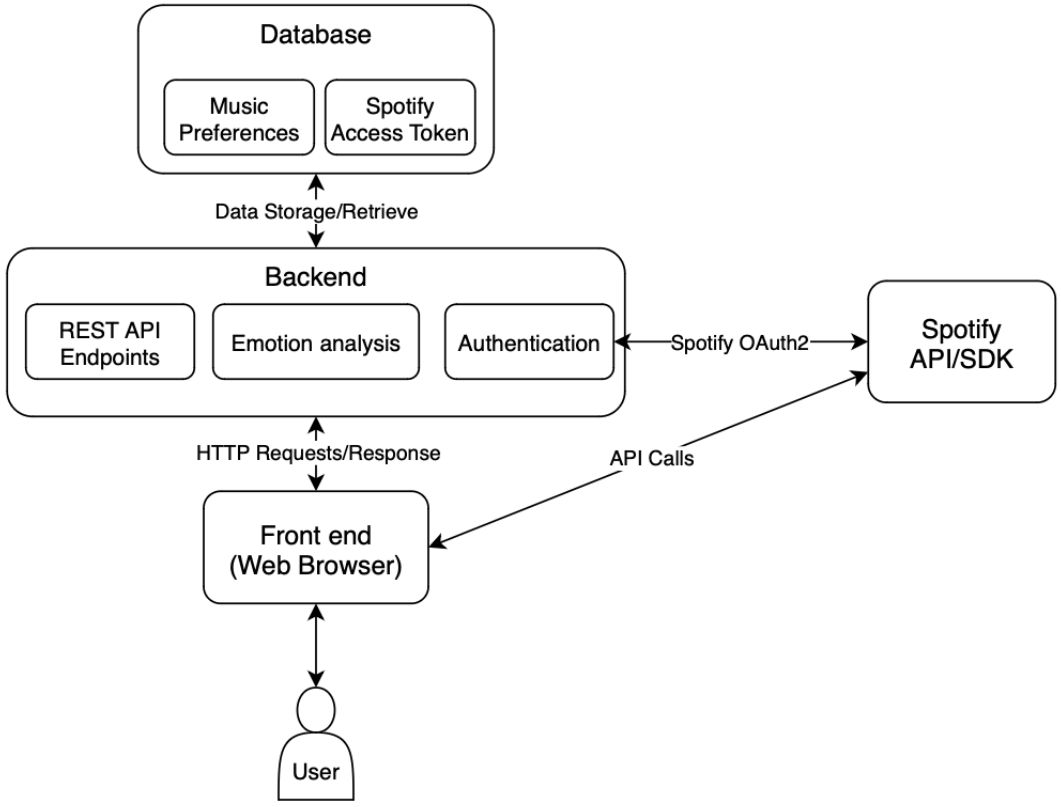


Figure 4.3: Architecture Overview

4.2.2 Spotify API

This section will outline the need to use a third-party music API, in particular the Spotify API to achieve the goals of the project.

An issue when creating a music platform are the restrictions we have when it comes to obtaining and playing music due to licensing and copyright issues. There is no way of obtaining large music libraries that will offer a diverse range of music that will satisfy the needs of the users. Similarly, there needs to be a way to access large amounts of artist and song information. For example, for every song we will need to know meta data such as song length, song name, artist name, song art etc, this adds up to be a vast amount of data that we do not have direct access to. Using a third-party music API, we will gain access to millions of songs and thousands of artists with minimal effort which greatly improves the capability of the application that is being built.

A potential downside to using a third-party API is the potential over reliance on the API. If an endpoint goes down or has an error, unfortunately there is nothing we can do to fix the problem but wait for the third-party to fix the issue. This is something that occurred a few times during development, where fetching an endpoint returned a 500-error code which indicated an internal server error which is out of our applications control. Another potential issue is public APIs will often have rate limits to control the amount of traffic to ensure the API cannot be abused with excessive requests or to charge a fee for additional calls. It is important to pick an API that provides enough calls, so it does not limit the speed to development and one that does not cost extra money to use.

When assessing different third-party music APIs, the basic requirements for the API to be even considered are the ability to play music from a vast music library, access to large amount of song and artist data and for reasonable rate limits. Only two music APIs can be considered following these basic requirements and they are the Spotify API and the SoundCloud API.

When comparing both the APIs song catalogues, they both boast a very large library size. Spotify claim to have over 100 million songs and SoundCloud have over 375 million songs. This is to be expected since getting music onto SoundCloud is far easier than Spotify. To publish music on Spotify you need to go through an application process through an external provider whilst anyone can upload onto SoundCloud whenever they want. This results in a wider range of options on SoundCloud especially when it comes to finding new upcoming artists. However, it is worth noting that whilst SoundCloud does have 3.75x the music of Spotify, 100 million songs are still an extremely large number of songs that the majority of people will not find limiting.

Both APIs have very reasonable rate limits, both of which should not hamper development too much. SoundCloud are very clear with their limits, you can get 15,000 calls per 24-hour window which is more than enough. Spotify are less clear with their limits, they simply claim that there is a limit within a 30 second rolling window and if you exceed the limit, you will receive errors with a 429 error code. This ambiguity is frustrating but after doing some extra research, the unknown limit should not be too low for the application we are building.

The Spotify API offers a very wide range of functionality such as many playlist operations, search, song recommendations and a sdk to play music. These all integrate with existing Spotify accounts seamlessly. However, SoundCloud offers far less functionality as there is no search feature, no song recommendations, fewer playlist operations and the tooling for streaming music is far less complete than that of Spotify's.

After weighing up both APIs, it was clear that the Spotify API would benefit the application more due to the greater functionality available despite the smaller song library, this should not be an issue for the majority of users. Especially the recommend a song endpoint, this will prove to be very useful in creating the emotion-based music recommendation feature. It is also worth noting that the Spotify API documentation is clearer and more exhaustive than the SoundCloud documentation which should aid the development process.

To get started using the Spotify API you need to make a create a Spotify Developer account. From there you can create an application where you will receive a *Client_id* and *Client_Secret*, these will both be used at a later time to fetch a user's access token.

4.2.3 Back-end Server

The server is written in python using the Django framework. The decision to write the back-end in python is motivated by the fact I have already used Python to create the emotion detection model using TensorFlow and MediaPipe, so to transfer the detection pipeline over to the back-end would be far easier. The reason to use the Django framework is the fact there are many built-in features such as object-relational mapping (ORM) for database management, routing for endpoints, ready to use making the development quicker and more efficient. There are also built in security features that help defend against common web attacks such as SQL injection and cross-cite scripting (XSS). On top of these benefits, I also have some experience with Django meaning I can easily transition into development without the learning curve of learning a new framework.

Authentication

For the user to be able to use the application they will have to own a Spotify account. To make calls on the behalf to the user we need to get access to their access token. This will be done using the Spotify OAuth2.0 flow shown in the diagram below:

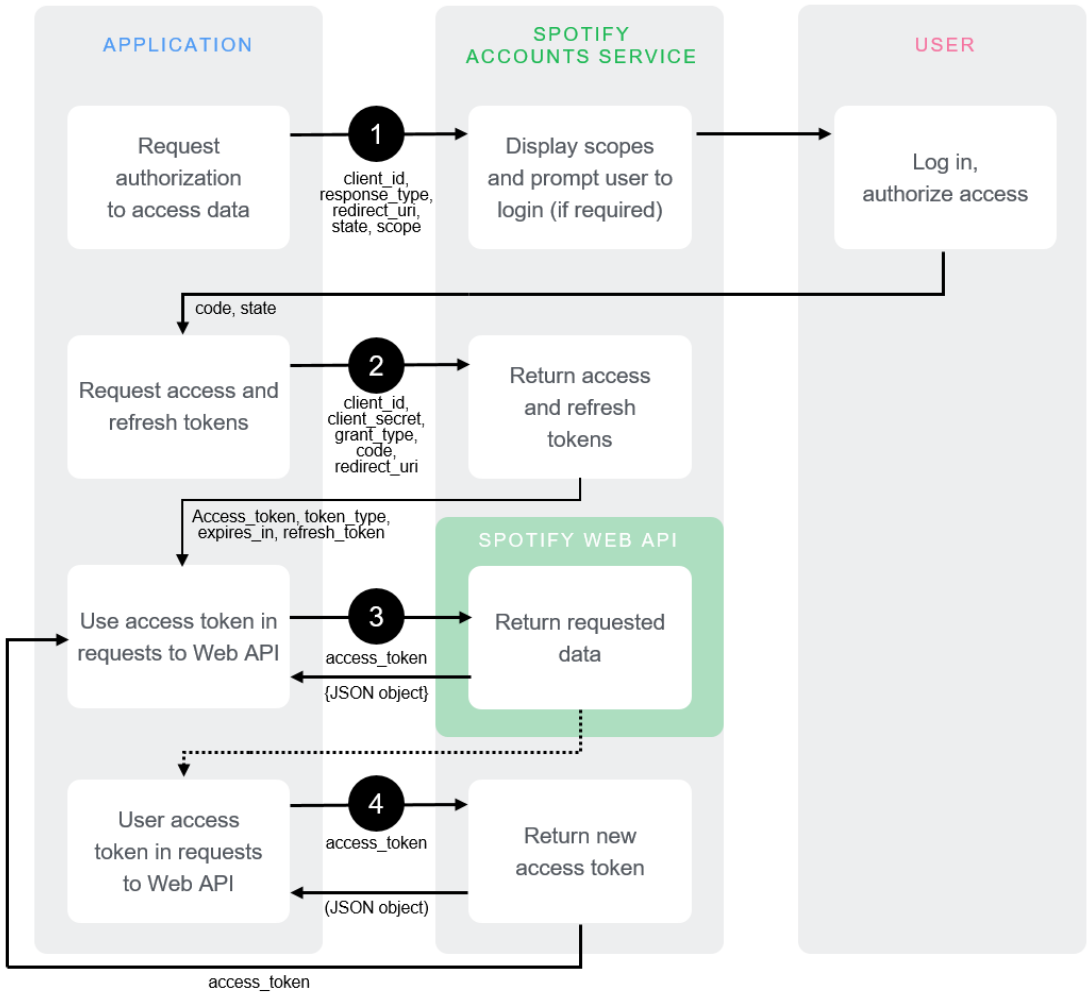


Figure 4.4: Spotify Authentication flow
source by: `https://developer.spotify.com/documentation/web-api/tutorials/code-flow`

When a user first starts the application, they will be met by the login screen, when they click login they will be redirected to Spotify with parameters in the URL including the *client_id*, the scope and redirect uri specifying where the user should be returned to after the login. Here the user can either login to an existing Spotify account or create a new account and then allow our application to perform actions on their behalf. Once the user has authorised the login they will be redirected to http:localhost:8000/spotify/callback with a code in the URL for us to extract. We then send the code back to the Spotify token API in a post request with the *redirect_uri*, *client_id*, *client_secret* and we should receive the *access_token*, *refresh_token*, *token_type* and the *expires_in*. Once we receive these items, they are then stored into the database, so the user only ever has to do this authentication flow once making it suitable for long-running applications. Now we have all the information we need about the user, they are redirected to the application where they can access the entire application.

The *access_token* is something we must send with every API call to Spotify for authentication. The endpoint (HTTP GET /spotify/token) will return the *access_token*, *refresh_token* and *expires_in* variable. The client side will make checks before every API call to see whether the token has expired by checking the time of the *expires_in* variable. If the current time is after the *expires_in* variable then it will call the endpoint (HTTP GET /spotify/refresh-token) where we make a post request to the Spotify token API with the *grant_type : refresh_token*, *client_id*, *client_secret* and the *refresh_token*. The Spotify API will return a new set of tokens which we switch out for the old set in the database.

If the user wishes to logout of the application, they will call the endpoint (HTTP GET /spotify/logout) which deletes all tokens from the database meaning the user will have to repeat the entire authentication flow again if they wish to login.

Emotion Classification

Below is the high-level view of the emotion classification pipeline which is part of the (HTTP GET /api/get-emotion) endpoint.

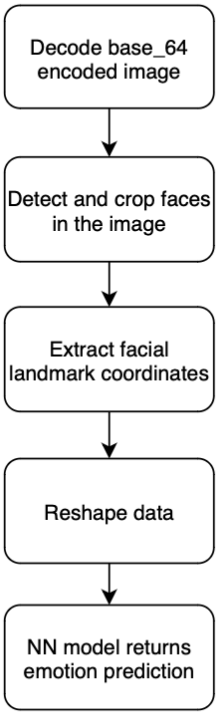


Figure 4.5: Emotion Classification Pipeline

The endpoint (HTTP GET /api/get-emotion) receives a base_64 encoding of an image where it is decoded into an image by the computer vision library cv2. Now we have an image to evaluate, the next step is to detect and crop any faces in the image. Using the cv2 library again, there is a Haar Cascade classifier to detect faces and crop them. The reason we need to crop the images is to match the data in the dataset because the neural network was trained on images of cropped faces so for an accurate prediction, we need to mimic this training data by also cropping the image to just the face. If no face has been detected, then a response will be sent back to the client indicating that no face has been detected.

Now we have the input image in the correct form, it is ready for data extraction. Using the same technique used when extracting data for training the model, MediaPipe is used to extract all the coordinates of key points on the face. The final step of the data preparation is to match the data shape to the input layer of the neural network. The input shape of the neural network is (478, 3), symbolising the 478 coordinates all containing a x, y, z value so we can simply use the np.reshape() function to achieve this shape. Finally, we make a prediction using the model which will return four values, each value corresponds to the likely hood of each emotion being detected. We evaluate which emotion has the greatest likelihood and we return that emotion back to the client where it will be used to recommend music.

Loading a neural network can be time consuming, hence we cannot have a situation where the

model has to be loaded every time there is a prediction. To overcome this, the model is loaded once when the server is started up by using the *__init__* function. The *__init__* is a constructor that is called when a new instance of a class is created, which in this case is when the server is first started up.

Music Preferences

One of the main goals of the project is to allow users to choose what kind of music can be played for an associated emotion. As part of this goal, we will store the music preferences of the user in the database so it can be used in the music recommendation client-side component. For each of the four emotions we can store up to five total songs/artists/genres. The client side will use the Spotify song recommendation API endpoint which you can only send a maximum of five songs/artists/genres to get a corresponding song recommendation to be played. The client will send JSON of the emotions and the corresponding songs/artists/genres for each emotion to the endpoint (HTTP GET /api/set-preferences), where we make use of the Object-Relational Mapping feature of Django to easy save the object into the database.

4.2.4 Client Overview

The front-end was built using React, a JavaScript library created by Facebook. This library uses a component-based architecture which allows us to break down the UI into smaller, reusable components. Each component will contain its own logic and functionality making it easier to manage and maintain code as the complexity of the UI increases. Also, when trying to meet the non-functional requirement stating the application must be fully responsive to change, this is where React will benefit us the most. This is achieved using a concept known as the virtual DOM, it is a lightweight representation of the actual DOM that exists in memory. When changes are made to the UI, it will first update the virtual DOM instead of the actual DOM and those changes are reflected in the UI. This makes dealing with state changes within the UI far easier and faster to deal with because interacting with this virtual DOM is quicker than dealing with the actual DOM. The front-end also uses Babel which is a JavaScript compiler that allows us to write JavaScript using the latest syntax without having to worry about compatibility with older browsers. Babel transpiles the modern JavaScript into equivalent code that can run on old browsers, which helps achieve our goal of being accessible across a wide range of browsers. WebKit is used as the browser engine responsible for rendering the content to the user. It takes the transpiled code from Babel and executes it in the browser. Webkit is good for us because it is optimised for a large range of browsers helping our accessibility goals and response times.

In order to create an application that is aesthetically pleasing the Tailwind CSS framework will be used. Tailwind CSS is a utility-first CSS framework that offers a wide range of pre-built utility classes to style HTML elements. This will lead to quicker more efficient development because there is no need to build up custom CSS in separate files, instead you can quickly apply some Tailwind styles to an element to get the same result. The utility classes provided by Tailwind are highly customisable and have a wide range of functionality making tasks such as animations and effects easier to implement. Throughout the application, Heroicons was used for all the icons due to their wide range of icons, all free of cost, and easy integration with React code.

Login

If the user is visiting the site for the first time, they will be prompted to login. From the user's perspective they are redirected to Spotify where they can either login or create a new Spotify account. They will then agree to allow the application to act on their account and be returned to the application where they can access the entire application. The *access_token* acquired from this process will be stored in a React context layer. The context layer provides a way for data to be available to all the components in the application without having to explicitly passed down

the component tree. This allows the entire application to make fetch requests to the Spotify API using the *access_token*. The *expires_in* variable is also stored in the context layer allowing all fetch requests using the *access_token* to check whether the token has expired before making the call. The token is stored permanently in the database so users will only have to login in once to get access to the application.

Music Controller

To be able to play and control music in the application, the Spotify web playback SDK was used in the music controller showing in 4.6. The SDK allows us to create a fully functional music player that can stream music in the web browser and allow users to interact with the current player state. To use the SDK, the first step is to install the SDK using a <**script**> tag. Once that has been correctly embedded, we can initialise the player where a new <**script**> tag is used to send the users *access_token* and the player name. The SDK will emit events to the browser every time the state of the player changes, we can attach the add listener method to each of the possible events that will listen for changes. An example of a state change would be a new song starting, the event listener will be updated with the new song information such as song name, song length etc, and we can use this to dynamically update the UI to reflect these new changes. Once the listeners have been attached, we can finally connect to the player allowing the user to interact with the music.

A lot of the functional requirements for the music controller can be easily achieved using functions supplied by the web player. Actions such as toggle play and skip song are all very easy to implement by assigning onClick() events of buttons to functions such as player.togglePlay().

Setting up the ability to seek through the song proved a bit more challenging. It involved updating an input field with the slider attribute, every second if the song was currently playing. If the user decides to change the position of the current song, we will need to calculate the position of the thumb of the slider and convert that into the song position depending on how long the song was. For example, if a song is 60 seconds long and you move the thumb to the centre of the slider, you would take 50% of 60 resulting in a seek to position 30 seconds.

To be able to allow user's to add the current playing song to a playlist, there is a button which when pressed will open a pop-up containing all the users owned playlists. The user can then select all the playlists they wish to add the song to, and this action will be completed. This process makes use of the Spotify API, when the pop-up is first opened, a get request is made that will return all the user's playlists. We can then make a POST request to add the song to all the selected playlists.

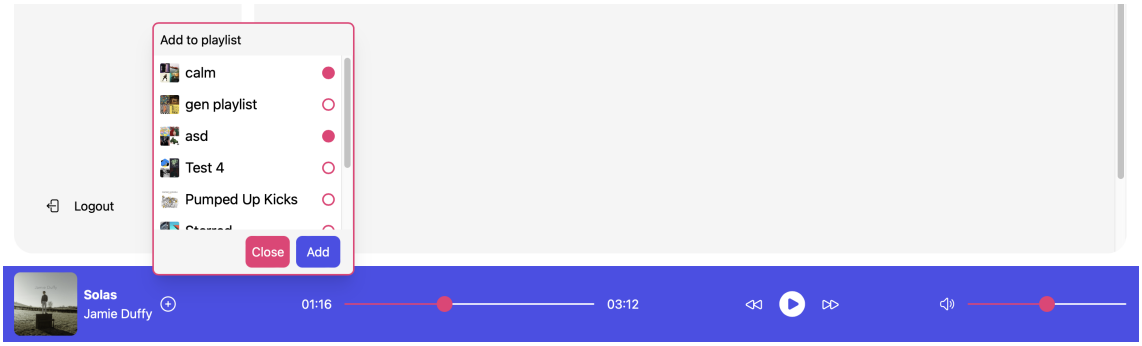


Figure 4.6: Music Controller UI

Music Preferences

To allow users to have personalised music choices, this feature gives the ability to input their desired songs/artists/genres for a corresponding emotion shown in 4.7. This feature makes use of the Spotify search API endpoint. When a user wants to add a new song/artist/genre to their

preference list they are prompted to search for it and whatever they enter into the input field is sent as a query to the Spotify search endpoint which returns the top results which are then displayed to the user. Every time a user adds or removes an item, the updated results are sent in a POST request to the set-preferences endpoint. As default the user will have some generic preferences loaded in for them so they can start using the emotion-based music recommendation straight away with no setup, they can then change these default preferences as they wish.

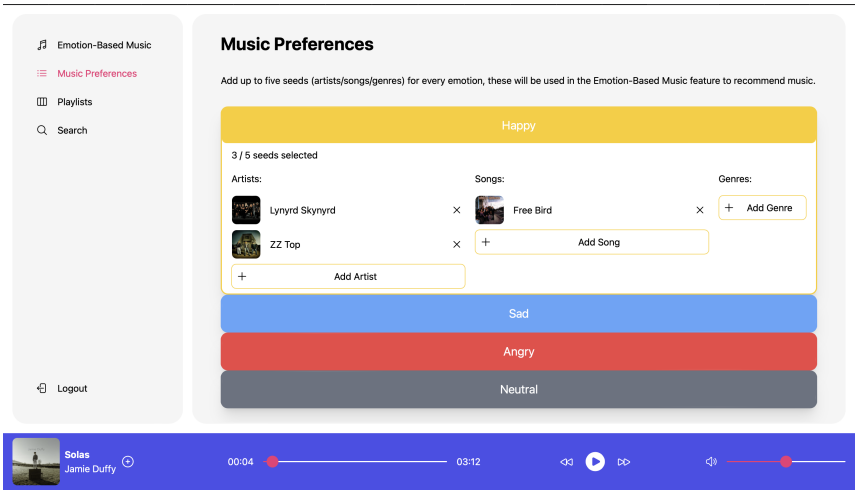


Figure 4.7: Preference Screen UI

Emotion-based music recommendation

Here is an overview of the live emotion-based music recommendation flow that will allow users to listen to music based off their current emotion.

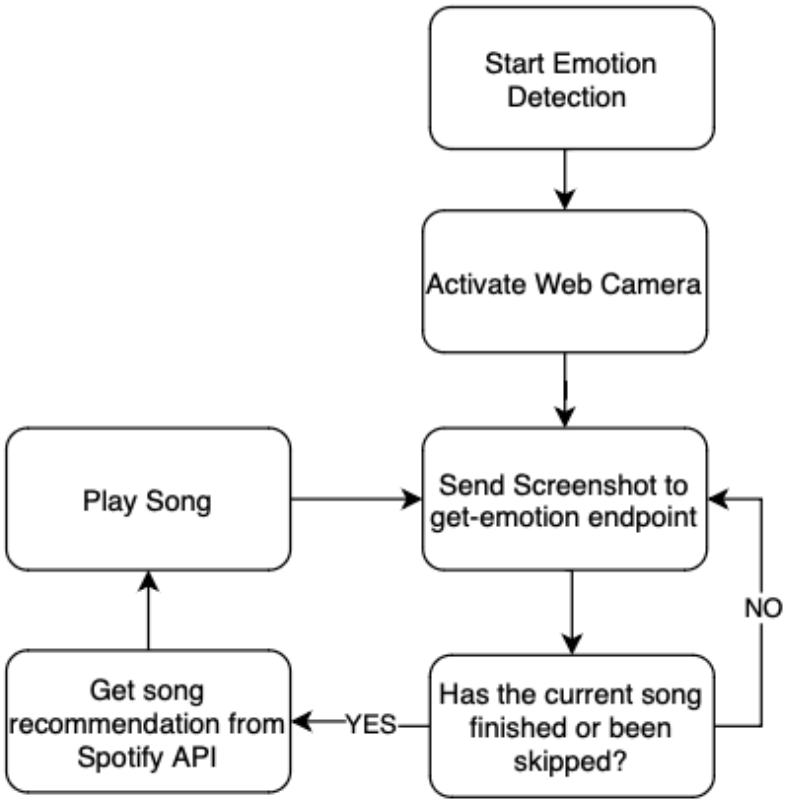


Figure 4.8: Music Recommendation Flow

To start the process, the user must start the live emotion detection, they will then be prompted to allow the application to use their web-cam. The website has to ask for user's permission to access resources outside of the browser in order to protect the privacy of the user. Once they have accepted, the live emotion detection will begin.

Every half second the application will send a screen shot of the web-cam feed to the get-emotion endpoint in a GET request. The endpoint will return an emotion which will be displayed for the user to see. If no song is currently playing or a current song has ended, the application will look at the current emotion and send the music preferences for that emotion to the Spotify get song recommendation. If no face is being detected, the application will use the last detected emotion. The application will then play the recommended song and the cycle repeats until the user turns off the live emotion detection.

It may be the case the user does not or cannot use the camera for live-emotion detection. If this is the case, they can manually select an emotion and music will start playing. This does not benefit from any live emotion detection however, they still benefit from personalised music recommendations for that chosen mood.

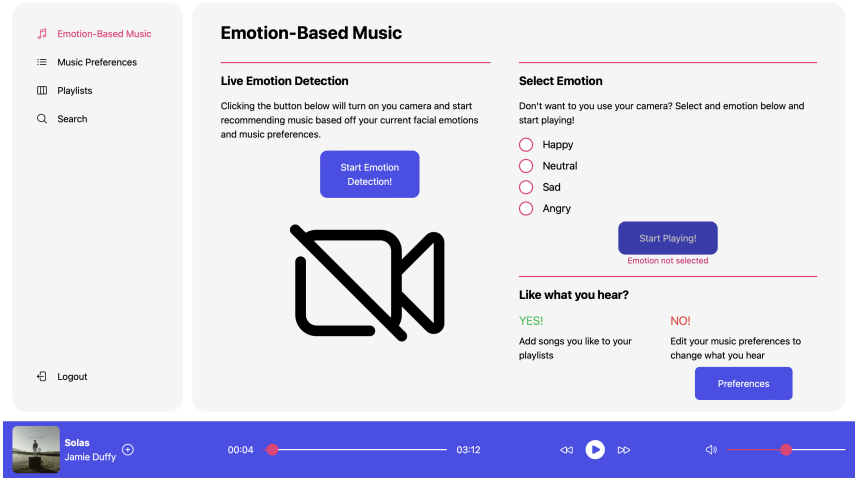


Figure 4.9: Music Recommendation UI

Playlists

When trying to fulfil the goal of creating a music platform with a wide range of functionality, a playlist feature was identified to allow users save songs and curate custom listening experiences. To create the playlist feature, we will use the Spotify API because it provides all the functionality we will need to manage playlists.

The API allows us to display all the users playlists and the contents of the playlists (figure 4.10) and perform actions such as removing a song from a playlist or creating a playlist. A user is also able to listen to the contents of the playlist in order or by turning on the shuffle which will randomise the order of the music. In addition to creating a empty playlist, the user has the option of generating a playlist based off their music preferences as shown in 4.11. Through the evaluation of the application conducted at the end, it was identified that users would want a playlist generation feature to get around the difficult and laborious task of creating playlists. This feature works by prompting the user to input 5 songs/artists/genres and getting a complete playlist with up to 100 songs in return. This feature works very similarly to the music recommendation feature where we send some music preferences to the recommend song Spotify API and receive a song to play, but in this case, we return a range of songs which will be added to a playlist.

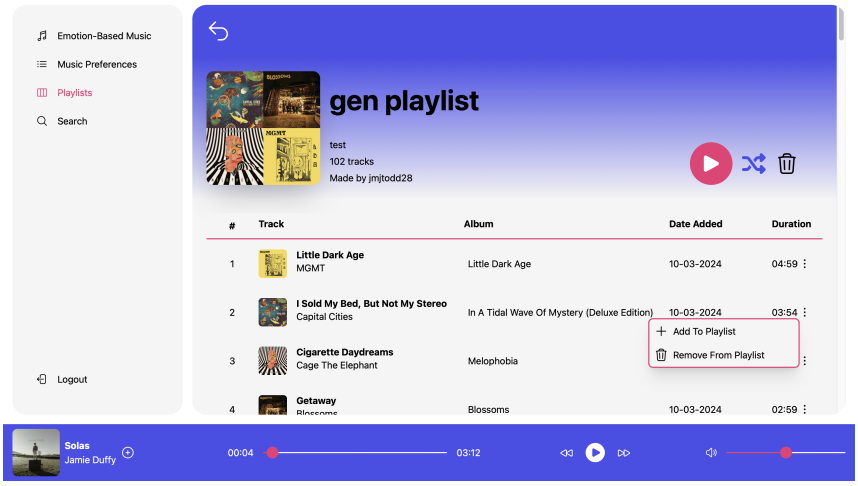


Figure 4.10: Playlist View UI

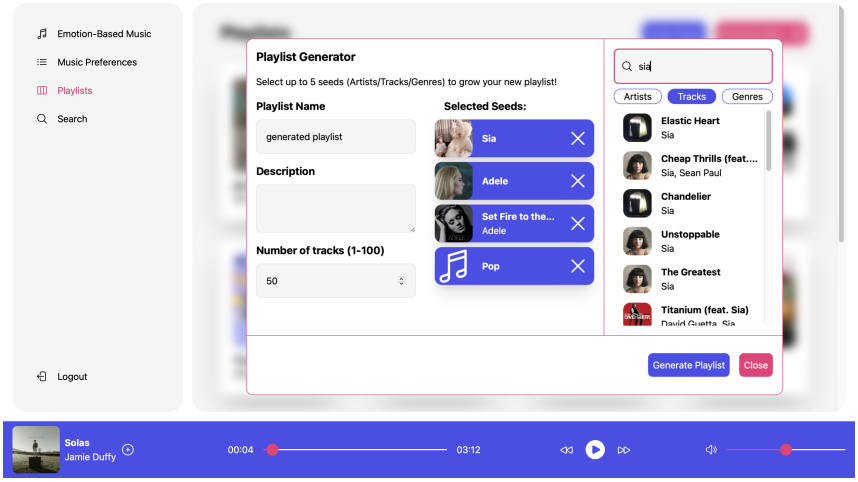


Figure 4.11: Generate Playlist UI

Search Feature

Adding upon the functionality of the application, it was identified a search feature was needed so users can directly access and play any song or view any artist. This also uses the Spotify search API, to get relevant search results. The search endpoint offers seven search types, album, artist, playlist, track, show, episode, audiobook. For the functionality the application needs, users can only search for tracks, playlists and artists. Users can search for a song to either play or add to an existing playlist. When a user clicks on a searched artist, they are met with an artist page containing their top songs and albums which can both be played. This shows the strength of the API since this is all information that would be impossible to obtain for such a range of artists and songs without the use of the API. There is also the option to allow users to search other people's playlists which they can add to their own libraries. Similarly to the playlist generation feature, this helps circumvent the potential difficulties in creating playlists. The main search view UI can be seen in 4.12

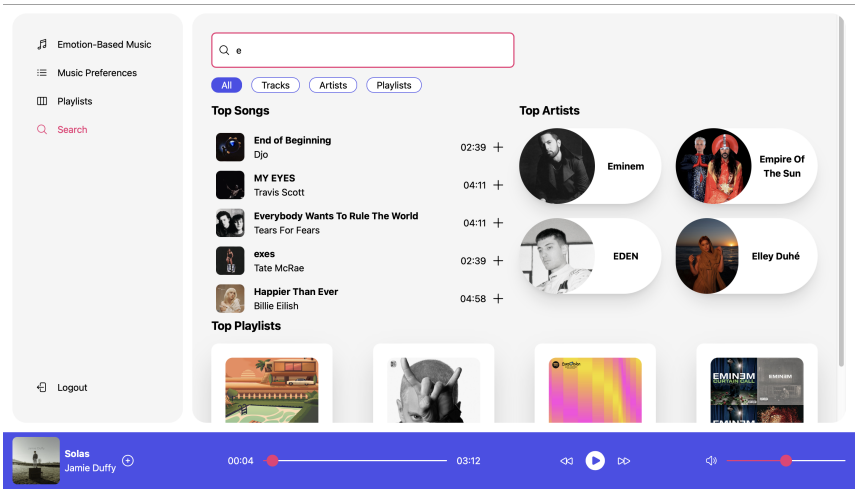


Figure 4.12: Main Search UI

CHAPTER 5

Evaluation

After conducting a literature review and a user study, a set of requirements were created. This section will evaluate how successfully those requirements have been fulfilled after the implementation process. This will be done by evaluating the machine learning models, which of the original functional requirements have been met and a user study to see how people responded to the application.

5.1 Model Evaluation

Quantitative Analysis

To evaluate which model was most suitable for the web application we will explore how the two models compare when comparing metrics such as evaluation accuracy, loss and precision and then a physical test of the two models to detect facial emotions.

Figures 5.1 and 5.2 both show the training of the neural networks. Both graphs show the loss and the accuracy of the models for both the training and validation sets. It should be noted that the ck+ model ran for far more epochs than the ck+ model because of the data sizes. Every epoch, training does not take place on all the data but a specified batch of the data, which for the FER dataset is a larger quantity leading to longer training times. Both models were trained until there was no noticeable improvements to the validation accuracy or the validation loss. The graph for the ck+ model is also more erratic than the Fer-2013 model, this could be caused by the more limiting nature of the ck+ dataset. One of the downsides of the ck+ dataset is its lack of size and variation, which could lead to over-fitting as the model learns specific patterns in the training set. This consequently results in limited generalisation leading to far greater fluctuations in performance. We observe the ck+ model achieves a evaluation accuracy of 0.95 and a loss of 0.15 whilst the FER-2013 model obtained an accuracy of 0.63 and a loss of 0.92. As discussed before we would expect a better performance for the ck+, but both set of accuracy's are typical of those datasets as seen in the literature review.

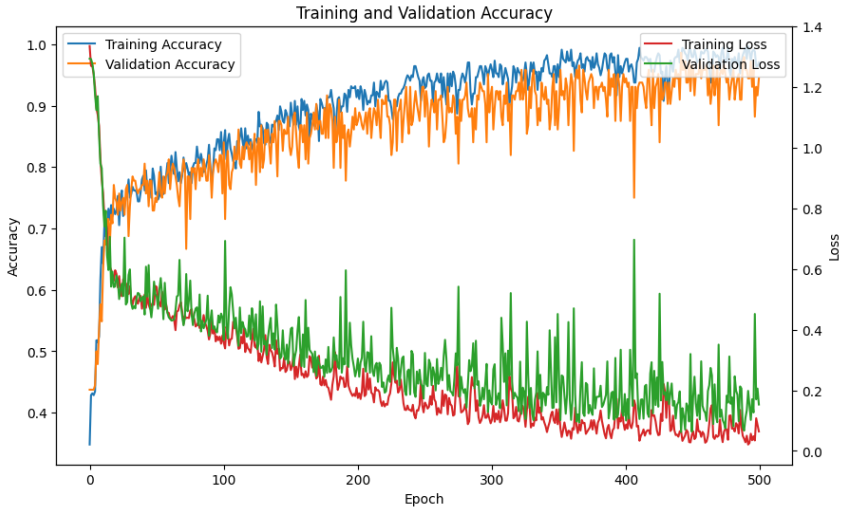


Figure 5.1: Training ck+ model

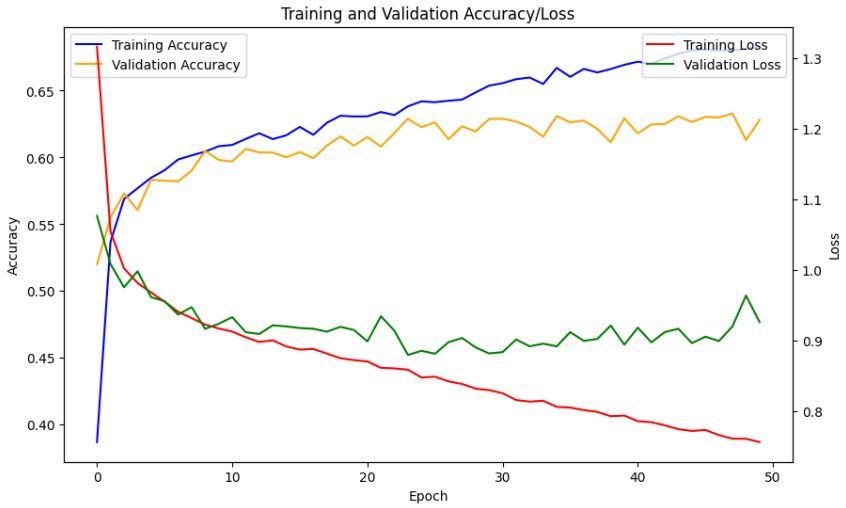
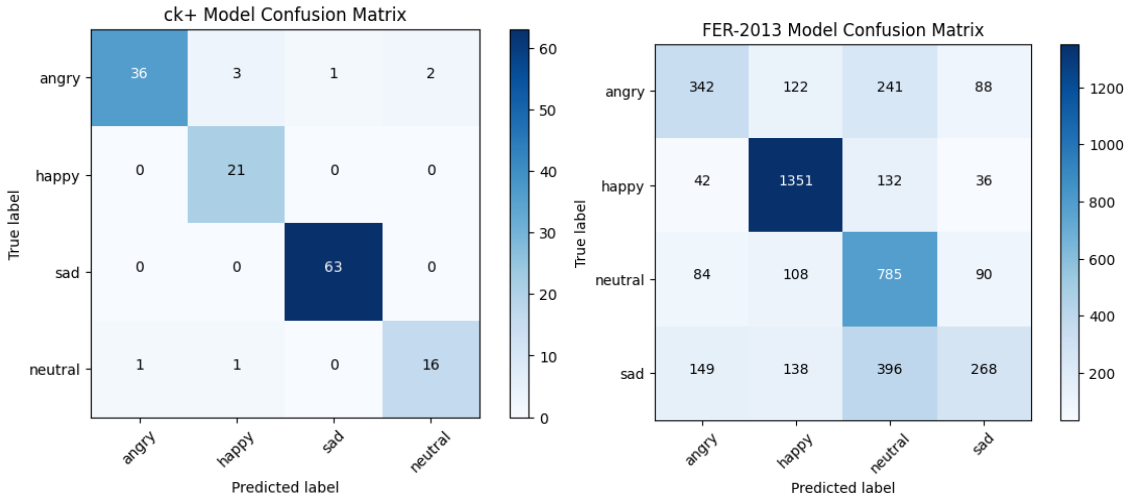


Figure 5.2: Training FER-2013 model

To calculate metrics such as precision, recall and F1-score we must first create a confusion matrix that will supply all the information we need. A confusion matrix used to show true positives, true negatives, false positives and false negatives for any classification model. Figure 5.3 shows the two confusion matrices of the two different models.



((a)) ck+ Confusion Matrix ((b)) Fer-2013 Confusion Matrix

Figure 5.3: Comparison of Confusion Matrix

It is important to evaluate metrics other than accuracy and loss as they may not give the complete picture of the model's performance. Here are the different metrics that will used to evaluate the models using the confusion matrix:

- **Precision** - This is a measure of the accuracy for positives predictions, it is simply the ratio of true positives to the sum of true positives and false positives. It helps provide insight into how the model is able to make accurate positives predictions for each emotion.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** - This metric evaluates the models ability to capture all positive instances in a dataset and can represent the sensitivity to positive instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1-score** - This is the harmonic mean of precision and recall, it is a measure that takes into consideration both the precision and recall. This balances the trade-offs between the precision and the recall and combines them into a single value which can help gain insight to the overall performance of the model.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Support** - this metric corresponds to the number of occurrences of each emotion in the dataset, it helps to provide some context and understanding to the metrics calculated.

Emotion Label	**Precision**		**Recall**		**F1-Score**		**Support**	
	Fer	**ck+**	**Fer**	**ck+**	**Fer**	**ck+**	**Fer**	**ck+**
Angry	0.55	0.97	0.43	0.86	0.49	0.91	793	42
Happy	0.79	0.84	0.87	1.00	0.82	0.91	1561	21
Neutral	0.51	0.89	0.74	0.89	0.60	0.89	1067	18
Sad	0.56	0.98	0.28	1.00	0.37	0.99	951	63
Accuracy	-	-	-	-	-	-	4372	144
Macro Avg	0.60	0.92	0.58	0.94	0.57	0.93	4372	144
Weighted Avg	0.63	0.95	0.63	0.94	0.61	0.94	4372	144

Table 5.1: Performance Metrics for Models

When evaluating these metrics, we learn a few things about the models. Following on from the pattern of the ck+ model having better evaluation accuracy and loss due to the nature of the dataset, this is reinforced here as all the precision, recall and F1-scores are better than the FER-2013 dataset. Both models can correctly identify happiness easier than the other emotions, this can be attributed to the happiness facial emotion having the most distinguishable features compared to the other emotions.

The FER-2013 model displays a poor ability to detect the sadness emotion as shown by the recall and the F1-score. Looking at the confusion matrix it is clear there are a lot of false positives where the model predicts neutral for the sad samples. This could be attributed to the fact sadness has a lot of shared characteristics with the neutral emotion, sadness can often look like or similar to a neutral expression. The reason the ck+ model does not suffer from this problem might again be due to the nature of the dataset. When users are prompted to make a facial emotion, they may display the most extreme case of that emotion which may not be natural, hence in the ck+ dataset each emotion has more clear separation from each other. This is not the case as much in the FER-2013 dataset where the conditions are less controlled, and the expression can be perceived as more natural.

Qualitative Analysis

This section will evaluate whether the better generalisation properties of the FER-2013 model will be of greater use than the high performance ck+ model in a controlled setting. To achieve this, I created a Python program that will use the laptop camera to mimic how the models will be used in the application. Using the CV2 library, we can capture the webcam feed and for every frame we can crop the frame so it is just the face and extract the facial landmark coordinates using MediaPipe. This is the same process used in the data extraction step, but we are doing it for all the frames in a live video feed. We can then pass the coordinates into the model and display the predicted emotion.

During these tests it was clear to see the FER-2013 model performed the best as it was able to distinguish between emotions and in particular the subtle emotions more successfully detected than the ck+ model. Something that has not really been considered is the potential body language of a person for every emotion, for example if you are sad, you may be more likely to be looking down. This is where the poor generalisation properties of the ck+ model showed as it struggled with different face orientations whilst the FER-2013 model was more likely to make a correct facial emotion detection for different face angles relative to the camera. It was observed that the FER-2013 model did sometimes miscategorise sad as neutral, but it was not as prevalent as the metrics may initially indicate. The FER-2013 model also generalised better for a wider range of emotions, for example it can detect a subtle smile as happiness whilst the ck+ model struggled to make such observations. Following from this analysis it was decided that the FER-2013 model will be used for the emotion detection in the application.

5.2 Technical

This section will discuss the functional requirements which have been identified in section 3.2.1. Using the MoSCoW method proved effective because it helped identify what the application needed resulting in a clear vision of what work needs to be done. All the "*could have*" and "*should have*" requirements were met meaning the application fulfilled all the basic functionality it set out to satisfy. R.7, R.15, and R.26 "*could have*" requirements have not been implemented due to time restraints. Regarding the non-functional requirements, these have been evaluated via user feedback and will be explored more when analysing the results of the user feedback in the next section.

5.3 User Feedback

It is important to gather user feedback in the evaluation process to verify whether the application was successful in providing a good listening experience. Users can help identify strengths and weaknesses for future improvements on the application. This was done letting people try the application in person and then fill out a questionnaire where they will give their feedback.

The general structure of the survey was to go through the application feature by feature and ask some generic questions such as "was this feature aesthetically pleasing" and some more feature specific questions such as "Did the real time emotion analysis improve your listening experience?". Users would have the option to answer between a range of 1-5. There would also be an optional text field they could fill in to give additional feedback. At the end of the survey the users were also asked some general application questions such as "What would you rate the overall experience of the app?". A total of 8 people used the application and filled in the questionnaire, with the subjects coming from a mixture of University of Birmingham (UoB) students and non-UoB students.

Feedback was generally very positive towards the emotion-based music recommendation feature. Users gave the emotion detection an average of 4.6/5 and were generally impressed with its ability to detect even subtle emotion changes. A complaint regarding the emotion detection was the application sometimes struggled to pick up the sad emotion and wrongly labelled it as the neutral emotion. This behaviour was potentially expected as this was a potential weakness identified during the original model evaluation step. Following on from the detection users felt the music recommendations were suitable given their music preferences and the current mood giving this a 4.75/5. Overall users believed that the real time emotion analysis benefited their listening experience with an average score of 4.75/5. One criticism is the potential lack of flexibility the feature had, some users would also like the option to assign playlists to a certain emotion instead of having music recommended for them so they can control exactly what is played.

When discussing whether the other features helped the overall experience, for every feature users were asked "Does this feature help improve the experience of the overall app?". Generally, users enjoyed the other features with the playlist feature scoring an average 4.75/5, the search feature scoring an average 4.5/5 and the music preferences scoring an average 4.6/5. This was good to see as one of the original goals was to provide a complete music platform with good functionality. An area for improvement was identified in the playlist feature. Users enjoyed giving music preferences and having appropriate music being played back at them, and they wondered if this could be used to implement a playlist generation feature. Creating playlists can be a laborious task and by using the Spotify API get song recommendation we were able to implement a playlist generation feature after this feedback. Unfortunately, due to time restraints there was no additional user feedback for this new feature.

Assessing how easy the application was to navigate and how aesthetically pleasing, the interface for inputting music preferences needed to be improved. This interface scored an average 3/5 for looks and usability. The main complaint was the lack of signposting leaving users confused what that page was for initially leading to some confusion. Using this feedback some amendments have been made to try and make it clearer what the feature is for but similarly to the playlist feature generation there is no feedback to whether this was successful. The music controller at the bottom of the screen scored the highest with a score of 4.875/5 regarding to its usability and aesthetic. Overall, the application fulfilled the non-functional requirement of being easy to use and aesthetically pleasing.

Overall, the users rated the overall experience of the application an average of 4.625/5 which is very positive. Users gave feedback that helped with making short term amendments and creating long-term goals for the application. There was also the possibility to report any bugs in the application which was also useful in identifying issues in the application that could be fixed to further better the user experience for the future. There were some concerns about the safety of the application moving forwards with respect to the live emotion detection. Some users said they would potentially feel uncomfortable having their laptop camera on for extended periods of time due to privacy concerns. They trusted that this project was not malicious with the use of their web

cams, however they would not feel the same if they did not know who created the web site or could not potentially check the code for malice. Also, whilst users liked the facial emotion detection, they felt like it is a bit limiting, as the app could only assess emotions if you are looking at the camera, otherwise there is no emotion detection. Getting past this hurdle would involve other methods of capturing user's emotions which do not use the camera.

CHAPTER 6

Conclusion

The goal of this project was to create an application which connected users with music through the use of emotions. During the research phase, it was identified that there is a lack of music systems which can provide live emotion detection to adapt with the music needs of the user. To tackle this problem, a web application was built that tried to emulate some of the key functionality of existing popular, music platforms with an emotion-based music recommendation feature. To do this facial emotion detection, a neural network was developed to classify between the emotions. When evaluating the results of the project, users enjoyed this concept of using live facial emotion detection to recommend music and the feedback was positive. Overall, I believe this project was successful in producing an application which provided a more meaningful and personalised listening experience for users using emotion analysis.

Looking forward beyond the scope of this project there are some ways in which the application can be improved. Fulfilling all the requirements that were not met during the implementation step is something that could potentially benefit the quality of the application and getting feedback on newly implemented features. It is also always beneficial to get feedback from a wider range of people. This could be achieved by fully deploying the application onto the internet, allowing users to try the application through their own devices.

This project used facial emotion detection, but this may not suit everybody as identified in the feedback. Instead, you could perform actions such as speech and text analysis with the help of natural language processing techniques, evaluating the user's activity through the use of a smart watch or even how they interact with the keyboard and mouse are all ways to potentially evaluate a user's emotions. Regarding the facial emotion model used, this will always be a place for improvement, with the goal of having a more effective classification model which would lead to a better user experience. This could be done by exploring other classification techniques or look to optimise the methodology used in this project. This could be achieved by changing the way we do the data extraction, tuning of hyper-parameters, model architecture and using any new datasets that are released in the future.

References

[1] M. Athavle, D. Mudale, U. Shrivastav, and M. Gupta. Music recommendation based on face emotion recognition. *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, 2(2):1–11, 2021.

[2] J. Chen, D. Chen, Y. Gong, M. Yu, K. Zhang, and L. Wang. Facial expression recognition using geometric and appearance features. In *Proceedings of the 4th international conference on internet multimedia computing and service*, pages 29–33, September 2012.

[3] S. M. Florence and M. Uma. Emotional detection and music recommendation system based on user facial expression. In *IOP conference series: Materials science and engineering*, volume 912, page 062007. IOP Publishing, 2020.

[4] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D. H. Lee, and Y. Zhou. Challenges in representation learning: A report on three machine learning contests. In *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013, Proceedings, Part III*, volume 20, pages 117–124. Springer Berlin Heidelberg, 2013.

[5] S. L. Happy and A. Routray. Automatic facial expression recognition using features of salient facial patches. *IEEE transactions on Affective Computing*, 6(1):1–12, 2014.

[6] C. Hewitt and H. Gunes. Cnn-based facial affect analysis on mobile devices. *arXiv preprint arXiv:1807.08775*, 2018.

[7] M. Hu, H. Wang, X. Wang, J. Yang, and R. Wang. Video facial emotion recognition based on local enhanced motion history image and cnn-ctslstm networks. *Journal of Visual Communication and Image Representation*, 59:176–185, 2019.

[8] A. Jaiswal, A. K. Raju, and S. Deb. Facial emotion detection using deep learning. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–5. IEEE, June 2020.

[9] T. Kanade, J. F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *Proceedings fourth IEEE international conference on automatic face and gesture recognition (cat. No. PR00580)*, pages 46–53. IEEE, March 2000.

[10] A. d. T. Lopes, E. De Aguiar, A. De Souza, and T. Oliveira-Santos. Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. *Pattern recognition*, 61:610–628, 2017.

[11] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE, 2010.

[12] Mediapipe. https://mediapipe-studio.webapps.google.com/studio/demo/face_landmarker. Accessed: November 19th, 2023.

[13] C. S. Pereira, J. Teixeira, P. Figueiredo, J. Xavier, S. L. Castro, and E. Brattico. Music and emotions in the brain: familiarity matters. *PloS one*, 6(11):e27241, 2011.

[14] E. Pranav, S. Kamal, C. S. Chandran, and M. H. Supriya. Facial emotion recognition using deep convolutional neural network. In *2020 6th International conference on advanced computing and communication Systems (ICACCS)*, pages 317–320. IEEE, March 2020.

[15] P. J. Rentfrow. The role of music in everyday life: Current directions in the social psychology of music. *Social and personality psychology compass*, 6(5):402–416, 2012.

[16] Spotify. https://www.spotify.com. Accessed: November 20th, 2023.

[17] M. T. Susan Hallam, Ian Cross. *The Oxford Handbook of Music Phychology*. Oxford University Press, 2009.

[18] Tensorflow. https://www.tensorflow.org. Accessed: November 19th, 2023.

[19] Y. Q. Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4:128–148, 2014.

Appendix A: Additional UI

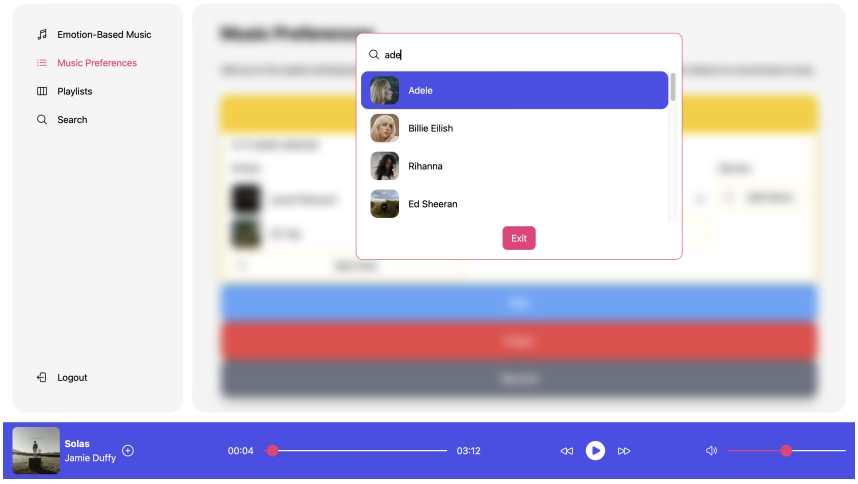


Figure 1: Artist Search in Music Preferences

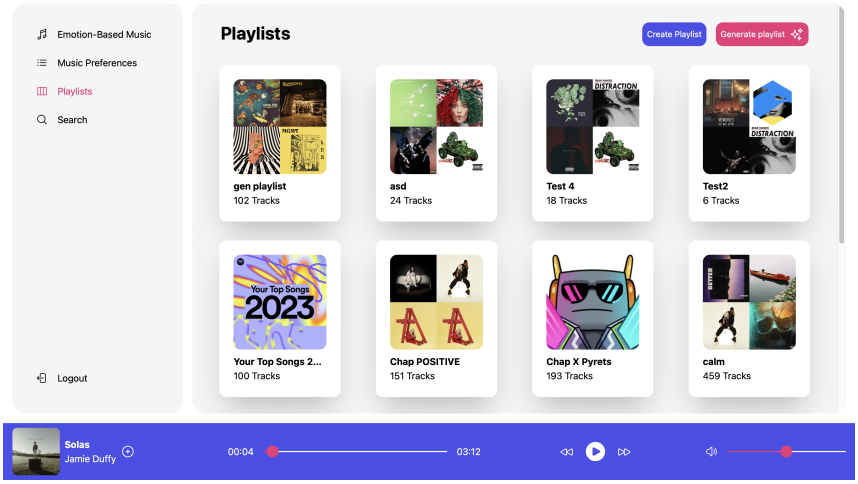


Figure 2: Playlists View

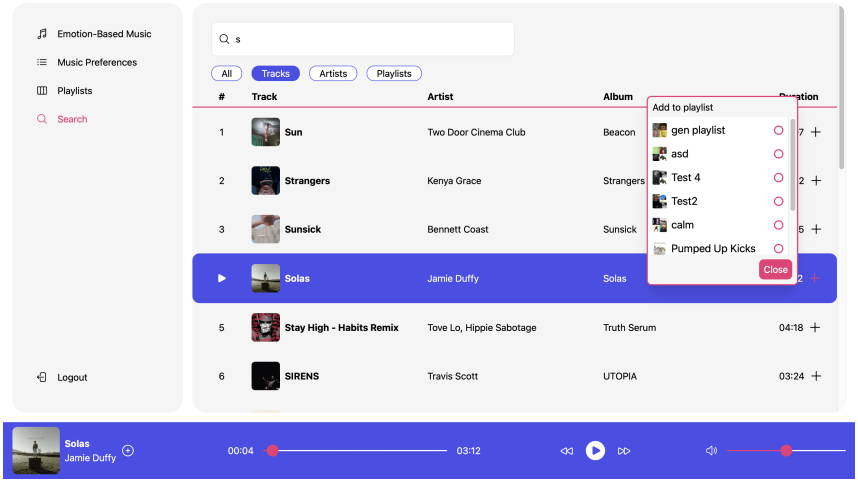


Figure 3: Track Search

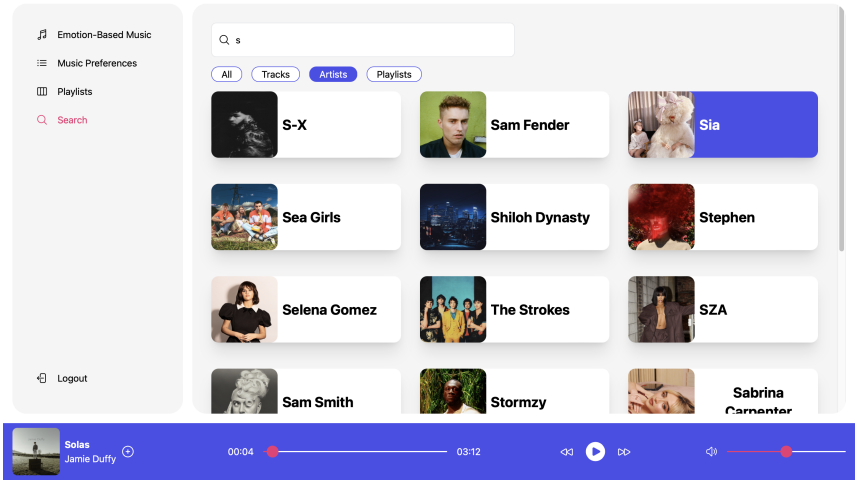


Figure 4: Artist Search

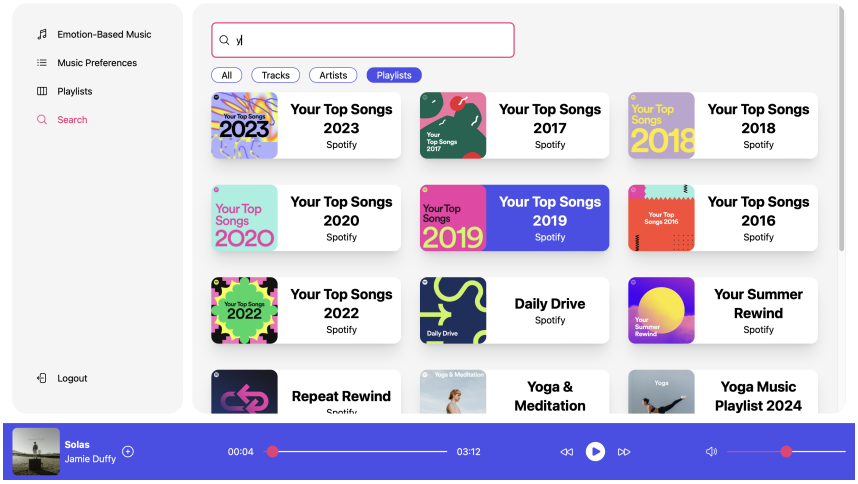


Figure 5: Playlist Search