Emotion Based Music Player

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University - Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Sri Charan Nimmagadda
Summer 2017

Committee Members

Dr. Ajay K. Katangur
Committee Chairperson

Dr. Dulal C. Kar
Committee Member
ABSTRACT

Recent studies confirm that humans respond and react to music and that music has a high impact on person’s brain activity. The average American listens up to four hours of music every day [2]. People tend to listen to music based on their mood and interests. This project focuses on creating an application to suggest songs for user based on their mood by capturing facial expressions. Facial expression is a form of nonverbal communication. Computer vision is an interdisciplinary field that helps convey a high-level understanding of digital images or videos to computers. In this system, computer vision [18] components are used to determine the user’s emotion through facial expressions. Once the emotion is recognized, the system suggests a play-list for that emotion, saving a lot of time for a user over selecting and playing songs manually. Emotion-Based Music Player also keeps track of user’s details like number of plays for each song, sorts songs based on category and interest level, and reorganizes the play-list every time. The system also notifies user about the songs that are never played so that they can be deleted or modified.

Keywords: Emotion recognition, Computer vision, Camera, Music, Categorization, recommendations.
TABLE OF CONTENTS

Abstract ......................................................................................... ii
Table of Contents ................................................................. iii
List of Figures ............................................................................. v
1 BACKGROUND AND RATIONALE ............................................. 1
   1.1 Related Work ................................................................. 1
2 Project Overview ................................................................. 3
   2.1 Problem Statement ......................................................... 3
   2.2 Motivation .................................................................. 3
   2.3 Product Description ..................................................... 4
   2.4 Product Scope ............................................................... 4
   2.5 Functionality ................................................................. 5
   2.6 System Requirements ................................................... 5
   2.7 Existing System ............................................................ 6
3 SYSTEM ARCHITECTURE AND DESIGN ............................... 7
   3.1 System Overview ........................................................... 9
4 IMPLEMENTATION ............................................................. 11
   4.1 MongoDB ................................................................. 11
   4.2 Express.js ................................................................. 12
   4.3 Angular.js ................................................................. 12
   4.4 Node.js ................................................................. 13
   4.5 Cordova ................................................................. 13
   4.6 Affectiva SDK ............................................................. 13
4.7 Firebase ................................................................. 13

4.8 System Flow ........................................................... 14
   4.8.1 User Login and Sign-up screen .............................. 14
   4.8.2 Loading of SDK ................................................. 16
   4.8.3 Emotion Capture Phase ....................................... 18
   4.8.4 EMO-Phase .................................................... 20

4.9 EMO-algorithm Scenarios .......................................... 24
   4.9.1 EMO-algorithm Scenario 1 .................................. 24
   4.9.2 EMO-algorithm Scenario 2 .................................. 26
   4.9.3 EMO-algorithm Scenario 3 .................................. 27
   4.9.4 EMO-algorithm Scenario 4 .................................. 28
   4.9.5 Adding Song .................................................. 29
   4.9.6 Saving Song .................................................. 34

4.10 Recommendations ................................................ 35

5 TESTING AND EVALUATIONS ........................................ 36

5.1 Login test .......................................................... 36

5.2 Test Case for Capture and Display Play-list based on Emotion . 38

5.3 Test Case for Adding or Editing a Song ........................ 40

5.4 Test Case Scenarios for EMO-algorithm ....................... 41
   5.4.1 EMO-algorithm Test Case Scenario 1 .................... 41
   5.4.2 EMO-algorithm Test Case Scenario 2 .................... 42
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>System Architecture of Emotion-Based Music Player</td>
<td>7</td>
</tr>
<tr>
<td>3.2</td>
<td>System Flow of Emotion-Based Music Player</td>
<td>9</td>
</tr>
<tr>
<td>4.3</td>
<td>Mean Stack Flow</td>
<td>11</td>
</tr>
<tr>
<td>4.4</td>
<td>Two way data binding in Angular</td>
<td>12</td>
</tr>
<tr>
<td>4.5</td>
<td>Signup</td>
<td>15</td>
</tr>
<tr>
<td>4.6</td>
<td>Login</td>
<td>16</td>
</tr>
<tr>
<td>4.7</td>
<td>loading SDK</td>
<td>17</td>
</tr>
<tr>
<td>4.8</td>
<td>Starting the detector</td>
<td>18</td>
</tr>
<tr>
<td>4.9</td>
<td>Emotion Feedback to user</td>
<td>20</td>
</tr>
<tr>
<td>4.10</td>
<td>Song-List for Happy emotion</td>
<td>23</td>
</tr>
<tr>
<td>4.11</td>
<td>EMO-algorithm scenario 1</td>
<td>24</td>
</tr>
<tr>
<td>4.12</td>
<td>EMO-algorithm scenario 2</td>
<td>26</td>
</tr>
<tr>
<td>4.13</td>
<td>EMO-algorithm scenario 3</td>
<td>27</td>
</tr>
<tr>
<td>4.14</td>
<td>EMO-algorithm scenario 4</td>
<td>28</td>
</tr>
<tr>
<td>4.15</td>
<td>Add songs</td>
<td>29</td>
</tr>
<tr>
<td>4.16</td>
<td>Song Upload from local files</td>
<td>30</td>
</tr>
<tr>
<td>4.17</td>
<td>User’s input for category</td>
<td>31</td>
</tr>
<tr>
<td>4.18</td>
<td>User’s input for Interest</td>
<td>33</td>
</tr>
<tr>
<td>4.19</td>
<td>Saving song on cloud</td>
<td>34</td>
</tr>
<tr>
<td>4.20</td>
<td>Recommendations for User</td>
<td>35</td>
</tr>
</tbody>
</table>
5.21 Wrong Email Format .............................................. 36
5.22 Authentication error .............................................. 37
5.23 Sad emotion detected ............................................. 38
5.24 Sad emotion paly-list detected ................................. 39
5.25 Category and Interest level are required ..................... 40
5.26 EMO-algorithm test case scenario 1 .......................... 41
5.27 EMO-algorithm test case scenario 2 .......................... 42
5.28 EMO-algorithm test case scenario 3 .......................... 43
1. BACKGROUND AND RATIONALE

Music is an important entertainment medium. With advancement of technology, the optimization of manual work has gained a lot of attention. Currently, there are many traditional music players that require songs to be manually selected and organized. User, have to create and update play-list for each mood, which is time consuming. Some of the music players have advanced features like providing lyrics and recommending similar songs based on the singer or genre. Although some of these features are enjoyable for user, there is room to improve in the field of automation when it comes to music players. Selecting songs automatically and organizing these based on the user’s mood gives user’s a better experience. This can be accomplished through the system reacting to the user’s emotion, saving time that would have been spent entering information manually.

Emotions can be expressed through gestures, speech, facial expressions, etc. For the system to understand a user’s mood, we use facial expression [1]. Using the mobile device’s camera, we can capture the user’s facial expression. There are many emotion recognition systems which take captured image as input and determine the emotion. For this application, we are using Affectiva SDK [7] for recognition of emotion. The system includes a novel algorithm [EMO-algorithm] that organizes songs based on the user’s emotions and preferences. This algorithm suggests user’s songs to play based on their emotion.

1.1 Related Work

Currently, there are no dedicated applications to suggest songs based on emotion of music listeners. There are also very few applications that focus on the user preferences and recommendations, and these are not customizable, like AllMusic [3].
Other applications suggest predefined (not user-specific) song play-lists. Applications like moodfuse [12] include features like manual selection of songs, partly shuffle, play-list. Some popular music applications like Saavn [16], Spotify provide users defined play-lists that need to be created and updated manually. All of these applications focus on general categorization rather than specificity to every user. A dedicated application that focuses more on user preferences, priorities, and the creation of dynamic play-list is required to optimize the user experience. It should contain user specific play-list generated based on usage and should be efficient in categorization.

Many widely used facial expression categorization techniques like Viola and Jones [19] etc., can be used for initial phase to capture and determine the user’s emotion, but these techniques have high computational requirement. The alternative is to use a cloud-based web service [7] that processes computation in the cloud. The current system uses, Affectiva SDK for emotion recognition, a system that has already analyzed emotions from over five million faces. This SDK helps the application capture and determine emotion from a picture. Later, this emotion can be used to organization of user’s play-list.
2. PROJECT OVERVIEW

2.1 Problem Statement

Music listeners have tough time creating and segregating the play-list manually when they have hundreds of songs. It is also difficult to keep track of all the songs: sometimes songs that are added and never used, wasting a lot of device memory and forcing the user to find and delete songs manually. User’s have to manually select songs every time based on interest and mood. User’s also have difficulty to re-organize and playing music when play-style varies. Currently in existing application, music is organized using play-list, and play-list songs cannot be modified or altered in one click. User’s have to manually change or update each song in their play-list every time. The sequence of songs in a play-list might not be the same every time, and songs that a user wants to listen frequently might not be given priority or might be left out from the list. Currently, there are no applications that allows users to play songs on-the-go without selecting songs manually or from a play-list.

2.2 Motivation

As a music lover, I’ve always felt that music players should do far more things than just playing songs and allowing users to create play-lists. A music player should be intelligent and act according to user’s preferences. A music player should help users organize and play the songs automatically without putting much effort into selection and re-organization of songs. The Emotion-Based Music Player provides a better platform to all the music listeners, and ensures automation of song selection and periodic updating of play-lists. This helps users organize and play songs based on their moods. The player should also give recommendation for users to change songs on-the-go. It calculates song-weight based on EMO-algorithm (discussed in chapter
4) to help users have more customized and organized play-lists.

2.3 Product Description

The Emotion Based Music player requires the user to have a profile to access the application. The user needs to grant permissions for the application to access the device’s camera and media. The application allows users to upload songs and give feedback on the song. Emotion-Based Music Player saves the user profile on the device and keeps the profile logged-in until user logs out of the device manually. As soon as the user opens the application, the device’s camera opens and begin capturing images. The system will determine emotions and create play-lists for the user based on emotion captured. The application also allows user’s to easily customize the play-lists. It recommends songs for the user that may fit their current emotion, helping the user automate the initial song selection. The recommendations are based on the previous information about the user’s preferences and usage.

2.4 Product Scope

Emotion Based Music player is a useful application for music listeners with a smart phone and an Internet connection. The application is accessible by anyone who creates a profile on the system. The application is designed to meet the following needs of the users as described below

1. Creating an account or signing up, signing in

2. Adding songs

3. Removing songs

4. Updating songs
5. Personalized play-list

6. Recommendations

7. Capturing emotion using camera

2.5 Functionality

The main functionality of this application is to give users more features to automate the selection and organization of the songs based on user preferences. Initially, the user uploads each song and gives details about the song, like category and interest level. Songs are listed and categorized based on user’s interest. The Emotion-Based Music Player gives user’s feedback about their usage. It also notifies user’s of songs that do not fit in the categories listed earlier.

2.6 System Requirements

The following are the minimum requirements to develop this application

1. Hardware requirements

   • Processor: 2 GHz
   • RAM: 1 GB

2. Browser

   • Chrome 51 or higher
   • Firefox 47 or higher
   • Opera 37
   • Edge 10586
3. Database

- Firebase
- NoSQL

4. API : Affective Emotion Recognition API

2.7 Existing System

Currently, there are many existing music player applications. Some of the interesting applications among them are:

- Saavan and Spotify – These application [16] gives good user accessibility features to play songs and recommends user with other songs of similar genre

- Moodfuse - In this application [12], user should manually enter mood and genre that wants to be heard and moodfuse recommends the songs-list

- Steromood - User should select his mood manually by selecting the moods from the list and the application [17] plays music from YouTube

- Musicover - This application [3] has High quality songs and comprehensive music recommendations. It also suggest predefined play-list for the user.
3. SYSTEM ARCHITECTURE AND DESIGN

The system architecture of Emotion-Based Music player is shown in Fig 3.1. The application is built using the architectural pattern of Model-View-Controller [13]. It is also widely used architecture. Here, the application is divided into three main logical components: the model, the view and the controller.

- **View**: The top layer is where the end-user communicates with the application through clicking buttons, typing details, accessing camera, selecting radio button, uploading songs, etc. This layer is responsible for displaying all data or a
portion of data to user based on the requirement of the application. This layer also acts as a bridge between the user and application itself. Angular 4 [4] is used in this application for displaying the output or response of the system to the user.

- Controller: This middle layer of the application contains the business logic, and the main functionality of the application. As soon as the user interacts with the application, the response is processed in this layer. From log-in to displaying play-list, all the functions that run in background belong to this layer. This mainly consists of all the functions and EMO-algorithm (discussed in later sections) which helps in segregating songs and sending output to view layer.

- Model: This layer is responsible for maintaining the user’s data. Emotion-Based Music Player uses Google Firebase [8] for storing user data. Firebase is very useful to maintain user’s profiles and preferences. The application also stores some temporary data on the device.

A MEAN [10] stack approach is used in the development of Emotion-Based Music Player. Here, the MEAN stack follows MVC architecture, with everything written in JavaScript [9]. Both the server-side and client-side executions are performed using JavaScript to increase the speed of the application.
3.1 System Overview

This section illustrates the design and functional phase of the application. Emotion-Based Music Player is installed on a mobile device, where the user can access their customized play-lists and play songs based on their emotions. Fig 3.2 depicts the overview of the application.

1. Login/signUp phase: Users have to create a profile in order to store personal date. If the user already has an account, they can log-in to their account to access customized play-lists as well as songs. Once user logs-in, their profile is saved on the application, until they manually log-out. While the user adds songs, their input (i.e. category and interest level) is taken by the system.
2. **Emotion Capture phase:** As soon as the authentication phase is done, the application will ask user’s permission to access media and photos and will access camera to capture the user’s image.

3. **Affdex API:** After the image is captured, the application sends image captured to Affdex SDK [7]. There, the captured image is processed and the image feedback is sent to the application.

4. **Emo-phase:** In this phase, the application receives the image information and recognizes the emotion based on the defined threshold. This emotion is sent to the database to fetch the emotion play-list.

5. **Display phase:** Here, the songs are organized based on EMO-algorithm and the user can play any song from the list displayed. The user has the option to add, remove, modify the songs and also can change category and interest level of a song at anytime in the application. The application also has a recommendation tab where the system notifies the user of songs that are rarely played.
4. IMPLEMENTATION

The system uses the following technologies to develop Emotion-Based Music Player. A MEAN [10] stack is used in the development of the Emotion-Based Music Player, with everything written in JavaScript, where M stands for MongoDB, E stands for Express.js, A stands for Angular, N stands for Node.js.

![Mean Stack Flow](image)

Fig. 4.3. Mean Stack Flow

4.1 MongoDB

MongoDB [11] is a free and open-source cross platform document-oriented database program. It is also classified as NoSQL [15] database. It uses JSON-documents with schemas. MongoDB supports various functions like field, range and also can search using regular expressions. It contains user defined JavaScript functions which can return specific fields of documents. MongoDB also gives good results if configured to return some random sample of results of any given size.
4.2 Express.js

Express.js is also called as Express. It is also open-source software, which is used to build web applications and APIs using Node.js framework. Express is mainly used for routing. It helps applications to maintain the MVC architecture on the server side. Express helps in managing everything from routes to handling requests and views.

4.3 Angular.js

AngularJS [4] is a structural framework for dynamic web applications. It is open-source software. It extends the HTML vocabulary. The AngularJS directives are used to define the behavior of presentation components. These directives will specify what exactly each component of the page should function as. This framework adapts and extends traditional HTML to present dynamic content through two-way data binding that allows for the automatic synchronization of model and view as shown in Fig 4.4.

![Two-Way Data Binding](image)

Fig. 4.4. Two way data binding in Angular
4.4 Node.js

Node.js [14] is a distributed development project maintained by Node.js foundation. Node.js is an open-source JavaScript run time environment for executing code on server-side. It enables JavaScript to be used for server-side scripting and runs the script on server-side to produce dynamic content before the web-page is sent to the user’s web browser.

4.5 Cordova

Apache Cordova [5] is a mobile application development framework originally developed by Nitobi. Adobe Systems. This framework allows users to create CSS [6], HTML and JavaScript to work on mobile devices. The result is hybrid, meaning that they are neither native or web-based applications. So the same code can be used for both mobile and web application.

4.6 Affectiva SDK

Affectiva [7] is an emotion measurement technology company which provide SDKs and APIs for extracting human emotions using facial cues [19] or physiological responses. Affectiva’s technology can enable applications to use a webcam to track a user’s emotion. It already measured over 5.3 million faces [7]. Affectiva also provides various SDK’s for facial recognition. The image processing is done on the cloud and result is returned to the users.

4.7 Firebase

Firebase [8] is a mobile and web application development platform. It has many features available for developers, based on developers needs they can mix and match
to fit their needs related to the database. Firebase is known for its real time database and also consists of sync features.

4.8 System Flow

The Emotion-Based Music Player needs a user account to store the user preferences. Once the user logs-in and the application loads the required components and the detector, the user can click camera button to capture their image. Once the image is captured, it is sent to the SDK, which returns the emotion to the application. The system is currently designed to recognize four basic emotions: happiness, sadness, anger and disgust. After the emotion is recognized, the system runs the Emo-algorithm to categorize and display the play-list to the user. This algorithm considers multiple criteria for segregating and displaying songs to the user, which will later be discussed in detail. The user can change song details like category and interest level at any time in the application. The system also notifies the user of songs that are infrequently played so that they can change songs or delete the songs if it does not fit in the category.

4.8.1 User Login and Sign-up screen

If the user is accessing the application for the first time, they need to register for the application as shown in Fig 4.5. The user must give valid email address, name and password. If the user forgets their password an email to reset the password is sent to registered email address. All the information is validated and account creation is finished. Users cannot create multiple accounts with the same email.
Fig. 4.6 depicts the user log-in screen. The user should provide a registered email address and corresponding password to log-in to the application and the information will be saved on cloud. The application saves the user’s authentication information as the default. The user can log-out of the application any time.
Once the user successfully logs-in to the application, the system loads the SDK for capturing the user’s image. The user is redirected to an error page if they enter the wrong credentials or if they are non-registered.

4.8.2 Loading of SDK

As soon as user logs-in to the application, the system loads the required components and loads the Affectiva SDK. If the user is logging in for the first time, the user’s profile and preferences are created during this phase, as shown in Fig 4.7.
In this phase, the system creates space on the local device for loading the SDK. This SDK is a detector used to capture the user’s image. An Internet connection is required for this phase to load the detector.
4.8.3 Emotion Capture Phase

*Emotion Detector*

Once the SDK is loaded, the starting detector screen appears as shown in Fig 4.8 and opens device’s camera to capture the emotion. If the application is running for the first time, the application asks permissions to access media and camera.

![Fig. 4.8. Starting the detector](image)

The captured image quality depends on the user’s device. Each device has its own capturing techniques, and the Emotion-Based Music Player inherits all the
features of the device’s regular camera.

If the user’s device has beautification features, zooming, or timer techniques included with default camera, then these features are also loaded by the detector and can be used to increase image quality. After the picture is captured, the application prompts the user to accept or retake the captured image. Then the captured image is sent to the SDK to detect the image’s emotion.

**Post Detection**

In the post-detection phase, the user’s emotion is calculated by the SDK and the user is alerted of the result, as shown in Fig 4.9. The application supports four emotions: happiness, sadness, anger, disgust.

Various alert messages are displayed when an emotion is detected. These alerts notify the user of the emotion recognized by the SDK or if no emotion is detected among the four supported by the system. It also gives the user feedback by displaying an error message when image capturing is not completed properly.

During this phase, before alerting to the user, the application runs EMO-algorithm (discussed in next section) and fetches results for the user in application’s backend. The EMO-algorithm uses user’s details and usage to generate a custom segregated play-list each time an image is captured. The segregation process is discussed in further detail in the next section.
After user presses OK, the application redirects to music player screen for the detected emotion.

4.8.4 EMO-Phase

This is most important phase of the application. In this phase, we use the EMO-algorithm, which calculates the order in which songs should be organized or segregated in the play-list. The user’s customized music list is created based on the system’s knowledge of the user. The system provides users with more customized
play-lists based on their usage, play-styles, and preferences.

**EMO-algorithm**

**Data:** interest level and category while adding song, count of song, user changing song category, skip the song

**Result:** Customized music-list based on emotion recognized initialization;

while category and interest details exists for song A do

check all the details for song A ;

if song has played before then

if song has played more than 5 times then

g et all-details of song A;

calculate the weight of the song A ( )

if(song A is not skipped)

{ interest level (60%) + (song count)(40 %) ;
}

else

{ interest level(60%) + (song count -1)(40 %) ;
}

else

| calculate the song based on initial input |

end

else

calculate the song based on interest level

end

d

Algorithm 1: EMO-algorithm to calculate the weight of song
In this phase, the segregation of songs in a category is calculated. When the user adds a song to the application, the song’s details are also taken by the application, which comprises of song category and user interest level. Once user begins using the application, the EMO-algorithm considers many factors for customizing the playlist. Factors that are considered for calculating the weight of the song are:

1. Category and interest level

2. Number of times song has been played
   - If song has never been played
   - If song has been played less than 5 times
   - If song has been played more than 5 times

3. Number of time song has been skipped

The EMO-algorithm uses the following formula to calculate the song weight, based mainly on song based on interest level and number of times each song is played:

\[ \text{Song Weight} = (60\%) \text{ number of times played} + (40\%) \text{ song interest level} \]

This algorithm is designed to work even if the user skips the song. The system assumes that user is not interested in the song and reduces the weight.

Using the Algorithm 1, the application categorizes songs of each emotion and displays them to the user as shown in Fig 4.10

Each user has their own customized play-list. The user can add new songs to the music player, and by using the EMO-algorithm, all the songs are reorganized at run time. The song weights are calculated every time the user runs the application.
Once the user’s emotion is determined, this whole process runs in the background and will fetch songs, organizes and re displays the play-list to the user. The user can change the song category or interest level at anytime in the application and the system keeps track of the data.

Fig. 4.10. Song-List for Happy emotion
4.9 EMO-algorithm Scenarios

The following are the scenarios to depict the segregation of play-list using EMO-algorithm

4.9.1 EMO-algorithm Scenario 1

In Fig 4.11, the user played song1 more number of times than song4. According to the EMO-algorithm, once user starts listening to the songs, the algorithm
considers both song count and interest level for song weight.

Here, song1 with low interest level and played more times tops the list based on the song weights, the weights of songs are as shown below:

Weight of song1 = 9 * (0.6) + 0.6 = 6.0
Weight of song4 = 6 * (0.6) + 1.8 = 5.4

In the following series of test scenarios from Fig 4.12 - Fig 4.14, the application is tested using many songs to demonstrate how the application handles situations of equal weights.
4.9.2 EMO-algorithm Scenario 2

In Fig 4.12, the user played song1(Low)(count-19) and song4(High)(count-17). The weights of both songs are equal:

Weight of song1 = 19 * (0.6) + 0.6 = 12.0
Weight of song4 = 17 * (0.6) + 1.8 = 12.0
Weight of song2 = 7 * (0.6) + 1.2 = 5.4
Based on the EMO-algorithm, song1 is on top of list, since its interest level is high as shown in Fig 4.12.

4.9.3 EMO-algorithm Scenario 3

In Fig 4.13, user played song4 one more time than in the previous scenario [Fig 4.12], and the weight for the song4 has changed. In this instance, the algorithm calculates based on weights and the weight of song4 has increased from 12.0 to 12.6. The song weights are calculated as shown below.
Weight of song4 = 18 * (0.6) + 1.8 = 12.6
Weight of song1 = 19 * (0.6) + 0.6 = 12.0

So song4 is on top of the list. Based on this, in Fig 4.13, song4 is above song1.

4.9.4 EMO-algorithm Scenario 4

![Happy Songs Playlist]

Fig. 4.14. EMO-algorithm scenario 4

In Fig 4.14, the user has played song2 19 times and its weight is more than song4 and song1. Now song2 comes on top of the list, as shown as Fig 4.14.
4.9.5 Adding Song

User’s will be provided the option to add songs in the applications. Songs are added manually – users upload and provide details for each song. These details consist of song category and interest level.

![Add songs](image)

Fig. 4.15. Add songs

As soon as the user clicks on "Add song", the media is accessed to check the local devices files. The application searches for songs by checking the file extension (i.e., .mp3), allowing user to add songs from various locations. Options include
adding from Google Drive, audio files, file manager, and downloads folder, as shown in Fig 4.15. Fig 4.16 shows the process of choosing and adding songs.

![internal storage screenshot]

**Fig. 4.16. Song Upload from local files**

User can add songs as follows:

1. Once the emotion is detected and the user is in play-list

2. In the recommendation tab

3. In the music tab, where all the songs are listed
The user needs to provide interest level and category for each song, as shown in Fig 4.17

![Image of user input for category](image)

**Fig. 4.17. User’s input for category**

Currently, the application supports four emotions. Each song can be added under one of these four categories.

The emotions are:

1. Happy
2. Sad
3. Anger

4. Disgust

These four categories of emotions, each has a defined play-list for each user. A song can be placed in multiple categories based on user interest. For instance, there might a song the user likes to hear when they are either angry or sad. In this scenario, the user can add the song and set its interest level in both the angry category and the sad category.

Another important field for the user to provide when uploading a song is interest level. Interest is how much the user likes the song. For instance, the user might like a song but not be crazy about it, and could set the song interest as either medium or low interest level. The user can modify category and interest for a song at any point.

The user can set a song at three interest levels as shown in Fig 4.18, which have pre-defined weights relative to each other to help the user in listing the songs in the play-list.
The three interest levels are:

1. High Interest
2. Medium Interest
3. Low Interest

These play a vital role in calculation of song weight. If the weight changes, the organization of the songs also changes.
4.9.6 Saving Song

Once the user enters the song details, the application uploads the song to cloud and saves the user’s preferences. Internet connectivity is required for this part. The applications display the message "‘loading songs..please wait’" at this time, as shown in Fig 4.19.

Fig. 4.19. Saving song on cloud
4.10 Recommendations

The Emotional-Based Music Player gives the user song recommendations for each emotion so the user is aware of songs that are never played or less played, as shown in Fig 4.20. It gives the user option to edit the songs if they fit in other categories or to delete the song if the user is not interested.

Fig. 4.20. Recommendations for User
5. TESTING AND EVALUATION

To test the application, the following are the testing categories formulated.

5.1 Login test

The user must be able to create a profile, and if they already have a profile, they must be able to log-in and see their personal data.

![Wrong Email Format](image)

Fig. 5.21. Wrong Email Format

If the user forgets his password, he must be able to reset password. The test for
this scenario is to check if email is in proper format and credentials are validated. If user does not enter email properly, then the error as shown in Fig 5.21 is displayed.

If the user tries to log-in to application using wrong credentials, the error message as shown in Fig 5.22 is displayed. There are similar test cases for the sign-up process to check the redundancy of user accounts. User’s can have multiple accounts with same user name, but the email address should be different.
5.2 Test Case for Capture and Display Play-list based on Emotion

Based on the emotion captured using the camera, the user is notified by the emotion that is detected as shown in Fig 5.23.

![Fig. 5.23. Sad emotion detected](image)

The system recognizes and alerts the user’s emotion as ’Sad’ based on Affectiva SDK as shown in Fig 5.23.
Based on the emotion captured the play-list that is associated with the emotion is displayed to the user as shown in Fig 5.24.

The system displays ‘Sad Play-list’ as shown in Fig 5.24, since SDK recognizes Sad.

Fig. 5.24. Sad emotion paly-list detected
5.3 Test Case for Adding or Editing a Song

The user must provide the category and interest level of the song that is uploaded. If the user does not provide this information, they will see the error shown in Fig 5.25. All fields are mandatory while adding and editing a song. The weight is calculated based on these fields every time.

Fig. 5.25. Category and Interest level are required
5.4 Test Case Scenarios for EMO-algorithm

The following are the test case scenarios that were tested to check EMO-algorithm.

5.4.1 EMO-algorithm Test Case Scenario 1

In the above test scenario, songs were added by the user but never played. In this situation, the songs are categorized based on the interest level of the user according to the EMO-algorithm as shown in Fig 5.26. Songs that are of high interest...
are at the top of the list and songs that are of low interest are at the bottom of list.

5.4.2 EMO-algorithm Test Case Scenario 2

Fig. 5.27. EMO-algorithm test case scenario 2

In Fig 5.27, songs are played less than five times (the threshold value). Here, the application will segregate the songs based on only interest level and will not consider the number of times the song has played. The song order obtained from algorithm is song4 (High)(count-3), song2(Medium)(count-2), song1(Low)(count-3).
5.4.3 EMO-algorithm Test Case Scenario 3

In Fig 5.28, songs are played more than five times. According to the EMO-algorithm, the application will categorize and segregate based on both interest level and number of times song is played. The application will consider the value of plays since the threshold value has been crossed. Song1(Low)(count-8) has overtaken song2(Medium)(count-5) for second place.

Fig. 5.28. EMO-algorithm test case scenario 3
6. CONCLUSION AND FUTURE WORK

The Emotion-Based Music Player is used to automate and give a better music player experience for the end user. The application solves the basic needs of music listeners without troubling them as existing applications do: it uses technology to increase the interaction of the system with the user in many ways. It eases the work of the end-user by capturing the image using a camera, determining their emotion, and suggesting a customized play-list through a more advanced and interactive system. The user will also be notified of songs that are not being played, to help them free up storage space.

6.1 Future Work

The application can be improved by modifying and adding few functionality.

- Current application uses Affectiva SDK that has a lot of limitations, creating custom emotion recognition system that can be merged into the current application improves functionality and performance of the system.

- Making the application run without needing an internet connection.

- Including other emotions

- Playing songs automatically

- Optimizing the EMO-algorithm by including additional features which helps system to categorize user based on many other factors like location and suggesting the user to travel to that location and play songs accordingly.
BIBLIOGRAPHY AND REFERENCES


