AAM MUSIC PLAYER

Aseem Garg, Ankit Singh, Manthan Bhardwaj, Mr. Muthamil Selvan.s, RAVI TEJA,

[1] *Under Graduate Student, Department of Computer Science and Engineering, SRMIST, Chennai, India*
[2]*Under Graduate Student, Department of Computer Science and Engineering, SRMIST, Chennai, India*
[3] *Under Graduate Student, Department of Computer Science and Engineering, SRMIST, Chennai, India*
[4] *Faculty of SRM institute, Department of Computer Science and Engineering, SRMIST, Chennai, India*

ABSTRACT

AAM MUSIC PLAYER is an application that lets you upload, store, and play all of your music from the cloud. AAM Music player is a web application based on python language. You can now manage and listen to your music from any device, anywhere in the world. How does it work? To get started, first create a new album. When adding an album cover logo, it's best to have a resolution of at least 512x512 and to use common image formats such as JPG, JPEG, or PNG. After an album is created you will then be able to add/upload songs. Currently supported file types are WAV, MP3, and OGG. Once songs are added to an album you are then able to play, favorite, and delete them. Searching for songs in music player, You can also search for music using the search feature at the top of every page. Any relevant albums will appear at the top of the results page, and the results for individual songs will appear below. Music player core Python module this Python module provides a high-level core Music player interface where you are supposed to provide all the remaining high-level logic like the user interface, the playlist logic, and the audio data.

Keyword : *- music player, media player, music*

1 Introduction

Research within the area of tune psychology has shown that tune induces a clear emotional reaction in its listeners . Musical alternatives have been validated to be surprisingly correlated with character developments and moods. The meter, wood, rhythm and pitch of music are managed in regions of the mind that address emotions and temper.
Absolutely, a person’s affective reaction to a music fragment relies upon on a massive set of outside elements, along with gender,age ,tradition ,private preferences, emotion and context .But, those external variables set apart, human beings are capable of constantly categorize songs as being glad, unhappy, enthusiastic or comfortable.
Present day studies in emotion based recommender structures makes a speciality of principal elements, lyrics and audio capabilities ,acknowledging the language barrier, we awareness our efforts on audio characteristic extraction and evaluation of modern-day american and british english songs which will map the ones capabilities to 4 basic moods. Automated track type the use of a few temper categories yields promising outcomes. Facial expressions are the most historic and natural manner of conveying emotions, moods and feelings. For the cause of this paper, we categorize facial expressions into 4 distinctive emotional categories, viz. Glad, sad, indignant and impartial.The primary objective of this paper is to layout a cost-effective track player which mechanically generates a sentiment aware playlist primarily based on the emotional nation of the user. The software is designed to consume minimal machine sources. The emotion module determines the emotion of the consumer. Relevant and essential audio facts from a track is extracted by means of the tune category module. The advice module combines the consequences of the emotion module and the tune type module to propose songs to the user. This device presents drastically better accuracy and performance than existing systems. Furthermore, song brings us exclusive feelings together with unhappy,calm, and glad. As a result, we inspired to design an interactive subject matter-based totally track visualization of tune participant using processing. Processing is a programming language and environment based totally on java programming language. It's miles used for various functions, from medical statistics visualization to

art installations, in which one among them is song visualization. One Technique to generative visualizations is to begin with the frequency distribution of the audio data and alter and use This information in a few summary way. However, there are other algorithms and audio records that could be used inclusive of root Mean square (rms) and beat detection algorithms.

2 RELATED WORKS

Numerous methodologies were proposed to classify the behavioural and emotional state of the person. Mase et al. Centered on using actions of facial muscular tissues even as tian et al. Attempted to recognize actions units (au) developed through ekman and friesen in 1978 using permanent and transient facial capabilities. With evolving methodologies, the usage of convolutional neural networks (cnns) for emotion reputation has turn out to be increasingly popular. Song has also been categorised using lyrical evaluation. Even as this tokenized technique is rather easier to put into effect, on its personal it isn't appropriate to categorise songs as it should be. Another apparent challenge with this approach is the language barrier which restricts type to a unmarried language.Another method for tune mood class is using acoustic features like pace, pitch and rhythm to perceive the sentiment conveyed with the aid of the song. This approach includes extracting a hard and fast of features and using those feature vectors to find styles feature to a specific temper.

2.2 Audiovisual Synesthesia

Synesthesia is an extraordinary neurological circumstance that leads stimulation in one sensory pathway to trigger an experience in any other. Basically, a brief-circuiting inside the mind that enables bizarre phenomena like identifying letters and numbers as inherently coloured (coloration-graphemic synesthesia or chromesthesia) or listening to sounds in reaction to a visual motion. More than 60 styles of synesthesia were recognized, with one of the most common being the crosssensory revel in of color and sound — "hearing" coloration or "seeing" tune
When a synesthete listens to track, they see shapes and shades as one-of-a-kind tones, pitches, frequencies, harmonies, and other factors of the melody. Humans with this type of synesthesia describe their sound perceptions with phrases like whirring, humming, wobbling, tapping, or beeping. It takes place automatically, doesn't require attempt or interest, and is commonly experienced considering the fact that youth someone with hearing-motion synesthesia might not even realize that she or he hears sounds that other humans do not. This neurological eccentricity have become one of the resources of awesome inventive notion. These days, many synesthetes enjoy sounds through a track visualizer for their very own pride. The parent 1 suggests one of the examples of the interpretation of what a synesthete experience or see while taking note of a music. Michal visualized john coltrane, an american jazz saxophonist's harmonic structure based totally on symmetrical styles the theme of this visualization defines a area and the musical improvisation is like someone drifting in an imaginary area. But, synesthetic tune visualization.
Does now not visualize the sound functions, rather, it visualizes the melody or the rhythm of the music. The melody's repeated pattern is related to the constructing's structure. The melody is played two times and the framework of the building inside the animation is finished handiest while constructed twice.

3. OVERVIEW

The proposed tune participant shown in figure 2 consists oftwo modules called a primary module and visualizer. The principle module is to browse, select and play a music in audio format, at the same time as the visualizer module is to allow a person to have interaction with player control capabilities. On this visualizer, we used the two fundamental tactics for visualizing the sounds that is real-time animation that response primarily based on low-degree functions, and static image that assist the subject matter as the background of the scene.The subject matter of this track visualization is the wet day. It includes rain drops, lightning and floods effects that Presented with cylinder, triangle, and freeform shapes. The rain drops are real-time animation based totally on the frequency characteristic,as for lightning, the consequences take after the beat detection and size of song primarily based on frequency spectrum as floods. Numerous styles from the derived shapes are animated in real-time from Low-level capabilities and shaped an inspired artwork that has represent a rainy day subject matter. Except sound features, we used blue and gray colorations, positions, sizes and other attributes to animate and render the shapes and consequences that make the tune player replicate the wet day subject. The music player cum with visualizer are designed and advanced based totally on the incremental model with 3 iterations. For the software program utilization, the java audio and graphical user interface (gui) libraries are used to guide the gui interfaces design and gaining access to the audio information. Those libraries are supported through processing that is a programming language that broadly utilized by artists and designers for different purposes in various environment. It runs on all commonplace operating structures, is loose and gives maximum state-of-the-art features. Consequently, this makes processing is a high-quality device for nonprogramming Information developer to put into effect such music player cum visualization with interplay interface featured.

4 Music classification module

In this section, we describe the procedure that was used to identify the mapping of each song with its mood. We extracted the acoustic features of the songs using LibROSA, aubiopitch and other state-of-the art audio extraction algorithms. Based on these features, we trained an artificial neural network which successfully classifies the songs in 4 classes with an accuracy of 97.69%.

4.1 Dataset Description

The dataset comprises of 390 songs spread across four moods. The distribution of the songs is as follows: magnificence a with 100 songs, class b with ninety three songs, elegance c with 100 songs and class d with 97 songs. The songs have been manually labelled and the class labels were validated by way of 10 paid topics. Class a accommodates of thrilling and energetic songs, elegance b has satisfied and pleased songs, elegance c includes unhappy and despair songs, and class d has calm and cozy songs

4.2 Preprocessing

All the songs were down sampled to a uniform bit-price of 128 kbps, a mono audio channel and resampled at a sampling frequency of 44 one hundred hz. We in addition cut up each song to achieve clips that contained the most significant Parts of the tune. The feature vectors had been then standardized in order that it had 0 imply and a unit variance. All the songs were down sampled to a uniform bit-price of 128 kbps, a mono audio channel and resampled

4.3 Feature Description

We recognized several temper touchy audio features by using studying present day works and the results from the 2007 mirex audio mood category challenge. The candidate capabilities for the extraction process belonged To unique instructions: spectral (rmse, centroid, rolloff, mfcc,kurtosis, and many others.), rhythmic (pace, beat spectrum, and so forth.), tonal mode and pitch. These kinds of descriptions are widespread. All the functions were extracted the use of python 2.7 and applicable applications. After figuring out all the capabilities, we used recursive characteristic Elimination (or rfe) to pick out the ones features that pleasant contribute to the accuracy of the model. Rfe works by way of recursively putting off attributes and building a version on the ones attributes that remain. It makes use of the model accuracy to discover which attributes (and aggregate of attributes) make contributions the most to predicting the target characteristic. The selected capabilities were pitch,spectral rolloff, mel-frequency cepstral coefficients, tempo,root mean square electricity, spectral centroid, beat spectrum,0-move price, quick-time fourier rework and kurtosis of the songs.

4.4 Model Description

A multi-layered neural community became skilled to assess the temper related to the music. The network contains an input layer, more than one hidden layers and a dense output layer.The enter layer has fixed and predetermined dimensions. It takes the ten feature vectors as input and uses relu operation to offer non-linearity to the dataset. This ensured that the model plays well in actual-world scenarios as nicely.The hidden layer is a traditional multi-layer perceptron,Which allowed us to make mixture of functions which brought about a better class accuracy. The output layer used a softmax activation feature which produces the output as a chance for each mood class.

5. Playlist Design

A multi-layered neural community became skilled to assess the temper related to the music. The network contains an input layer, more than one hidden layers and a dense output layer.The enter layer has fixed and predetermined dimensions. It takes the ten feature vectors as input and uses relu operation to offer non-linearity to the dataset. This ensured that the model plays well in actual-world scenarios as nicely.The hidden layer is a traditional multi-layer perceptron, Which allowed us to make mixture of functions which brought about a better class accuracy. The output layer used a softmax activation feature which produces the output as a chance for each mood class.

6. Coding

```
from django.contrib.auth import authenticate, login
from django.contrib.auth import logout
from django.http import JsonResponse
from django.shortcuts import render, get_object_or_404
from django.db.models import Q
from .forms import AlbumForm, SongForm, UserForm
from .models import Album, Song

AUDIO_FILE_TYPES = ['wav', 'mp3', 'ogg']
IMAGE_FILE_TYPES = ['png', 'jpg', 'jpeg']


def create_album(request):
    if not request.user.is_authenticated():
        return render(request, 'music/login.html')
    else:
        form = AlbumForm(request.POST or None, request.FILES or None)
        if form.is_valid():
            album = form.save(commit=False)
            album.user = request.user
            album.album_logo = request.FILES['album_logo']
            file_type = album.album_logo.url.split('.')[-1]
            file_type = file_type.lower()
            if file_type not in IMAGE_FILE_TYPES:
                context = {
                    'album': album,
                    'form': form,
                    'error_message': 'Image file must be PNG, JPG, or JPEG',
                }
                return render(request, 'music/create_album.html', context)
            album.save()
            return render(request, 'music/detail.html', {'album': album})
        context = {
            "form": form,
        }
        return render(request, 'music/create_album.html', context)


def create_song(request, album_id):
    form = SongForm(request.POST or None, request.FILES or None)
    album = get_object_or_404(Album, pk=album_id)
    if form.is_valid():
        albums_songs = album.song_set.all()
        for s in albums_songs:
            if s.song_title == form.cleaned_data.get("song_title"):
                context = {
                    'album': album,
                    'form': form,
                    'error_message': 'You already added that song',
                }
                return render(request, 'music/create_song.html', context)
        song = form.save(commit=False)
        song.album = album
        song.audio_file = request.FILES['audio_file']
        file_type = song.audio_file.url.split('.')[-1]
        file_type = file_type.lower()
        if file_type not in AUDIO_FILE_TYPES:
            context = {
                'album': album,
                'form': form,
                'error_message': 'Audio file must be WAV, MP3, or OGG',
            }
            return render(request, 'music/create_song.html', context)

        song.save()
        return render(request, 'music/detail.html', {'album': album})
    context = {
        'album': album,
        'form': form,
    }
    return render(request, 'music/create_song.html', context)


def delete_album(request, album_id):
    album = Album.objects.get(pk=album_id)
    album.delete()
    albums = Album.objects.filter(user=request.user)
    return render(request, 'music/index.html', {'albums': albums})


def delete_song(request, album_id, song_id):
    album = get_object_or_404(Album, pk=album_id)
    song = Song.objects.get(pk=song_id)
    song.delete()
    return render(request, 'music/detail.html', {'album': album})


def detail(request, album_id):
    if not request.user.is_authenticated():
        return render(request, 'music/login.html')
    else:
        user = request.user
        album = get_object_or_404(Album, pk=album_id)
        return render(request, 'music/detail.html', {'album': album, 'user': user})


def favorite(request, song_id):
    song = get_object_or_404(Song, pk=song_id)
    try
```

```
def login_user(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                albums = Album.objects.filter(user=request.user)
                return render(request, 'music/index.html', {'albums': albums})
            else:
                return render(request, 'music/login.html', {'error_message': 'Your account has been disabled'})
        else:
            return render(request, 'music/login.html', {'error_message': 'Invalid login'})
    return render(request, 'music/login.html')


def register(request):
    form = UserForm(request.POST or None)
    if form.is_valid():
        user = form.save(commit=False)
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        user.set_password(password)
        user.save()
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                albums = Album.objects.filter(user=request.user)
                return render(request, 'music/index.html', {'albums': albums})
    context = {
        "form": form,
    }
    return render(request, 'music/register.html', context)


def songs(request, filter_by):
    if not request.user.is_authenticated():
        return render(request, 'music/login.html')
    else:
        try:
            song_ids = []
            for album in Album.objects.filter(user=request.user):
                for song in album.song_set.all():
                    song_ids.append(song.pk)
            users_songs = Song.objects.filter(pk__in=song_ids)
            if filter_by == 'favorites':
                users_songs = users_songs.filter(is_favorite=True)
        except Album.DoesNotExist:
            users_songs = []
        return render(request, 'music/songs.html', {
            'song_list': users_songs,
            'filter_by': filter_by,
```

HTML

```
{% extends 'music/base.html' %}
{% block albums_active %}active{% endblock %}

{% block body %}
<div class="albums-container container-fluid">

  <!-- Albums -->
  <div class="row">
    <div class="col-sm-12">
      <h3>{{ user.username }}'s Albums</h3>
    </div>
    {% if albums %}
      {% for album in albums %}
        <div class="col-sm-4 col-lg-2">
          <div class="thumbnail">
            <a href="{% url 'music:detail' album.id %}">
              <img src="{{ album.album_logo.url }}" class="img-responsive">
            </a>
            <div class="caption">
              <h2>{{ album.album_title }}</h2>
              <h4>{{ album.artist }}</h4>

              <!-- View Details -->
              <a href="{% url 'music:detail' album.id %}" class="btn btn-primary btn-sm" role="button">View
Details</a>

              <!-- Delete Album -->
              <form action="{% url 'music:delete_album' album.id %}" method="post" style="display: inline;">
                {% csrf_token %}
                <input type="hidden" name="album_id" value="{{ album.id }}" />
                <button type="submit" class="btn btn-default btn-sm">
                  <span class="glyphicon glyphicon-trash"></span>
                </button>
              </form>

              <!-- Favorite Album -->
              <a href="{% url 'music:favorite_album' album.id %}" class="btn btn-default btn-sm btn-favorite"
role="button">
                <span class="glyphicon glyphicon-star {% if album.is_favorite %}active{% endif %}"></span>
              </a>

            </div>
          </div>
        </div>
        {% cycle '' '' '' '' '' '<div class="clearfix visible-lg"></div>' %}
      {% endfor %}
    {% else %}
      <div class="col-sm-12">
        <br>
        <a href="{% url 'music:create_album' %}">
          <button type="button" class="btn btn-success">
            <span class="glyphicon glyphicon-plus"></span>&nbsp; Add an Album
          </button>
        </a>
```

```
            </div>
        {% endif %}
    </div>

    <!-- If user searches and there are songs -->
    {% if songs %}
        <div class="row">
            <div class="col-sm-12">
                <h3>Songs</h3>
            </div>
            <div class="col-sm-12">
                <div class="panel panel-default">
                    <div class="panel-body">
                        <table class="table">
                            <thead>
                                <tr>
                                    <th>Song Title</th>
                                    <th>Artist</th>
                                    <th>Audio File</th>
                                    <th>Album</th>
                                    <th>Favorite</th>
                                </tr>
                            </thead>
                            <tbody>
                                {% for song in songs %}
                                    <tr>
                                        <td>{{ song.song_title }}</td>
                                        <td>{{ song.album.artist }}</td>
                                        <td>
                                            <a target="_blank" href="{{ song.audio_file.url }}">
                                                <button type="button" class="btn btn-success btn-xs">
                                                    <span class="glyphicon glyphicon-play"></span>&nbsp; Play
                                                </button>
                                            </a>
                                        </td>
                                        <td>
                                            <a href="{% url 'music:detail' song.album.id %}">
                                                <img src="{{ song.album.album_logo.url }}" class="img-responsive" style="width:
20px; float: left; margin-right: 10px;" />
                                            </a>
                                            <a href="{% url 'music:detail' song.album.id %}">{{ song.album.album_title }}</a>
                                        </td>
                                        <td>
                                            <a href="{% url 'music:favorite' song.id %}" class="btn-favorite"><span
class="glyphicon glyphicon-star {% if song.is_favorite %}active{% endif %}"></span></a>
                                        </td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    {% endif %}
```

```
</div>
{% endblock %}
```

7. Conclusion and future work

Through the improvement of tune participant on android platform, we get a clean understanding of basic manner of the device. The middle a part of the track participant is particularly composed of main interface, playlists, menus, play settings, report browsing and song seek. Grasping the development of the six elements, the track participant has had the initial scale. Primarily based on the characteristic of the six categories, add a few different small capabilities. Tune player device found out the fundamental function of participant: play, pause, and forestall, up/down a, quantity adjustment, lyrics show, play mode, tune search, document browser, playlists query, and other functions. This improvement implicated the popular cellular terminal development generation. This is the combination management of java language in the open source mobile platform based totally on linux device+ + sqlite database support+ sharepreference configuration file. The system realized the music participant programming. This layout of music player based on android system calls for intricate design of the music player framework, with the aid of adopting eclipse3.5 + java language as technical help of this system, with the android plug-in gear, and mixture of android sdk2.1 version lead to the comprehensive and easily layout and improvement of the mobile terminal.

8. References

[1] A. S. Bhat, V. S. Amith, N. S. Prasad, and D. M. Mohan, "An efficient classification algorithm for music mood detection in western and hindi music using audio feature extraction," in *2014 Fifth International Conference on Signal and Image Processing*, Jeju Island, 2014, pp. 359–364.

[2] "How music changes your mood", Examined Existence. [Online]. Available:http://examinedexistence.com/how-music-changes-yourmood/. Accessed: Jan. 13, 2017.

[3] Kyogu Lee and Minsu Cho, "Mood Classification from Musical Audio Using User Group-dependent Models."

[4] Daniel Wolff, Tillman Weyde, and Andrew MacFarlane, "Culture-aware Music Recommendation."

[5] Mirim Lee and Jun-Dong Cho, "Logmusic: context-based social music recommendation service on mobile device," Ubicomp'14 Adjunct, Seattle, WA, USA, Sep. 13–17, 2014.

[6] D. Gossi and M. H. Gunes, "Lyric-based music recommendation," in *Studies in computational intelligence*. Springer Nature, pp. 301–310, 2016.

[7] Bo Shao, Dingding Wang, Tao Li, and Mitsunori Ogihara, "Music recommendation based on acoustic features and user access patterns," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 8, Nov. 2009.

[8] Ying-li Tian, T. Kanade, and J. Cohn, "Recognizing lower. Face action units for facial expression analysis," in *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, Mar. 2000, pp. 484–490.

[9] Gil Levi and Tal Hassner, "Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns."

[10] E. E. P. Myint and M. Pwint, "An approach for mulit-label music mood classification," in *2010 2nd International Conference on Signal Processing Systems*, Dalian, 2010, pp. V1-290-V1-294.

[11] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, and Marcus Liwicki, "DeXpression: Deep Convolutional Neural Network for Expression Recognition."

[12] Ujjwalkarn, "An intuitive explanation of Convolutional neural networks," the data science blog, 2016. [Online]. Available: https://ujjwalkarn.me/ 2016/08/11/intuitive-explanation-convnets/. Accessed: Jan. 13, 2017.

[13] Ian J. Goodfellow et al., "*Challenges in Representation Learning: A report on three machine learning contests*."

[14] S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," in *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[15] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. R. Wriobel, "Human-Computer systems interaction: back-grounds and applications," 3, ch. *Emotion Recognition and Its Applications*, Cham: Springer International Publishing, 2014, pp. 51–62.

[16] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, . . ., and Adrian Holovaty, (2015). librosa: 0.4.1 [Data set]. Zenodo. http://doi.org/10.5281/zenodo.32193.

[17] The aubio team, "Aubio, a library for audio labelling," 2003. [Online]. Available: http://aubio.org/. Accessed: Jan. 13, 2017.

[18] J. S. Downie, *The music information retrieval evaluation exchange (mirex).* D-Lib Magazine, 12(12), 2006.

[19] Cyril Laurier, Perfecto Herrera, M Mandel and D Ellis, "Audio music mood classification using support vector machine."

[20] "Unsupervised feature learning and deep learning Tutorial," [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/. Accessed: Jan. 13, 2017.

[21] A. S. Bhat, V. S. Amith, N. S. Prasad, and D. M. Mohan, "An efficient classification algorithm for music mood detection in western and hindi music using audio feature extraction," in *2014 Fifth International Conference on Signal and Image Processing*, Jeju Island, 2014, pp. 359–364.