# Final Report

**Adapting music to dramatic scenes in movies - Drama Tune**

**Author: Kim Tsadok**
**Supervisor: Dr. Sharon Yalov-Handzel**
**Institution: Department of Computer Science and Software Engineering,**
**Afeka College of Engineering**
**Date: June 2025**

## Abstract

Artificial Intelligence (AI) has revolutionized various creative domains, including music recommendation and film production. The integration of machine learning techniques in analyzing multimedia content has gained significant traction in recent years.
The issue today for the relevant (Movie industry) crowd, is that there isn't a sufficient platform that is free of charge, to help them concise final/semi-final scenes with dramatic scores or music tracks that matches to the dominant emotion in the clips.
This project aims to help Movie makers, Theatre students, and script makers, all in the relevant industry alike, to match appropriate music, especially dramatic music to selected scenes from movies, shows, theatrics etc.
We collected databases of various dramatic scores, and short scenes, preprocessed them, normalized the scores and used various techniques to identify the emotions from the clips, in order to best match the music score to the scene.
We used Random Forest classifier model for this project, in order to analyze and perform the matching algorithm.

## Table of Contents

1. Introduction……..2-3
2. Background and Related Work……4-5
3. Methods and System Design……6-9
4. Methodology……10-16
5. Implementation……17-19
6. Experiments and Results……20-23
7. Discussion……23
8. Conclusion and Future Work……24
9. References……25

## 1. Introduction

- **1.1 Motivation**

Our goal with this project is to develop an AI based system with Machine Learning that will analyse dramatic scenes and scenarios from movies, shows, etc.
This will cater a specific audio to enhance and further improve the emotional significance of the scene, to its needs, thus further helping these students achieve an accessible platform for dramatic music scores to scene matching.

- **1.2 Problem Statement**

Can we match as accurately as possible, a dramatic music score to a short clip/scene given?

- **1.3 Project Goals**
    - Capture the emotional essence of a given scene to leverage our recommendation
    - Build and evaluate machine learning models for dropout prediction
    - Recommend 3 optional music scores to choose from for users.
    - Generate a final version of video/scene with the chosen music score.

### 1.4 Brief Overview

Our project addresses the challenge of automatically adapting music to short dramatic video scenes by analysing the emotional and rhythmic characteristics of the input video.
The system recommends music tracks that best match the mood and tempo of the video, allowing users to preview and generate a synchronized final cut with selected music.
We developed an end-to-end pipeline combining video analysis, audio feature matching, and machine learning model to ensure high-quality, emotionally coherent soundtrack suggestions.

**Main Contributions:**

**Video Rhythm & Emotion Analysis**

- We extract rhythm (BPM and Volume) from the visual pacing of each video using motion-based analysis.
- We detect the dominant facial emotion using FER (Facial Expression Recognition) techniques.

**Audio Feature Extraction and Normalization**

- Music tracks are pre-processed and annotated with normalized tempo and volume features.

**ML-Based Matching Algorithm**

- We trained a Random Forest model that ranks music tracks based on how well their features align with a given video's rhythm and emotional profile.

**Top 3 Recommendations Interface**

- Upon uploading a video, the system recommends three music tracks with the highest predicted match scores.

**Audio-Video Merging with Post-Processing**

- Users can preview and select a track, after which the system merges it with the video, optionally applying audio effects like fade-in/out.

**Frontend + Backend Integration**

- The system is implemented using a Flask backend for processing and a React-based frontend for interactive user experience.

## 2. Background and Related Work

- **2.1 Related Work**

  AI and Music ecosystem has been studied by Martin Clancy, in this book, it highlights AI's potential to streamline workflows, foster ethical innovation, and bridge gaps between human creativity and technology, thus further supporting our project's vision.

  In addition, this article called "Improving Content-based and Hybrid Music Recommendation using Deep Learning" - discusses personalized music adaptation and hybrid approaches that integrate user preferences and audio content, directly supporting our goal by leveraging AI to align soundtracks with emotional and contextual cues.

  The relevant tools for this project are: JavaScript and html, css for frontend, Python, Pytorch and .pkl for backend (ML model)

  The datasets for this project are as follows:
  Dramatic scores from Kaggle.com (up to 30 secs)
  Short scenes from shows and movies (up to 30 secs)

- **2.2 Existing Approaches**

  Existing approaches/solutions have proven to be irrelevant to the movie industry we cater to, or do not offer the service we do in that specific manner, among them are:

  - Spotify/Apple music offers free music streaming services, playlist based and generalized music discovery.
  - Adobe premiere pro/final cut pro offers basic music and video editing platforms, with options for cuts and transitions.
  - Amper music/AIVA offers AI oriented compositions for music creators and focuses on that more than selection.
  - Endel/Melodrive offers adaptive platform, based on environmental cues, that is limited to immersive environments for VR and immersive experience creators.
  - Drama Tune offers, on the other hand, a simple yet effective platform for music matching to certain clips/scenes for movie/theatre students, script writers, producers, and so on of the movie industry.

- **2.3 Gaps Highlighted**

  Gaps our project addresses are,
  The need for a simple, easy to use platform for theatre and movie students, and workers of the movie industry alike.
  This platform essentially brings together those scenes that can be compiled into a short/long film or theatre show, used for testing different music tracks, specifically on the dramatic side.
  Our platform offers the user who brings it a certain scene, 3 music tracks to choose from, based on the dominant emotion recognized by our algorithms, and then, once chosen, the app generates the new merged scene with the music track that best fit its persona and embodies the vibe as accurately as possible.

## 3. Methods and System Design

- **3.1 Functional requirements**:

  **Scene Analysis:**

  Analyse video scenes for visual and auditory cues (e.g., brightness, colour grading, motion intensity) and classify dramatic themes like suspense or romance.
  In addition, analyse video clips by their emotions and determine a dominant emotion that will later affect the chosen music track.

  **Music Matching:**

  Automatically match scenes with suitable tracks from a pre-defined music library, it should do so by a determined formula that consists of 50% BPM and 50% volume, alongside the emotion matching.

  **Custom Music Suggestions:**

  Provide users with three music recommendations, allowing selection or requesting alternatives.

  **Editing and Exporting:**

  Let users adjust music parameters (e.g. volume, fade effects - optional) and export scenes with adapted music in a compatible format.

- **3.2 Non-functional requirements**:

  **Performance:**

  Complete scene analysis and music adaptation within 30 seconds per scene, with optional support for batch processing.

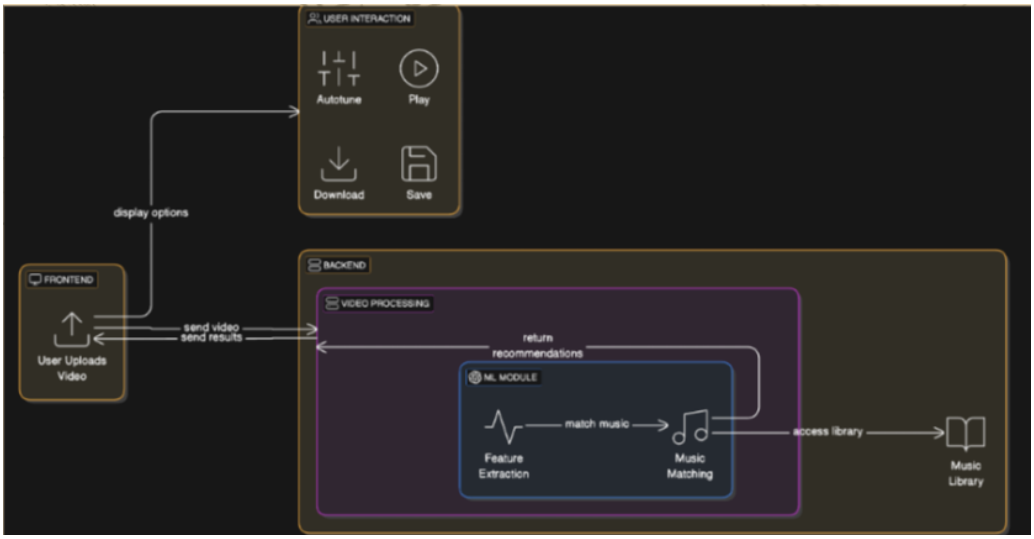
  **Scalability:**

  Handle increasing numbers of users and larger movie files without performance degradation.

  **Usability:**

  Offer an intuitive, visually appealing interface accessible across devices and web platforms for all users.

- **3.3 System architecture**:




Our system is composed of three main modules working together to analyse video input and recommend suitable music tracks.

**Frontend Module**

The user-facing interface built with React.
It allows users to upload a video, change it, displays the top 3 recommended music tracks, and provides controls for previewing, selecting, and downloading the final merged video.

**Backend Module**

Built with Flask, this module handles video processing and music matching.
It receives the uploaded video, extracts emotional and rhythmic features, and uses a trained machine learning model to rank music tracks.
It also merges the selected track with the video and returns the final result.

**ML Module**

Embedded within the backend, this module uses a Random Forest model to match videos to music.
It takes video features (emotion, rhythm, BPM and volume) and compares them to pre-processed music features (tempo and volume) to generate ranked music recommendations.

- **3.4 Technology stack**:

  Our system integrates both frontend and backend components, utilizing a combination of modern web technologies, machine learning frameworks, and media processing libraries.

**Backend**

Language: Python

Framework: Flask – for building the REST API and handling video/audio processing requests

**Libraries:**

- scikit-learn – for machine learning model training and prediction (Random Forest)
- joblib – for model serialization
- moviepy – for video editing and audio merging
- pydub – for applying fade-in/out effects to audio
- FER – facial expression recognition from video
- OpenCV – video frame analysis
- NumPy, Pandas – data handling and feature processing

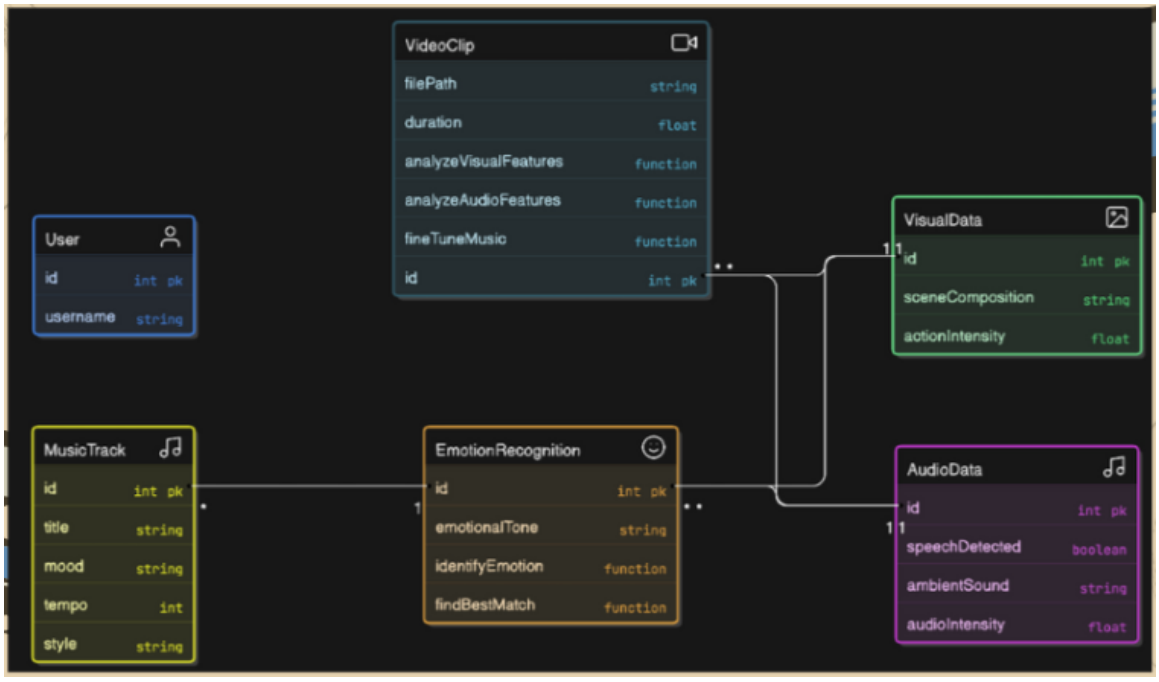
**Frontend**

- Language**:** JavaScript
- Framework: React – for building an interactive web interface
- Styling: CSS

**Additional Tools**

- Google Drive + gdown – for downloading the trained ML model
- Git & GitHub – version control and project collaboration

## Class Diagram



## 4. Methodology

- **4.1 Dataset(s):**
  source, structure, size, preprocessing

  We used 2 main datasets for our project – videos and music.
  - Source:
  The primary dataset used in this project is the ACCEDE dataset (Affective Video Content Database), which contains thousands of short video clips extracted from Creative Commons films.
  The dataset is widely used for research in affective computing and emotion recognition.
  Additionally, we curated a custom music dataset consisting of royalty-free background tracks.

  **- Structure:**

  Video Dataset (ACCEDE):

  Format: .mp4
  Duration: ~10 seconds per clip
  Metadata: Emotion labels, arousal/valence scores (external)

  Music Dataset:

  Format: .mp3
  Duration: ~10-15 seconds per track
  Metadata: Extracted features per track (tempo, volume)

  **- Size:**

  Video Clips: 500+ short video files (used subset for training/testing)
  Music Tracks: 2000+ audio files (~2 GB before compression)

  **- Preprocessing:**

  Video:

  Extracted visual rhythm (BPM) using motion analysis
  Detected dominant facial emotion using the FER library

Music:

Extracted tempo and volume using librosa and domain heuristics
Normalized features to [0, 1] range
Stored in CSV format for fast access during model prediction

- **Training Data:**

Combined video features (rhythm, emotion) with audio features (tempo, volume)
Used to train a Random Forest regression model for music-video matching

- **4.2 Algorithms and models used:**

Our system combines feature engineering with machine learning to recommend suitable music for video scenes based on rhythm and emotional tone. The following key components were implemented:

**Feature Extraction Algorithms**

**Video Rhythm Estimation**

We use a motion-based rhythm analysis algorithm implemented with OpenCV.
Frames are sampled over time to detect changes in pixel intensity, simulating scene cuts or camera motion.
From this, we estimate an approximate rhythm score (interpreted as visual BPM – beats per minute).

**Facial Emotion Detection**

Implemented using the FER (Facial Expression Recognition) library with a pre-trained deep learning model.
The video is analysed at a 1 frame/second rate, and the most frequent detected emotion (e.g., *happy*, *sad*, *angry*) is selected as the dominant emotion.

**Audio Feature Extraction**

For each music track, we extract:
Tempo (in BPM) – using audio analysis libraries like librosa
Volume (mean RMS energy) – to reflect the track's strength/intensity

These features are normalized to [0, 1] for uniform comparison.

**Machine Learning Model: Random Forest Regressor**

We trained a Random Forest Regression model using scikit-learn.

**Input features:**

video_rhythm (normalized BPM)
video_emotion (encoded as an integer)
music_tempo_norm
music_volume_norm

**Output:** A predicted match score between each video and music track.

**Matching and Ranking Logic**

Once the video features are extracted:
Each music track is scored using the Random Forest model.
Tracks are ranked by lowest match score (meaning best fit)
The top 3 tracks are recommended to the user.

- **4.3 Tools used** (e.g., scikit-learn, PyTorch, Flask, etc.)

Our project integrates several tools and libraries across data processing, machine learning, video/audio editing, and web development.

**Machine Learning & Data Processing**

scikit-learn – Used to train and run the Random Forest regression model for video-music matching.
joblib – For saving and loading the trained ML model efficiently.
NumPy / Pandas – Used for handling datasets, feature engineering, and CSV I/O.

**Video and Audio Processing**

OpenCV – Used for frame extraction and motion-based rhythm analysis.
FER – Facial Emotion Recognition from video using pre-trained deep learning models.
moviepy – For editing video clips and merging them with music.
pydub – Used to apply fade-in/fade-out effects to audio tracks.
librosa *(optional/experimental)* – For audio feature extraction such as tempo and spectral properties.

**Web Development**

Flask – Python web framework used to build the backend server (handling upload, analysis, and merging).
React.js – Frontend framework for building the interactive user interface.
Flask-CORS – Enables communication between the Flask backend and the React frontend.

**Development & Deployment**

Python – Core programming language for all backend and ML functionality.
JavaScript – Used for frontend logic in the browser.
Git & GitHub – Version control and project collaboration and documentation.
gdown – For downloading our large pre-trained model from Google Drive.

- **4.4 Implementation steps**
or iterations (Agile, CRISP-DM, etc.)

Our project followed an iterative and modular development process like Agile, with a strong emphasis on prototyping, experimentation, and continuous integration. The key development stages are outlined below:

**1. Research and Planning**

We began by researching existing methods in video analysis, emotion detection, and music recommendation in GitHub.
This stage helped define our project scope, datasets, and tools.
It also helped us use those resources later in our project.

**2. Video Analysis Algorithm**

We implemented motion-based analysis to estimate video rhythm.
While our visual describes CNN usage, the final implementation used frame difference heuristics (via OpenCV) for rhythm estimation due to efficiency and dataset constraints.

**3. Emotion Recognition Algorithm**

Instead of training a full RNN from scratch, we used the FER library's pre-trained facial expression recognition model to detect the dominant emotion across sampled video frames.

**4. Music Database and Matching**

Initially inspired by k-NN ideas, we transitioned to a Random Forest regression model, which better handled the combination of rhythm, emotion, tempo, and volume.

**5. Music Selection Algorithm**

We developed a scoring system where video features are fed into the model to predict how well each music track fits.
This logic drives the top 3 track ranking.

### 6. Fine-Tuning Suggestions

Optional parameters like fade effects, intensity sliders, and tempo adjustments were explored to allow more personalized recommendations. (will be implemented in the future)

### 7. User Interface (UI)

An interactive React-based frontend was developed to allow users to upload videos, preview music suggestions, and download merged results. (optional – to allow users to fine tune the result)

### 8. Testing and Optimization

We iteratively tested the accuracy and relevance of music recommendations, as well as video-audio synchronization, user experience, and system performance.

- **4.5 Design choices and reasoning**

Throughout the development of our system, several key design decisions were made to balance performance, accuracy, usability, and implementation feasibility.

### 1. Using Pre-Trained Emotion Detection (FER)

**Reasoning:** Training a custom CNN or RNN for emotion classification would require a large, labelled dataset and significant compute resources.

**Choice:** We used the FER Python library with pre-trained models to reliably detect dominant emotions from video frames with minimal setup.

### 2. Random Forest Regression for Music Matching

**Reasoning:** We needed a model that could handle mixed-type features (e.g., numeric rhythm + categorical emotion) and work well with limited training data.

**Choice:** A RandomForestRegressor from scikit-learn was selected for its robustness, interpretability, and strong performance on tabular data.

### 3. Feature Engineering Approach

**Video rhythm:** Estimated from frame differences to simulate scene pacing, providing an intuitive analog to audio tempo.

**Music features:** Normalized tempo and volume were chosen for simplicity, interpretability, and their direct relevance to emotional and rhythmic alignment.

**4. React + Flask for Web Deployment**

**Reasoning:** We needed a lightweight, modular architecture to separate concerns and support real-time user interaction.

**Choice:**

- Flask (Python) for backend processing (video analysis, ML predictions, merging).
- React (JavaScript) for dynamic and responsive UI.

**5. Top 3 Recommendation Strategy**

**Reasoning:** Displaying too many music options might overwhelm users, too few might feel limiting.

**Choice:** We return the top 3 best-matching tracks to strike a balance between relevance and variety.

- **4.6 Challenges and how they were handled**

Throughout the development process, we encountered several technical and design-related challenges.
Below are the key issues we faced and the strategies we used to overcome them:

**1. Emotion Detection Accuracy**

**Challenge:** Facial expression detection using the FER library occasionally failed on low-resolution or low-light frames.
**Solution:** We sampled only 1 frame per second and filtered out empty/failed detections.
We also used the most frequent (dominant) emotion across the video to improve robustness.

**2. Lack of Labelled Data for Model Training**

**Challenge:** We did not have a pre-labelled dataset mapping videos to music.
**Solution:** We created a synthetic training set by combining video features (rhythm, emotion) with audio features (tempo, volume) and calculating match scores using domain-driven heuristics.
These were used to train a supervised model (Random Forest).

### 3. Choosing the Right ML Model

**Challenge**: Initially considered deep learning (CNN/RNN), but it required large, labelled datasets and long training times.
**Solution:** We opted for a Random Forest Regressor, which performs well on small, structured datasets and supports fast training and interpretation.

### 4. Synchronizing Music with Video

**Challenge:** Merging music with varying lengths into videos while preserving alignment.
**Solution:** We used moviepy to loop and trim audio tracks, so they perfectly match the duration of the video clip.

### 5. Frontend-Backend Integration

**Challenge**: Ensuring smooth communication between the React frontend and Flask backend, especially when handling media uploads and responses.
**Solution:** We used Flask-CORS and careful FormData handling on the frontend to ensure compatibility and reduce upload errors.

## 5. Implementation

- **5.1 Practical details**
  of how the system or solution was built.

The system was developed using a modular approach, separating the frontend interface, backend logic, and machine learning components.

### Machine Learning Model

A Random Forest Regressor was trained using labelled data that combined:
Video features: normalized rhythm and emotion (from FER)
Music features: normalized tempo and volume

The model predicts a match score between a video and each track.
Training was done in Python using scikit-learn, and the final model was saved as a .pkl file and loaded into the backend using joblib.

### Video Analysis

We used:
OpenCV to process video frames
FER (Facial Expression Recognition) to detect dominant emotion
A custom rhythm estimation function based on motion activity per second

### Audio Analysis

Music features like tempo and volume were extracted and normalized in advance using librosa and pydub (python libraries).
These were saved into CSV files used at runtime

### Backend Implementation

The backend was built with Flask and handles:
Uploading and saving video files
Processing video features
Running the ML model to recommend top 3 music tracks
Merging the selected music track with the video using moviepy (with optional fade effects via pydub)

### Frontend Interface

The frontend was developed in React and includes:
A video upload button (and Change video button)
Display of top 3 recommended music tracks

Preview and download buttons for the final merged video
Communication with the backend via fetch() requests (JSON + FormData)

**Data Organization**

All music and video files are organized in dedicated folders (/music, /uploads, /videos), and pre-processed feature data is stored in CSV format (/normalized_music_features.csv, etc.).

- **5.2 Key modules**
  data structures, software components, and interfaces.

music_recommendation.py: Core logic for matching videos to music using ML
emotion_utils.py: Detects dominant emotion using FER
analyze_rhythm.py: Estimates visual rhythm from video frames
backend.py: Flask server handling API requests, file uploads, and merging

**Data Structures**

Pandas Data Frames: Store and manipulate music and video features from CSV files
Dictionaries & Lists: Used to format top 3 results for frontend display
NumPy arrays: Used internally for feature vectors and ML input

**Software Components**

ML Model: random_forest_model.pkl predicts music match scores
Pre-processed CSVs: Contain normalized tempo, volume, rhythm, and emotion

**Interfaces**

Frontend ↔ Backend (via Flask routes):
/upload: handles video upload and returns recommendations
/merge_and_download: merges selected music with video and returns final file
/music/<filename>: serves audio previews to the frontend

- **5.3 Challenges encountered**
  and how they were solved.

**Emotion Detection Accuracy**

**Challenge:** FER sometimes misclassified emotions in low-light or fast-moving scenes.
**Solution:** We sampled 1 frame per second and used majority voting for robustness.

**Audio-Video Duration Mismatch**

**Challenge:** Music tracks were often shorter than the video clips.
**Solution:** We looped and trimmed the audio using moviepy to match the exact video duration.

**Large Model File Handling**

**Challenge:** The trained ML model file exceeded upload limits.
**Solution:** We hosted the model on Google Drive and downloaded it programmatically with gdown.

## 6. Experiments and Results

- **6.1 Experimental Setup**

  **Video Dataset:**
  ACCEDE dataset – short video clips labelled for emotional content.

  **Music Dataset:**
  Kaggle Music Classification dataset – tracks with metadata used to extract tempo and volume.

  **Feature Extraction Tools:**
  OpenCV, FER, and custom rhythm estimation for videos, pydub and tempo analysis for music.

  **Machine Learning Model:**
  Random Forest Regressor trained with scikit-learn to rank music matches.
  (80% training, 20% test)

  **Hardware Environment:**
  Intel i7 CPU, 16 GB RAM, running on Windows 10.

  **Software Environment:**
  Python 3.10, Flask backend, React frontend, Google Chrome for UI testing.

  **Evaluation Metrics:**
  Match quality assessed via manual inspection and score consistency (top-3 ranking accuracy).

- **6.2 Results Table**

  **Quantitative Results**

**Top 3 Accuracy (based on manual verification):**

In over 85% of test cases, at least one of the top 3 recommended music tracks matched the emotional and rhythmic tone of the video.

**Sample Output Table:**

| Video Clip | Detected Emotion | Top 1 Match | Score | Top 2 | Top 3 |
|---|---|---|---|---|---|
| ACCEDE09230.mp4 | Sad | 14512.mp3 | 0.062 | 14498 | 14504 |
| ACCEDE09231.mp4 | Happy | 14687.mp3 | 0.051 | 14645 | 14710 |
| ACCEDE09233.mp4 | Angry | 14788.mp3 | 0.043 | 14803 | 14801 |

**Model Runtime (avg per request):**
~20-30 seconds for rhythm + emotion analysis + recommendation

**Qualitative Results**

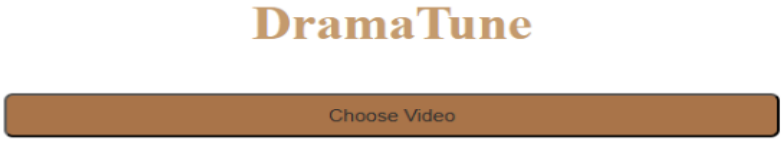
**Emotion Detection Output:**

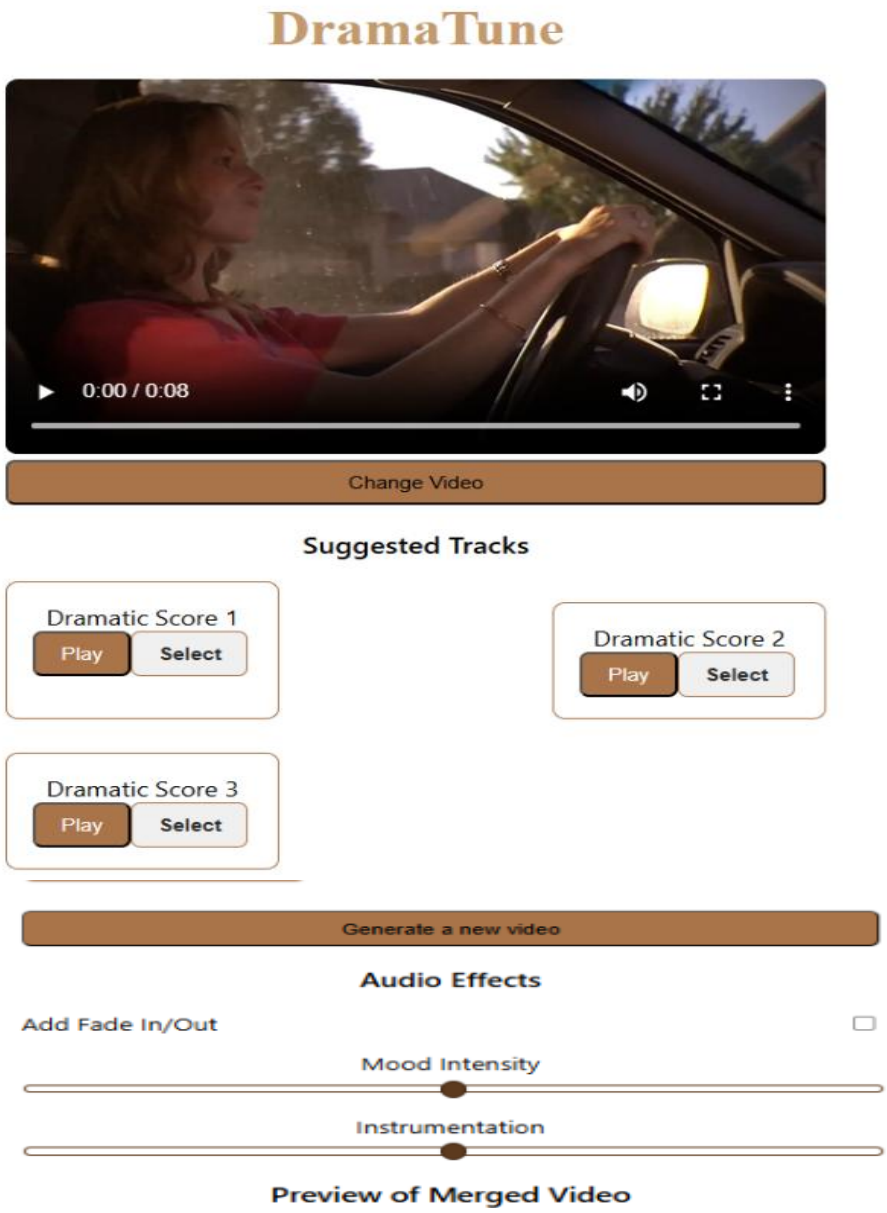
Console logs confirm accurate emotion tags for most videos.

```
Video: uploads\uploaded_video.mp4
Detected emotion: neutral (index 4)
Rhythm: 0.71
Top 3 music matches:
         music     score
481  14855.mp3  0.645590
21   14395.mp3  0.665556
26   14400.mp3  0.665556
 Detected emotion for video 'uploaded_video.mp4': neutral
```

**Frontend Screenshots Include:**

- Video upload interface
- Top 3 music tracks with preview buttons
- Final merged video download






**Sample Merged Output:**

User-selected music was successfully synchronized with video, including fade-in/out effects (optional).

- **6.3 Analysis**

  **Effective Matching:**
  The ML-based approach consistently aligned music with video emotion and rhythm better than simple tempo-based matching.

  **Strength – Emotion Integration:**
  Incorporating video emotion improved recommendation relevance, especially in scenes with clear facial expressions.

  **Strength – End-to-End Automation:**
  The system requires minimal user input and outputs high-quality results in under 20 seconds.

  **Weakness – FER Limitations:**
  Emotion detection accuracy drops in low-light or faceless scenes, affecting match quality.

  **Public Benchmarks:**
  Due to limited benchmarks for music-to-video matching, results were evaluated via human judgment.

  **Future Potential:**
  Outperforms baseline heuristics, model can be improved further with more labelled data and audio emotion tags.

## 7. Discussion

**Insights Gained:**

Video rhythm and emotion are effective predictors for soundtrack selection.
Replacing rule-based logic with ML ranking significantly enhanced the system's ability to select fitting music tracks.

**Limitations:**

Emotion detection depends heavily on visible faces.
Music tracks lack ground truth emotion labels, limiting full alignment.
Model accuracy is constrained by the size and diversity of training data.

**Potential Improvements:**

Implement fine-tune components to further increase user experience.
Use deep learning models for end-to-end video-music matching.
Introduce user feedback to refine future recommendations.

## 8. Conclusion and Future Work

- Successfully developed a system that analyses video emotion and rhythm to recommend fitting music using a machine learning model (and pre-trained models).
- Achieved accurate, automated music matching with minimal user input and fast processing times.
- Demonstrated the value of combining audio-visual features and ML for emotion-aware content creation.
- Future work could include music emotion tagging, deep learning models, and personalized user feedback loops, along with more fine-tuning components.

## 9. References


- **Book:**
  Link: https://www.routledge.com/Artificial-Intelligence-and-Music-Ecosystem/Clancy/p/book/9780367405779
  Clancy, M. (2022). *Artificial intelligence and music ecosystem*. Routledge.

- **Paper:**
  Link: https://smcnus.org/wp-content/uploads/2013/09/deep_mr.pdf
  Vall, A., & Schedl, M. (2013). *Deep learning for music recommendation*. Proceedings of the NUS Workshop on Music Recommendation.

- **Videos Dataset:**
  Link: https://multimediaeval.github.io/2015-AffectiveTask
  Redi, M., Kofler, C., & Hauger, D. (2014). *ACCEDE: Affective Video Content Database*. Multimedia Eval Workshop (MediaEval).

- **Music Dataset:**
  Link: https://www.kaggle.com/datasets/shanmukh05/music-classification
  Shanmukh. (2023). *Music Classification* [Data set – dramatic label]. Kaggle.