

SONG RECOMMENDATION SYSTEM USING FACE RECOGNITION & EMOTION DETECTION

A Project Report for Industrial Internship

Submitted by:

ABHAY SONI

RAMAN GUPTA

VEDANSHI GUPTA

In partial fulfilment of the requirements for the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY



Under the guidance of

MR. MAHENDRA DATTA

Project Carried Out

At



Ardent Computech Pvt. Ltd. (An ISO 9001:2015 Company)
SDF Building, Module 132, Ground Floor, Sector V, Salt Lake,
Kolkata 700091

In Association With



1. Title of the Project: Water Quality
2. Project Member: Shreyoshi Baral
3. Name and Address of the Guide: Mahendra Datta

Ardent Computech Pvt. Ltd. (An ISO 9001:2015 Company)

Module No.-132, SDF Building, Sector V, Salt Lake, Kolkata Pin - 700091.

4. Educational Qualification of the Guide: B. Tech, MAKAUT.
5. Working / Training experience of the Guide:

PROJECT VERSION CONTROL HISTORY

VERSION	PRIMARY AUTHOR	DESCRIPTION OF VERSION	DATE COMPLETED
Final	Abhay Soni	project Report	
Final	Raman Gupta	project Report	
Final	Vedanshi Gupta	project Report	

Date: -

Name and Signature of the Students:

1) Abhay Soni

2) Raman Gupta

3) Vedanshi Gupta

Signature of Approver
MR. MAHENDRA DUTTA
Project Proposal Evaluator

Approved

Not Approved

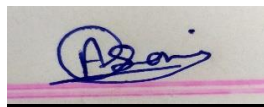
DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **“SONG RECOMMENDATION SYSTEM USING FACE RECOGNITION & EMOTION DETECTION”** in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

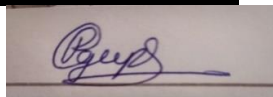
Date: -

Name and Signature of the Students:

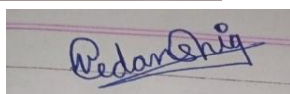
1) Abhay Soni



2) Raman Gupta



3) Vedanshi Gupta



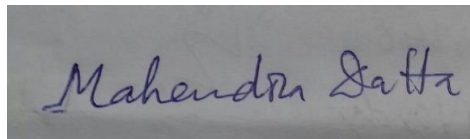
Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of the minor project entitled “**Water Quality**” is a record of bonafide work, carried out by **ABHAY SONI, RAMAN GUPTA & VEDANSHI GUPTA** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DUTTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **Mr. MAHENDRA DUTTA**, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

ABSTRACT

The Song Recommendation System based on human emotions is an innovative application designed to provide personalized song recommendations by leveraging emotion detection techniques. By analyzing users' facial expressions, voice tone, and physiological signals, the system identifies their emotional states, such as happiness, sadness, excitement, or relaxation. Using a sophisticated recommendation algorithm, the system matches users' emotional states with songs that possess corresponding emotional characteristics, enabling a more tailored and engaging music listening experience. The system continuously learns from user feedback and preferences, refining its recommendations over time to align with individual emotional preferences.

The Song Recommendation System based on human emotions offers a unique approach to music discovery by incorporating the emotional aspect of music listening. By considering users' emotional states, the system provides music suggestions that align with their current emotional needs, helping to uplift their mood, evoke specific feelings, or provide comfort. This personalized approach enhances user satisfaction and engagement, fostering a deeper connection between users and the music they listen to. With ongoing advancements in emotion detection and machine learning techniques, the Song Recommendation System based on human emotions holds the potential to revolutionize the way people discover and interact with music, offering a more meaningful and emotionally resonant music listening experience.

LIST OF FIGURES

S. No.	FIGURE	Page No.
1.	Flow Chart	14
2.	Use Case Model	15
3.	Conceptual Level Class Diagram	15
4.	Conceptual Level Activity Diagram	16
5.	Data Flow Diagram (Level 0)	16
6.	Data Flow Diagram (Level 1)	17
7.	Data Design (ER-Diagram)	17
8.	Sequence Diagram	18

TABLE OF CONTENTS

Overview

History of Python

Environment Setup

Basic Syntax

Variable Types

Functions

Modules

Packages

Artificial Intelligence

- Machine Learning

Machine Learning

- Supervised and Unsupervised Learning
- NumPy
- SciPy
- Scikit-learn
- Pandas
- Regression Analysis
- Matplotlib
- Clustering

SONG RECOMMENDATION SYSTEM

CHAPTER 1 – INTRODUCTION

1.1 Introduction

1.2 Problem Statement

1.3 Need for proper System

1.4 Objective

1.5 Modules of the System

1.6 Scope

CHAPTER 2 - LITERATURE SURVEY

2.1 Existing System

2.2 Proposed System

2.3 Feasibility Study

- 2.3.1 Technical Feasibility
- 2.3.2 Economical Feasibility
- 2.3.3 Operational Feasibility

CHAPTER 3 – REQUIREMENTS ANALYSIS

- 3.1 Method used for Requirement Analysis
- 3.2 Data Requirements
- 3.3 Functional Requirements
- 3.4 Non-Functional Requirements
- 3.5 System Specification
 - 3.5.1 Hardware Specification
 - 3.5.2 Software specification

CHAPTER 4 – DESIGN

- 4.1 Software Requirement Specification
 - 4.1.1 Glossary
 - 4.1.2 Supplementary Specification
 - 4.1.3 Use Case Model
- 4.2 Conceptual Level Class Diagram
- 4.3 Conceptual Level Activity Diagram
- 4.4 Data Flow Diagram (Level 0,1,2)
- 4.5 Data Design (ER-Diagram)
- 4.6 Sequence Diagram
- 4.7 Testing

CHAPTER 5 – CONCLUSION & FUTURE WORK

- 5.1 Limitation of Project
- 5.2 Future Enhancement

CHAPTER 6 - BIBLIOGRAPHY & REFERENCES

- 6.1 Reference Books
- 6.2 Other Documentations & Resources
- 6.3 Snapshot of python code

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few keywords, a simple structure, and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy-to-Maintain: Python's source code is fairly easy-to-maintaining.

A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

*If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");***

However, in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

**And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield**

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:  
    print "True"  
else:  
    print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python-h
```

```
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10      # An integer assignment
weight=10.60    # A floating point
name="Ardent"   # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- ☐ String
- ☐ List
- ☐ Tuple
- ☐ Dictionary
- ☐ Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

<u>SL.NO.</u>	<u>Function And Description</u>
<u>1.</u>	int(x [,base]) Converts x to an integer, base specifies the base if % is a string.
<u>2.</u>	long(x [,base]) Converts x to a long integer, base specifies the base if % is a string.
<u>3.</u>	float(x) Converts x to a floating-point number,
<u>4.</u>	complex(real [,imag]) Creates a complex number.
<u>5.</u>	str(x) Converts object x to a string representation.
<u>6.</u>	repr(x) Converts object x to an expression sting.
<u>7.</u>	eval(str) Evaluates a string and returns an object.
<u>8.</u>	tuple(s) Converts s to a tuple.
<u>9.</u>	list(s) Converts s to a list.

FUNCTIONS

Defining a Function

- `def function name(parameters):`
 `"function_docstring"`
 `function suite`
 `return [expression]`

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

Function definition is here

```
def change me(mylist):  
    "This changes a passed list into this function"  
    mylist.append([1,2,3,4]);  
    print"Values inside the function: ",mylist  
    return
```

Now you can call changeme function

```
mylist=[10,20,30];  
change me(mylist);  
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;           # This is global variable.
```

Function definition is here

```
def sum( arg1, arg2 ):
```

Add both the parameters and return them."

```
total= arg1 + arg2;    # Here total is local variable.  
print"Inside the function local total: ", total  
return total;
```

Now you can call sum function

```
sum(10,20);  
Print"Outside the function global total: ", total
```

When the above code is executed, it produces the following result –

```
Inside the function local total: 30  
Outside the function global total: 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):  
    print "Hello : ", par  
    return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [... moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub- packages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots ():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- ☐ *Phone/Isdn.py* file having function *Isdn ()*
- ☐ *Phone/G3.py* file having function *G3 ()*

Now, create one more file `__init__.py` in *Phone* directory –

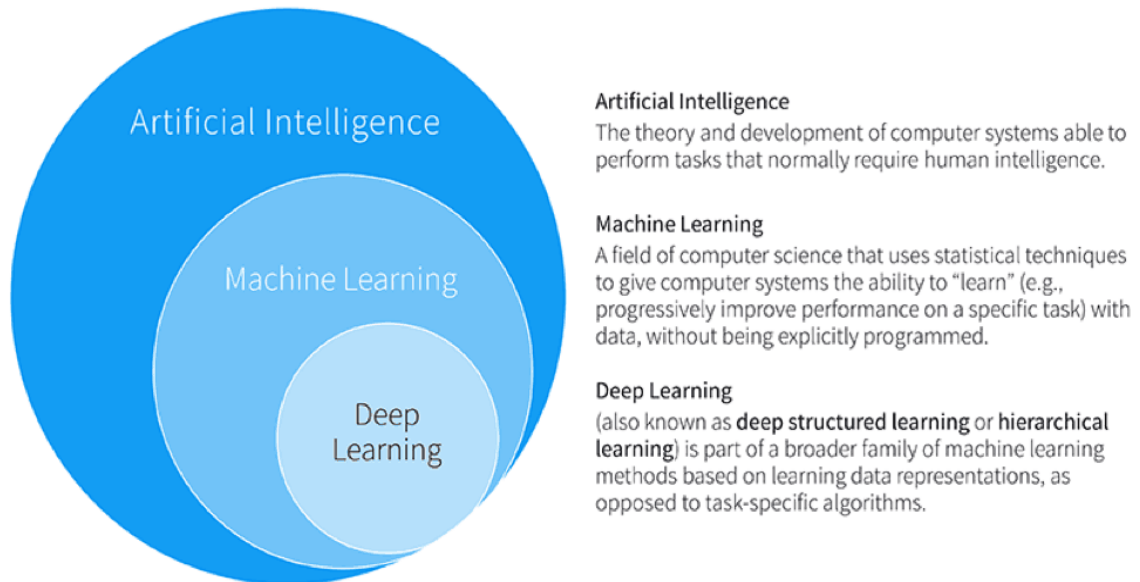
- ☐ *Phone/__init__.py*

To make all of your functions available when you've imported *Phone*, you need to put explicit import statements in `__init__.py` as follows –

```
from Pots import Pots  
from Isdn import Isdn  
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is “*The science and engineering of making intelligent machines, especially intelligent computer programs*”.

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems – The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.

To Implement Human Intelligence in Machines – Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing – It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems – These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

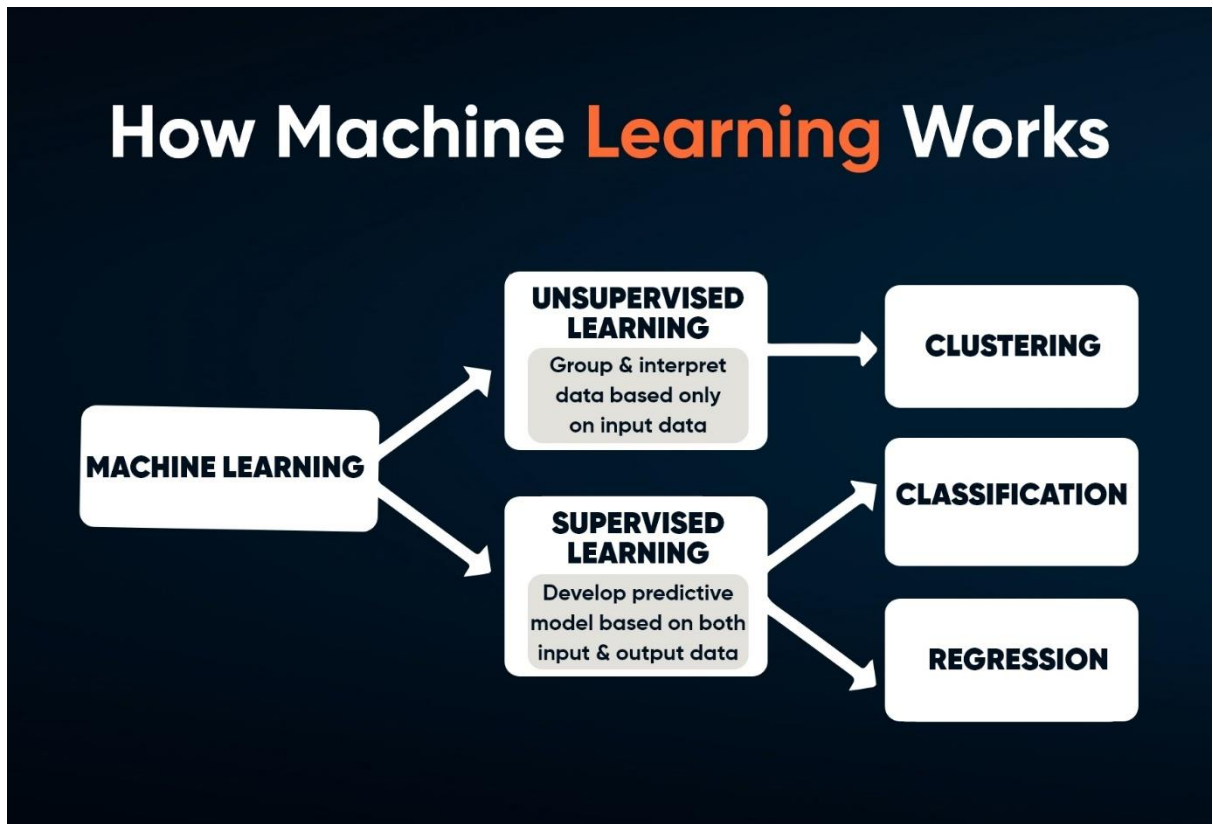
Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

Speech Recognition – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's voice due to cold, etc.

Handwriting Recognition – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system: -

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.^[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the C Python reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

```
a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])
```

```
print 'Our array is:'
```

```
Print a
```

```
print '\n'
```

```
print 'The items in the second column are:'
```

```
print a[:,1]
```

```
print '\n'
```

```
print 'The items in the second row are:'
```

```
print a[1...]
```

```
print '\n'
```

```
print 'The items columns 1 onwards are:'
```

```
print a [...,1:]
```

OUTPUT

Our array is:

```
[[1 2 3]
```

```
[3 4 5]
```

```
[4 5 6]]
```

The items in the second column are:

[2 4 5]

The items in the second row are:

[3 4 5]

The items column 1 onwards are:

[[2 3]

[4 5]

[5 6]]

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions

- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

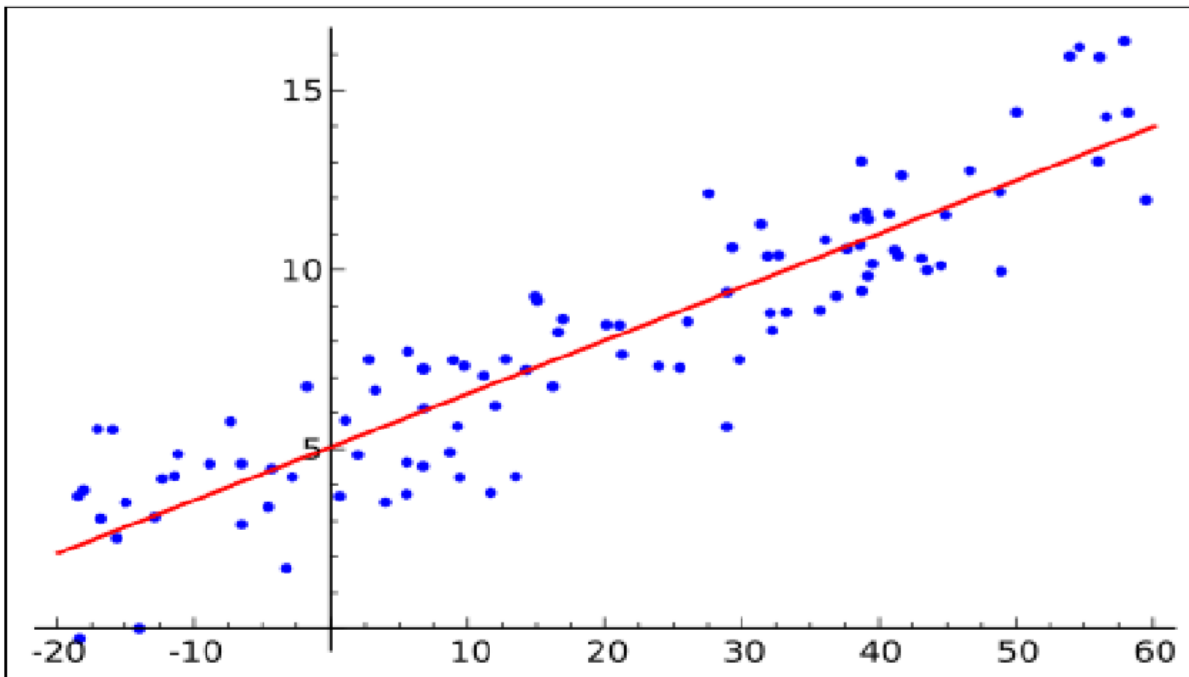
Data Structures

The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However, this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model^[1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x .

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted $E(y | x)$, and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function $E(y | x)$ is linear in the unknown parameters that are estimated from the data.

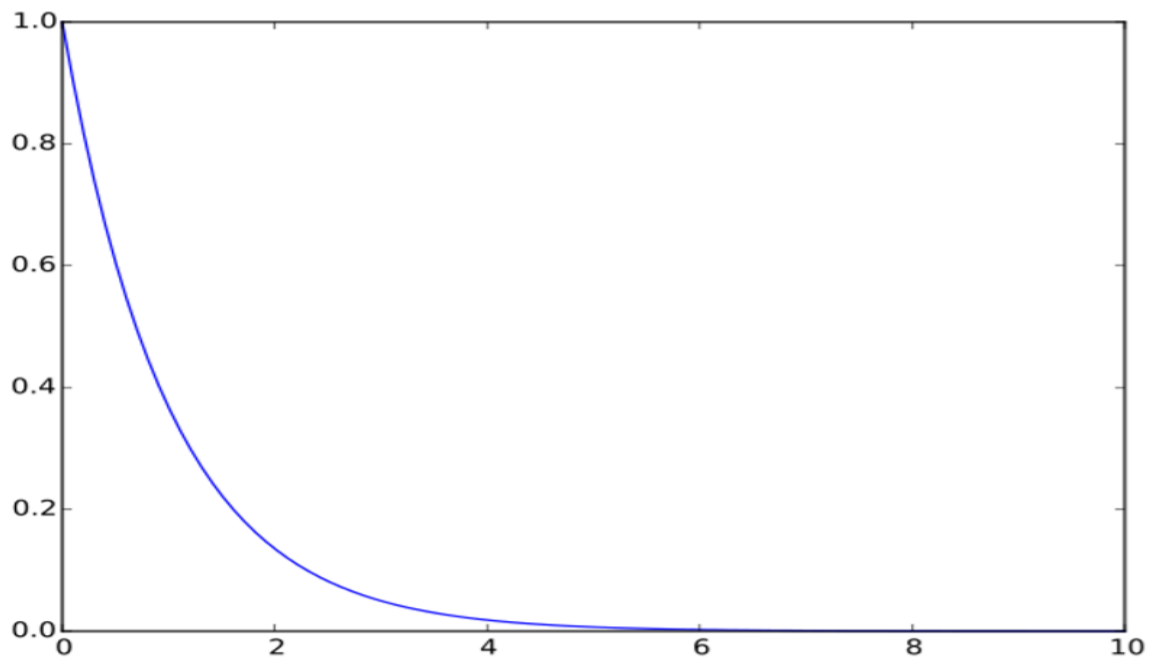
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

EXAMPLE

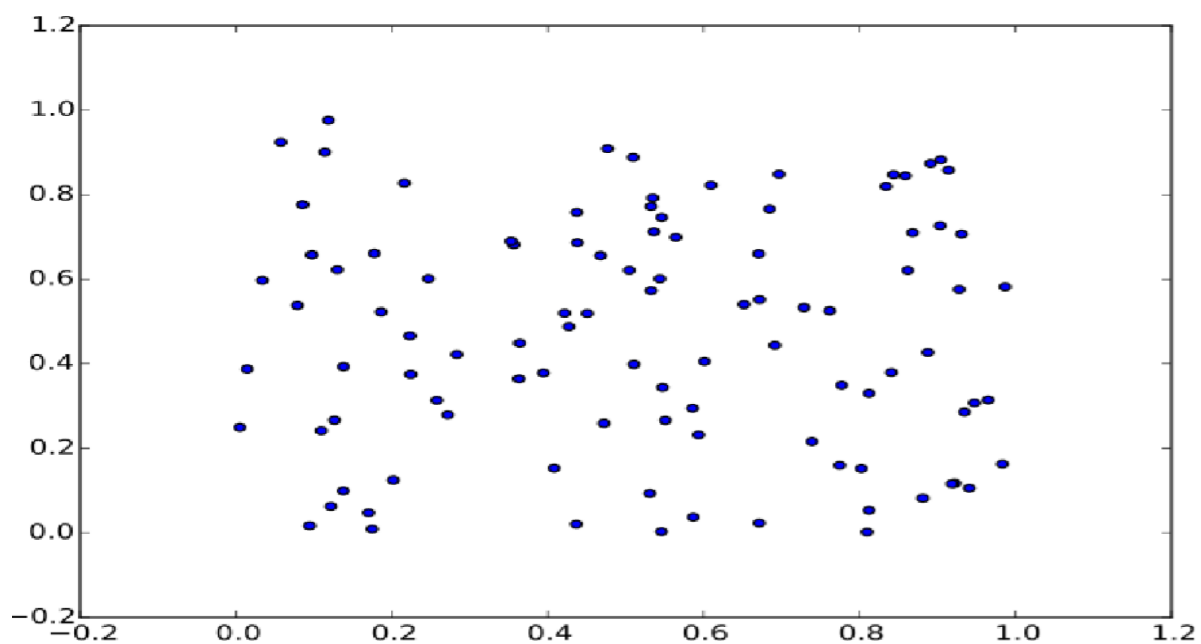
➤ LINE PLOT

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>>a=np.linspace(0,10,100)
>>>b=np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



➤ SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimportrand
>>>a=rand(100)
>>>b=rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

Song Recommendation System

Using Face Recognition & Emotion Detection

Chapter 1- Introduction

1.1 Introduction

People tend to express their emotions, mainly by their facial expressions. Music has always been known to alter the mood of an individual. Capturing and recognizing the emotion being voiced by a person and displaying appropriate songs matching the one's mood and can increasingly calm the mind of a user and overall end up giving a pleasing effect. The project aims to capture the emotion expressed by a person through facial expressions. A music player is designed to capture human emotion through the web camera interface available on computing systems. The software captures the image of the user and then with the help of image segmentation and image processing techniques extracts features from the face of a target human being and tries to detect the emotion that the person is trying to express. The project aims to lighten the mood of the user, by suggesting songs that match the requirements of the user by capturing the image of the user. Since ancient times the best form of expression analysis known to humankind is facial expression recognition. The best possible way in which people tend to analyze or conclude the emotion or the feeling or the thoughts that another person is trying to express is by facial expression. In some cases, mood alteration may also help in overcoming situations like depression and sadness. With the aid of expression analysis, many health risks can be avoided, and also there can be steps taken that help bring the mood of a user to a better stage.

1.2 Problem Statement

Using traditional music players, a user had to manually browse through his playlist and select songs that would soothe his mood and emotional experience. In today's world, with ever increasing advancements in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed (seek & time compression), local playback, streaming playback with multicast streams and including volume modulation, genre classification etc.

Although these features satisfy the user's basic requirements, yet the user has to face the task of manually browsing through the playlist of songs and select songs based on his current mood and behavior. That is the requirements of an individual, a user sporadically suffered through the need

and desire of browsing through his playlist, according to his mood and emotions.

1.3 Need for Proper System

You need a Music Recommendation System if you want to:

1. Increase satisfaction and engagement of your customers
2. Make your platform maximum personalized
3. Provide quality and immersive customer streaming experience
4. Build trust and listening habits of your users
5. Make it convenient to use your service, with no need to waste time finding new songs
6. Boost sales and subscription rate
7. Encourage up-sells and cross-sells
8. Automate curating and play listing audio
9. Get insights about users' behaviour and make data-based marketing decisions

1.4 Objectives

The main concept of this project is to automatically play songs based on the emotions of the user. It aims to provide user-preferred music with emotion awareness. In existing system user want to manually select the songs, randomly played songs may not match to the mood of the user, user has to classify the songs into various emotions and then for playing the songs user has to manually select a particular emotion. These difficulties can be avoided by using Song Recommendation System using Emotions.

Song Recommendation System using Emotions is a novel approach that helps the user to automatically recommend songs based on the emotions of the user. It recognizes the facial emotions of the user and plays the songs according to their emotion.

The human face is an important organ of an individual's body and it especially plays an important role in extraction of an individual's behaviors and emotional state. The webcam captures the image of the user. It then extracts the facial features of the user from the captured image. Facial expression categorized into 2, smiling and not smiling. According to the emotion, the music will be played from the predefined directories.

1.5 Modules of the System

Emotion Extraction Module –

The image of the user is captured with the help of a camera/webcam. Once the picture captured, the frame of the captured image from webcam feed is converted to a grayscale image to improve the performance of the classifier, which is used to identify the face present in the picture. Once the conversion is complete, the image is sent to the classifier algorithm which, with the help of feature extraction techniques can extract the face from the frame of the web camera feed. From the extracted face, individual features are obtained and are sent to the trained network to detect the emotion expressed by the user. These images will be used to train the classifier so that when a completely new and unknown set of images is presented to the classifier, it is able to extract the position of facial landmarks from those images based on the knowledge that it had already acquired from the training set and return the coordinates of the new facial landmarks that it detected. The network is trained with the help of CK extensive data set. This is used to identify the emotion being voiced by the user.

Audio Extraction Module –

After the emotion of the user is extracted the music/audio based on the emotion voiced by the user is displayed to the user, a list of songs based on the emotion is displayed, and the user can listen to any song he/she would like to. Based on the regularity that the user would listen to the songs are displayed in that order.

Emotion - Audio Integration Module –

The emotions which are extracted for the songs are stored, and the songs based on the emotion are displayed on the web page built using PHP and MySQL. For example, if the emotion or the facial feature is categorized under happy, then songs from the happy database are displayed to the user.

1.6 Scope

Facial expressions are a great indicator of the state of a mind for a person. Indeed, the most natural way to express emotions is through facial expressions. Humans tend to link the music they listen to, to the emotion they are feeling. The song playlists though are, at times too large to sort out automatically. It would be helpful if the music player was “smart enough” to sort out the music based on the current state of emotion the person is feeling. The project sets out to use various techniques for an emotion recognition system, analyzing the impacts of different techniques used. By using this song recommendation system, we can easily play the songs according to the emotion of the user.

CHAPTER 2 - LITERATURE SURVEY

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Steps in Machine Learning Machine learning is a field of computer science that gives computers the ability to learn without being programmed explicitly. The power of machine learning is that you can determine how to differentiate using models, rather than using human judgment. The basic steps that lead to machine learning and will teach you how it works are described below in a big picture:

- Gathering data
- Preparing that data
- Choosing a model
- Training
- Evaluation
- Hyper parameter tuning

- Prediction.

2.1 Existing System

The features available in the existing Music players present in computer systems are as follows:

i. Manual selection of Songs ii. Party Shuffle iii. Playlists iv. Music squares where user has to classify the songs manually according to particular emotions for only four basic emotions. Those are Passionate, Calm, Joyful and Excitement.

Using traditional music players, a user had to manually browse through his playlist and select songs that would soothe his mood and emotional experience .In today's world, with ever increasing advancements in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed (seek & time compression),local playback, streaming playback with multicast streams and including volume modulation, genre classification etc.

Although these features satisfy the user's basic requirements, yet the user has to face the task of manually browsing through the playlist of songs and select songs based on his current mood and behavior. That is the requirements of an individual, a user sporadically suffered through the need and desire of browsing through his playlist, according to his mood and emotions.

Limitations in Existing System:

- It requires the user to manually select the songs.
- Randomly played songs may not match to the mood of the user.
- User has to classify the songs into various emotions and then for playing the songs user has to manually select a particular emotion.

2.2 Proposed System

Here we propose an Emotion based music system. This is a music player/system which play songs according to the emotion of the user. It aims to provide user preferred music with emotion awareness. This system is based on the idea of automating much of the interaction between the music player and its user. The emotions are recognized using a machine learning method Support Vector Machine (SVM) algorithm. In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It finds an optimal boundary between the possible outputs. The training dataset which we used is Olivetti faces which contain 400 faces and its desired values or

parameters. The webcam captures the image of the user. It then extracts the facial features of the user from the captured image. The training process involves initializing some random values for say smiling and not smiling of our model, predict the output with those values, then compare it with the model's prediction and then adjust the values so that they match the predictions that were made previously. Evaluation allows the testing of the model against data that has never been seen and used for training and is meant to be representative of how the model might perform when in the real world. According to the emotion, the music will be played from the predefined directories.

2.3 Feasibility Study

2.3.1 Technical Feasibility

A song prediction system, as you've described it, would likely require a significant amount of technical expertise and resources to develop. The feasibility of such a system would depend on a number of factors, including:

1. **Data availability:** A song prediction system would require access to large amounts of music data, including information about lyrics, melodies, and other musical features. Depending on the scope and complexity of the system, acquiring and organizing this data could be a significant challenge.
2. **Algorithm development:** Developing algorithms that can accurately predict the characteristics of a song based on its lyrics, melody, and other features is a complex task. It would require expertise in fields such as natural language processing, machine learning, and music theory.
3. **Computing resources:** A system capable of processing and analysing large amounts of music data would require significant computing resources, including high-performance computing clusters and large-scale data storage.

Given these challenges, it's important to carefully consider the technical feasibility of a song prediction system before including it in a software requirements specification document. It may be necessary to engage with experts in fields such as data science and music theory to help assess the feasibility of the project and develop a detailed plan for its implementation.

2.3.2 Economic Feasibility

The economic feasibility of a song prediction system would depend on a careful cost-benefit analysis, taking into account the development costs, ongoing maintenance costs, and potential revenue streams. The development costs of the system would depend on factors such as the complexity of the algorithms and the size of the data sets required. Once developed, ongoing maintenance costs such as hosting, data storage, and technical support would also need to be factored in. Potential revenue streams could include licensing fees, subscription fees, or advertising revenue if the system is monetized. The economic feasibility of the project would depend on whether the potential revenue streams could cover the development and maintenance costs, making it economically viable.

2.3.3 Operational Feasibility

The operational feasibility of a song prediction system would depend on factors such as the availability of required resources, the level of user acceptance, and the compatibility with existing systems and processes. The system would need to be user-friendly, providing accurate predictions and integrating well with existing music streaming platforms. The availability of required resources, such as large datasets and computing resources, would need to be considered, and the development team would need to have the necessary technical expertise. End-users should be involved in the development process and user testing should be conducted to ensure that the system meets their needs. If these factors can be adequately addressed, then the system may be operationally feasible.

CHAPTER 3 – REQUIREMENTS ANALYSIS

3.1 Method used for Requirement Analysis

The requirements analysis for a song prediction system would typically involve the following steps:

- **Understanding the business goals and objectives:** The first step would be to understand the goals and objectives of the song prediction system. This would involve identifying the target audience, the purpose of the system, and the expected outcomes.
- **Identifying the functional requirements:** The next step would be to identify the functional requirements of the system. This would involve defining the features and capabilities of the system, such as the ability to analyze user preferences, recommend songs, and personalize the user experience.
- **Defining the non-functional requirements:** Non-functional requirements define the qualities of the system that are not directly related to its functionality, such as performance, scalability, security, and usability. These requirements need to be defined to ensure that the system is robust, efficient, and user-friendly.
- **Gathering stakeholder requirements:** It is important to gather requirements from stakeholders, such as end-users, business owners, and developers. This can be done through interviews, surveys, focus groups, and other forms of data gathering.

3.2 Data Requirements

The data requirements for a song prediction system would depend on the specific features and functionality of the system. However, some common data requirements for a song prediction system may include:

- **User profile data:** This includes data about the user's demographics, location, musical preferences, listening history, and other relevant information. This data is used to personalize the song recommendations and improve the accuracy of the prediction algorithm.
- **Song metadata:** This includes data about the song itself, such as the artist, genre, tempo, mood, and other attributes. This data is used to identify patterns and correlations between songs, and to create a database of songs that can be recommended to users.

- **User feedback data:** This includes data about the user's response to the song recommendations, such as whether they liked or disliked a particular song, or whether they skipped it. This data is used to improve the accuracy of the prediction algorithm and to refine the song recommendations over time.
- **External data sources:** This includes data from external sources such as social media, streaming services, and music blogs. This data can be used to identify emerging trends and to keep the song recommendation database up to date.
- **Audio data:** This includes data about the audio features of the song, such as the tempo, key, and pitch. This data can be used to identify similarities and differences between songs and to improve the accuracy of the prediction algorithm.

Overall, the data requirements for a song prediction system would depend on the specific goals and objectives of the system, as well as the preferences and behaviors of the target audience.

3.3 Functional Requirements

Functional requirements are statement of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situation:

- The dataset train by support vector classifier
- Machine learns support vector classification using support vector machine.
- Learn and identify image capture by web cam.

3.4 Non-Functional Requirements

Nonfunctional requirements define system properties and constraints it arises through user needs, because of budget constraints or organizational policies, or due to the external factors such as safety regulations, privacy registration and so on. Nonfunctional requirements are:

- Reliability
- Maintainability
- Portability
- Extensibility
- Reusability
- Simplicity
- Resource Utilization

3.5 System Specification

3.5.1 Hardware Specification

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The hardware requirements required for this project are:

- Intel i5
- 4GB RAM
- Webcam
- Speaker

3.5.2 Software Specification

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed. The software requirements that are required for this project are:

- Python 2.7
- Open CV 3.1

CHAPTER 4 – DESIGN

4.1 Software Requirement Specification

4.1.1 Glossary

Glossary of terms related to a Song Prediction System:

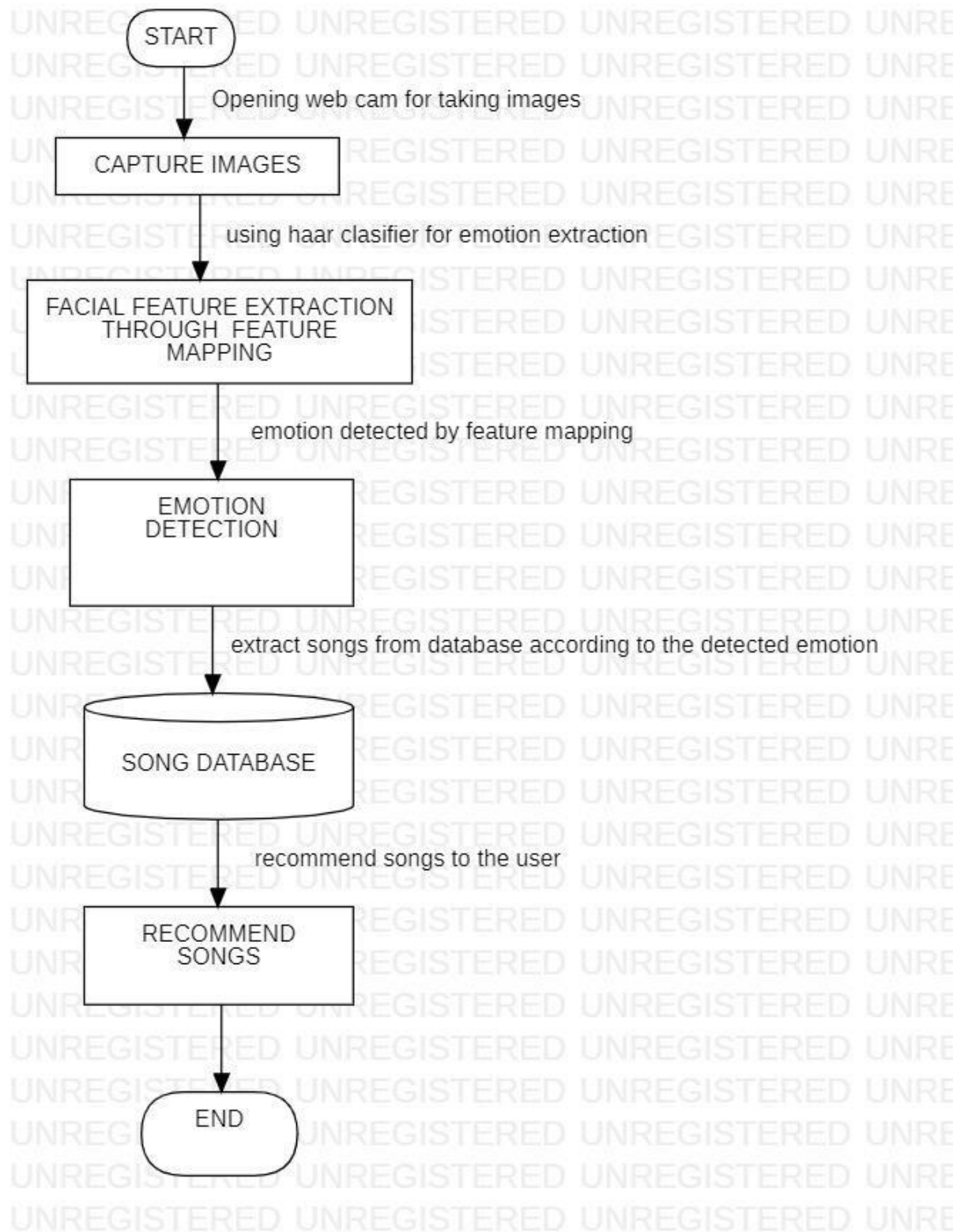
1. Song Prediction System: A software that predicts songs a user may want to listen to based on their listening history and preferences.
2. Machine Learning: A type of artificial intelligence that enables software to learn from data and improve its performance without being explicitly programmed.
3. Algorithm: A set of instructions that a computer program follows to perform a specific task, such as predicting songs.
4. Data: Information used by the Song Prediction System to make predictions, such as the user's listening history, song preferences, and relevant information.
5. Recommender System: A software that provides personalized recommendations to users based on their past behaviour and data.
6. Collaborative Filtering: A type of recommender system that predicts what a user may like based on the preferences of similar users.
7. Content-Based Filtering: A type of recommender system that predicts what a user may like based on the features of the songs they have listened to.
8. Hybrid Filtering: A type of recommender system that combines collaborative filtering and content-based filtering.
9. User Profile: A collection of information about the user that the Song Prediction System uses to make predictions.
10. Accuracy: A measure of how well the Song Prediction System predicts the songs the user wants to listen to, often expressed as a percentage.

4.1.2 Supplementary Specification

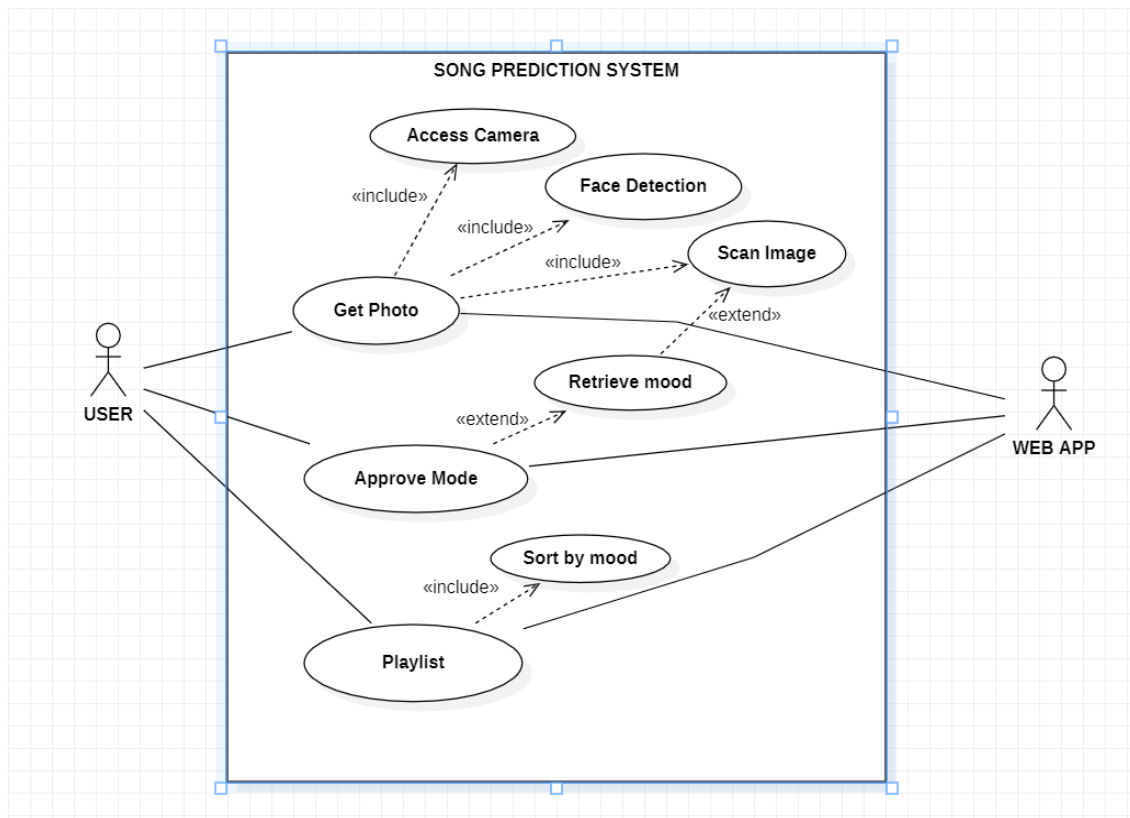
Some possible supplementary specification for a Song Prediction System are as follows:

1. **Compatibility:** The Song Prediction System should be compatible with major operating systems, web browsers, and mobile devices.
2. **User Interface:** The Song Prediction System should have a user-friendly interface that allows users to easily navigate and interact with the system.
3. **Integration:** The Song Prediction System should integrate with popular music streaming services and online music libraries to access a wide range of songs and data.
4. **Privacy and Security:** The Song Prediction System should adhere to privacy and security regulations and protect user data from unauthorized access or misuse.
5. **Machine Learning Models:** The Song Prediction System should use accurate and up-to-date machine learning models to predict songs that users want to listen to.
6. **Recommendations:** The Song Prediction System should provide personalized and relevant recommendations to users based on their listening history, preferences, and other relevant data.
7. **Feedback and Improvement:** The Song Prediction System should allow users to provide feedback on the recommended songs and continuously improve its prediction accuracy.
8. **Scalability:** The Song Prediction System should be scalable to handle large volumes of data and user requests without compromising its performance.
9. **Performance:** The Song Prediction System should have fast response times and minimal downtime to ensure a smooth and uninterrupted user experience.
10. **Documentation:** The Song Prediction System should have clear and concise documentation that explains its features, functionalities, and usage instructions for both users and developers.

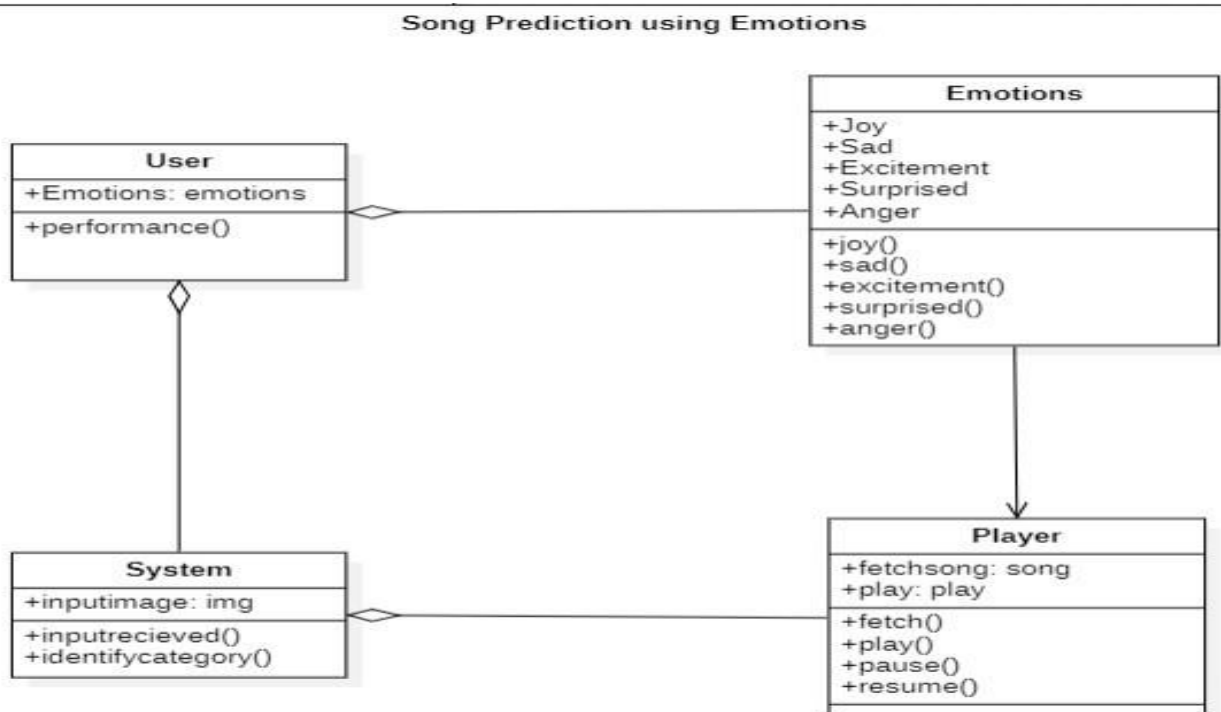
Flow-chart Diagram



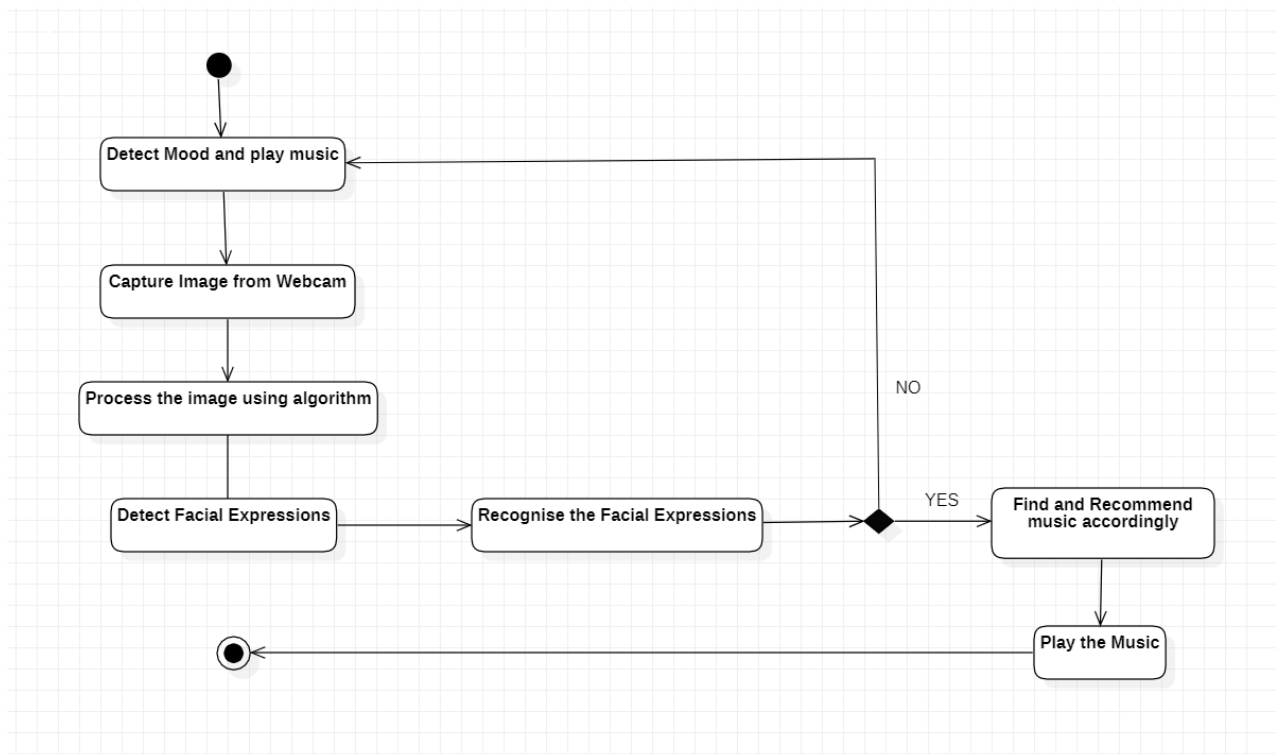
4.1.3 Use Case Model



4.2 Conceptual Level Class Diagram

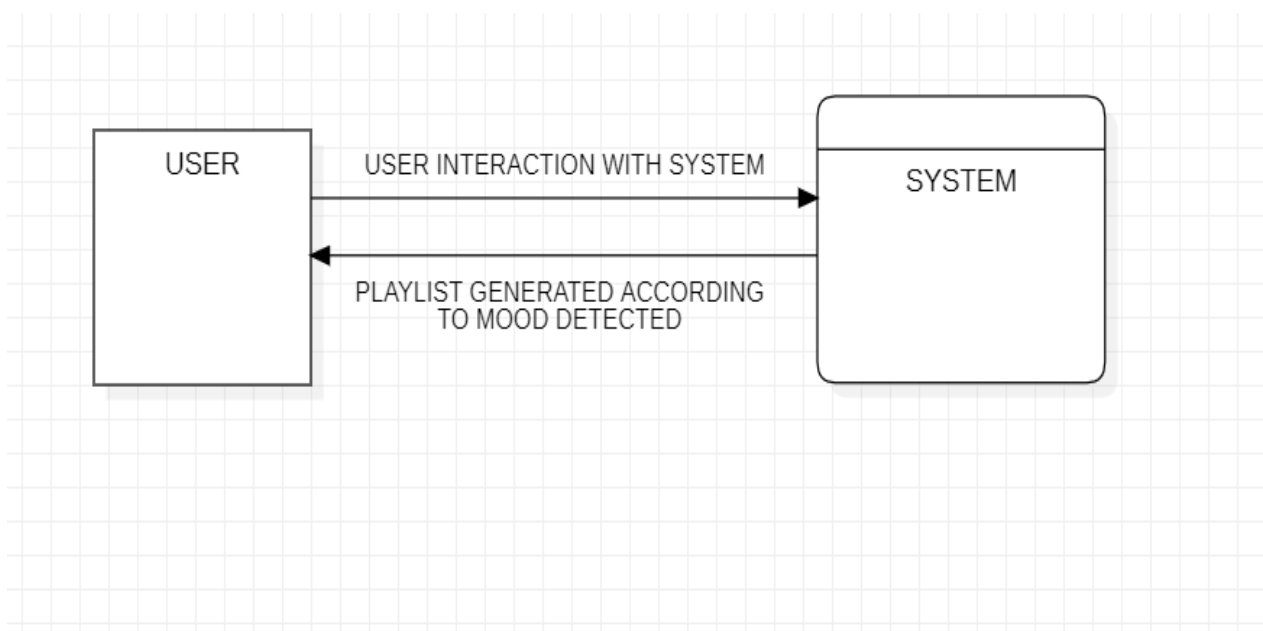


4.3 Conceptual Level Activity Diagram

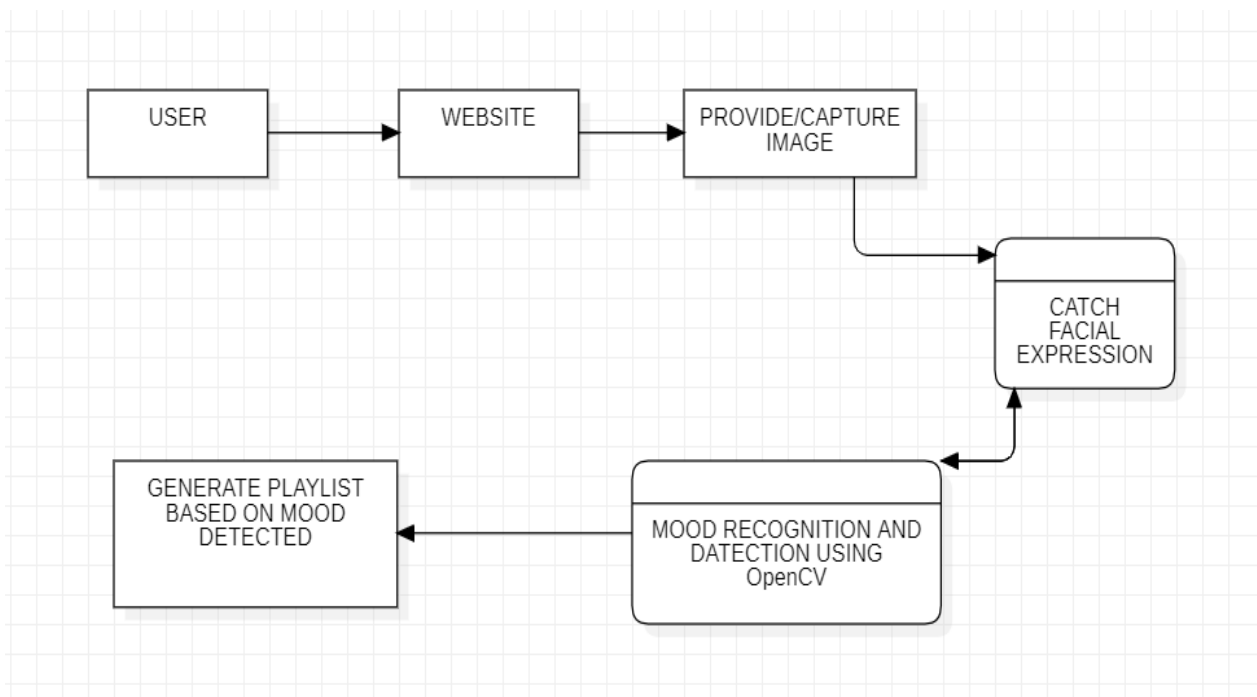


4.4 Data Flow Diagram

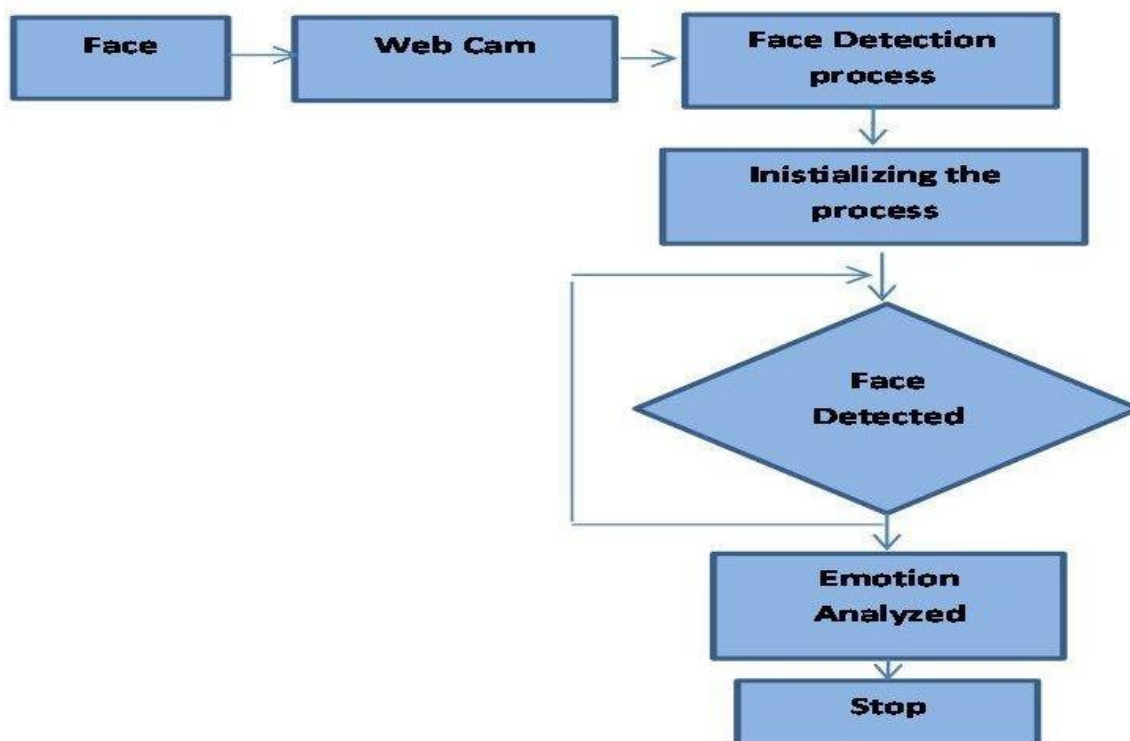
Level 0:



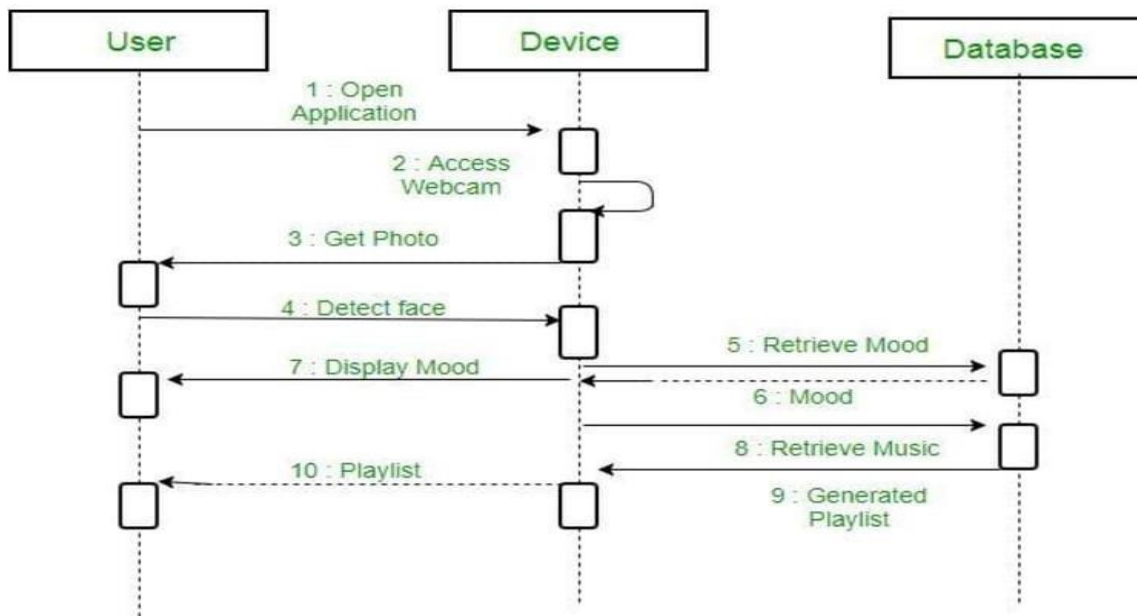
Level 1:



4.5 Data Design (ER-Diagram)



4.6 Sequence Diagram



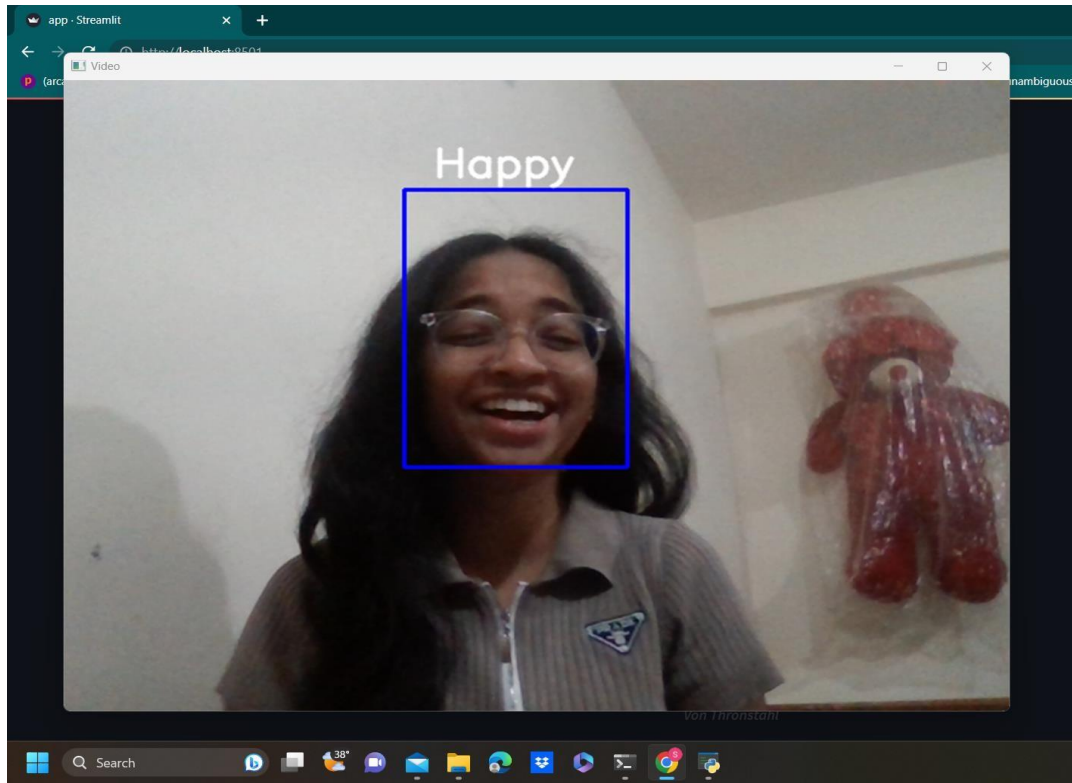
4.7 Testing

Testing a Song Prediction System can be done in several ways:

1. **Usability Testing:** This type of testing focuses on evaluating how user-friendly the system is. It involves asking users to perform certain tasks, such as searching for a song based on their current mood, and observing their behavior and feedback. Usability testing can provide valuable insights into how well the system's interface and features meet users' needs and expectations.
2. **A/B Testing:** A/B testing is a method of comparing two different versions of a system to see which one performs better. In the context of a Song Prediction System, this could involve presenting users with different song recommendations based on the same emotional input and measuring which set of recommendations results in more engagement, longer listening times, or higher user satisfaction.
3. **Expert Evaluation:** Expert evaluation involves having domain experts evaluate the system to determine its effectiveness in meeting its intended purpose. In the case of a Song Prediction System that uses human emotions, domain experts could be music therapists, psychologists, or other professionals with knowledge of the relationship between music and emotions. They could evaluate the system based on factors such as the accuracy and relevance of the recommendations, the quality of the emotional analysis, and the overall user experience.
4. **User Surveys:** User surveys can be used to gather feedback on the Song Prediction System from a large group of users. Surveys can ask questions about the accuracy of the recommendations, the relevance of the emotional analysis, the overall usability of the system, and any suggestions or improvements users may have.

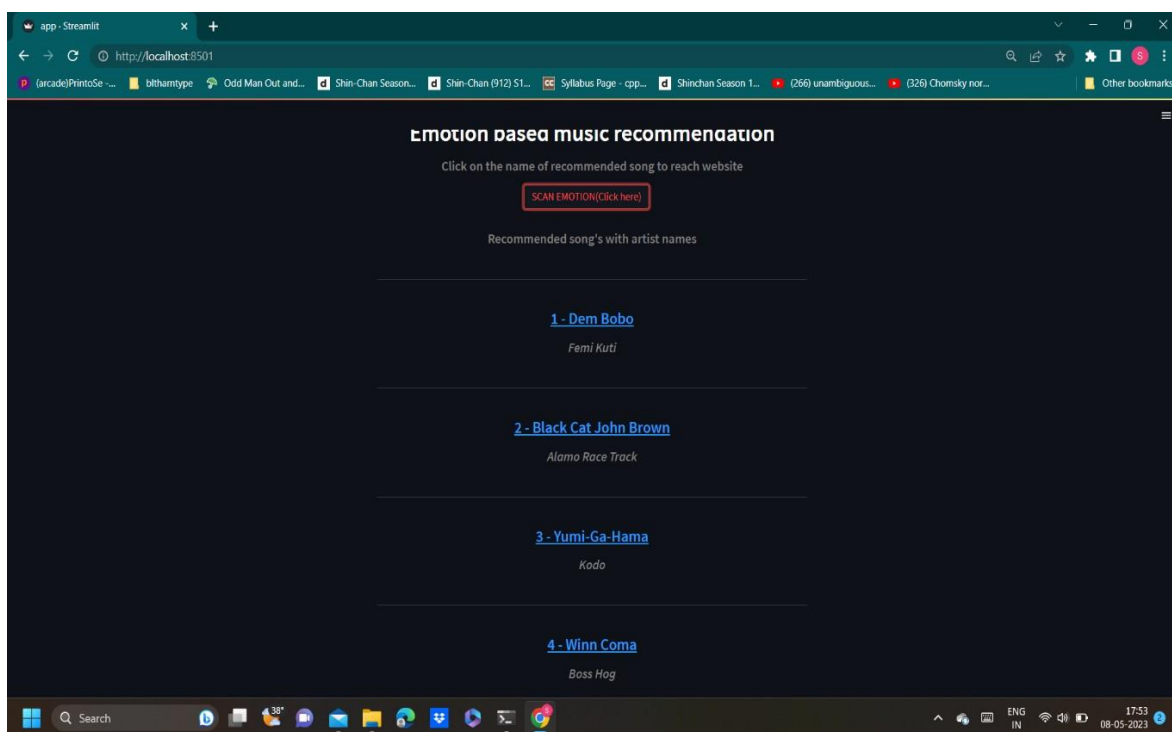
Test Cases

1. Emotion detected: - “Happy ”

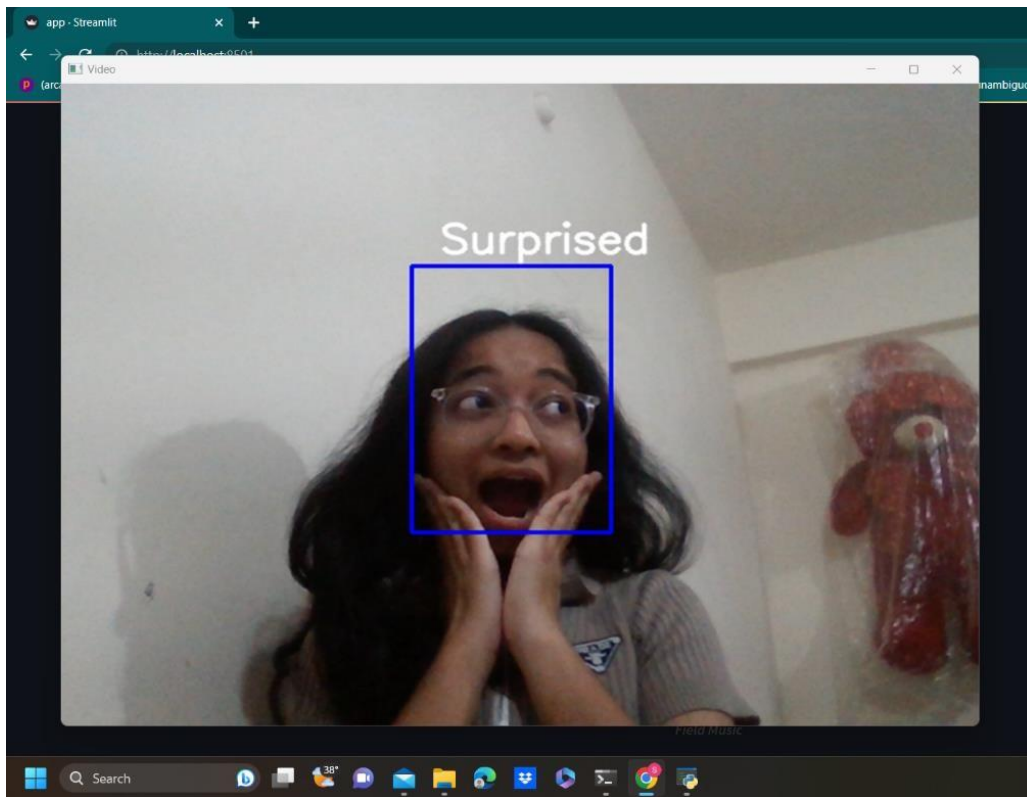


OUTPUT: -

System recommended the songs based on the emotions detected.

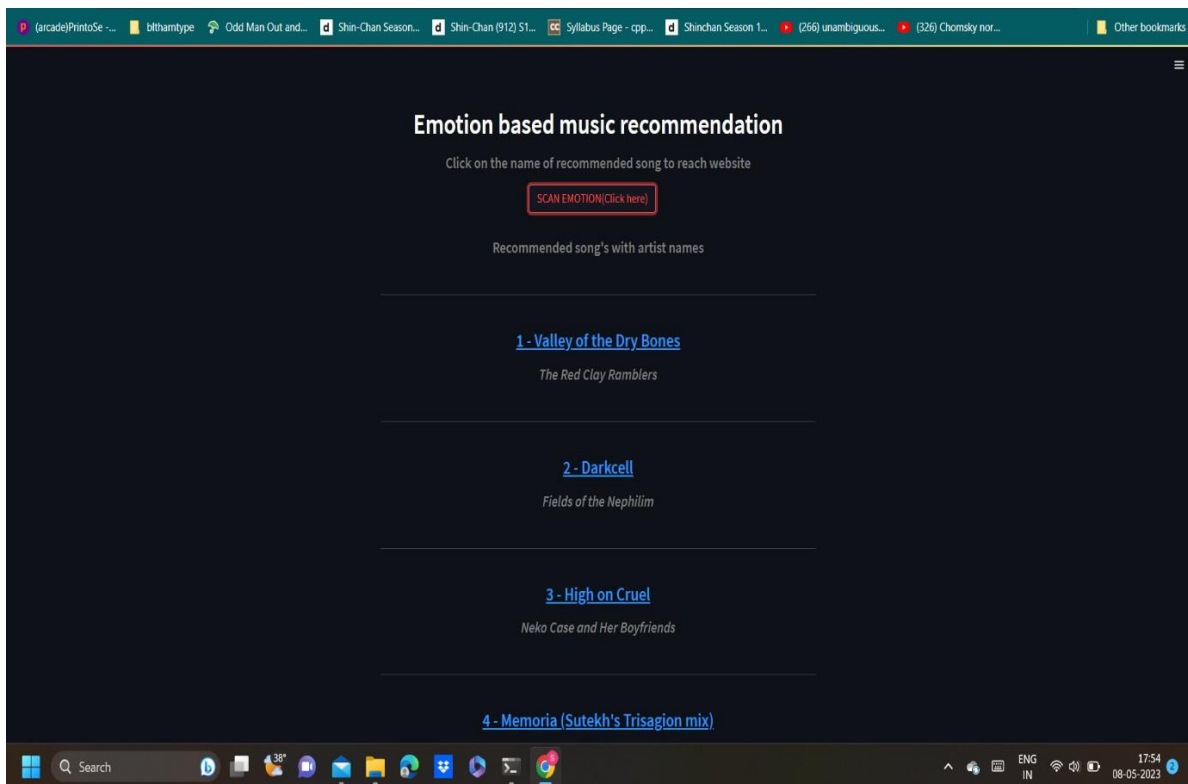


2. Emotion detected: - “Surprised”

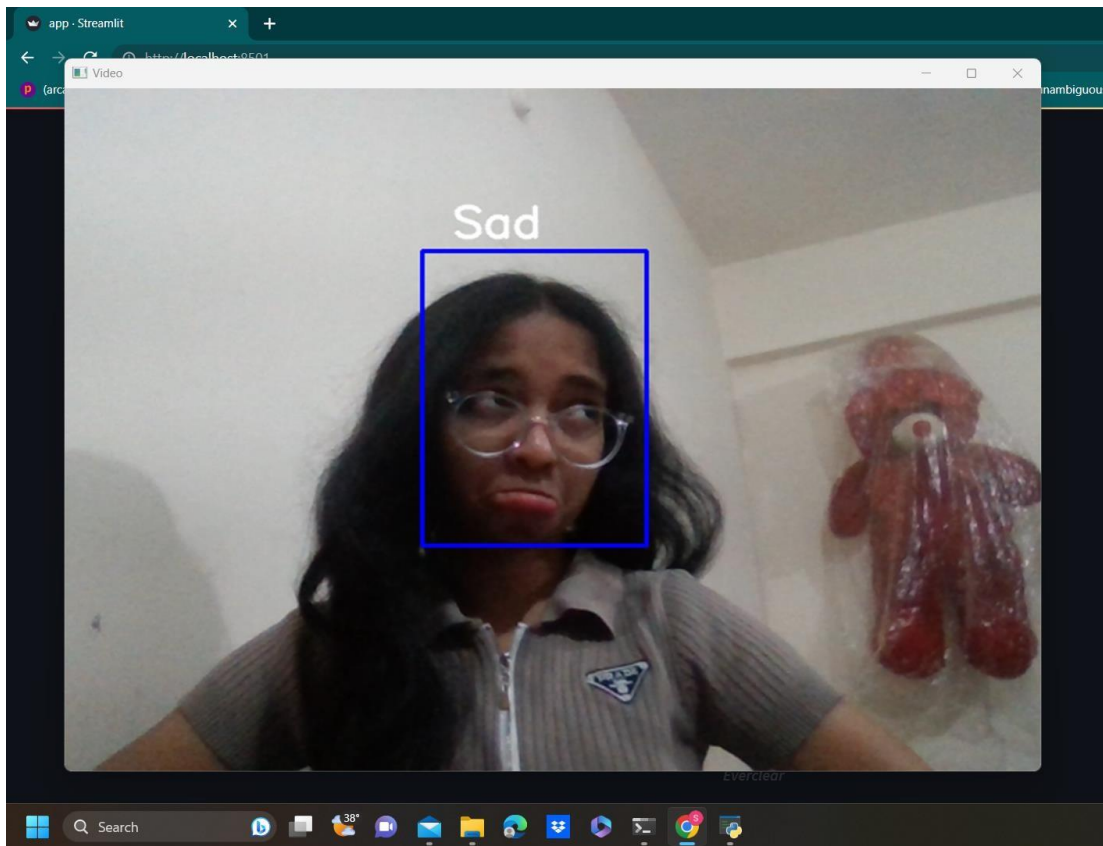


OUTPUT: -

System recommended the songs based on the emotions detected.

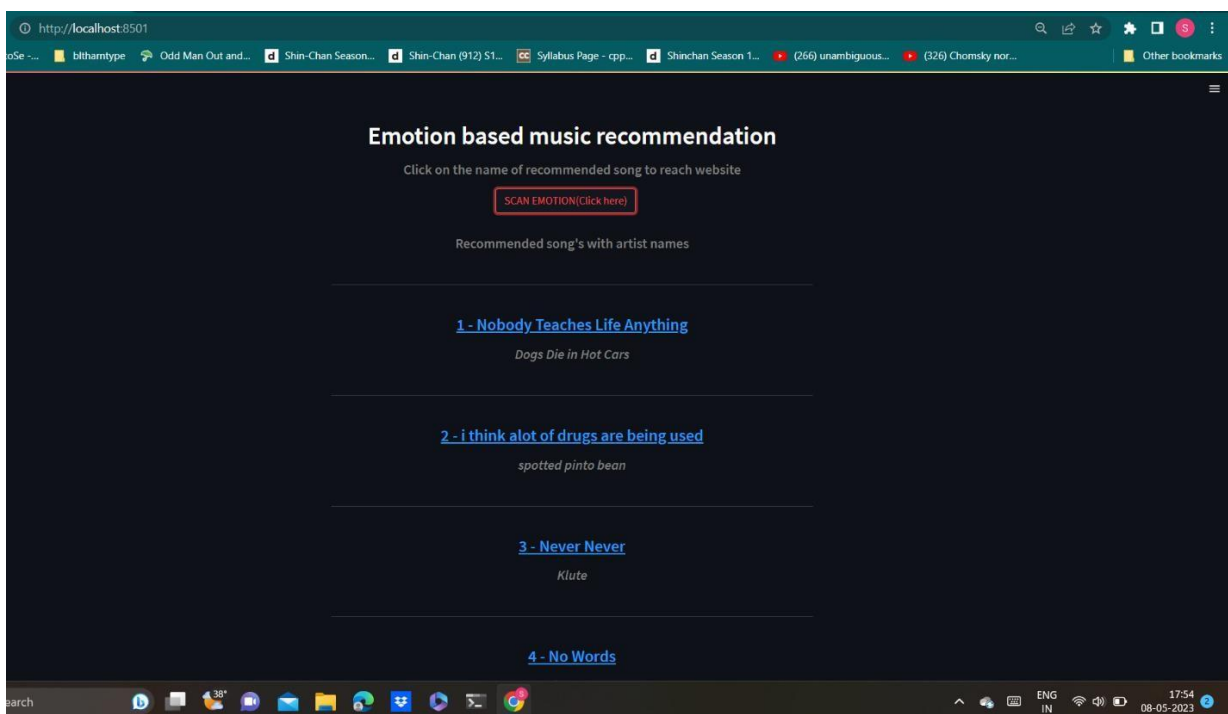


3. Emotion detected: - “Sad ”

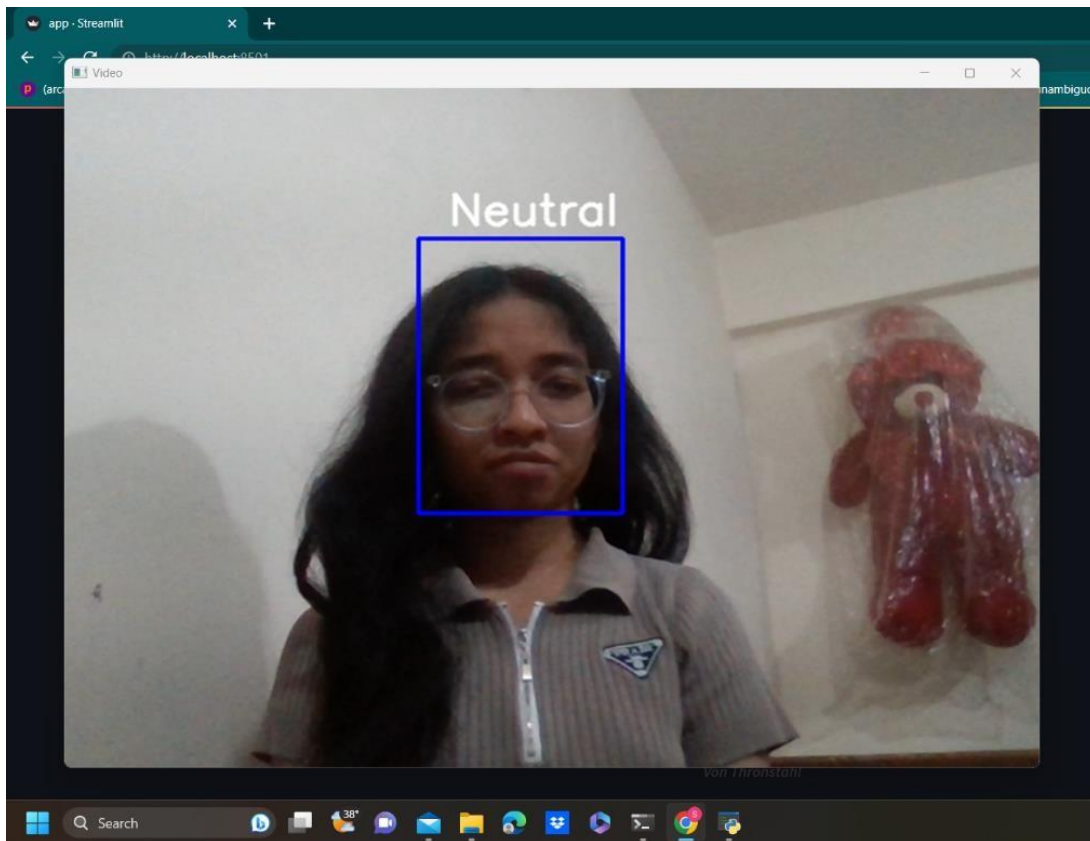


OUTPUT: -

System recommended the songs based on the emotions detected.

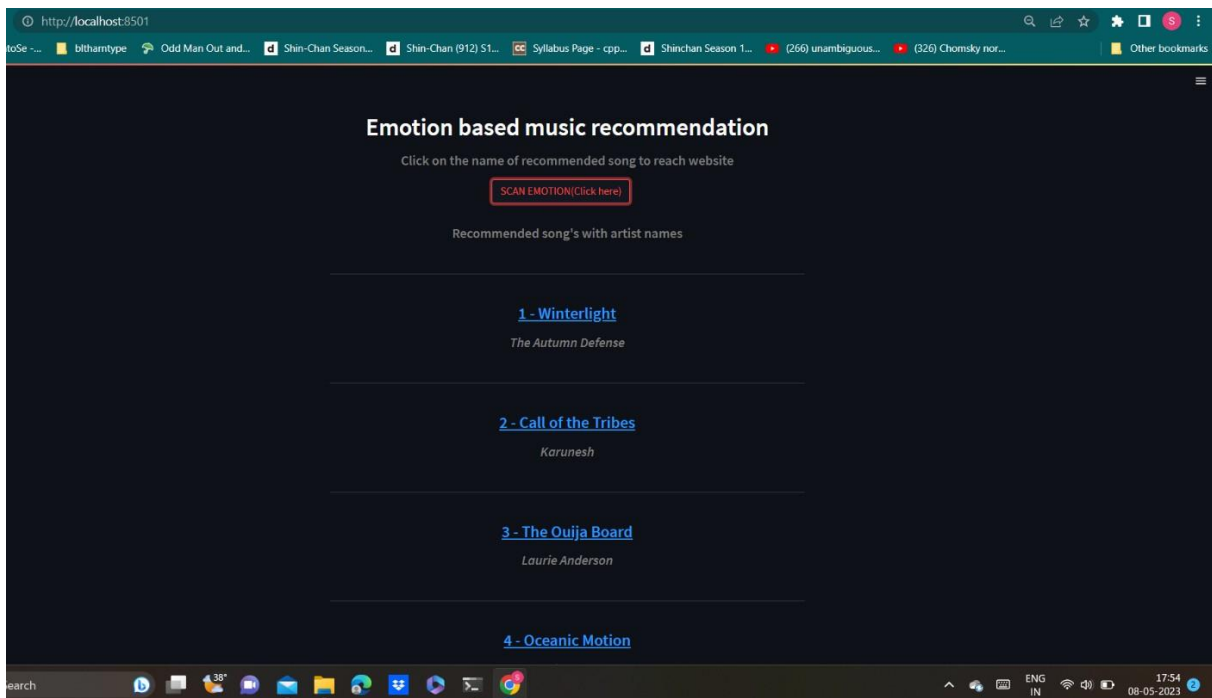


4. Emotion detected: - “Neutral”

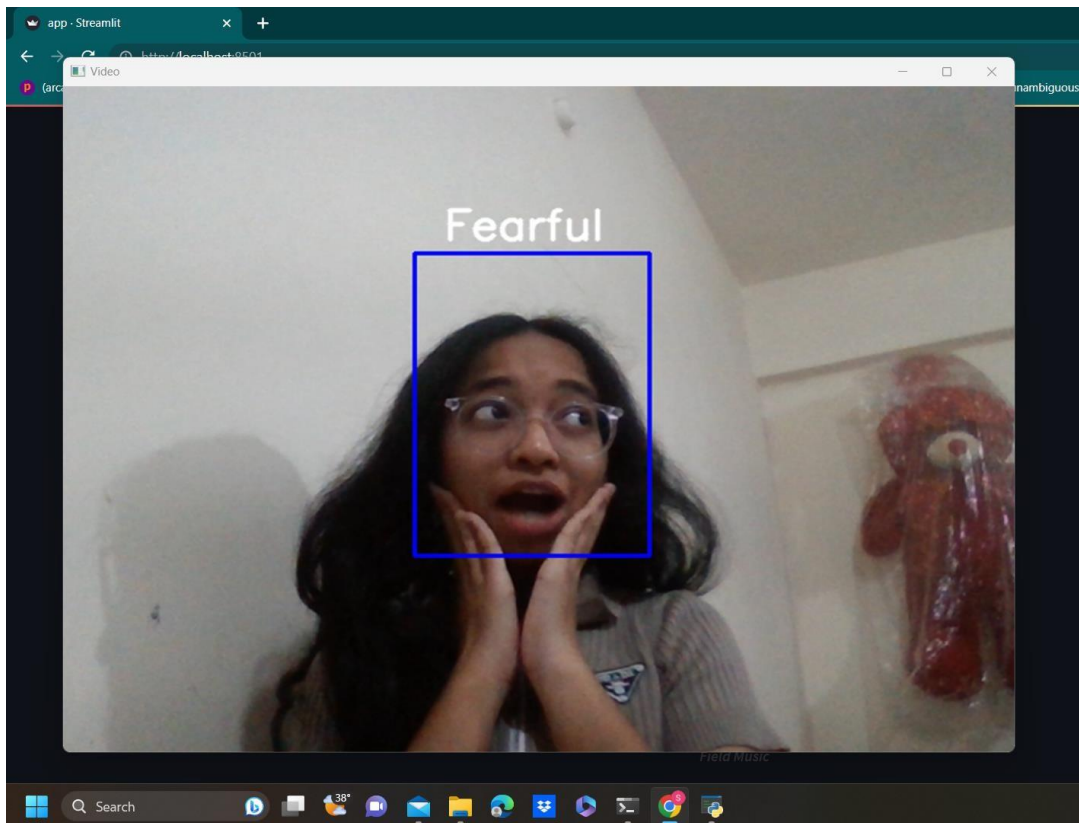


OUTPUT: -

System recommended the songs based on the emotions detected.

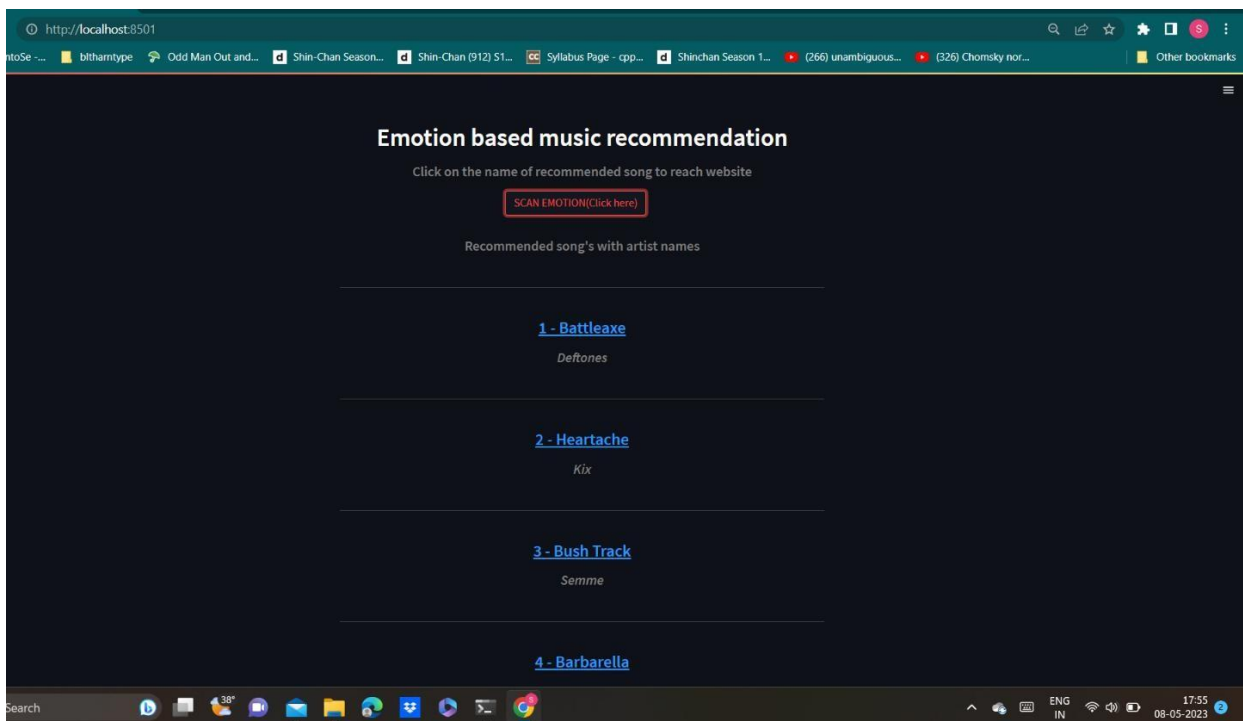


5. Emotion detected: - “ Fearful ”



OUTPUT: -

System recommended the songs based on the emotions detected.



Chapter 5 – CONCLUSION & FUTURE WORK

5.1 Limitation of Project

Here are some limitations of a Song Recommendation System:

1. **Limited emotional detection accuracy:** The accuracy of the system's emotional analysis depends on the quality of the algorithms used to detect emotions from user input. Emotion detection algorithms are not perfect and can have limitations, such as not being able to accurately detect subtle or complex emotions.
2. **Limited music database:** The effectiveness of the system's song recommendations depends on the size and diversity of the music database used by the system. If the music database is limited, the system may not be able to provide accurate recommendations for all emotions or may recommend the same songs repeatedly.
3. **Individual variability:** Human emotions can be highly variable and subjective, which can make it difficult for the system to accurately detect and interpret them. What one person considers happy or sad may differ from what another person considers happy or sad.
4. **Privacy concerns:** To detect emotions, the system may need to collect and analyze personal data from users, such as their facial expressions or voice recordings. This can raise privacy concerns for some users and may limit the number of people willing to use the system.
5. **User dependency:** The effectiveness of the system's recommendations relies heavily on the user providing accurate emotional input. If the user is unable to accurately express their emotions or does not input their emotions at all, the system may not be able to provide relevant recommendations.

5.2 Future Enhancement

Here are some potential future enhancements for a Song Recommendation System:

1. **Integration with wearable devices:** Future versions of the system could integrate with wearable devices that measure biometric data, such as heart rate, to further enhance emotional analysis. By incorporating this additional data, the system may be able to provide even more accurate recommendations.
2. **Personalized recommendations:** The system could be enhanced to provide personalized recommendations based on the user's past listening history and preferences. By analyzing a user's music listening behavior, the system could learn their individual tastes and provide recommendations that better match their preferences.
3. **Multi-modal input:** The system could be enhanced to allow users to input their emotions through multiple modalities, such as speech, text, or gestures. This would provide users with more flexibility in how they express their emotions, potentially leading to more accurate emotional analysis.
4. **Integration with social media:** Future versions of the system could integrate with social media platforms to analyze a user's posts, comments, and interactions for emotional content. By incorporating this data, the system could better understand a user's emotional state and provide more accurate recommendations.
5. **Expansion of music database:** To provide more diverse and accurate recommendations, the system could be enhanced to expand its music database to include music from more genres, languages, and cultures.

Chapter 6 - BIBLIOGRAPHY & REFERENCES

6.1 Reference Books

Here are some reference books that could be helpful for developing a Song Recommendation System:

1. "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking" by Foster Provost and Tom Fawcett - This book provides an introduction to data mining and predictive analytics, which are key components of building a recommendation system.
2. "Programming Collective Intelligence: Building Smart Web 2.0 Applications" by Toby Segaran - This book covers machine learning techniques and algorithms, which are essential for building a recommendation system.
3. "Machine Learning: A Probabilistic Perspective" by Kevin Murphy - This book provides a detailed introduction to machine learning algorithms and probabilistic models, which are important for building recommendation systems that can learn and adapt over time.
4. "Applied Predictive Modeling" by Max Kuhn and Kjell Johnson - This book covers the practical aspects of building predictive models, including data preparation, feature selection, model selection, and evaluation.
5. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili - This book provides a practical introduction to machine learning using the Python programming language, which is commonly used for building recommendation systems.

6.2 Other Documentations & Resources

Here are some examples of the additional documentation and resources I mentioned earlier:

1. Datasets:

- Million Song Dataset: <http://millionsongdataset.com/>
- FMA: A Dataset For Music Analysis: <https://github.com/mdeff/fma>
- EmoReact: A Multimodal Dataset for Emotion Recognition in Music Videos:

<https://zenodo.org/record/3561314#.YJ0PXVUzaM9>

2. Code repositories:

- TensorFlow Music:

https://github.com/tensorflow/magenta/tree/master/magenta/models/music_vae

- PyTorch-VAE: <https://github.com/AntixK/PyTorch-VAE>

- Music Recommender System: <https://github.com/madisonmay/Music-Recommender-System>

3. Blog articles and tutorials:

- How to Build a Song Recommender Using Deep Learning:

<https://towardsdatascience.com/how-to-build-a-song-recommender-using-deep-learning-keras-and-tensorflow-3a451f6c7a6f>

- Emotion Detection in Music: A Tutorial: <https://towardsdatascience.com/emotion-detection-in-music-a-tutorial-1d347b38f29b>

- Building a Music Recommendation System with Python:

<https://www.analyticsvidhya.com/blog/2021/02/building-a-music-recommendation-system-with-python/>

4. Online communities and groups:

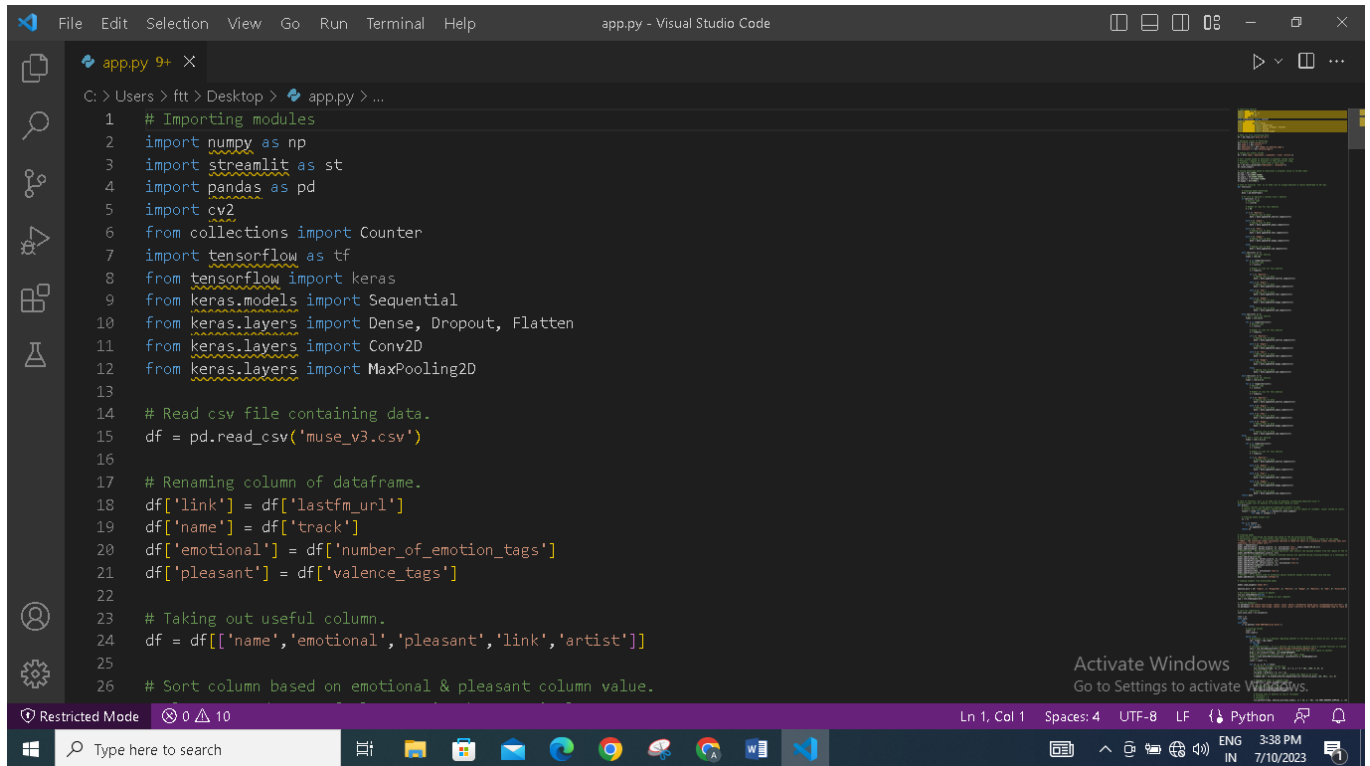
- Reddit's r/MachineLearning: <https://www.reddit.com/r/MachineLearning/>

- Facebook's Music Recommendation Systems group:

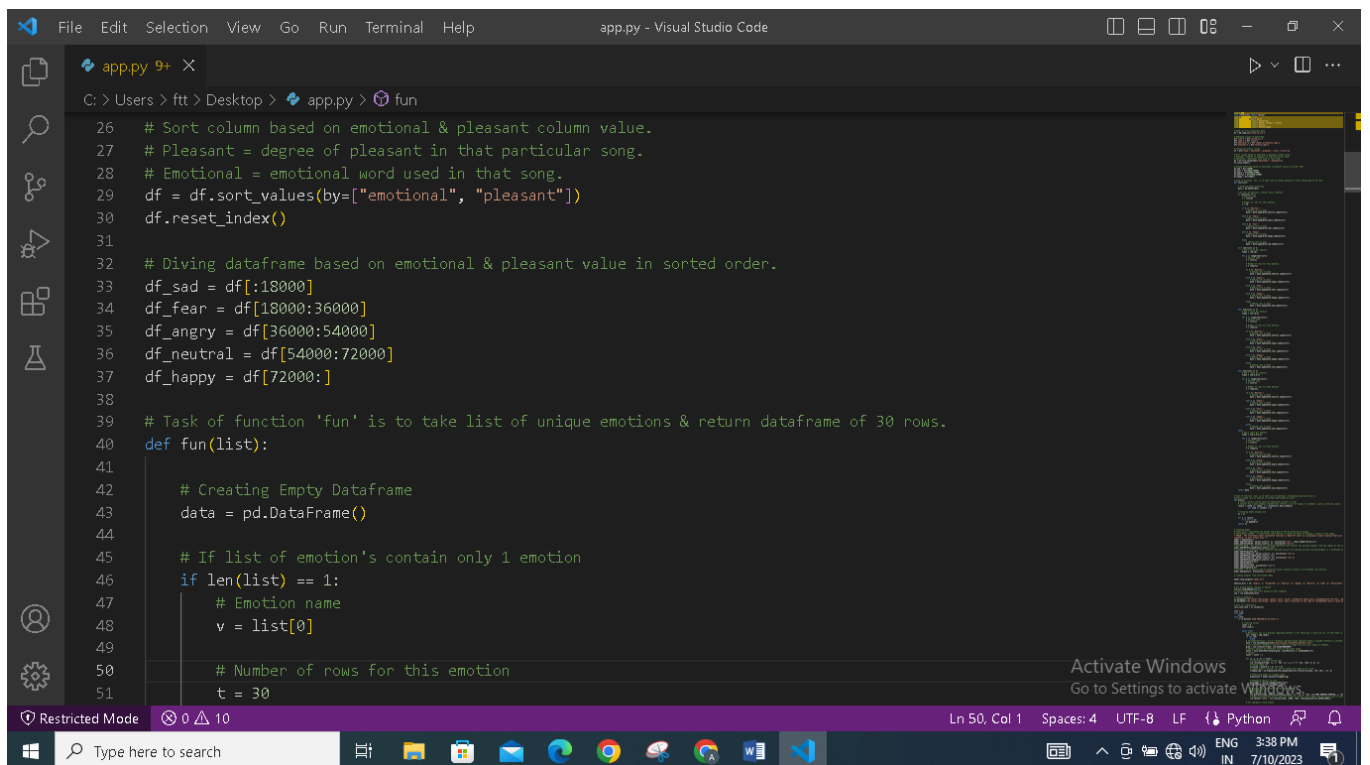
<https://www.facebook.com/groups/504960876523322/>

- Slack's Music Technology group: <https://musictectonics.slack.com/>

6.3 Snapshot of python code:



```
app.py 9+ X
C:\> Users > ftt > Desktop > app.py > ...
1  # Importing modules
2  import numpy as np
3  import streamlit as st
4  import pandas as pd
5  import cv2
6  from collections import Counter
7  import tensorflow as tf
8  from tensorflow import keras
9  from keras.models import Sequential
10 from keras.layers import Dense, Dropout, Flatten
11 from keras.layers import Conv2D
12 from keras.layers import MaxPooling2D
13
14 # Read csv file containing data.
15 df = pd.read_csv('muse_v3.csv')
16
17 # Renaming column of dataframe.
18 df['link'] = df['lastfm_url']
19 df['name'] = df['track']
20 df['emotional'] = df['number_of_emotion_tags']
21 df['pleasant'] = df['valence_tags']
22
23 # Taking out useful column.
24 df = df[['name', 'emotional', 'pleasant', 'link', 'artist']]
25
26 # Sort column based on emotional & pleasant column value.
```



```
app.py 9+ X
C:\> Users > ftt > Desktop > app.py > fun
26 # Sort column based on emotional & pleasant column value.
27 # Pleasant = degree of pleasant in that particular song.
28 # Emotional = emotional word used in that song.
29 df = df.sort_values(by=["emotional", "pleasant"])
30 df.reset_index()
31
32 # Diving dataframe based on emotional & pleasant value in sorted order.
33 df_sad = df[:18000]
34 df_fear = df[18000:36000]
35 df_angry = df[36000:54000]
36 df_neutral = df[54000:72000]
37 df_happy = df[72000:]
38
39 # Task of function 'fun' is to take list of unique emotions & return dataframe of 30 rows.
40 def fun(list):
41     # Creating Empty Dataframe
42     data = pd.DataFrame()
43
44     # If list of emotion's contain only 1 emotion
45     if len(list) == 1:
46         # Emotion name
47         v = list[0]
48
49         # Number of rows for this emotion
50         t = 30
```

File Edit Selection View Go Run Terminal Help app.py - Visual Studio Code

app.py 9+ X

C:\> Users > fit > Desktop > app.py > fun

```
50 # Number of rows for this emotion
51 t = 30
52
53 if v == 'Neutral':
54     # Adding rows to data
55     data = data.append(df_neutral.sample(n=t))
56
57 elif v == 'Angry':
58     # Adding rows to data
59     data = data.append(df_angry.sample(n=t))
60
61 elif v == 'fear':
62     # Adding rows to data
63     data = data.append(df_fear.sample(n=t))
64
65 elif v == 'happy':
66     # Adding rows to data
67     data = data.append(df_happy.sample(n=t))
68
69 else:
70     # Adding rows to data
71     data = data.append(df_sad.sample(n=t))
72
73 elif len(list) == 2:
74     # Row's count per emotion
75     times = [20,10]
```

Activate Windows
Go to Settings to activate Windows.

Restricted Mode 0 10 Ln 74, Col 1 Spaces: 4 UTF-8 LF Python ENG IN 7/10/2023

File Edit Selection View Go Run Terminal Help app.py - Visual Studio Code

app.py 9+ X

C:\> Users > fit > Desktop > app.py > fun

```
76
77 for i in range(len(list)):
78     # Emotion name
79     v = list[i]
80
81     # Number of rows for this emotion
82     t = times[i]
83
84     if v == 'Neutral':
85         # Adding rows to data
86         data = data.append(df_neutral.sample(n=t))
87
88     elif v == 'Angry':
89         # Adding rows to data
90         data = data.append(df_angry.sample(n=t))
91
92     elif v == 'fear':
93         # Adding rows to data
94         data = data.append(df_fear.sample(n=t))
95
96     elif v == 'happy':
97         # Adding rows to data
98         data = data.append(df_happy.sample(n=t))
99
100 else:
101     # Adding rows to data
```

Activate Windows
Go to Settings to activate Windows.

Restricted Mode 0 10 Ln 100, Col 1 Spaces: 4 UTF-8 LF Python ENG IN 3:39 PM 7/10/2023


```

153
154     elif v == 'fear':
155         # Adding rows to data
156         data = data.append(df_fear.sample(n=t))
157
158     elif v == 'happy':
159         # Adding rows to data
160         data = data.append(df_happy.sample(n=t))
161
162     else:
163         # Adding rows to data
164         data = data.append(df_sad.sample(n=t))
165
166 else:
167     # Row's count per emotion
168     times = [10,7,6,5,2]
169
170     for i in range(len(list)):
171         # Emotion name
172         v = list[i]
173
174         # Number of rows for this emotion
175         t = times[i]
176
177     if v == 'Neutral':
178         # Adding rows to data
179         data = data.append(df_neutral.sample(n=t))

```

Activate Windows
Go to Settings to activate Windows.

Restricted Mode 0 10

Ln 177, Col 1 Spaces: 4 UTF-8 LF Python

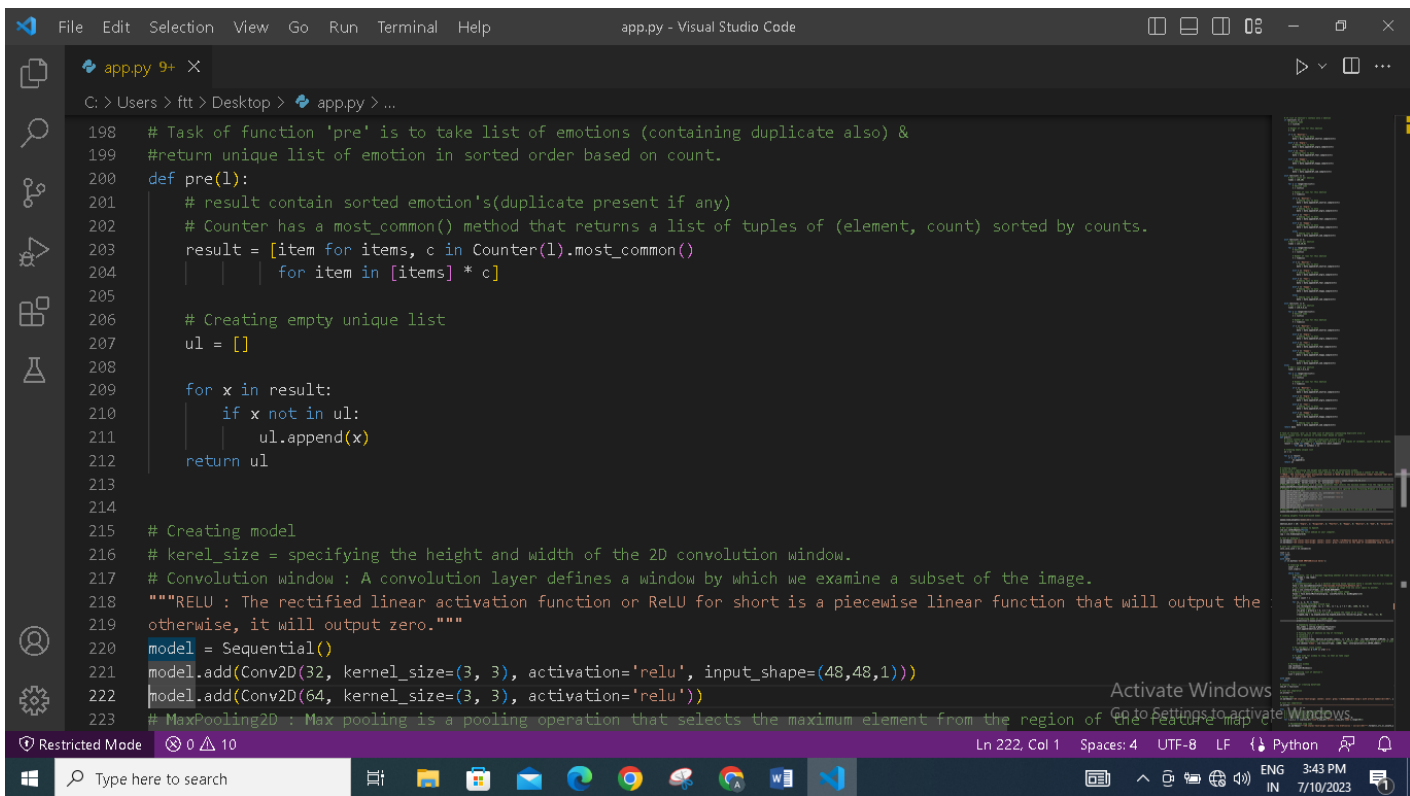
3:41 PM 7/10/2023

The image shows a Visual Studio Code editor window with a Python file named 'app.py'. The code defines a function 'pre' that takes a list of emotions as input and returns a sorted list of unique emotions based on their frequency. The function uses pandas to create a DataFrame, sample data, and then uses 'Counter' to count the occurrences of each emotion. The results are sorted by count and returned as a list of tuples.

```
C:\> Users > ftt > Desktop > app.py > pre

177         # Adding rows to data
178         data = data.append(df_neutral.sample(n=t))
179
180     elif v == 'Angry':
181         # Adding rows to data
182         data = data.append(df_angry.sample(n=t))
183
184     elif v == 'fear':
185         # Adding rows to data
186         data = data.append(df_fear.sample(n=t))
187
188     elif v == 'happy':
189         # Adding rows to data
190         data = data.append(df_happy.sample(n=t))
191
192     else:
193         # Adding rows to data
194         data = data.append(df_sad.sample(n=t))
195     return data
196
197
198 # Task of function 'pre' is to take list of emotions (containing duplicate also) &
199 #return unique list of emotion in sorted order based on count.
200 def pre(l):
201     # result contain sorted emotion's(duplicate present if any)
202     # Counter has a most_common() method that returns a list of tuples of (element, count) sorted by count
```

At the bottom of the editor, there is a status bar showing 'Ln 201, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'Python'. The Windows taskbar is visible at the very bottom, showing the time as 3:42 PM on 7/10/2023.

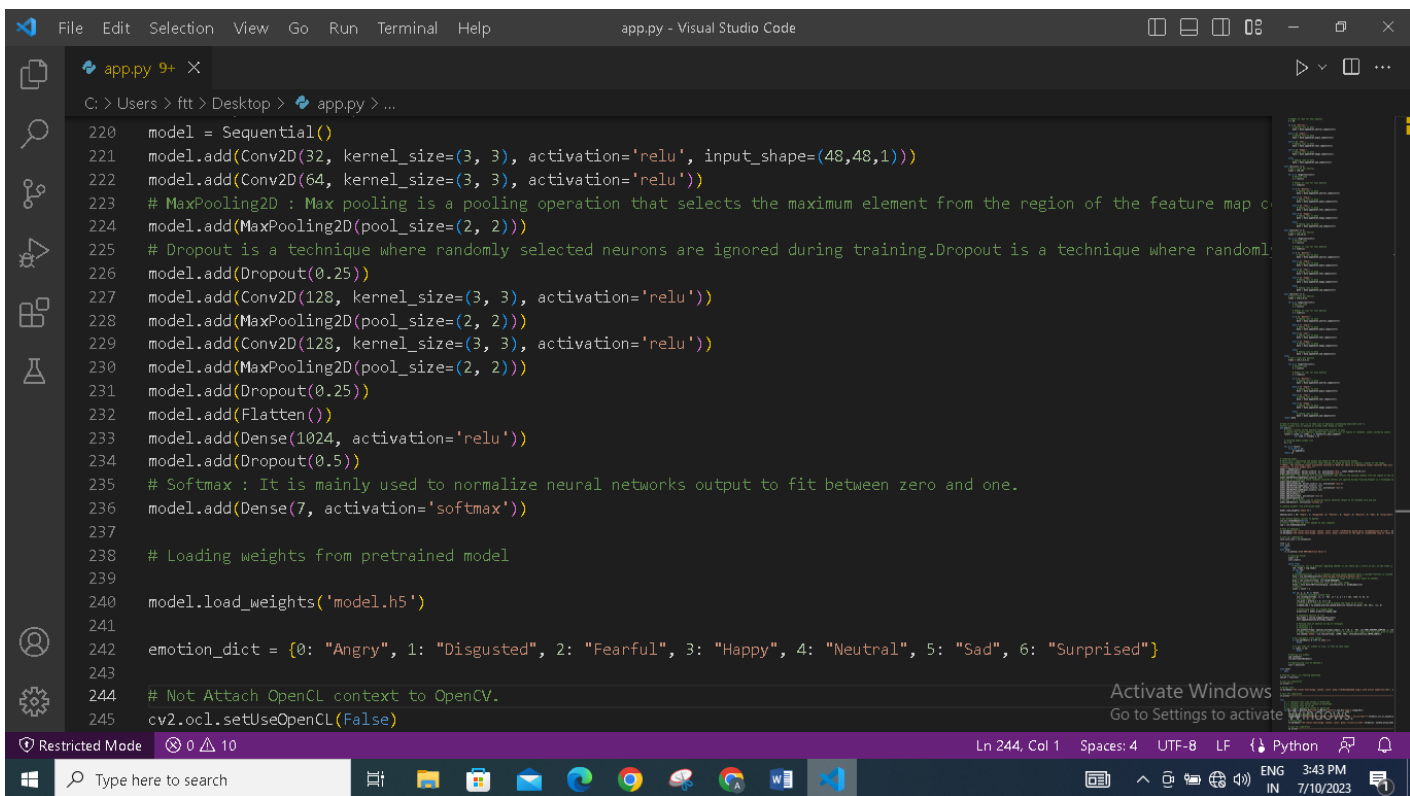


```
File Edit Selection View Go Run Terminal Help
app.py - Visual Studio Code

app.py 9+ X
C:\> Users > ftt > Desktop > app.py > ...

198 # Task of function 'pre' is to take list of emotions (containing duplicate also) &
199 #return unique list of emotion in sorted order based on count.
200 def pre(l):
201     # result contain sorted emotion's(duplicate present if any)
202     # Counter has a most_common() method that returns a list of tuples of (element, count) sorted by counts.
203     result = [item for items, c in Counter(l).most_common()
204               | for item in [items] * c]
205
206     # Creating empty unique list
207     ul = []
208
209     for x in result:
210         if x not in ul:
211             ul.append(x)
212     return ul
213
214
215 # Creating model
216 # kernel_size = specifying the height and width of the 2D convolution window.
217 # Convolution window : A convolution layer defines a window by which we examine a subset of the image.
218 """RELU : The rectified linear activation function or ReLU for short is a piecewise linear function that will output the
219 otherwise, it will output zero."""
220 model = Sequential()
221 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
222 model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
223 # MaxPooling2D : Max pooling is a pooling operation that selects the maximum element from the region of the feature map
```

Ln 222, Col 1 Spaces: 4 UTF-8 LF Python 3:43 PM 7/10/2023

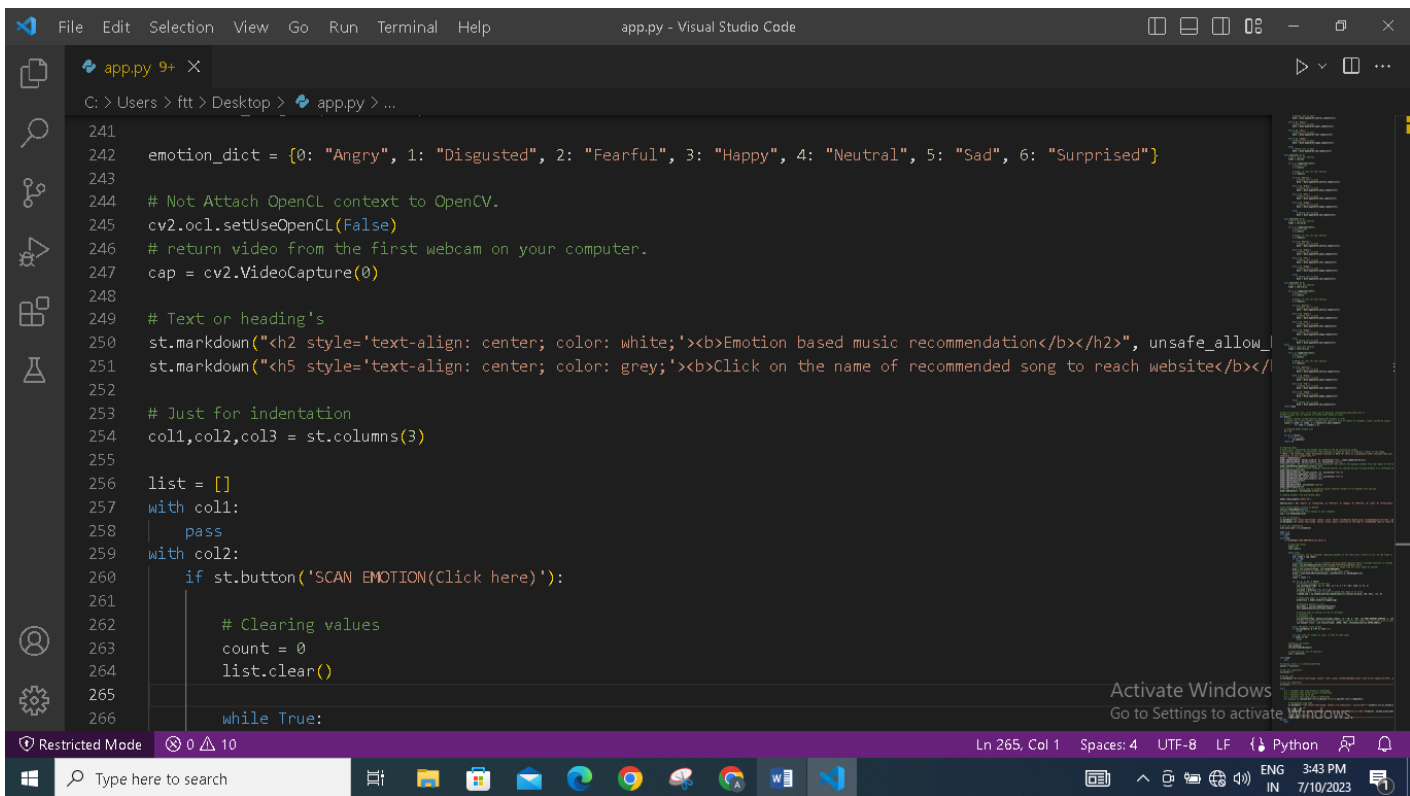


```
File Edit Selection View Go Run Terminal Help
app.py - Visual Studio Code

app.py 9+ X
C:\> Users > ftt > Desktop > app.py > ...

220 model = Sequential()
221 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
222 model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
223 # MaxPooling2D : Max pooling is a pooling operation that selects the maximum element from the region of the feature map
224 model.add(MaxPooling2D(pool_size=(2, 2)))
225 # Dropout is a technique where randomly selected neurons are ignored during training. Dropout is a technique where randomly
226 model.add(Dropout(0.25))
227 model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
228 model.add(MaxPooling2D(pool_size=(2, 2)))
229 model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
230 model.add(MaxPooling2D(pool_size=(2, 2)))
231 model.add(Dropout(0.25))
232 model.add(Flatten())
233 model.add(Dense(1024, activation='relu'))
234 model.add(Dropout(0.5))
235 # Softmax : It is mainly used to normalize neural networks output to fit between zero and one.
236 model.add(Dense(7, activation='softmax'))
237
238 # Loading weights from pretrained model
239
240 model.load_weights('model.h5')
241
242 emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}
243
244 # Not Attach OpenCL context to OpenCV.
245 cv2ocl.setUseOpenCL(False)
```

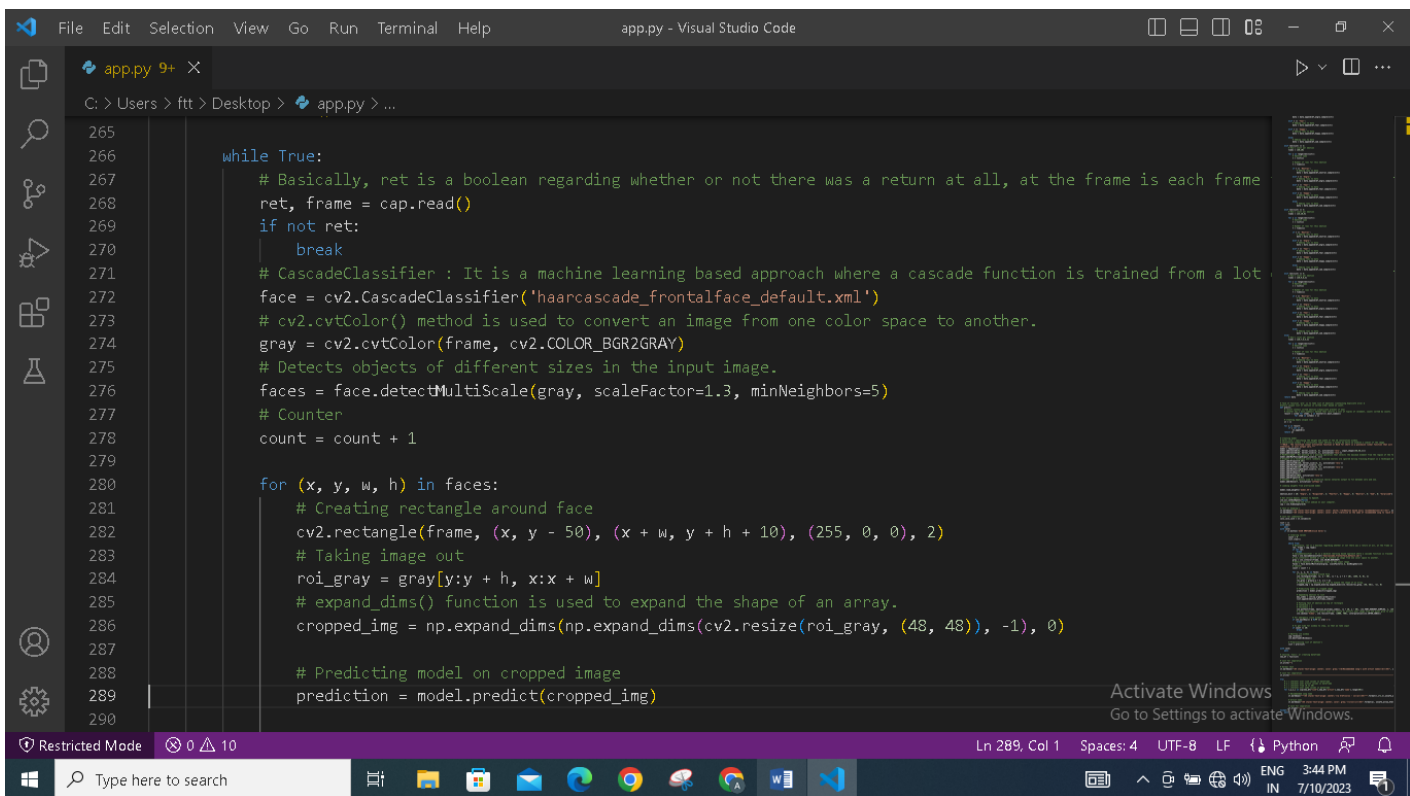
Ln 244, Col 1 Spaces: 4 UTF-8 LF Python 3:43 PM 7/10/2023



```
File Edit Selection View Go Run Terminal Help app.py - Visual Studio Code
C: > Users > ftt > Desktop > app.py > ...
241
242 emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}
243
244 # Not Attach OpenCL context to OpenCV.
245 cv2.oc1.setUseOpenCL(False)
246 # return video from the first webcam on your computer.
247 cap = cv2.VideoCapture(0)
248
249 # Text or heading's
250 st.markdown("<h2 style='text-align: center; color: white;'>b>Emotion based music recommendation</b></h2>", unsafe_allow_
251 st.markdown("<h5 style='text-align: center; color: grey;'>b>Click on the name of recommended song to reach website</b></
252
253 # Just for indentation
254 col1,col2,col3 = st.columns(3)
255
256 list = []
257 with col1:
258     pass
259 with col2:
260     if st.button('SCAN EMOTION(Click here)'):
261
262         # Clearing values
263         count = 0
264         list.clear()
265
266 while True:
```

Activate Windows
Go to Settings to activate Windows.

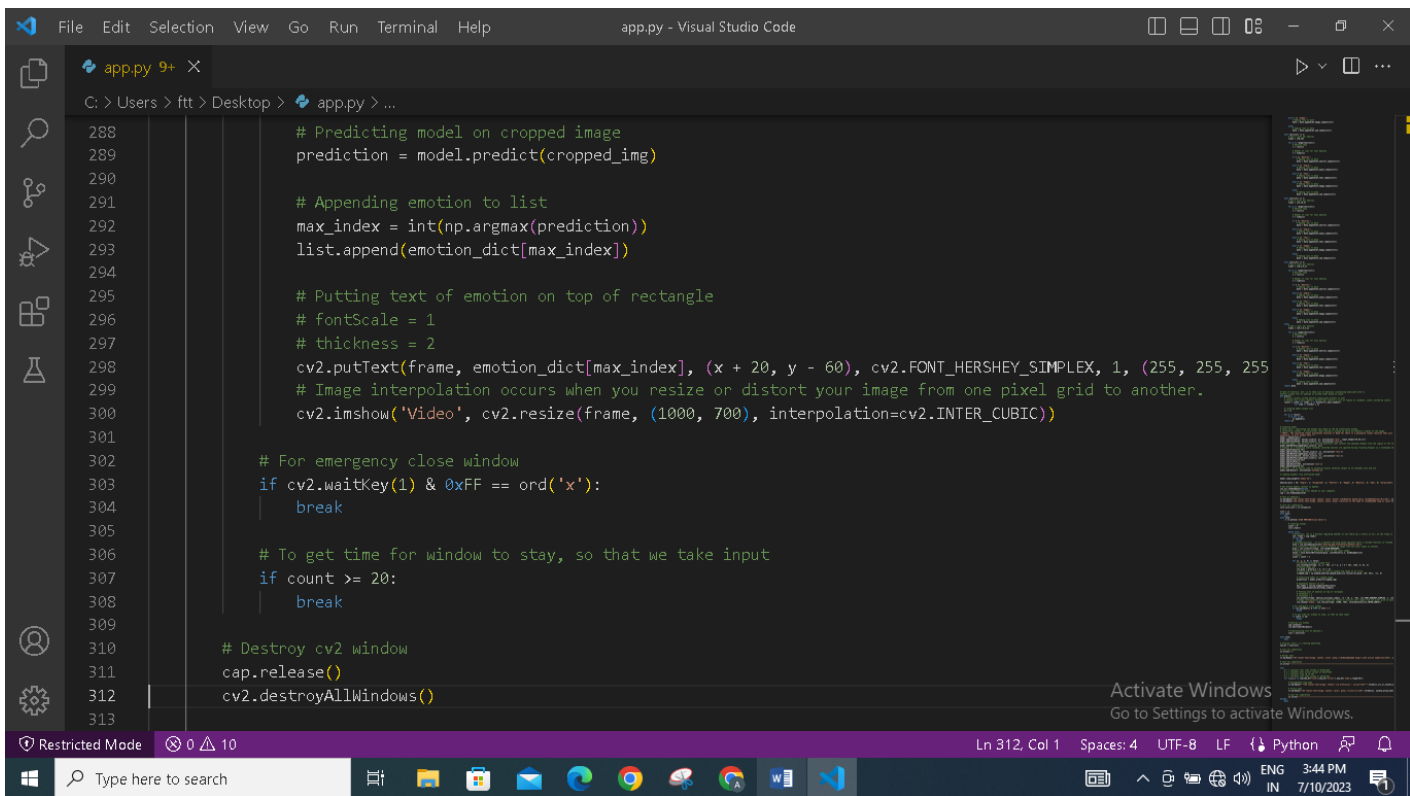
Ln 265, Col 1 Spaces: 4 UTF-8 LF Python ENG IN 3:43 PM 7/10/2023



```
File Edit Selection View Go Run Terminal Help app.py - Visual Studio Code
C: > Users > ftt > Desktop > app.py > ...
265
266 while True:
267     # Basically, ret is a boolean regarding whether or not there was a return at all, at the frame is each frame
268     ret, frame = cap.read()
269     if not ret:
270         break
271     # CascadeClassifier : It is a machine learning based approach where a cascade function is trained from a lot
272     face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
273     # cv2.cvtColor() method is used to convert an image from one color space to another.
274     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
275     # Detects objects of different sizes in the input image.
276     faces = face.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
277     # Counter
278     count = count + 1
279
280     for (x, y, w, h) in faces:
281         # Creating rectangle around face
282         cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
283         # Taking image out
284         roi_gray = gray[y:y + h, x:x + w]
285         # expand_dims() function is used to expand the shape of an array.
286         cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
287
288         # Predicting model on cropped image
289         prediction = model.predict(cropped_img)
290
```

Activate Windows
Go to Settings to activate Windows.

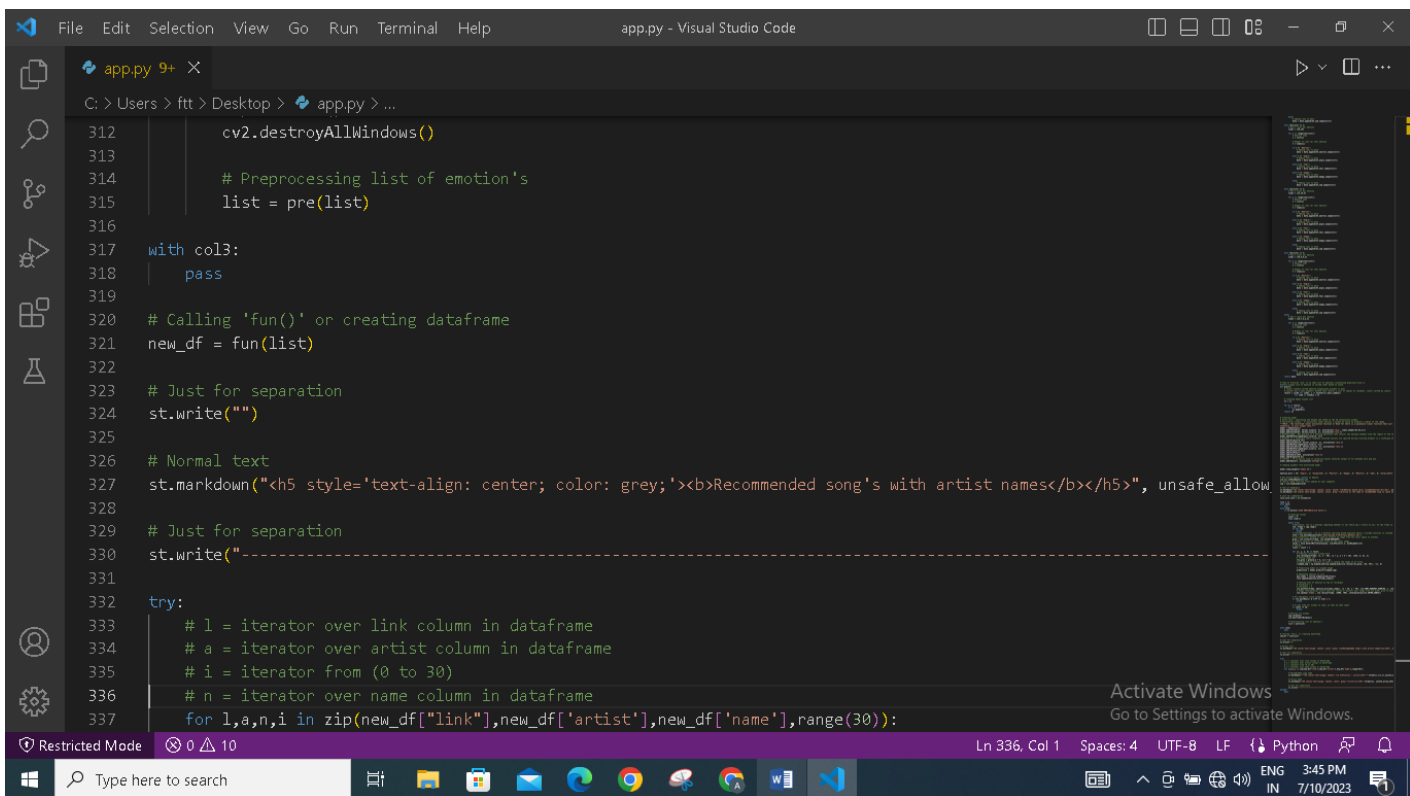
Ln 289, Col 1 Spaces: 4 UTF-8 LF Python ENG IN 3:44 PM 7/10/2023



The screenshot shows the Visual Studio Code editor with a Python file named `app.py`. The code is currently at line 313. The code includes comments and logic for predicting emotions on a cropped image, appending the result to a list, and displaying the emotion on a video frame. It also includes a loop to wait for a key press and a final cleanup of the video window.

```
288 # Predicting model on cropped image
289 prediction = model.predict(cropped_img)
290
291 # Appending emotion to list
292 max_index = int(np.argmax(prediction))
293 list.append(emotion_dict[max_index])
294
295 # Putting text of emotion on top of rectangle
296 # fontScale = 1
297 # thickness = 2
298 cv2.putText(frame, emotion_dict[max_index], (x + 20, y - 60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255)
299 # Image interpolation occurs when you resize or distort your image from one pixel grid to another.
300 cv2.imshow('Video', cv2.resize(frame, (1000, 700), interpolation=cv2.INTER_CUBIC))
301
302 # For emergency close window
303 if cv2.waitKey(1) & 0xFF == ord('x'):
304     break
305
306 # To get time for window to stay, so that we take input
307 if count >= 20:
308     break
309
310 # Destroy cv2 window
311 cap.release()
312 cv2.destroyAllWindows()
```

The status bar at the bottom indicates the file is in Restricted Mode, with 0 errors and 10 warnings. The cursor is at line 312, column 1. The system tray shows the time as 3:44 PM on 7/10/2023.



The screenshot shows the Visual Studio Code editor with the same Python file `app.py`, now at line 337. The code continues with preprocessing the list of emotions, creating a new dataframe, and writing the results to a web page. It includes a try block to iterate over the data and generate HTML output.

```
312 cv2.destroyAllWindows()
313
314 # Preprocessing list of emotion's
315 list = pre(list)
316
317 with col3:
318     pass
319
320 # Calling 'fun()' or creating dataframe
321 new_df = fun(list)
322
323 # Just for separation
324 st.write("")
325
326 # Normal text
327 st.markdown("<h5 style='text-align: center; color: grey;'><b>Recommended song's with artist names</b></h5>", unsafe_allow
328
329 # Just for separation
330 st.write("-----")
331
332 try:
333     # l = iterator over link column in dataframe
334     # a = iterator over artist column in dataframe
335     # i = iterator from (0 to 30)
336     # n = iterator over name column in dataframe
337     for l,a,n,i in zip(new_df["link"],new_df["artist"],new_df["name"],range(30)):
```

The status bar at the bottom indicates the file is in Restricted Mode, with 0 errors and 10 warnings. The cursor is at line 337, column 1. The system tray shows the time as 3:45 PM on 7/10/2023.

```
File Edit Selection View Go Run Terminal Help app.py - Visual Studio Code
C:\Users> ftt > Desktop > app.py > ...
329 # Just for separation
330 st.write("-----")
331
332 try:
333     # l = iterator over link column in dataframe
334     # a = iterator over artist column in dataframe
335     # i = iterator from (0 to 30)
336     # n = iterator over name column in dataframe
337     for l,a,n,i in zip(new_df["link"],new_df['artist'],new_df['name'],range(30)):
338
339         # Recommended song name
340         st.markdown("""<h4 style='text-align: center;'><a href={}>{} - {}</a></h4>""".format(l,i+1,n),unsafe_allow_html=T
341
342         # Artist name
343         st.markdown("<h5 style='text-align: center; color: grey;'><i>{}</i></h5>".format(a), unsafe_allow_html=True)
344
345         # Just for separation
346         st.write("-----")
347 except:
348     pass
349
350
351
352
353
```

Activate Windows
Go to Settings to activate Windows.

Restricted Mode 0 10 Ln 353, Col 1 Spaces: 4 UTF-8 LF Python 3:45 PM 7/10/2023

Thank You