

# **The Groove Connection**

## **A Project Report**

Submitted in partial fulfillment of the  
Requirements for the award of the Degree of  
**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

**Poornimaa Shanmuganathan**

**Under the esteemed guidance of**

**Ms. Nahid Shaikh**

**Assistant Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Shree Narayan Guru College of Commerce**

*(Affiliated to University of Mumbai)*

**Mumbai, 400089**

**MAHARASHTRA**

**2022-23**

## PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No.: .....

Roll no: **30**

1. Name of the Student: Poornima Shanmuganathan

2. Title of the Project: The Groove Connection

3. Name of the Guide: Nahid Shaikh

4. Teaching experience of the Guide: 7 year

5. Is this your first submission?    Yes ☒    No ☐

Signature of the Student

Signature of the Guide

Date: .....

Date: .....

Signature of the Coordinator

Date: .....

**Shree Narayan Guru College of Commerce**  
*(Affiliated to University of Mumbai)*  
**MUMBAI-MAHARASHTRA-400089**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that the project entitled, "**The Groove Connection**", is Bonafide work of **Poornima Shanmuganathan** bearing Seat.No: \_\_\_\_\_ submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

**Internal Guide**

**Coordinator**

**External Examiner**

**Date:**

**College Seal**

# **ABSTRACT**

Music plays a very important role in a human's daily life. Everyone wants to listen to music of their individual taste, mostly based on their mood. Users always face the task of manually browsing the music and to create a playlist based on their current mood.

The proposed project is very efficient which generates a music playlist based on the current mood of users. Facial expressions are the best way of expressing the ongoing mood of the person. The objective of this project is to suggest songs for users based on their mood by capturing facial expressions.

Facial expressions are captured through a webcam and such expressions are fed into a learning algorithm which gives the most probable emotion. Once the emotion is recognized, the system suggests a play-list for that emotion, thus saving a lot of time for a user. Once the emotion is detected by CNN algorithm then the data will be transferred in you tube search query then the you tube will generates a song list according to the facial expression of the user.

# ACKNOWLEDGEMENT

It is a privilege for us to have been associated with Ms Nahid Shaikh, our guide, during this project work. We have been greatly benefited by their valuable suggestions and ideas.

I would also like to extend my gratitude to the principal “Dr.Ravindran Karathadi” and our IT Coordinator Mrs. Tulsi Adal for her support and facilities provided to us for the same. I take this opportunity to thank all our classmates for their company during the course work and for useful discussion we had with them.

I would also like to thank all those who directly and indirectly helped in completion of this project. I would be failing in our duties if we do not make a mention of our family members including our parents for providing moral support, without which this work would not have been completed.

Date:

Place:

Completed

by,

Poornima Shanmuganathan

# DECLARATION

I hereby declare that the project entitled, “**The Groove Connection**” done at **SREE NARAYANA GURU COLLEGE OF CHEMBUR**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as a final semester project as part of our curriculum.

**Poornima Shanmuganathan**

# TABLE OF CONTENTS

Serial no	Topic	Page no
<b>1.</b>	<b>Introduction</b>	
1.1	Background	
1.2	Objective of the project	
1.3	Purpose	
1.4	Scope	
1.5	Applicability	
<b>2.</b>	<b>Survey Of Technology</b>	
2.1	Existing System	
2.2	Technology	
<b>3.</b>	<b>Requirements And Analysis</b>	
3.1	Problem Definition	
3.2	Requirement Specification	
3.3	Planning and Scheduling	
3.4	Software and Hardware requirement	
3.5	Preliminary Product Design	
<b>4.</b>	<b>System Design</b>	
4.1	Basic Modules	
4.2	Data Design	
4.3	System Design	
<b>5.</b>	<b>Implementation and Testing</b>	

<b>5.1</b>	<b>Code (Place Core segments)</b>	
<b>5.2</b>	<b>Testing Approach</b>	
<b>5.2.1</b>	<b>Unit Testing (Test cases and Test Results)</b>	
<b>5.2.2</b>	<b>Integration System (Test cases and Test Results)</b>	
<b>6</b>	<b>Results and Discussions (Output Screens)</b>	
<b>7</b>	<b>Conclusion and Future Work</b>	
<b>8</b>	<b>References</b>	



# **CHAPTER 1: INTRODUCTION**

## **Introduction**

Music plays an important role in our daily life. Users have to face the task of manually browsing the music.

Computer vision involves seeing or sensing a visual stimulus, making sense of what it has seen and also extracting complex information that could be used for other machine learning activities.

We will implement our use case using the Haar Cascade classifier. Haar Cascade classifier is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper. This project recognizes the facial expressions of users and plays songs according to emotion. Facial expressions are the best way of expressing the mood of a person. The facial expressions are captured using a webcam and face detection is done by using Haar cascade classifier. The captured image is input to CNN which learns features and these features are analyzed to determine the current emotion of the user then the music will be played according to the emotion. In this project, seven emotions are considered for classification which includes 'Angry', 'Happy', 'Neutral', 'Sad', 'Surprise'

This project consists of 4 modules-face detection, feature extraction, emotion detection, songs classification. Face detection is done by Haar cascade classifier, feature extraction and emotion detection are done by CNN. Finally, the songs are played according to the emotion recognized.

## **1.1 Background**

As we get the information from various research papers and other sources we analyze that many peoples have difficulties to find the songs which are the same as their emotion. People search songs according to their mood to play their suitable song but this will consume a lot of time that their mood spoils and frustrated arbitrarily.

So to solve this problem, I came up with a web application which will detect your emotions and play songs according to their mood. This is time saving and very fun to use.

User service is extremely important. We want each user to have a pleasant experience, and it is the intention of our staff to answer questions with expertise and to offer advice that I feel is needed.

## **1.2 Objectives**

The project aims to lighten the mood of the user, by playing 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.

By using Facial Emotion Recognition , Businesses can process images and videos in real-time for monitoring video feeds. Automating video analytics, thus saving costs and making life better for their users.

Listening to music is an easy way to alter mood or relieve stress.

Music increases memory and retention as well as maximizes learning capabilities.

## **1.3 Purpose, Scope, and Applicability**

### **1.3.1 Purpose**

As a music listener, I've always felt that music players should do way more things than just playing songs and allowing users to make play-lists. A music playlist should be intelligent and act in keeping with the user's preferences.

A music playlist should help users organize and list the songs automatically without putting much effort into selection and re-organization of songs. The Generating Playlist Using Facial Expressions provides a better platform to any or all the music listeners, and ensures automation of generating a playlist according to the emotion of the user.

This helps users to generate a playlist according to their moods. It recommends the top playlist based on the mood of the user using Youtube.

### **1.3.2 Scope**

Music plays an important part of our life. It gives us relief and reduces stress.

The goal of this project is to generate a music playlist based on the facial expressions of the users.

This discusses how convolutional neural networks (CNN) is used for generating music playlist according to the facial expressions of the user.

### **1.3.3 Applicability**

- The project is web based application in python.
- “The Groove Connection” is a web-based application that allows the user to play songs according to their mood.

## **CHAPTER 2: SURVEY OF TECHNOLOGIES**



## **This project is developed by using this technologies**

**HTML:** HTML (Hyper Text Markup Language) is the code that is used to structure a web page and its content

**CSS:** CSS (Cascading Style Sheets) is used to style and layout web page

**JavaScript:** Javascript is to create dynamic and interactive web content like applications and browsers

**Python:** Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis

**MySQL:** MySQL is a tool used to manage databases and servers, so while it's not a database, it's widely used in relation to managing and organizing data in data.

**Flask:** Flask is used for developing web applications using python. It is a light-weight python web framework that encourage rapid development and clean, pragmatic design.

### **Library used in this project:**

**Keras :**Keras is a high-level neural network library that runs on top of TensorFlow. TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks.

**OpenCV:**OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.

**NumPy:**NumPy is an open-source library available in Python, which helps in mathematical,scientific, engineering, and data science programming.

## **CHAPTER 3: REQUIREMENTS AND ANALYSIS**

### **3.1 Problem Definition**

Music is an essential part of human life. Music is the pleasant sound that leads us to experience harmony and higher happiness. With the advancement in technology, music has significantly progressed and increased in terms of quality and volume.

The type of music people create and listen to differs according to place and culture. The taste of music even differs from person to person and even in the moods of the same person. So, it is very useful if we could determine some method to find what kind of music a person might be interested in listening to and use this finding to recommend music to him/her. Collaborative filtering is one the most popular filtering techniques today and with this project we aim to enhance it.

For this, we have assumed that the personality of the user might be one of the key factors in his/her music listening habit. Hence, via this project, we are to see if the personality of the user might have any impact on the collaborative filtering enhancement assuming the correlation between the personality and music listening habit exists.

## **3.2 Requirements Specification**

### **3.2.1 HARDWARE REQUIREMENT**

The database server needed to have large capacity which will store all the information about the datasets, model to train and other functional or non-functional database activities . Thus the requirement of data base server is as follows:

PROCESSOR 2.5 GHz to 4.0GHz

MEMORY 4 GB

HARD DISK 500 GB

MONITOR ANY STANDARD

KEYBOARD ANY STANDARD

### **3.2.2 SOFTWARE REQUIREMENT**

We are using Python to initialize the actual application because it is a good tool with enough versatility for both face/emotion detection and youtube API and it require. The database system uses SQL server system so we need MySQL server database and its software.

OPERATING SYSTEM WINDOWS 10

LANGUAGE PYTHON

DATABASE MYSQL

BROWSER ANY OF CHROME, MOZILLA , OPERA

### **3.3 Planning and Scheduling**

It is concerned with development of projects for investment.

It identifies and addresses the task required for accomplishment of objectives.

After approving of my project topic I started my project planning.

It requires one or more than one week for requirement analysis.

It requires two weeks for analyzing the hardware requirement and software requirement.

It requires two weeks for studying the existing system.

It requires four to five weeks for detailed planning of the proposed system.

It takes two to three weeks to analyze whether the proposed system is feasible or not.

Scheduling is lying out the actual jobs of the project in the time order in which they have to be performed.

After planning has to be done we start with making the logic diagrams according to the requirement.

[illegible]

### 3.4 Software and Hardware Requirements

**Operating system Windows 10:** Windows 10 features built-in capabilities that allow corporate IT departments to use (MDM) software.

**Python:** Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of Whitespace.

**Flask:** *Flask* is used for developing web applications using python. Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

**Jinja template :**It is a text-based template language and thus can be used to generate any markup as well as source code. The Jinja template engine allows customization of tags, filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects.

**Keras:** Keras is an Open-Source Neural Network library written in Python that runs on top of Theano or TensorFlow. It is designed to be modular, fast and easy to use.

**OpenCV:** OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. OpenCV supports some models from deep

learning frameworks like TensorFlow, Torch, PyTorch according to a defined list of supported layers.

**Database Mysql SQLAlchemy:** MySQL is a relational database management system based on SQL – Structured Query Language. SQLAlchemy provides ways to interact with several database engines such as SQLite, MySQL, and PostgreSQL. It gives you access to the database's SQL functionalities. It also gives you an Object Relational Mapper (ORM), which allows you to make queries and handle data using simple Python objects and methods.

**Browser of Chrome:** Chrome is designed to be the fastest web browser. With one click, it loads web pages, multiple tabs, and applications with lightning speed. Chrome is fitted with V8, a faster and more powerful JavaScript engine. Chrome also loads web pages faster by using the WebKit open source rendering engine.

**Web browser :**A web browser takes you anywhere on the internet. It retrieves information from other parts of the web and displays it on your desktop or mobile device.

**Datasets Details:**The "*Face expression recognition dataset*" was prepared by Jonathan Oheix (Owner).The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image.This dataset consists of 35,887 grayscale images.

**CNN Algorithm:**A convolutional neural network (CNN) is a type of artificial neural network used primarily for image recognition and processing, due to its ability to recognize patterns in images. A CNN is a powerful tool but requires millions of labelled data points for training.



## **Hardware requirements:**

Processor: AMD Ryzen 3 3250U with Radeon Graphics (2.60 GHz)

RAM: 8.00 GB (5.94 GB usable)

HDD: 100 GB or higher

Architecture: 32-bit or 64-bit

Monitor: 15'' or 17'' color monitor

Mouse: Scroll or optical mouse

Keyboard: Standard 110 keys keyboard

### **3.5 Preliminary Product Description**

#### **Limitation Of Existing System**

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

i. Doesn't detect motion properly ii. Party Shuffle iii. Playlists iv. Music squares where the 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, the user has to face the task of manually browsing through the playlist of songs and selecting songs based on his current mood and behavior. That is the requirements of an individual, a user sporadically suffers through the need and desire of browsing through his playlist, according to his mood and emotions.

## **Features Of Proposed System**

**No Need Manual Browsing:** The user doesn't need to manually browse the songs based on his /her current mood and behavior which will save the user's time and emotions.

**Automatically Generate Playlist:** User's don't need to select any emotion subject(happy,sad,neutral and angry). Our system will automatically generate a playlist according to the retrieved emotions.

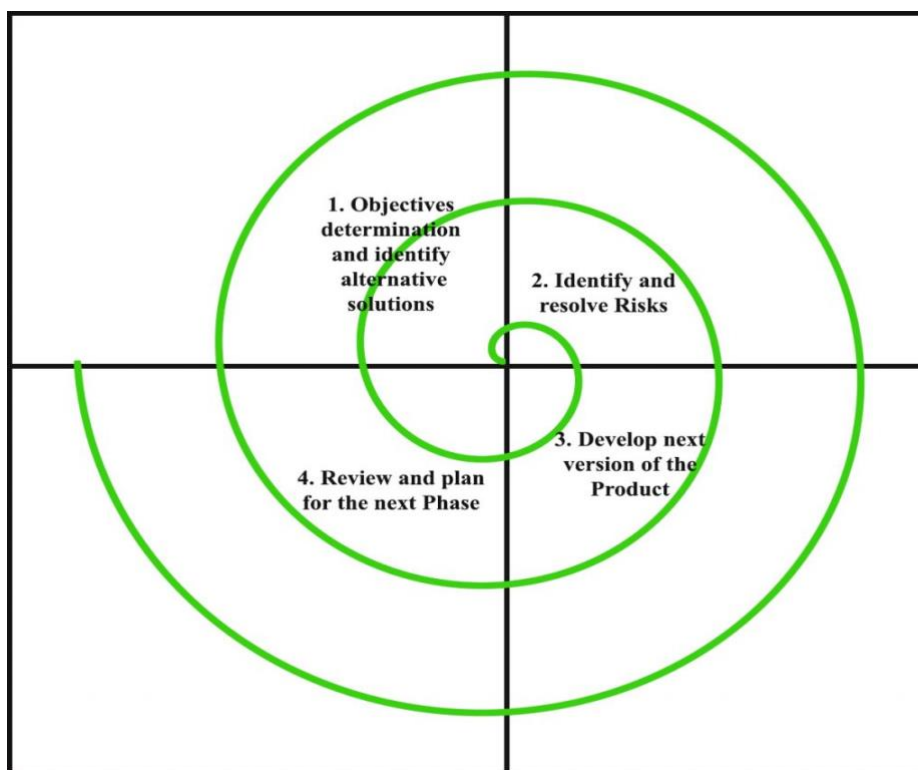
**User Friendly:** The user's don't have to face any complexity while using our system. It will be user friendly and super easy to use.

### 3.6 Conceptual Models

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling.

In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.

Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.



- Requirement analysis
- Design
- Coding
- Testing and risk analysis

## Requirement Analysis

The spiral model process starts with collecting business needs. In this, the following spirals will include the documentation of system requirements, unit requirements, and the subsystem needs. In this stage, we can easily understand the system requirements because the business analyst and the client have constant communication.

## Design

The second stage of the spiral model is designed, where we will plan the logical design, architectural design, flow charts, decision tree, and so on.

## Coding

After the compilation of the design stage, we will move to our next step, which is the coding stage. In this, we will develop the product based on the client's requirement and getting the client's feedback as well. This stage refers to the construction of the real application in every cycle.

The design details of an application are known as the build with having version numbers. After that, these builds are transferred to the client for their responses.

## Testing and Risk Analysis

Once the development is completed successfully, we will test the build at the end of the first cycle and also analyze the risk of the software on the different aspects such as managing risks, detecting, and observing the technical feasibility. And after that, the client will test the application and give feedback.

### 3.6.1 USE CASE DIAGRAM:

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external system
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system

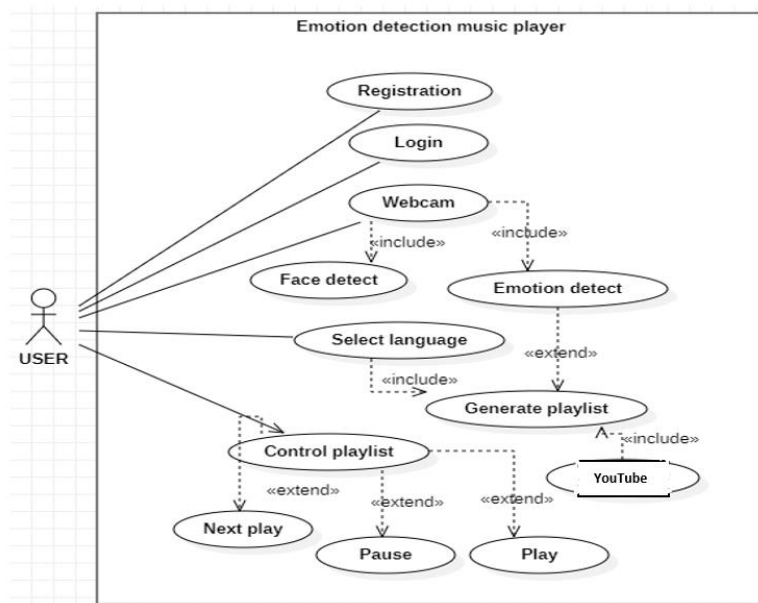
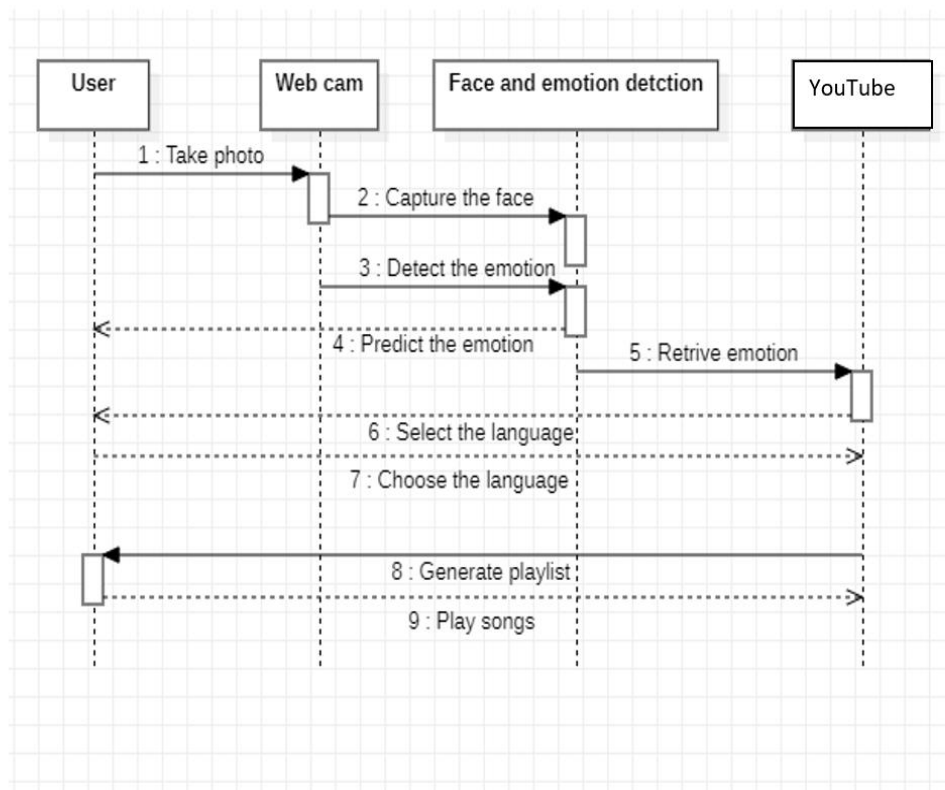


Figure 1 Use case diagram

### 3.6.2 SEQUENCE DIAGRAM:

In the Unified Modelling Language (UML), A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

*Figure 2 Sequence Diagram*



*Figure 2 Sequence Diagram*

### 3.6.3 ACTIVITY DIAGRAM:

An activity diagram is a behavioral diagram i.e, it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

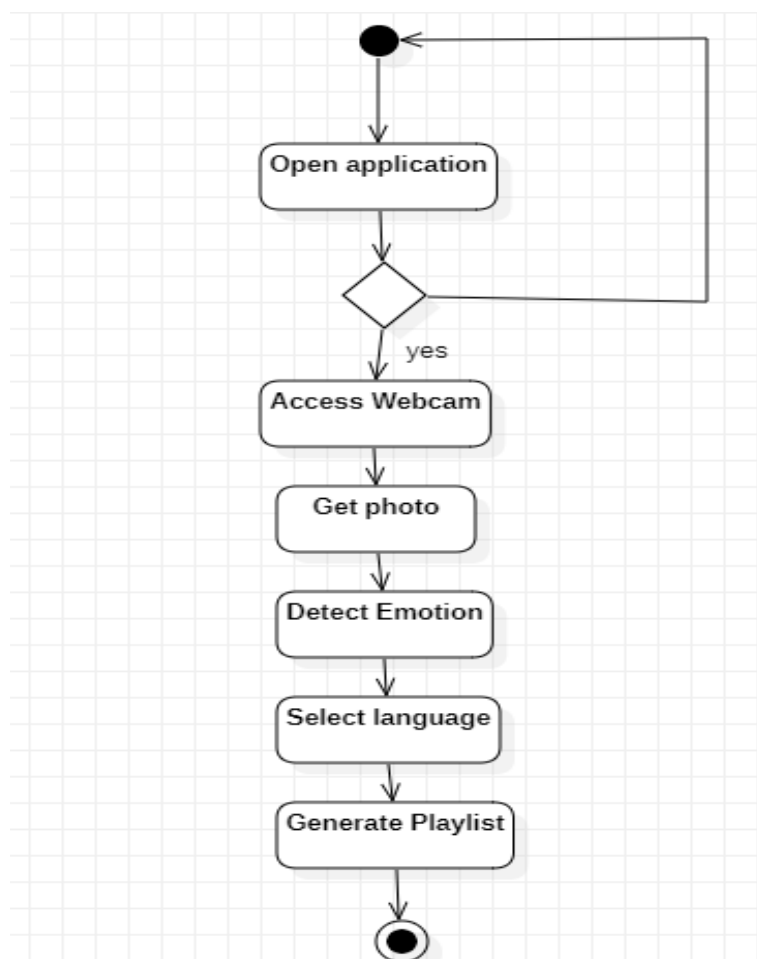


Figure 3 Activity Diagram



### 3.6.4 CLASS DIAGRAM:

A description of a group of objects all with similar roles in the system, which consists of:

- **Structural features** (attributes) define what objects of the class "know"
  - Represent the state of an object of the class
  - Are descriptions of the structural or static features of a class
- **Behavioral features** (operations) define what objects of the class "can do"
  - Define the way in which objects may interact
  - Operations are descriptions of behavioral or dynamic features of a class

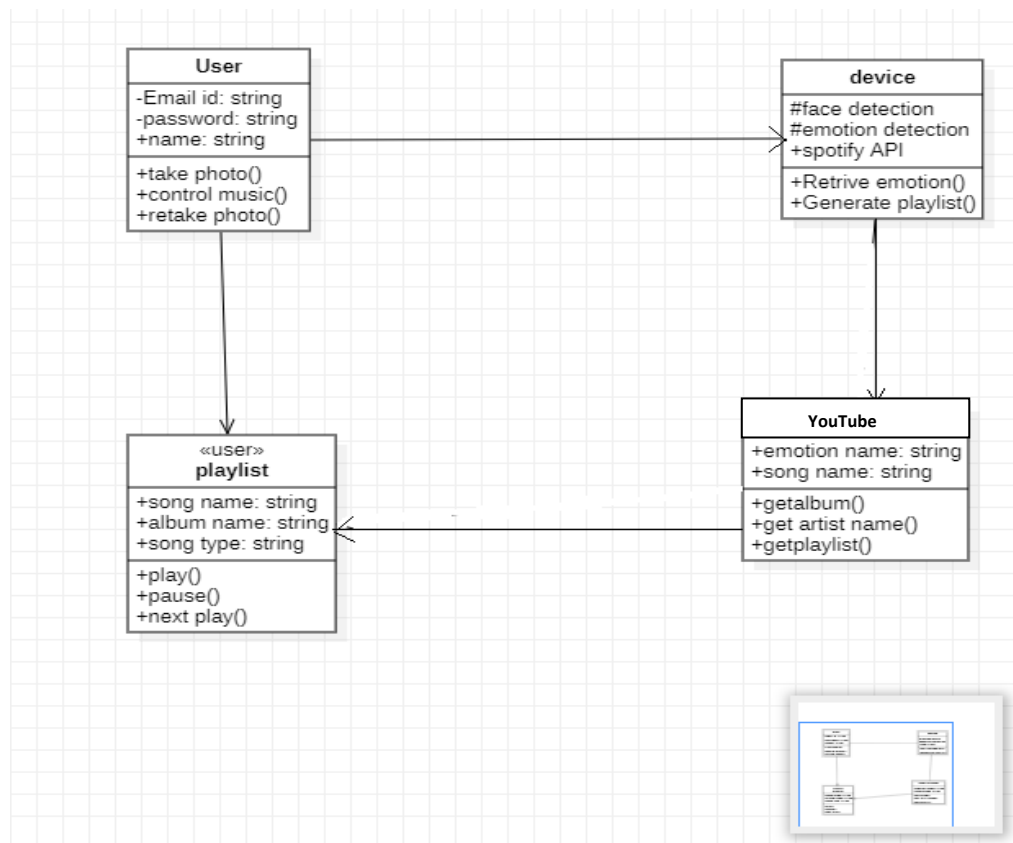


Figure 4 Class Diagram

### 3.6.5 E-R DIAGRAM:

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system.

An ERD uses data modeling techniques that can help define business processes and serve as the foundation for a relational database.

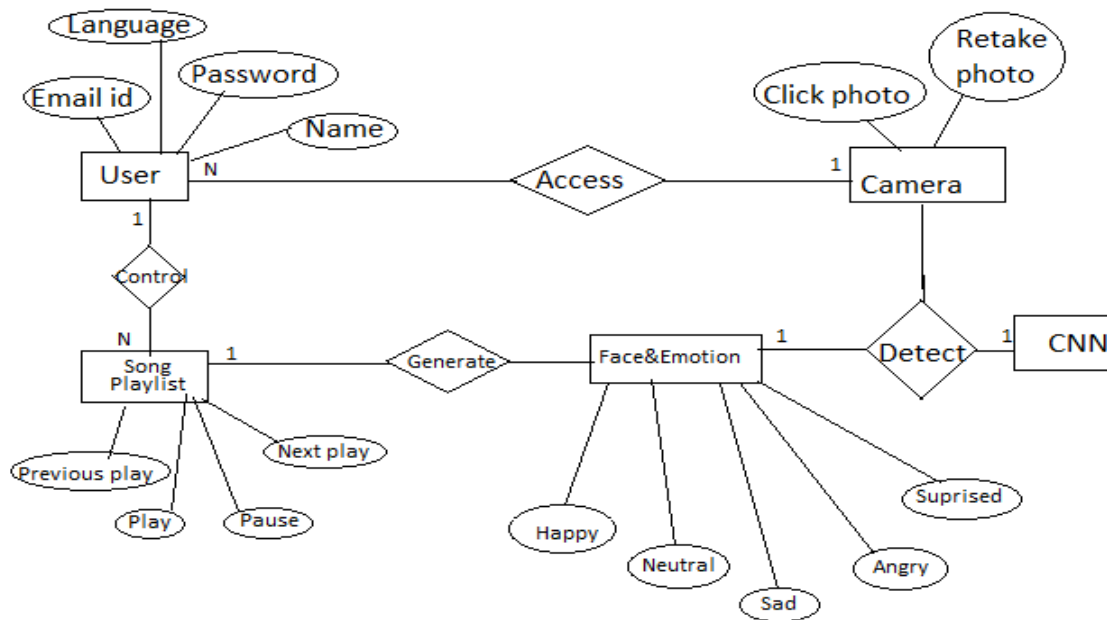


Figure 5 ER Diagram

## **CHAPTER 4: SYSTEM DESIGN**

## 4.1 Basic Modules

**User:** Take photo or retake it

Login and Registration

Change playlist

shuffle the songs

select option on singer

select option on language

**Admin:** Admin login

View the logged in user

View the registered user

View user to contact with admin

## 4.2 Data Design

**Table: details**

Serial no	Field name	Data type	Constraint	Description
1	Name	varchar(50)	NOT NULL	Store user name
3	Language	varchar(30)	NOT NULL	Store user selected language
4	Singer	varchar(50)	NOT NULL	Store device singer name
5	date	DATE	NOT NULL	Store the date

**Table: accounts**

Serial no	Name field	Data type	Constraint	Description
1	Name	varchar	NOT NULL	Store the Name
3	Email id	varchar	PRIMARY KEY	Store the email id
4	Password	varchar	NOT NULL	Store the password
5	date	DATE	NOT NULL	Store the date

**Table: contacts**

Serial no	Name field	Data type	Constraint	Descr iption
1	Name	varchar	NOT NULL	Store the Name
3	Email id	varchar	PRIMARY KEY	Store the email id
4	Password	varchar	NOT NULL	Store the password
5	phone_no	int	NOT NULL	Store the phone no
6	msg	varchar	NOT NULL	Store the message
7	date	DATE	NOT NULL	Store the date

## **4.2.2 Data Integrity and Constraints**

**Integrity Constraints in DBMS are of 4 types:**

- Domain Constraint.
- Entity Constraint.
- Referential Integrity Constraint.
- Key Constraint.

### **1) Domain Constraint.**

Domain Constraints are user-defined columns that help the user to enter the value according to the data type. And if it encounters a wrong input it gives the message to the user that the column is not fulfilled properly

### **2) Entity Constraint.**

An entity constraint describes, in terms of entity objects and attributes, the database-level relationships between tables and columns. You select the entity object's attributes and define the constraint in terms of database integrity constraints such as primary, foreign, check, or unique.

### **3) Referential Integrity Constraint.**

A referential integrity constraint is defined as part of an association between two entity types. The definition for a referential integrity constraint specifies the following information: The principal end of the constraint. (An entity type whose entity key is referenced by the dependent end.)

### **4) Key Constraint.**

A primary key constraint is a column or combination of columns that has the same properties as a unique constraint. You can use a primary key and foreign key constraints to define relationships between tables.

**Data integrity** is the overall accuracy, completeness, and consistency of data. Data integrity also refers to the safety of data in regard to regulatory compliance — such as GDPR compliance — and security. It is maintained by a collection of processes, rules, and standards implemented during the design phase.



# **CHAPTER 5:**

## **Implementation and Testing**

## H5 model code:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries

from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activation,MaxPooling2D
from keras.models import Model,Sequential
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers import SGD

picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"

expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
```

```
img = load_img(folder_path+"train/"+expression+"/"+  
                os.listdir(folder_path + "train/" + expression)[i], target_size=(picture_size,  
picture_size))  
plt.imshow(img)  
plt.show()
```

```
batch_size = 128
```

```
datagen_train = ImageDataGenerator()
```

```
datagen_val = ImageDataGenerator()
```

```
train_set = datagen_train.flow_from_directory(folder_path+"train",  
                                              target_size = (picture_size,picture_size),  
                                              color_mode = "grayscale",  
                                              batch_size=batch_size,  
                                              class_mode='categorical',  
                                              shuffle=True)
```

```
test_set = datagen_val.flow_from_directory(folder_path+"validation",  
                                          target_size = (picture_size,picture_size),  
                                          color_mode = "grayscale",  
                                          batch_size=batch_size,  
                                          class_mode='categorical',  
                                          shuffle=False)
```

Found 28821 images belonging to 7 classes.

Found 7066 images belonging to 7 classes.

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers import SGD
```

```
no_of_classes = 7
```

```
model = Sequential()
```

```
#1st CNN layer
```

```
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))
```

```
#2nd CNN layer
```

```
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))
```

```
#3rd CNN layer
```

```
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
```

```
model.add(Dropout (0.25))
```

```
#4th CNN layer
```

```
model.add(Conv2D(512,(3,3), padding='same'))
```

```
model.add(BatchNormalization())
```

```
model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```

```
model.add(Flatten())
```

```
#Fully connected 1st layer
```

```
model.add(Dense(256))
```

```
model.add(BatchNormalization())
```

```
model.add(Activation('relu'))
```

```
model.add(Dropout(0.25))
```

```
# Fully connected layer 2nd layer
```

```
model.add(Dense(512))
```

```
model.add(BatchNormalization())
```

```
model.add(Activation('relu'))
```

```
model.add(Dropout(0.25))
```

```
model.add(Dense(no_of_classes, activation='softmax'))
```

```
opt = Adam(lr = 0.0001)
```

```
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

```
2022-11-11 13:38:38.277807: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least one
NUMA node, so returning NUMA node zero
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #	
=====			
conv2d (Conv2D)	(None, 48, 48, 64)	640	
-----			
batch_normalization (Batch Normalization)	(None, 48, 48, 64)	256	
-----			
activation (Activation)	(None, 48, 48, 64)	0	max_pooling2d (MaxPooling2D)
(None, 24, 24, 64)	0		
-----			
dropout (Dropout)	(None, 24, 24, 64)	0	
-----			
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928	
-----			
batch_normalization_1 (Batch Normalization)	(None, 24, 24, 128)	512	
-----			
activation_1 (Activation)	(None, 24, 24, 128)	0	
-----			
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0	
-----			

dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_2 (Batch Normalization)	(None, 12, 12, 512)	2048
activation_2 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_3 (Batch Normalization)	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_4 (Batch Normalization)	(None, 256)	1024

activation_4 (Activation)	(None, 256)	0
---------------------------	-------------	---

---

dropout_4 (Dropout)	(None, 256)	0
---------------------	-------------	---

---

dense_1 (Dense)	(None, 512)	131584
-----------------	-------------	--------

---

batch_normalization_5 (Batch Normalization)	(None, 512)	2048
---	-------------	------

---

activation_5 (Activation)	(None, 512)	0
---------------------------	-------------	---

---

dropout_5 (Dropout)	(None, 512)	0
---------------------	-------------	---

---

dense_2 (Dense)	(None, 7)	3591
-----------------	-----------	------

=====

Total params: 4,478,727

Trainable params: 4,474,759

Non-trainable params: 3,968

```
from keras.models import load_model
```

```
from keras.layers import Lambda
```

```
import tensorflow as tf
```

```
from tensorflow.keras.optimizers import RMSprop
```

```
from tensorflow.keras.optimizers import Adam
```

```
from tensorflow.keras.optimizers import SGD
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
```

```
checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1,  
save_best_only=True, mode='max')
```

```
early_stopping = EarlyStopping(monitor='val_loss',
```



```
min_delta=0,  
patience=3,  
verbose=1,  
restore_best_weights=True  
)
```

```
reduce_learningrate = tf.keras.callbacks.ReduceLROnPlateau( monitor="val_loss", factor=0.5,  
patience=1, verbose=1)
```

```
callbacks_list = [early_stopping,checkpoint,reduce_learningrate]
```

```
epochs = 48model.compile(loss='categorical_crossentropy',  
optimizer = Adam(lr=0.001),  
metrics=['accuracy'])
```

```
history = model.fit_generator(generator=train_set,  
steps_per_epoch=train_set.n//train_set.batch_size,  
epochs=epochs,  
validation_data = test_set,  
validation_steps = test_set.n//test_set.batch_size,  
callbacks=callbacks_listPasses are enabled (registered 2)
```

Epoch 1/48

2022-11-11 13:38:44.845780: I tensorflow/stream\_executor/cuda/cuda\_dnn.cc:369] Loaded cuDNN version 8005

225/225 [=====] - 234s 1s/step - loss: 1.7958 - accuracy: 0.3107 - val\_loss: 2.0740 - val\_accuracy: 0.2881

Epoch 2/48

225/225 [=====] - 28s 123ms/step - loss: 1.4268 - accuracy: 0.4515 - val\_loss: 1.4600 - val\_accuracy: 0.4483

Epoch 3/48

225/225 [=====] - 26s 117ms/step - loss: 1.2745 - accuracy: 0.5123 - val\_loss: 1.3711 - val\_accuracy: 0.4828

Epoch 4/48

225/225 [=====] - 28s 123ms/step - loss: 1.1904 - accuracy: 0.5474 - val\_loss: 1.3955 - val\_accuracy: 0.4956

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 5/48

225/225 [=====] - 29s 127ms/step - loss: 1.0991 - accuracy: 0.5834 - val\_loss: 1.1422 - val\_accuracy: 0.5639

Epoch 6/48

225/225 [=====] - 28s 124ms/step - loss: 1.0497 - accuracy: 0.6024 - val\_loss: 1.0982 - val\_accuracy: 0.5884

Epoch 7/48

225/225 [=====] - 26s 117ms/step - loss: 1.0146 - accuracy: 0.6125 - val\_loss: 1.0600 - val\_accuracy: 0.6107

Epoch 8/48

225/225 [=====] - 27s 119ms/step - loss: 0.9749 - accuracy: 0.6296 - val\_loss: 1.1367 - val\_accuracy: 0.5827

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.

Epoch 9/48

225/225 [=====] - 26s 117ms/step - loss: 0.9066 - accuracy: 0.6579 - val\_loss: 1.0139 - val\_accuracy: 0.6310

Epoch 10/48

225/225 [=====] - 27s 121ms/step - loss: 0.8722 - accuracy: 0.6724 - val\_loss: 1.0154 - val\_accuracy: 0.6354

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.

Epoch 11/48

225/225 [=====] - 28s 126ms/step - loss: 0.8209 - accuracy: 0.6926 - val\_loss: 1.0022 - val\_accuracy: 0.6413

Epoch 12/48

225/225 [=====] - 28s 124ms/step - loss: 0.7943 - accuracy: 0.7013 - val\_loss: 1.0214 - val\_accuracy: 0.6314

Epoch 00012: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.

Epoch 13/48

225/225 [=====] - 28s 122ms/step - loss: 0.7672 - accuracy: 0.7140 - val\_loss: 0.9772 - val\_accuracy: 0.6493

Epoch 14/48

225/225 [=====] - 28s 126ms/step - loss: 0.7503 - accuracy: 0.7229 - val\_loss: 0.9819 - val\_accuracy: 0.6523

Epoch 00014: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.

Epoch 15/48

225/225 [=====] - 29s 129ms/step - loss: 0.7421 - accuracy: 0.7212 - val\_loss: 0.9908 - val\_accuracy: 0.6490

Epoch 00015: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.

Epoch 16/48

225/225 [=====] - 29s 128ms/step - loss: 0.7284 - accuracy: 0.7287 - val\_loss: 0.9828 - val\_accuracy: 0.6526

Restoring model weights from the end of the best epoch.

Epoch 00016: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.

Epoch 00016: early stopping

plt.style.use('dark\_background')

```
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

### **app.py code:**

```
from flask import Flask, render_template, request, redirect, url_for, jsonify, session, flash
from flask_sqlalchemy import SQLAlchemy
import numpy as np
import cv2
from keras.models import load_model
import webbrowser
from datetime import datetime

with open('config.json', 'r') as c:
```

```
params=json.load(c) ["params"]
```

```
app = Flask(__name__)
```

```
app.secret_key="super-secret-key"
```

```
app.config["SQLALCHEMY_DATABASE_URI"] = "mysql://root:@localhost/groove"
```

```
# initialize the app with the extension
```

```
db= SQLAlchemy(app)
```

```
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
```

```
info = { }
```

```
haarcascade = "haarcascade_frontalface_default.xml"
```

```
label_map = ['Anger','Neutral','Happy','Suprise','Sad']
```

```
print("+"*50, "loadin gmmodel")
```

```
model = load_model(r'C:\Users\veathavalli\Downloads\model (2).h5')
```

```
cascade = cv2.CascadeClassifier(cv2.data.haarcascades +  
'haarcascade_frontalface_default.xml')
```

```
@app.route('/')
```

```
def content():
```

```
    return render_template('content.html')
```

```
# accounts class table of login and register
```

```
class Accounts(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
```

```
email = db.Column(db.String(20))
username = db.Column(db.String(80))
password = db.Column(db.String(120))
date = db.Column(db.String(12))
```

```
#login routes
```

```
@app.route("/login",methods=["GET", "POST"])
```

```
def login():
```

```
    if request.method == "POST":
        email = request.form["email"]
        passw = request.form["passw"]
        login = Accounts.query.filter_by(email=email,password=passw).first()
        if login is not None:
            return render_template("afterlogin.html",login=login,date=datetime.now())
        return render_template("newlogin.html")
```

```
#register routes
```

```
@app.route("/register", methods=["GET", "POST"])
```

```
def register():
```

```
    if request.method == "POST":
        uname = request.form['uname']
        mail = request.form['email']
        passw = request.form['passw']

        register = Accounts(username = uname, email = mail, password =
passw,date=datetime.now())
```

```
db.session.add(register)
```

```
db.session.commit()
```

```
return redirect(url_for("newlogin"))
```

```
return render_template("newregister.html")
```

```
#logout routes
```

```
@app.route('/logout')
```

```
def logout():
```

```
    session.clear
```

```
    return redirect(url_for("content"))
```

```
#dashbord routes
```

```
@app.route('/dashboard',methods=["GET","POST"])
```

```
def dashboard():
```

```
    if request.method=="POST":
```

```
        username=request.form.get('uname')
```

```
        userpass=request.form.get('passw')
```

```
        if (username == params['admin_user'] and userpass == params['admin_pass']):
```

```
            contacts = Contacts.query.all()
```

```
            accounts = Accounts.query.all()
```

```
            details = Details.query.all()
```

```
            return
```

```
render_template("dashboard.html",params=params,contacts=contacts,accounts=accounts,details=details)
```

```
return render_template("admin.html",params=params)
```

```
#contact class table
```

```
class Contacts(db.Model):
```

```
    sno = db.Column(db.Integer, primary_key=True)
```

```
    name = db.Column(db.String(80))
```

```
    phone_num = db.Column(db.String(80))
```

```
    msg = db.Column(db.String(120))
```

```
    date = db.Column(db.String(12))
```

```
    email = db.Column(db.String(20))
```

```
#contact routes
```

```
@app.route('/contact',methods=['GET','POST'])
```

```
def contact():
```

```
    if(request.method=='POST'):
```

```
        name=request.form.get('name')
```

```
        email=request.form.get('email')
```

```
        phone=request.form.get('phone')
```

```
        message=request.form.get('message')
```

```
        entry =
```

```
Contacts(name=name,phone_num=phone,msg=message,date=datetime.now(),email=email)
```

```
        db.session.add(entry)
```

```
        db.session.commit()
```

```
    return render_template('contact.html')
```

```
@app.route('/home')
```

```
def home():
```



```
    return render_template('afterlogin.html')
```

```
@app.route('/about')
```

```
def about():
```

```
    return render_template('about.html')
```

```
#aboutme routes
```

```
@app.route('/aboutme')
```

```
def aboutme():
```

```
    return render_template('aboutme.html')
```

```
# afterlogin routes
```

```
@app.route('/afterlogin')
```

```
def afterlogin():
```

```
    return render_template('afterlogin.html')
```

```
#contact class table
```

```
class Details(db.Model):
```

```
    sno = db.Column(db.Integer, primary_key=True)
```

```
    language = db.Column(db.String(80))
```

```
    singer = db.Column(db.String(80))
```

```
    date = db.Column(db.String(12))
```

```
#after clicking 'start app' button index page will open
```

```
@app.route('/index')
```

```

def index():

    return render_template('index.html')


#choose singer routes
@app.route('/choose_singer', methods = ["POST"])
def choose_singer():
    info['language'] = request.form['language']
    print(info)
    if(request.method=='POST'):
        language =request.form.get('language')
        singer=request.form.get('singer')
        entry = Details(singer=singer,language=language,date=datetime.now())
        db.session.add(entry)
        db.session.commit()
    return render_template('choose_singer.html', data = info['language'])


# emotion routes
@app.route('/emotion_detect', methods=["POST"])
def emotion_detect():
    info['singer'] = request.form['singer']
    if(request.method=='POST'):
        language =request.form.get('language')
        singer=request.form.get('singer')
        entry = Details(singer=singer,language=language,date=datetime.now())
        db.session.add(entry)
        db.session.commit()

```

```
found = False
```

```
cap = cv2.VideoCapture(0)
```

```
while not(found):
```

```
    _, frm = cap.read()
```

```
    gray = cv2.cvtColor(frm,cv2.COLOR_BGR2GRAY)
```

```
    faces = cascade.detectMultiScale(gray ,1.3, 5)
```

```
    for x,y,w,h in faces:
```

```
        found = True
```

```
        roi = gray[y:y+h, x:x+w]
```

```
        cv2.imwrite("static/face.jpg", roi)
```

```
roi = cv2.resize(roi, (48,48))
```

```
roi = roi/255.0
```

```
roi = np.reshape(roi, (1,48,48,1))
```

```
prediction = model.predict(roi)
```

```
print(prediction)
```

```
prediction = np.argmax(prediction)
```

```
prediction = label_map[prediction]
```

```
cap.release()
```

```
link =  
f"https://www.youtube.com/results?search_query={info['singer']}{prediction}{info['language']}" + "official song"  
webbrowser.open(link)
```

```
return render_template("emotion_detect.html", data=prediction, link=link)
```

```
if __name__ == "__main__":  
    app.run(debug=True)
```

### **index.html(for selecting language):**

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
    <title>Start App</title>  
  
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css')}} ">  
  
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"  
rel="stylesheet" integrity="sha384-  
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"  
crossorigin="anonymous">  
  
</head>  
  
<body style="background-color: #edffec;">
```

```
<div class="title">
```

```
  <h1 class="display-4">Select Language</h1>
```

```
</div>
```

```
<form action="{ {url_for('choose_singer')}}" method="POST">
```

```
  <label><input type="radio" name="language" value="tamil" checked="true"><div  
class="box">Tamil</div></label>
```

```
  <label><input type="radio" name="language" value="english"><div  
class="box">English</div></label>
```

```
  <br>
```

```
  <label><input type="radio" name="language" value="hindi"><div  
class="box">Hindi</div></label>
```

```
  <label><input type="radio" name="language" value="korea"><div  
class="box">Korea</div></label>
```

```
  <label><input type="radio" name="language" value="telugu"><div  
class="box">Telugu</div></label>
```

```
  <label><input type="radio" name="language" value="malayalam"><div  
class="box">Malayalam</div></label>
```

```
  <br>
```

```
  <input type="submit" name="btn" value="Next" class="btn">
```

```
</form>
```

```
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-  
beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-  
JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"  
crossorigin="anonymous"></script>
```

</body>

</html>

**choose\_singer.html(for choosing singer name):**

<!DOCTYPE html>

<html>

<head>

<title>Singer</title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

</head>

<body style="background-color: #edffec;">

<form action="{{ url\_for('emotion\_detect') }}" method="POST">

{ %if data == "tamil"% }

<div class="title">

<h1 class="display-4">Tamil Singer</h1>

</div>

<label><input type="radio" name="singer" value="Anirudh" checked="true"><div class="box">Anirudh</div></label>

<label><input type="radio" name="singer" value="K. S. Chithra"><div class="box">K. S. Chithra</div></label>

<br>

<label><input type="radio" name="singer" value="Sujatha"><div class="box">Sujatha</div></label>

<label><input type="radio" name="singer" value="A. R. Rahman"><div class="box">A. R. Rahman</div></label>

<label><input type="radio" name="singer" value="Chinmayi"><div class="box">Chinmayi</div></label>

<label><input type="radio" name="singer" value="Karthik"><div class="box">Karthik</div></label>

{ %elif data == "english"% }

<div class="title">

<h1 class="display-4">English singer</h1>

</div>

<label><input type="radio" name="singer" value="Ed sheeran" checked="true"><div class="box">Ed sheeran</div></label>

<label><input type="radio" name="singer" value="Ariana grande"><div class="box">Ariana grande</div></label>

<br>

<label><input type="radio" name="singer" value="Taylor Swift"><div class="box">Taylor Swift</div></label>

<label><input type="radio" name="singer" value="Harry Styles"><div class="box">Harry Styles</div></label>

<label><input type="radio" name="singer" value="Shawn Mendes"><div class="box">Shawn Mendes</div></label>

<label><input type="radio" name="singer" value="Dua Lipa"><div class="box">Dua Lipa</div></label>

{ %elif data == "hindi"% }

```
<div class="title">
```

```
<h1 class="display-4">Hindi singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="Shreya Ghoshal" checked="true"><div  
class="box">Shreya Ghoshal</div></label>
```

```
<label><input type="radio" name="singer" value="Arijit Singh"><div class="box">Arijit  
Singh</div></label>
```

```
<br>
```

```
<label><input type="radio" name="singer" value="Sonu Nigam"><div class="box">Sonu  
Nigam</div></label>
```

```
<label><input type="radio" name="singer" value="Neha kakkar"><div class="box">Neha  
kakkar</div></label>
```

```
<label><input type="radio" name="singer" value="Sunidhi Chauhan"><div  
class="box">Sunidhi Chauhan</div></label>
```

```
<label><input type="radio" name="singer" value="KK"><div  
class="box">KK</div></label>
```

```
{ %elif data == "korea"% }
```

```
<div class="title">
```

```
<h1 class="display-4">korean singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="BTS" checked="true"><div  
class="box">BTS</div></label>
```

```
<label><input type="radio" name="singer" value="BlacK Pink"><div class="box">Black  
Pink</div></label>
```

```
<br>
```

```
<label><input type="radio" name="singer" value="IU"><div  
class="box">IU</div></label>
```

```
<label><input type="radio" name="singer" value="EXO"><div  
class="box">EXO</div></label>
```



```
<label><input type="radio" name="singer" value="Red velvet"><div class="box">Red velvet</div></label>
```

```
<label><input type="radio" name="singer" value="Twice"><div class="box">Twice</div></label>
```

```
{ %elif data == "telugu"% }
```

```
<div class="title">
```

```
<h1 class="display-4">Telugu singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="Sooraj Santosh" checked="true"><div class="box">Sooraj Santosh</div></label>
```

```
<label><input type="radio" name="singer" value="Satya Yamini"><div class="box">Satya Yamini</div></label>
```

```
<br>
```

```
<label><input type="radio" name="singer" value="Anudeep"><div class="box">Anudeep</div></label>
```

```
<label><input type="radio" name="singer" value="Mohana"><div class="box">Mohana</div></label>
```

```
<label><input type="radio" name="singer" value="Prudhvi Chandra"><div class="box">Prudhvi Chandra</div></label>
```

```
<label><input type="radio" name="singer" value="Mounima"><div class="box">Mounima</div></label>
```

```
{ %elif data == "malayalam"% }
```

```
<div class="title">
```

```
<h1 class="display-4">Malayalam singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="Vineeth" checked="true"><div class="box">Vineeth</div></label>
```

```
<label><input type="radio" name="singer" value="Vijay Yesudas"><div
class="box">Vijay Yesudas</div></label>
```

```
<br>
```

```
<label><input type="radio" name="singer" value="Sithara"><div
class="box">Sithara</div></label>
```

```
<label><input type="radio" name="singer" value="Manjari"><div
class="box">Manjari</div></label>
```

```
<label><input type="radio" name="singer" value="Sujatha"><div
class="box">Sujatha</div></label>
```

```
<label><input type="radio" name="singer" value="Sushin Shyam"><div
class="box">Sushin Shyam</div></label>
```

```
{ %endif% }
```

```
<br><br>
```

```
<input type="submit" name="btn" value="Take a shot" class="btn">
```

```
</form>
```

```
</body>
```

```
</html>
```

**emotion\_detect.html(display the image and emotion label):**

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Singer</title>
```

```
<link rel="stylesheet" type="text/css" href="{ {url_for('static', filename='style.css')}}">
```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
```

eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"  
crossorigin="anonymous">

</head>

<body style="background-color: #edffec;">

<form action="{ {url\_for('emotion\_detect')}}" method="POST">  
 {%if data == "tamil"% }

<div class="title">  
 <h1 class="display-4">Tamil Singer</h1>

</div>  
  
 <label><input type="radio" name="singer" value="Anirudh" checked="true"><div  
class="box">Anirudh</div></label>  
  
 <label><input type="radio" name="singer" value="K. S. Chithra"><div class="box">K.  
S. Chithra</div></label>  
  
 <br>  
  
 <label><input type="radio" name="singer" value="Sujatha"><div  
class="box">Sujatha</div></label>  
  
 <label><input type="radio" name="singer" value="A. R. Rahman"><div class="box">A.  
R. Rahman</div></label>  
  
 <label><input type="radio" name="singer" value="Chinmayi"><div  
class="box">Chinmayi</div></label>  
  
 <label><input type="radio" name="singer" value="Karthik"><div  
class="box">Karthik</div></label>

{%elif data == "english"% }  
  
 <div class="title">  
 <h1 class="display-4">English singer</h1>

</div>

<label><input type="radio" name="singer" value="Ed sheeran" checked="true"><div class="box">Ed sheeran</div></label>

<label><input type="radio" name="singer" value="Ariana grande"><div class="box">Ariana grande</div></label>

<br>

<label><input type="radio" name="singer" value="Taylor Swift"><div class="box">Taylor Swift</div></label>

<label><input type="radio" name="singer" value="Harry Styles"><div class="box">Harry Styles</div></label>

<label><input type="radio" name="singer" value="Shawn Mendes"><div class="box">Shawn Mendes</div></label>

<label><input type="radio" name="singer" value="Dua Lipa"><div class="box">Dua Lipa</div></label>

{ %elif data == "hindi"% }

<div class="title">

<h1 class="display-4">Hindi singer</h1>

</div>

<label><input type="radio" name="singer" value="Shreya Ghoshal" checked="true"><div class="box">Shreya Ghoshal</div></label>

<label><input type="radio" name="singer" value="Arijit Singh"><div class="box">Arijit Singh</div></label>

<br>

<label><input type="radio" name="singer" value="Sonu Nigam"><div class="box">Sonu Nigam</div></label>

<label><input type="radio" name="singer" value="Neha kakkar"><div class="box">Neha kakkar</div></label>

<label><input type="radio" name="singer" value="Sunidhi Chauhan"><div class="box">Sunidhi Chauhan</div></label>

```
<label><input type="radio" name="singer" value="KK"><div
class="box">KK</div></label>
```

```
{ %elif data == "korea"% }
```

```
<div class="title">
```

```
<h1 class="display-4">korean singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="BTS" checked="true"><div
class="box">BTS</div></label>
```

```
<label><input type="radio" name="singer" value="BlacK Pink"><div class="box">Black
Pink</div></label>
```

```
<br>
```

```
<label><input type="radio" name="singer" value="IU"><div
class="box">IU</div></label>
```

```
<label><input type="radio" name="singer" value="EXO"><div
class="box">EXO</div></label>
```

```
<label><input type="radio" name="singer" value="Red velvet"><div class="box">Red
velvet</div></label>
```

```
<label><input type="radio" name="singer" value="Twice"><div
class="box">Twice</div></label>
```

```
{ %elif data == "telugu"% }
```

```
<div class="title">
```

```
<h1 class="display-4">Telugu singer</h1>
```

```
</div>
```

```
<label><input type="radio" name="singer" value="Sooraj Santosh" checked="true"><div
class="box">Sooraj Santosh</div></label>
```

```
<label><input type="radio" name="singer" value="Satya Yamini"><div
class="box">Satya Yamini</div></label>
```

<br>

<label><input type="radio" name="singer" value="Anudeep"><div  
class="box">Anudeep</div></label>

<label><input type="radio" name="singer" value="Mohana"><div  
class="box">Mohana</div></label>

<label><input type="radio" name="singer" value="Prudhvi Chandra"><div  
class="box">Prudhvi Chandra</div></label>

<label><input type="radio" name="singer" value="Mounima"><div  
class="box">Mounima</div></label>

{ %elif data == "malayalam"% }

<div class="title">

<h1 class="display-4">Malayalam singer</h1>

</div>

<label><input type="radio" name="singer" value="Vineeth" checked="true"><div  
class="box">Vineeth</div></label>

<label><input type="radio" name="singer" value="Vijay Yesudas"><div  
class="box">Vijay Yesudas</div></label>

<br>

<label><input type="radio" name="singer" value="Sithara"><div  
class="box">Sithara</div></label>

<label><input type="radio" name="singer" value="Manjari"><div  
class="box">Manjari</div></label>

<label><input type="radio" name="singer" value="Sujatha"><div  
class="box">Sujatha</div></label>

<label><input type="radio" name="singer" value="Sushin Shyam"><div  
class="box">Sushin Shyam</div></label>

{ %endif% }

```
<br><br>
<input type="submit" name="btn" value="Take a shot" class="btn">
</form>

</body>
</html>
```

### **content.html(The main page):**

```
{% extends 'layout.html' %}
{% block body %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>The Groove Connection</title>
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='content.css')}}">
</head>
<body>

  <div class="maincontent">

    <div class="titles">

      <h2>The Groove Connection!!!</h2>
```

```

    <p>Click here to check your current mood & seek your engaging song!!</p>
</div>

<div id="outer">

    <div class="button_slide slide_left">

        <a href="" onclick="myFunction()">START APP</a>

    </div>

</div>

</div>

</body>
<script>
function myFunction() {
    alert("You need to login first!!");
}
</script>
</html>
{ %endblock% }

```

### **layout.html(navigation bar in all page):**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>The Groove Connection</title>

```



```

    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='layout.css') }}">
</head>
<body>
    <div class="full-page">

        <div class="navbar">

            <nav>

                <ul id="menuitems">
                    <li><a href="/">HOME</a></li>
                    <li><a href="/aboutme">ABOUT ME</a></li>
                    <li><a href="/contact">CONTACT ME</a></li>
                    <li><a href="/dashboard">ADMIN</a></li>
                    <li><a href="/login">LOGIN</a></li>

                </ul>

            </nav>

        </div>
    { % block body % } { % endblock % }
</div>

</body>

</html>

```

**style.css(css for content page):**

```
body{
    text-align: center;
}

h1{
    font-family: sans-serif;
    color: #DB7093;
}

.title{
    margin-top: 20px;
}

input[type="radio"]{
    display: none;
}

.box{
    background-color: #C71585;
    height: 250px;
    width: 400px;
    padding: 30px;
    text-align:center;
    font-size: 40px;
    font-family: sans-serif;
    line-height: 220px;
    margin: 10px 10px;
    border: 2px solid #FF1493;
    border-radius: 45%;
```

```
}
```

```
.box:hover{
```

```
    background-color: #FF69B4;
```

```
    transition-duration: 0.8s;
```

```
}
```

```
input[type=radio]:checked + .box{
```

```
    background-color: #FF69B4;
```

```
    color: white;
```

```
    transition-duration: 1s;
```

```
}
```

```
img{
```

```
    margin-top: 50px;
```

```
    height: 450px;
```

```
    border : 2px solid #FF1493;
```

```
    border-radius: 30px;
```

```
}
```

```
input[type="submit"]{
```

```
    background-color: #f656ab;
```

```
    border: none;
```

```
    color: deeppink;
```

```
    padding: 16px 32px;
```

```
    text-align: center;
```

```
text-decoration: none;
display: inline-block;
font-size: 16px;
margin: 4px 2px;
transition-duration: 0.4s;
cursor: pointer;
height: 50px;
width: 140px;
color: black;
}

.btn{
    background-color: palevioletred;

    border: 2px solid palevioletred;
}

.btn:hover{
    background-color: deeppink;
    color: deeppink;
}
```

## 5.2 Testing Approach

### Unit Testing

Unit Testing was done by for verification on the website i.e. the module. Using detailed design and the process specification testing is done uncover error with in the boundary of the module.

All modules must be successful in the unit test.

- **Entry module:** Various cases of error like invalid agents etc are verified.
- **View module:** Only those reports could he viewed that are valid. This property is ensured.

### Acceptance Testing

Website is acceptance testing

### User login:

Test Scenario	Test Case	Test Step	Valid data	Expected Result	Actual Result	Pass/Fail
Check Login Functionality	Check response on Entering Valid Username & Password	Correct User name and password 1. Launch application 2. Enter username 3. Enter password 4. Click on login button	Username: As per registration Password: As per Default name and password given in the Data base	login must be successful	Login successful	Pass
	Should not take null value in the field	User name and password 1. Launch application 2. Enter username 3. Enter password 4. Click on login button	Null value	login must be Unsuccessful	Login Unsuccessful	Pass
	Should not login while enter	User name and password 1. Launch application 2. Enter username 3. Enter password 4. Click on Login button	Invalid Data	login must be Unsuccessful	Login Unsuccessful	Pass

### Admin login:

Test Scenario	Test Case	Test Step	Valid data	Expected Result	Actual Result	Pass/Fail
Check Login Functionality	Check response on Entering Valid Username & Password	Correct User name and password 5. Launch application 6. Enter username 7. Enter password 8. Click on login button	Username: As per given data in json file Password: As per given data in json file	login must be successful	Login successful	Pass
	Check response on Entering valid Username & Password	Should not login with out user name and password	Null value	login must be fail	Login fail	Pass
	Check response on Entering valid Username & Password	Wrong username and password	Wrong password and username	login must be fail	Login fail	Pass

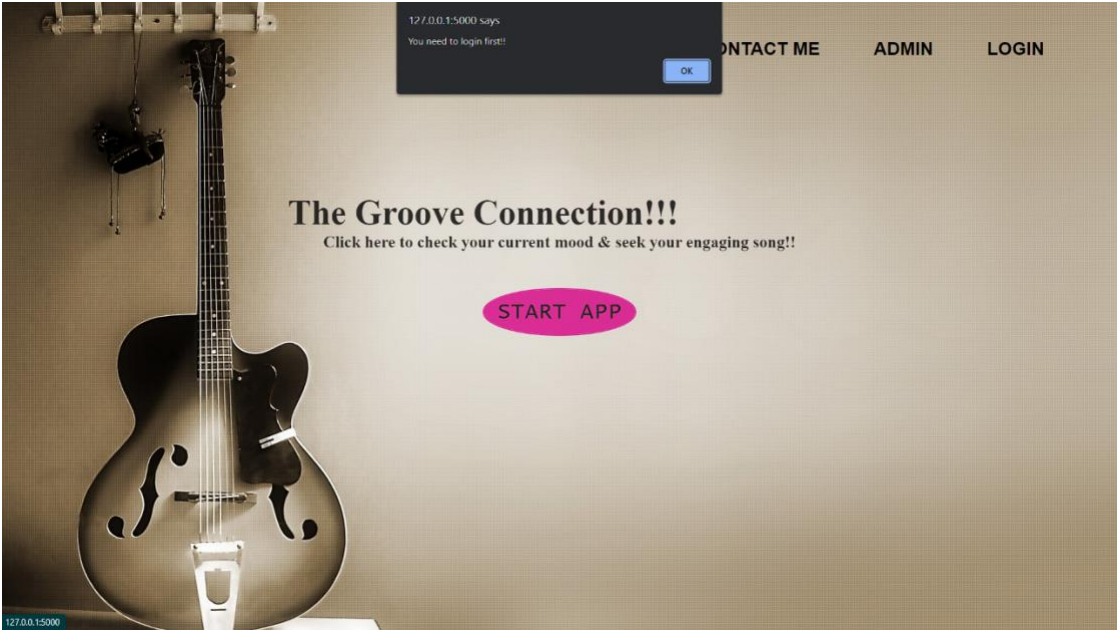
### User Registration:

Test Scenario	Test Case	Test Step	Valid data	Expected Result	Actual Result	Pass/Fail
Check Login Functionality	Check response on Entering Valid information	1.Lauch Application 2. Ener Name 3.Enter Email 4.Enter Password 5.Enter Mail ID	Valid Data	Registration must be successful	Registration successful	Pass
	Check response on Entering Null value information	1.Lauch Application 2. Ener Name 3.Enter Email 4.Enter Password 5.Enter Mail ID	Null value	Registration must be Unsuccessful	Registration Unsuccessful	Pass
	Check response on Entering Wrong value information	1.Lauch Application 2. Ener Name 3.Enter Email 4.Enter Password 5.Enter Mail ID	Wrong Data	Registration must be Unsuccessful	Registration Unsuccessful	Pass

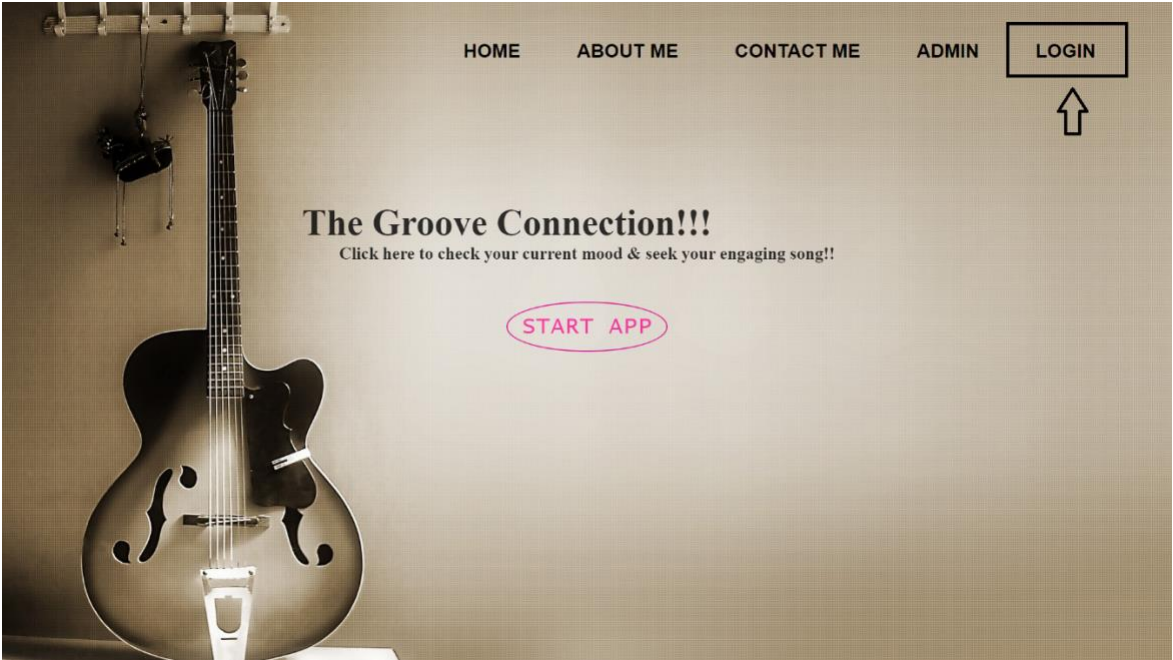


## **CHAPTER 6: Result and discussion**

Home page:



To access Start app login first:



**To login register first:**

## Registration

Please fill in this form to create an account.

**Username**


**Email**

**Password**

Register

Already have an account?  
[Log in](#)

## Login Form



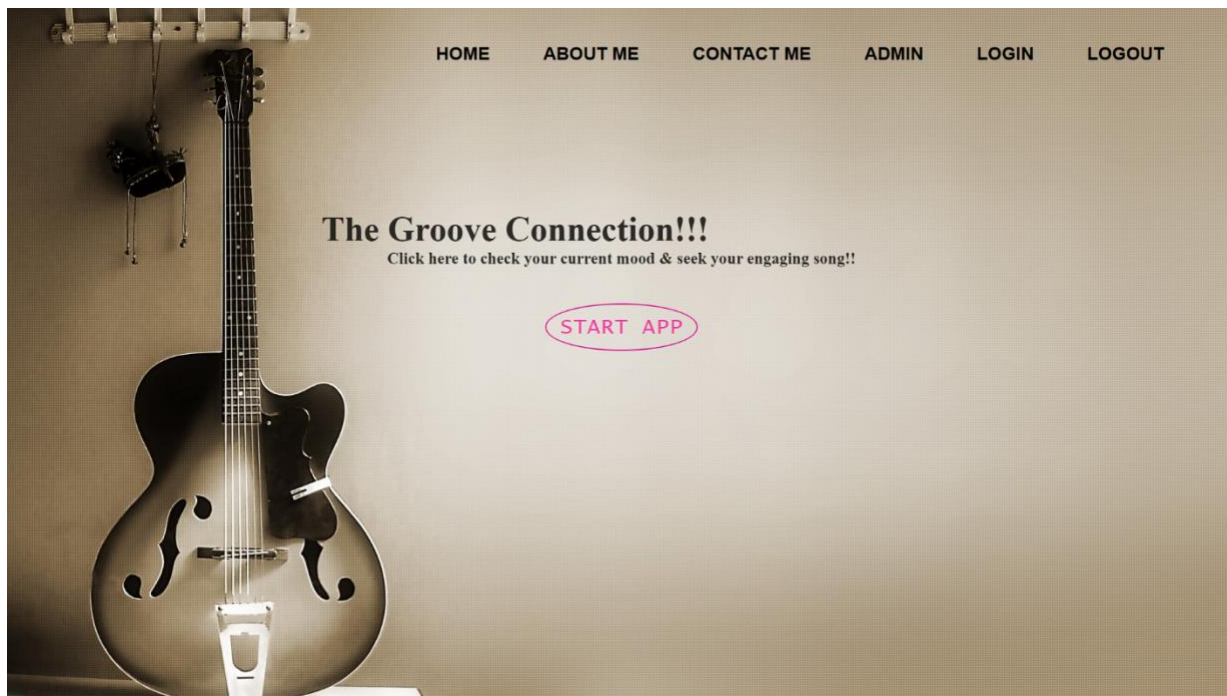
**Email**

**Password**

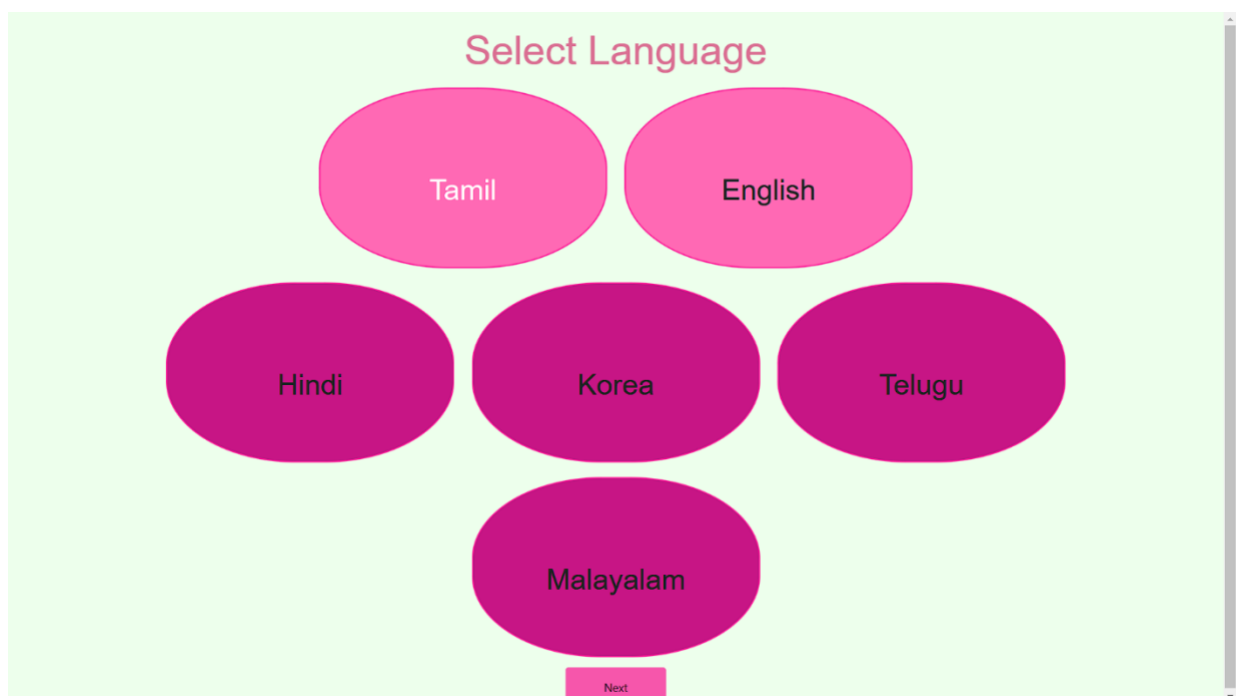
Login

Don't have an account?  
[Sign in](#)

**Now you can access Startapp:**



**Select your language:**

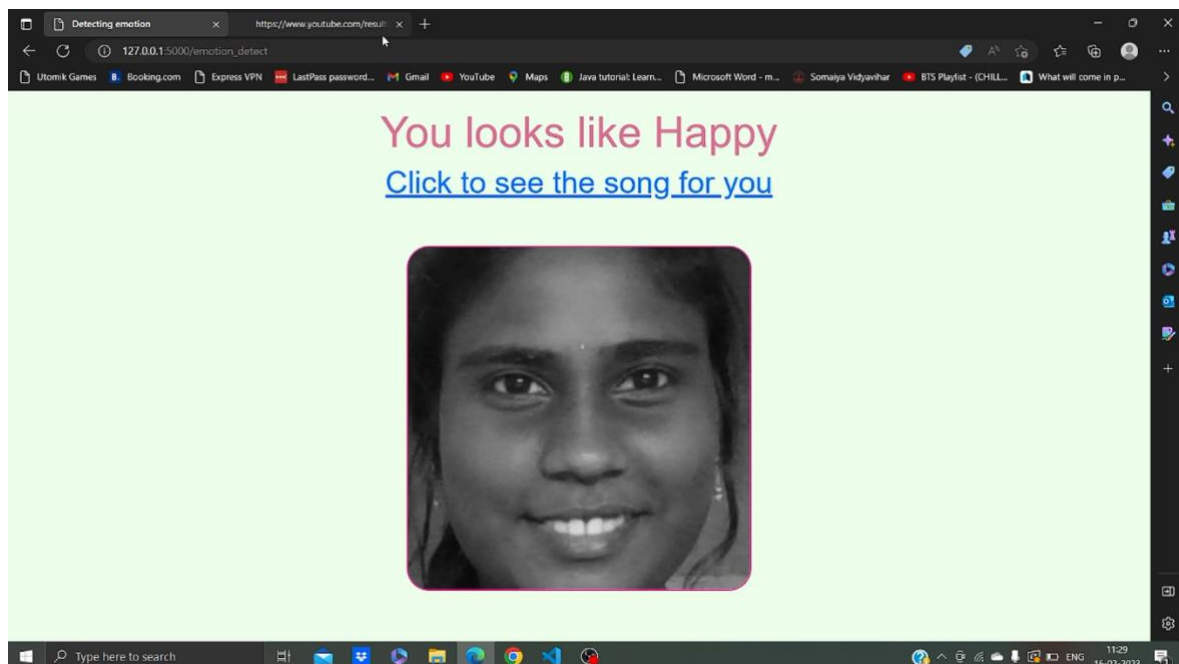


**Now select your favorite singer:**

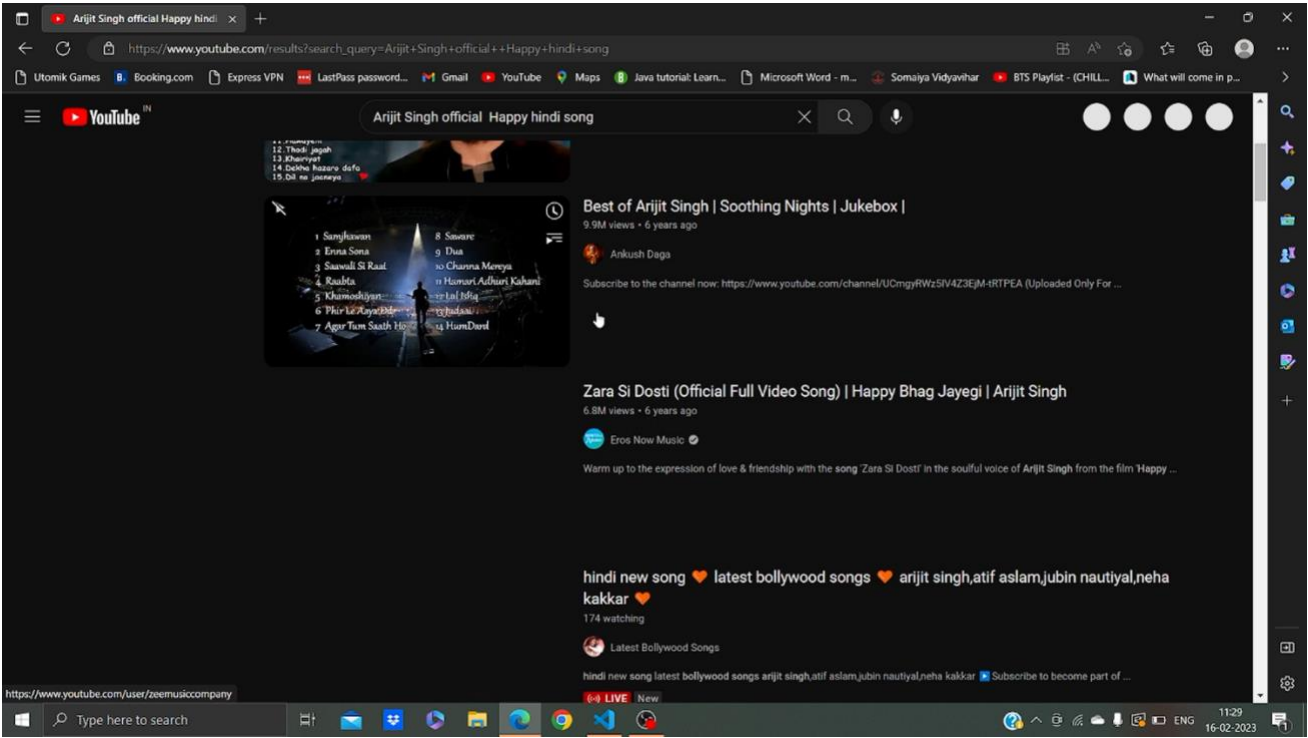


After clicking “take a shot” button the camera will capture your facial expression.

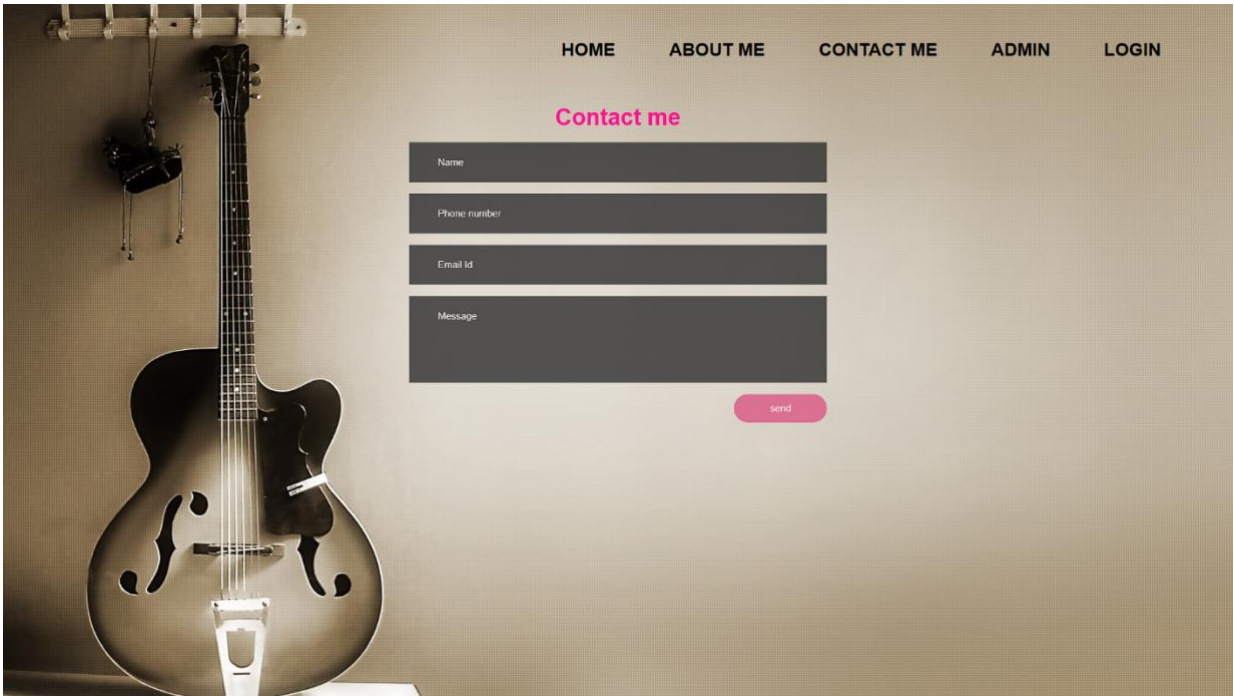
**The detected Emotion:**



The search query of the provided data and detected emotion:

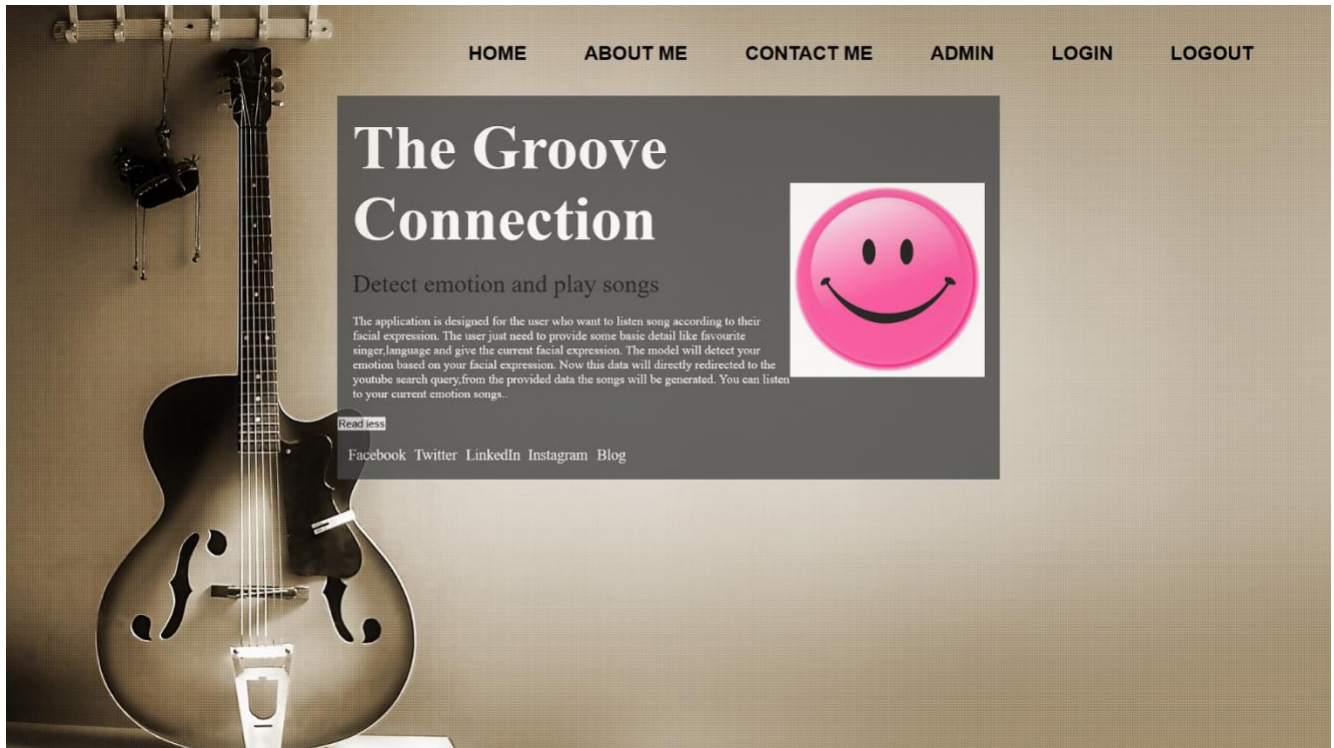


Contact page:





## About me page:



## Admin page:

The image shows an 'Admin login' form. The form is centered on a light grey background. At the top, the title 'Admin login' is written in a bold, black, sans-serif font. Below the title, there are two input fields. The first field is labeled 'Email' and contains the text 'Admin'. The second field is labeled 'Password' and contains a series of asterisks. To the right of the password field is a small, circular icon with an eye, indicating a toggle for password visibility. Below the input fields is a green rectangular button with the text 'Submit' in a white, sans-serif font.

Dashboard page:

HOMEABOUT MECONTACT MEADMINLOGINLOGOUT

Welcome back admin

CONTACTS

sno	name	phone_no	msg	date	email
1	poomima	12345678970	hi this is demo	2023-02-14 22:25:29	poomima@gmail.com
2	jimin	0987654321	hi im jimin	None	jimin@gmail.com
3	VEATHAVALLI NADAR	1357908645	hii im veathavalli	None	vea@gmail.com
4	park jimin	135792468	im a member of bts.	2023-02-15 20:11:40	jimin@gmail.com
5	pooja	1364837131	hi im pooja	2023-02-16 08:32:56	pooja@gmail.com
6	poomima	172383289	hii lets demo	2023-03-16 11:22:05	aqws@gmail.com
7	poomi	1234567890	this is final	2023-03-25 10:39:13	poomi@gmail.com

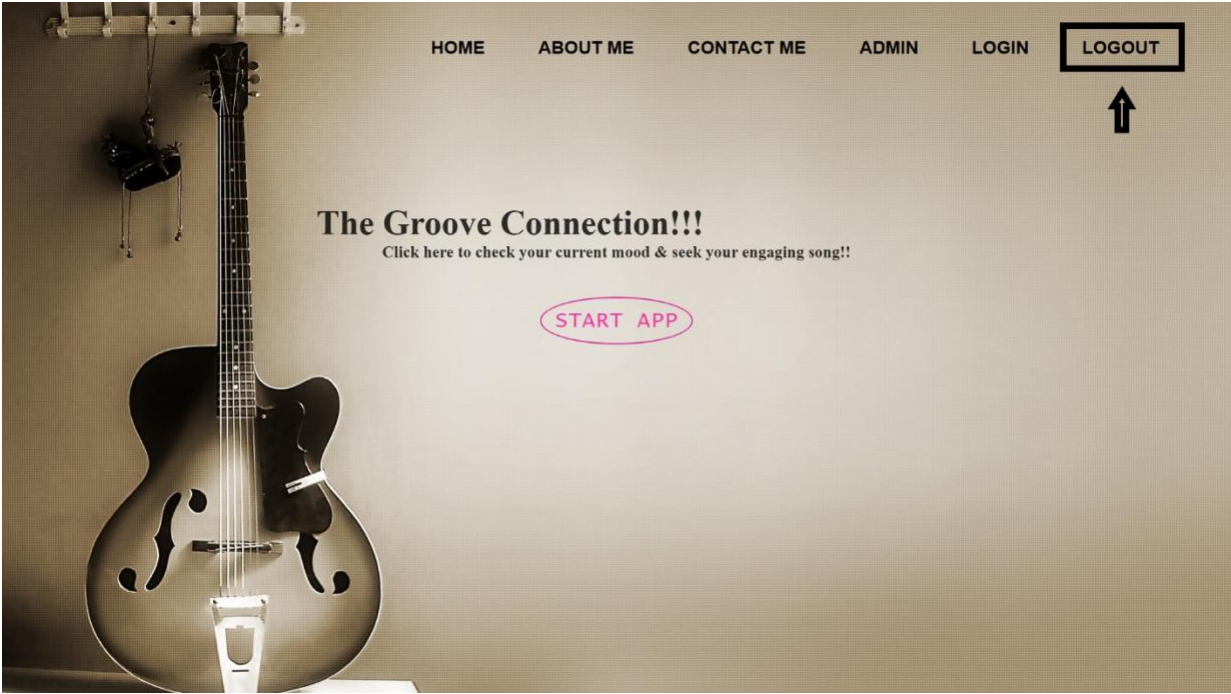
LOGIN

id	username	email	date
1	poomima	poomima@gmail.com	2023-03-29

DETAILS

sno	language	singer	date
19	tamil	None	2023-03-29
20	tamil	None	2023-03-29
21	None	Anirudh	2023-03-29
22	None	Anirudh	2023-03-29
23	hanna	hanna	2023-04-01

Logout:





## **Chapter 7: Conclusion and Future work**

# Conclusion

## **The Groove Connection consists of 2 Views:**

Admin

User

Admin has the authorities to manage server, view the registered user , contacts us information, selected language's and singer's name.

User can registered, login and logout. Users can select languages and singers name.

User can hear their mood based songs without manually typing. It is fun project that will make user full of joy while using our web app. User can contact the admin for help, manual info or any issues by filling the contact page form.

## **Future Enhancement**

Based on the current working of the website the update will include validation process, more accuracy of detection. Instead of recognizing emotion from facial expression the update would be by voice recognizing. User can get their suggested song from YouTube by using voice recognizing.

More language and more singer option will be provided in future scope.

## Reference

<https://www.w3schools.com/>

<https://medium.com/>

<https://www.quora.com/>

<https://www.github.com/>

<https://www.youtube.com/AkshitMadan>

<https://www.youtube.com/Edureka!>

<https://www.youtube.com/CodeWithHarry>