MUSIC RECOMMENDATION APPLICATION BASED ON FACIAL EXPRESSIONS

A PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

In partial fulfillment of requirements for the award of the Degree

BACHELOR OF TECHNOLOGY

By

Abdul Saherabegum	**17HP1A0583**
Akhila Umma	**17HP1A0561**
Guthula Jayasri Sai Nikitha	**17HP1A0571**

UNDER THE ESTEEMED GUIDANCE OF

Ms. MD ARSHA SULTANA (M.Tech, Asst. Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by A.I.C.T.E, New Delhi and Affiliated to JNTU Kakinada

Accredited by NAAC & An ISO 9001 : 2015 Certified Institution)

ALC Campus, Vijayawada, Andhra Pradesh – 520008

www.andhraloyola.org

2020-2021

ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Approved by A.I.C.T.E, New Delhi and Affiliated to JNTU Kakinada

Accredited by NAAC & An ISO 9001 : 2015 Certified Institution)

ALC Campus, Vijayawada, Andhra Pradesh – 520008

CERTIFICATE

This is to certify that the project report entitled **"MUSIC RECOMMENDATION APPLICATION BASED ON FACIAL EXPRESSIONS"** is submitted by **Ms. ABDUL SAHERABEGUM, Ms. AKHILA UMMA** and **Ms. GUTHULA JAYASRI SAI NIKITHA** to the JNTU KAKINADA in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Science and Technology is a bonafide work carried out by them under my supervision during the year 2020-2021.

Guide

Ms. MD ARSHA SULTANA,
M.Tech, Asst. Professor, CSE Dept

Head of the Department

Dr. CH RAJENDRA BABU
Ph.D, Associate Professor, CSE Dept

Signature of the External Examiner

ACKNOWLEDGEMENT

Firstly, we would like to convey our heartfelt gratitude to Almighty for the blessings on us to carry out this project work without any disruption.

We are extremely thankful to **Ms. MD Arsha Sultana** our guide throughout Project. Her wise approach made us to learn the minute details of the subject. Her matured and patient guidance paved a way for completing our project with the sense of satisfaction and pleasure. We also thank her for the at most independence and freedom of thought given to us during various phases of the project.

We are very much grateful **to Dr. CH. Rajendra Babu**, Professor and H.O.D of C.S.E, Departments, for his valuable guidance which helped us to bring out this project successfully.

We are thankful to our project coordinators **Dr. A Srinivasa Rao, Dr. CH. Rathna Jyothi** for their valuable guidance which helped us to bring this project successfully.

We would like to express our gratitude to the Director **Rev. Fr. Dr. Francis Xavier S. J** who has been a constant encourangement for us during the period of our prject.

We are thankful to our Principal **Dr. O. MAHESH** for his inspiration, intensive help and valuable support in every step of our project.

We would also like to convey our gratitude to our Technical Staff, for their support in every step of this project. We convey our sincere thanks to all the faculty and friends who directly or indirectly helped us for the successful completion of this project.

PROJECT ASSOCIATES

Abdul Saherabegum (17HP1A0583)

Akhila Umma (17HP1A0561)

Guthula Jayasri Sai Nikitha (17HP1A0571)

DECLARATION

We **Ms. ABDUL SAHERABEGUM, Ms. AKHILA UMMA** and **Ms. GUTHULA JAYASRI SAI NIKITHA,** hereby declare that the project report entitled **"MUSIC RECOMMENDATION APPLICATION BASED ON FACIAL EXPRESSIONS"** is an original work done in the Department of **Computer Science and Engineering**, **Andhra Loyola Institute of Engineering and Technology**, Vijayawada, during the academic year 2020-2021, in partial fulfillment for the award of the Degree of **Bachelor Of Technology** in Computer Science and Engineering. We assure that this project is not submitted in any other University or College.

Roll No	Name of the Student	Signature
17HP1A0583	**Abdul Saherabegum**	
17HP1A0561	**Akhila Umma**	
17HP1A0571	**Guthula Jayasri Sai Nikitha**	

ABSTRACT

Music is the form of art, which is known to have a greater connection with a person's emotion. It has got a unique ability to lift up one's mood. Our project focuses on building an efficient music recommendation system which determines the emotion of user using Facial Recognition techniques. Facial Recognition has widely attracted attention due to its enormous application value and market potential. Face recognition Technology is being implemented in various fields like security system, digital video processing, forensic investigation and many such technological advances. Our algorithm for Facial recognition being Viola Jones is implemented because it's proved to be more proficient than most of the existing systems. Moreover, on a larger dimension, recommending ready made playlists based on the user's mood would render savage of time and labor invested in performing the process manually. The overall concept of the system is to recognize facial emotion, mood and recommend songs efficiently. The proposed system will be both time and cost efficient.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In course of history, Music is the greatest creation of mankind. In this 21st century, we see so many people attached to their headphones listening to music at any given time. People tend to listen to different types of genres in different occasions depending on their mood. Creating customized playlists for different occasions at any given time depending on the user's mood will be very beneficial for the user. By using the advanced technology such as face detection and emotion recognition, recommendations can be possible for the better usage.

For our application, Human face is the key. We know how a face plays a great role in knowing the emotional status or the mood of the person. The mood of the person can be predicted, up to certain accuracy, with the help of certain features visible on the face. With today's technologies, extracting or filtering the inputs can be done directly with the help of a webcam or an external device along with a few softwares. This input can further be used for many applications and further studies. These technologies can be used to extract the moods of the user and those moods can be stored in a database. This information might then be used to make a group or pre-made playlists of songs and audio tracks that match the mood or the emotion generated from the information and data which was gathered. This reduces the time-consuming, conventional method of manually selecting songs, orrefining them into playlists and helps to create a suitable playlist based on the emotional status or the mood of the person.

With our application, each and every Music enthusiast irrespective of age group, gender, ethnicity and occasion who have accesss to Internet can use.

.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Only face detection and mood identification is being used for various other applications such as unlocking a smart phone, applying different filters for face, creating animoji or emoticons. Coming to music recommendation, it is available in Spotify but it recommends top hits from their respective country or Global charts and most streamed songs which are not based on emotion.

Drawbacks of the Existing System

- Music recommendation is not possible
- Playlist creation for different mood consumes lot of time
- Downloading songs and storing data in the phone
- No mood-based recommendations exist

2.2 PROPOSED SYSTEM

The proposed System would work by first giving a simple enough interface which gives an option to scan a live face or consider a default value.This application being one stop for both mood identification and recommendation of music, it can be highly beneficial for the user.

Advantages of Proposed Systems

- Mood enhancing playlists are recommended at any time.
- Time Conserving.
- Energy saving.
- Money saving.
- User friendly.

2.3 SYSTEM REQUIREMENTS SPECIFICATION

2.3.1 Software Specifications

- Operating system : Windows XP or MAC
- Frame work : Python 3.8
- Front-End : XML, PHP
- Database : MYSQL
- Model Design : Rational Rose

2.3.2Hardware Specifications

- Processor : Intel Pentium 4.0
- Ram : 2GB
- Hard disk : 500GB

CHAPTER 3

FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Economical and Social feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Economical Feasibility
- Social Feasibility

3.1 TECHNICAL FEASIBILITY

Technical feasibility refers to the ability of the process to take advantage of the current state of the technology in pursuing further improvement. We are using Python, HTML, CSS which are freely available and the technical skills required are manageable. Time limitations of the product development and the ease of implementing using these technologies are synchronized. The developed application must have a modest requirement, as only minimal or null changes are required for implementing.

3.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Using freely available resources to develop the application will be beneficial to cover it up within the budget. Only while upgrading the application, we need to purchase for customization.

- Cost benefit analysis
- Long-term returns
- Maintenance cost

3.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it is a necessary. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make them familiar with it. Their level of confidence must be raised so that they can also be able to make some constructive criticism, which is welcomed, as they will be the final user of the system.

Our application can be a good option for the music enthusiasts as it comes with so many good features. They don’t need to spend money to buy storage and download songs. They don’t need to organize playlists. They don’t need to think hard to which music they need to listen. With all these advantages our application can be very much useful to the user.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

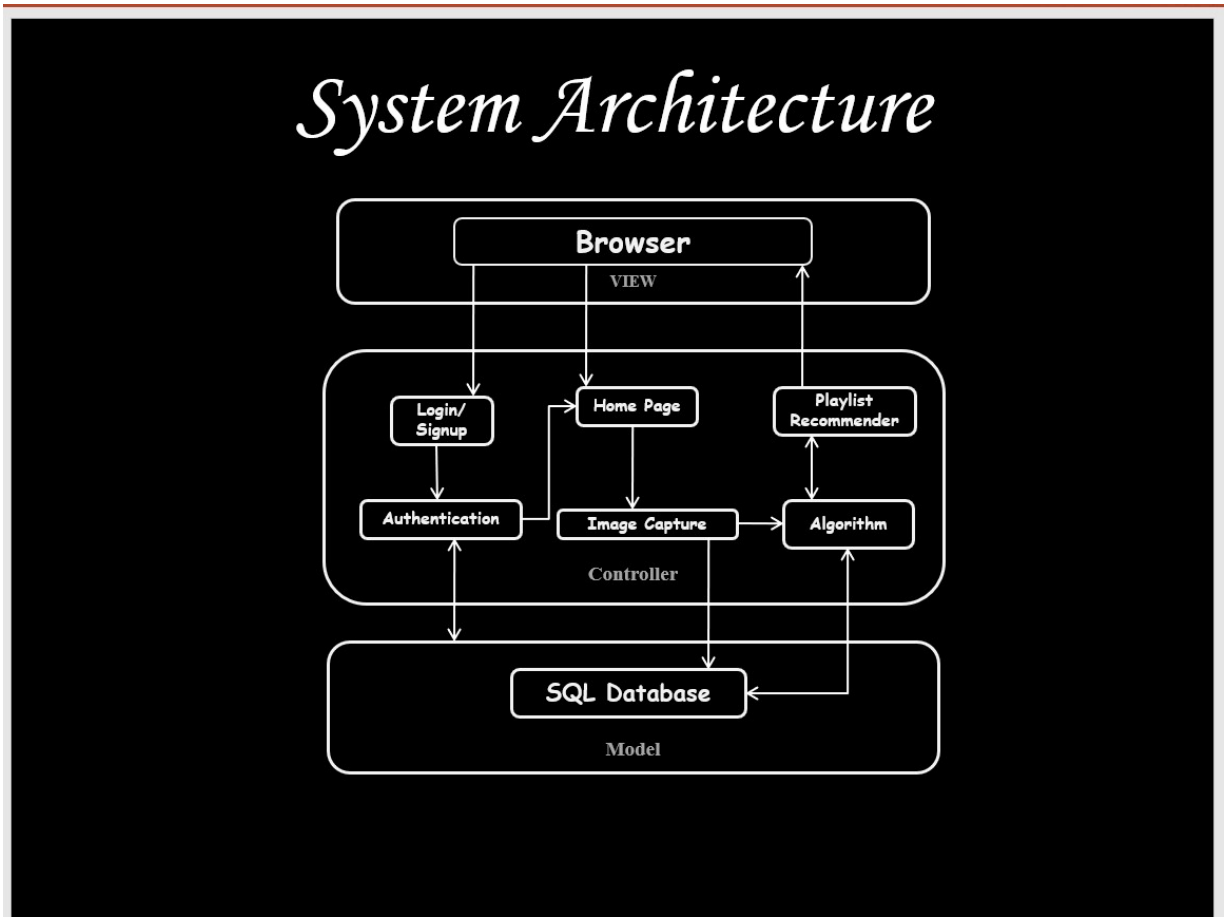

Fig.4.1 Architecture

4.2 DESIGN DIAGRAMS

Scenario Based Building

It vaguely represents a synthesis process explained in previous section. In mixing top-down and bottom-up design it often appears that we start in the middle of the problem and work our way both up and down there the two basic modern design strategies employed in software design are:

1. Top-Down Design

Top-down Design is basically a decomposition process, which focuses on the flow of control. At later stages it concerns itself with the code production. The first step is to study the overall aspects of the tasks at hand and to break it into a number of independent modules. The second step is to break each one of these modules further into independent sub-modules. The process is repeated one to obtain modules, which are small enough to group mentally and to code in a straightforward manner. One important feature is that at each level the details of the design at the lower level are hidden.

2. Bottom-Up Design

In a bottom-up design one first identifies and investigates parts of design that are most difficult and necessary designed decision are made the remainder of the design is tailored to fit around the design. In a complex problem; it is often difficult to decide how to modularize the various procedures in such cases one might consider a list of system inputs and decide what functions are necessary to process these inputs. This is called back to front design. Similarly, one can start with the required outputs and work backwards evolving so called front-back design. We have applied both the top down and bottom-up approach in our design approach.

4.2.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined.

Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

System boundary boxes (optional)

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

Include

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values.

Extend

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

Generalization

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

Associations

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Identified Use Cases

Now it’s time to identify the use cases. A good way to do this is to identify what the actors needs from the system. In banking system, a customer will need to open accounts, deposit and withdraw funds, request check books and similar functions. So, all of these can be considered as use cases. Top level use cases should always provide a complete function required by an actor. You can extend or include use cases depending on the complexity of the system. Once you identify the actors and the top-level use case you have basic idea of the system. Now you can fine tune it and add extra layers of detail to it.

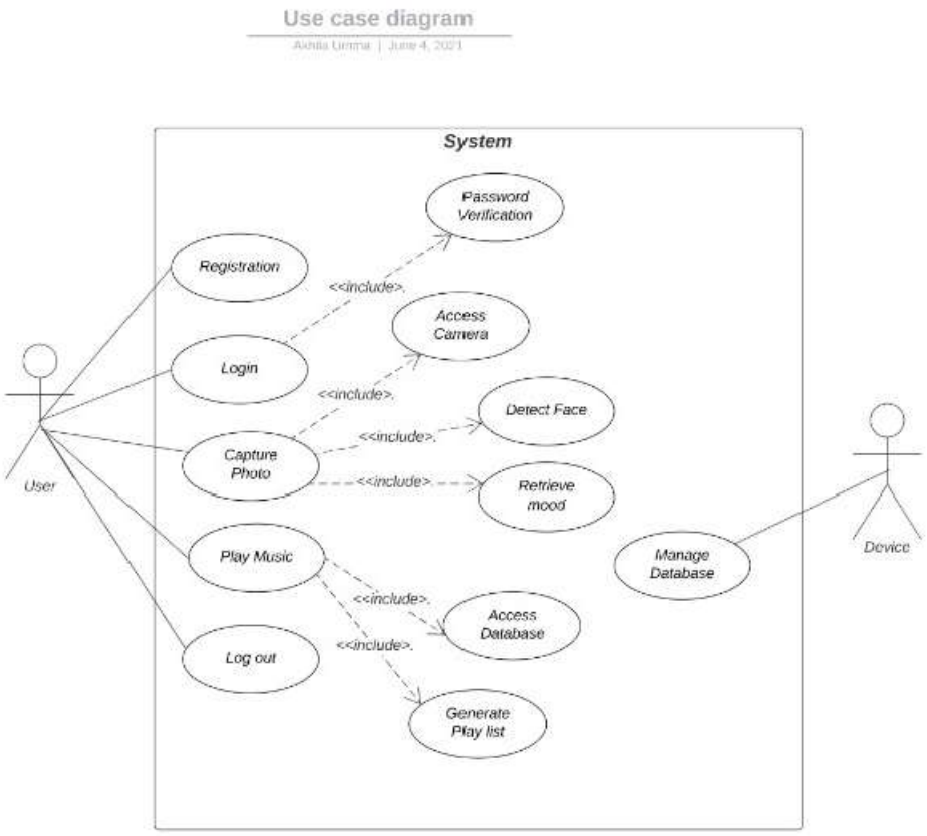

Fig.4.2 Use Case Diagram

4.2.2 Class Diagram

Class diagrams are visual representations of the static structure and composition of a particular system using the conventions set by the Unified Modeling Language (UML). Out of all the UML diagram types it is one of the most used ones. System designers use class diagrams as a way of simplifying how objects in a system interact with each other. Using class diagrams, it is easier to describe all the classes, packages, and interfaces that constitute a system and how these components are interrelated. For example, a simple class diagram may be used to show how an organization such as a convenient store chain is set up. On the other

hand, precisely detailed class diagrams can readily be used as the primary reference for translating the designed system into a programming code. In the class diagram these classes are represented with boxes which contain three parts:

- The top partition contains the name of the class.
- The middle part contains the class's attributes.
- The bottom partition shows the possible operations that are associated with the class.

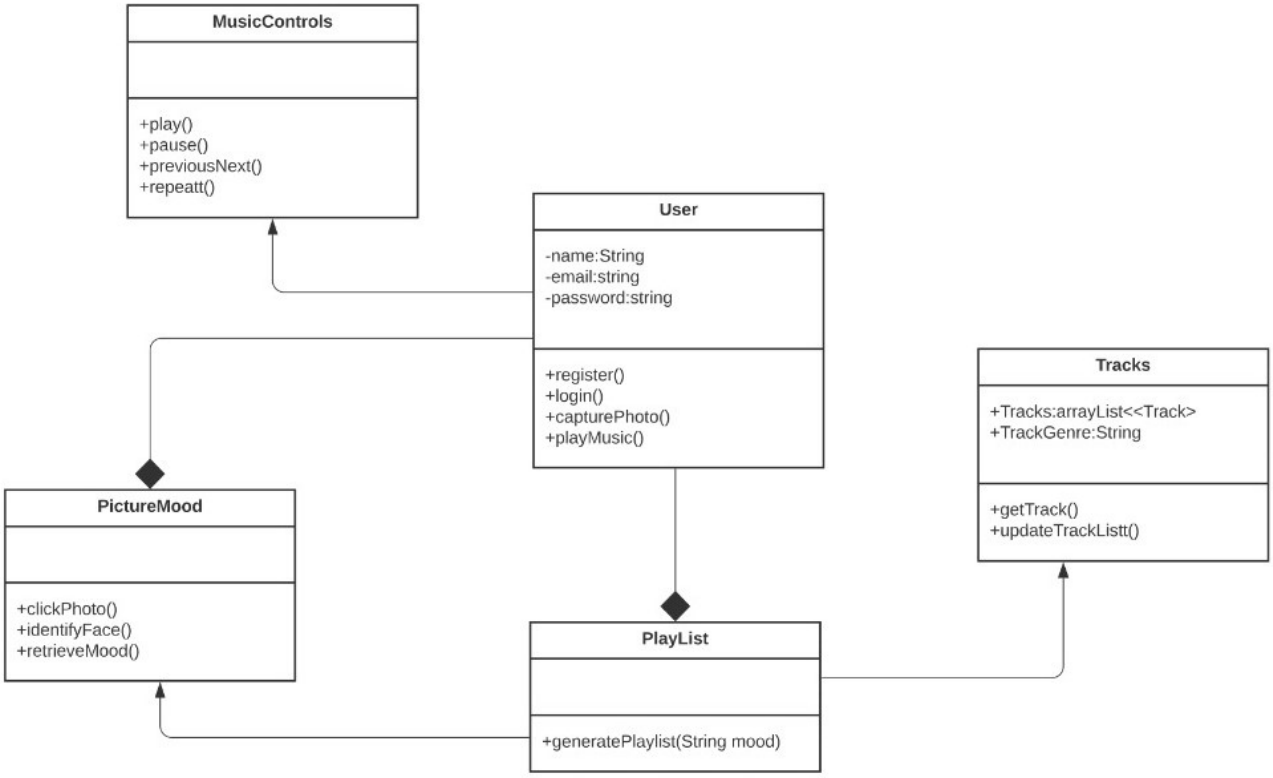

Fig.4.3 Class Diagram

4.2.3 Sequence diagram

A **Sequence diagram** is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages are written on horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response. Asynchronous calls are present in multithreaded applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent those processes are being performed in response to the message (Execution Specifications in UML).

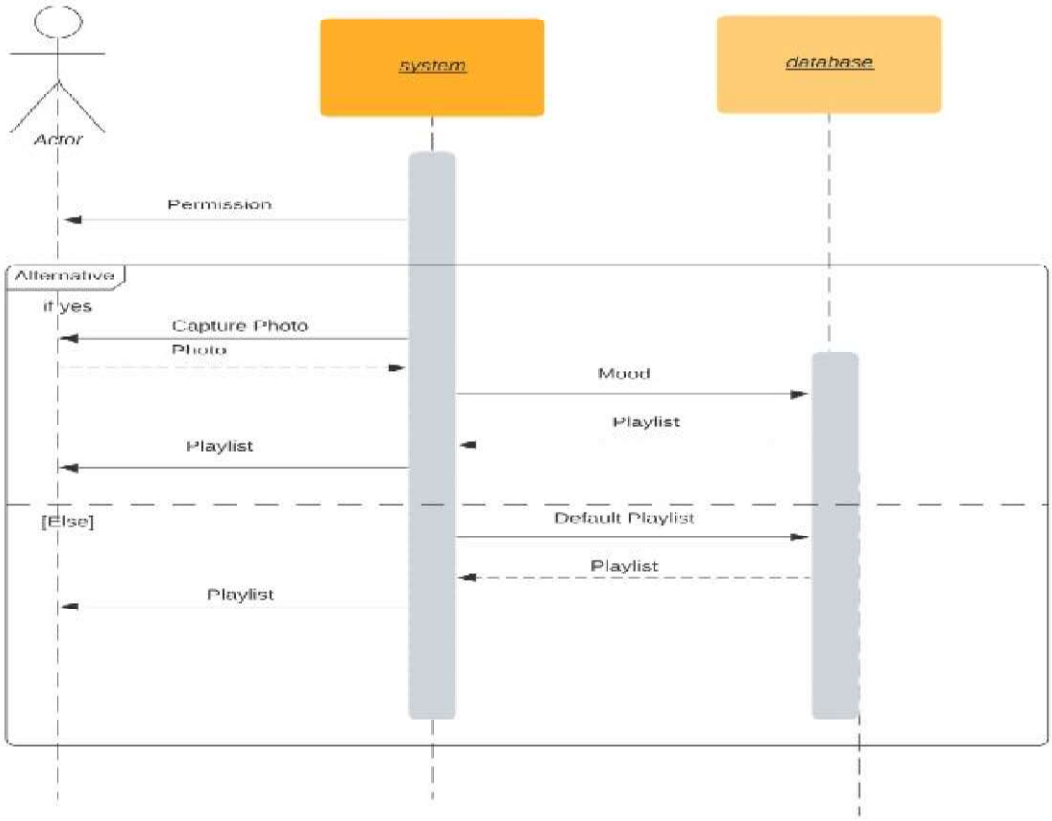

Fig.4.4 Sequence Diagram

4.2.4 Collaboration Diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in realtime. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagram.

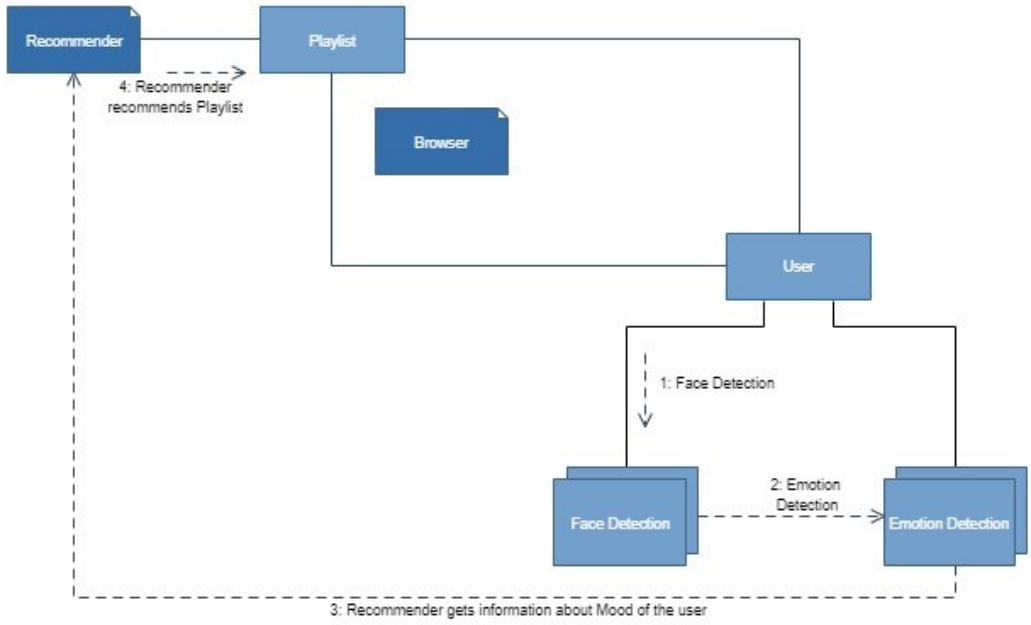

Fig.4.5 Collaboration Diagram

4.2.5 State Chart Diagram

A **state diagram** is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and tracks the different states of its objects through the system".

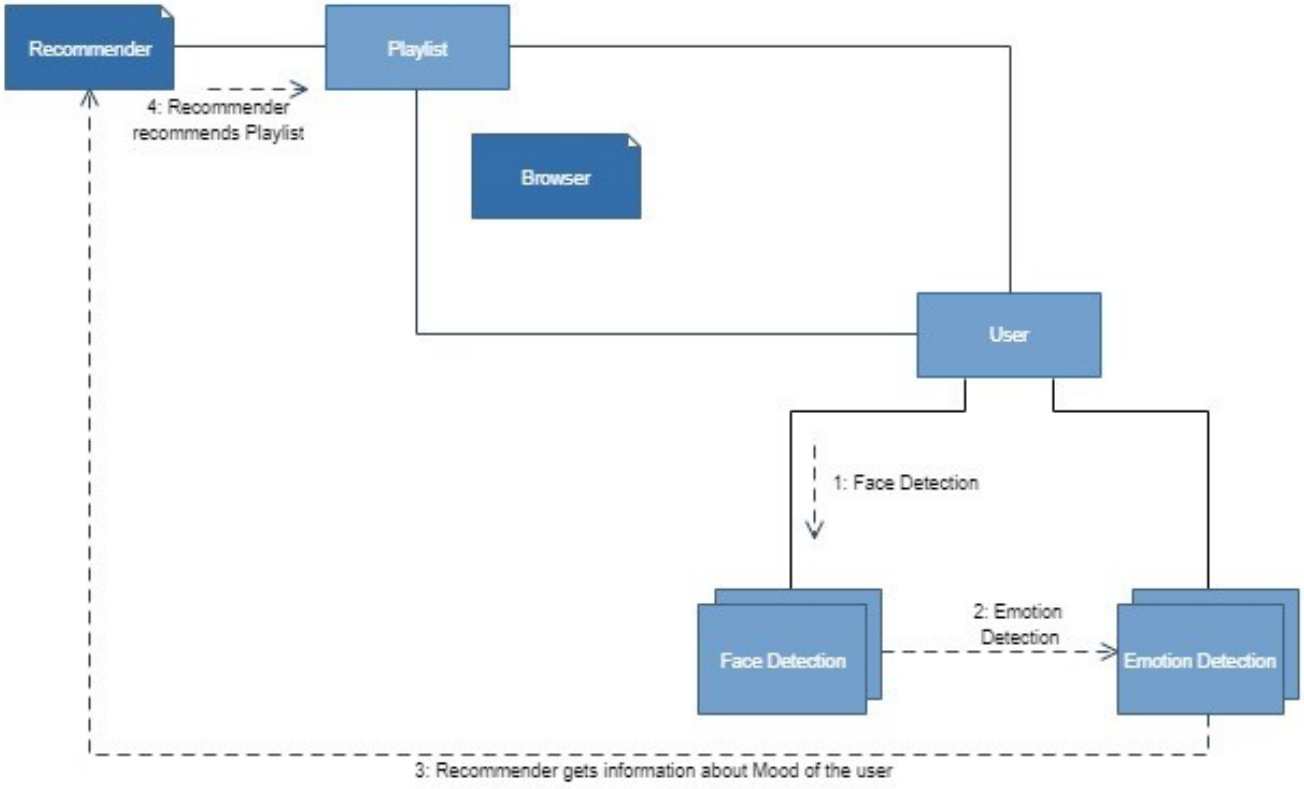

Fig.4.6 State Chart Diagram

4.2.6 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the UnifiedModelingLanguage, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represent actions;
- Diamonds represent decisions;
- Bars represent the start (split) or end (join) of concurrent activities;
- A black circle represents the start (initial state) of the workflow;
- An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen.

Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

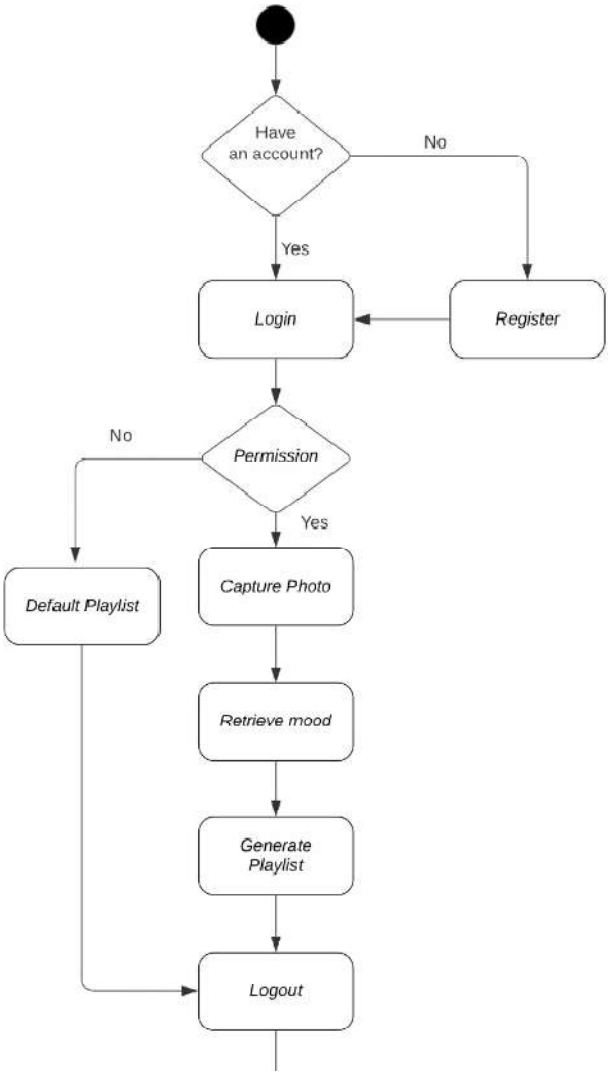

Fig.4.7 ActivityDiagram

4.2.7 Entity –Relationship Diagram

An **entity–relationship model** (**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships.

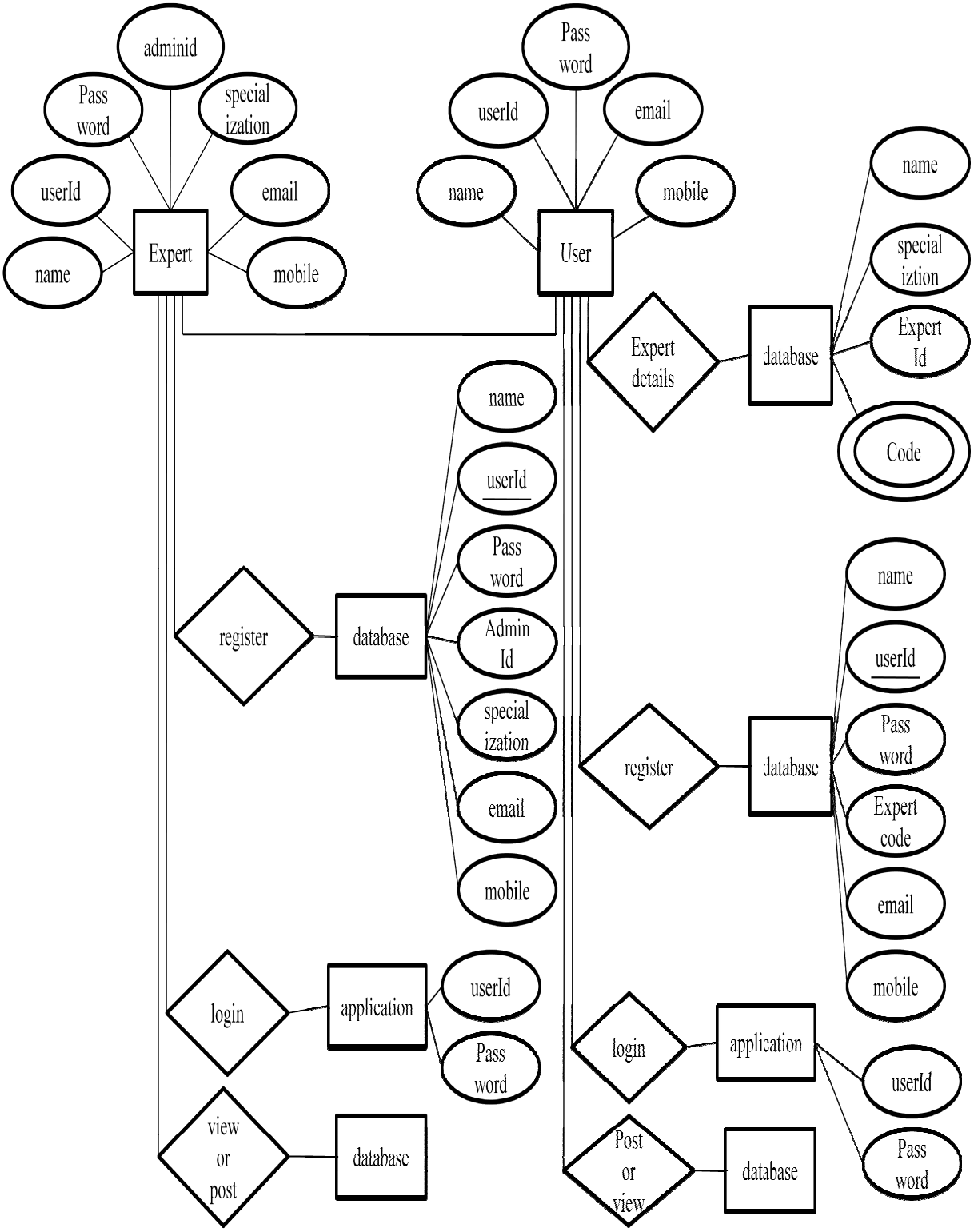

Fig.4.8 Entity –Relationship Diagram

4.3 DATABASE TABLES

4.3.3 User Registration

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	name	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
2	pass	varchar(40)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	repass	varchar(40)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	email	varchar(40)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Fullte

Print Propose table structure Track table Move columns Normalize

Fig.4.9 User Registration

CHAPTER 5

TECHNOLOGY DESCRIPTION

5.1 SOFTWARE DESCRIPTION

Introduction to Python

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming.

Feature of Python

The main properties of the python, which made python so popular, are as follow:

- Easy to code
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Portable language
- Integrated language
- Large Standard Library
- Dynamically Typed Language

Easy to code:

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

Free and Open Source:

Python language is freely available at the official websites what resources a class can access such as reading and writing to the local disk**.**

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior. Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules. Basic concepts of OOPs are:

- Object.
- Class.
- Inheritance.
- Polymorphism.
- Abstraction.
- Encapsulation.

GUI Programming Support:

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

Extensible feature:

Python is an **Extensible** language. We can write some Python code into C or C++ language and also, we can compile that code in C/C++ language.

Python is Portable language:

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

Python is Integrated language:

Python is also an integrated language because we can easily integrated python with other languages like c, c++, etc.

Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. Unlike other languages C, C++, Java, etc. there is no need to compile python code; this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode**.**

Large Standard Library:

Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python such as regular expressions, unit-testing, web browsers, etc.

Dynamically Typed Language:

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

Packages:

- **OpenCV**

  OpenCV Python is a library of Python bindings designed to solve computer vision problems. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

- **Pandas**

  Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

  Pandas is well suited for many different kinds of data:

  - Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
  - Ordered and unordered (not necessarily fixed-frequency) time series data.
  - Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
  - Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a pandas data structure

  The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

- **NumPy package**

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

- **Spotipy package**

Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform.

- **Keras**

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation**.**

- **Statistics**

Python has a built-in module that you can use to calculate mathematical statistics of numeric data. The statistics module was new in Python 3.4.

- **OS**

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality.

5.2 DATABASE DESCRIPTION

INTRODUCTION TO MYSQL:

MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Free-software open-source projects that require a full-featured database management system often use MySQL. Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Google(though not for searches), Facebook, Twitter,[Flickr and YouTube.

Advantages of MySQL

Whether you are a Web developer, CNE[SM], or a dedicated network administrator with an interest in building database applications, MySQL is easy to use, yet extremely powerful, secure, and scalable. And because of its small size and speed, it is the ideal database solution for Web sites.

Some of its advantages include the following:

- **It's easy to use:** While a basic knowledge of SQL is required—and most relational databases require the same knowledge—MySQL is very easy to use. With only a few simple SQL statements, you can build and interact with MySQL.
- **It's secure:** MySQL includes solid data security layers that protect sensitive data from intruders. Rights can be set to allow some or all privileges to individuals. Passwords are encrypted.
- **It's inexpensive:** MySQL is included for free with NetWare® 6.5 and available by free download from MySQL Web site.
- **It's fast:** In the interest of speed, MySQL designers made the decision to offer fewer features than other major database competitors, such as Sybase* and Oracle*. However, despite having fewer features than the other commercial database products, MySQL still offers all of the features required by most database developers.

- **It's scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **It manages memory very well:** MySQL server has been thoroughly tested to prevent memory leaks.
- **It supports Novell Cluster Services:** MySQL on NetWare runs effectively with Novell® Cluster Services™, letting you add your database solution to a Novell cluster. If one server goes down, MySQL on an alternate server takes over and your customers won't know that anything happened.
- **It runs on many operating systems:** MySQL runs on many operating systems, including Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others.
- **It supports several development interfaces:** Development interfaces include JDBC, ODBC, and scripting (PHP and Perl), letting you create database solutions that run not only in your NetWare 6.5 environment, but across all major platforms, including Linux, UNIX, and Windows.

PHP:

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.PHP is basically used for developing web-based software applications.

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

HTML:

HTML is the standard markup language for Web pages. With HTML you can create your own Website.HTML describes the structure of a web page semantically. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

CHAPTER 6

METHODOLGY

MODULE DESCRIPTION

There are two modules. They are

- Face detection
- Emotion detection
- Song recommendation

6.1 Face detection

After acquiring the image from webcam, the system will start to detect the face by applying the Viola-Jones algorithm. This Algorithm is considered as one of the first frameworks that recognize the objects in real time.

There are 2 stages in the Viola-Jones Algorithm:

- Training
- Detection

The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.

Viola-Jones outlines a box and searches for a face within the box. It is essentially searching for these Haar-like features.

There are 3 types of Haar-like features that Viola and Jones identified in their research

- Edge features
- Line-features
- Four-sided features

Viola-jones scans the images using a sub-window to detect the features of the face in the image. When the face is determined, the image is cropped to contain the face only, to enhance the proposed system performance. Also, the Viola-Jones is used to identify and crop

the left and right eyes and mouth separately. The outcome of this step is four images, face, right eye, left eye, and mouth images. These specific outcomes will be used to later for Emotion Classification.

6.2 Emotion detection

The detected human face is set to be processed further for recognizing the mood by using the Fisher Face classifier. Fisher Face is one of the popular algorithms used in face recognition, and is widely believed to be superior to other techniques, such as Eigen face because of the effort to maximize the separation between classes in the training process. Image recognition using Fisher Face method is based on the reduction of face space dimension using Principal Component Analysis (PCA) method, then apply Fishers Linear Discriminant (FDL) method or also known as Linear Discriminant Analysis (LDA) method to obtain feature of image characteristic. After obtaining the emotion, the user is asked to confirm the emotion. For any changes, the application allows the user to recapture the image.

6.3 Song Recommendation

Based on the user's mood which is identified as one of the emotions Happy, Sad, Neutral or Angry, the respective playlist will be displayed. For happy emotion, the cheerful and party type playlist will be recommended. For neutral emotion, classy and edm type music will be recommended to savor the moment. For emotions such as sad and angry, a playlist will be recommended with r&b, pop music which will be able to enhance the user mood to lighten up and feel at peace.

For no mood choice, default happy songs are played.

CHAPTER 7

CODING AND IMPLEMENTATION

7.1 CODE

Emotions.py(for detecting emotions)

```
import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input



USE_WEBCAM = True    # If false, loads video file source

# parameters for loading data and images

emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape

frame_window = 10
# To be used for the mode comparison with the emotion window
```

```
emotion_offsets = (20, 40)   # Hyper parameters for the bounding facial box

# loading models

face_cascade= cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')   # face
detection model

emotion_classifier = load_model(emotion_model_path)
 # 'fer2013' dataset trained model for emotions

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')

video_capture = cv2.VideoCapture(0)       # 0 for default camera or source selection

# Select video or webcam feed
cap = None
if (USE_WEBCAM == True):
   cap = cv2.VideoCapture(0) # Webcam source
else:
   cap = cv2.VideoCapture('./demo/dinner.mp4') # Video file source

while cap.isOpened(): # True:
   ret, bgr_image = cap.read()
```

```
gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
  # gray image for HOGs
rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
  # color image for video frame
  faces=face_cascade.detectMultiScale(gray_image,scaleFactor=1.1,
minNeighbors=5,minSize=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)
   for face_coordinates in faces:
 x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
          # extracting faces from gray image
gray_face = gray_image[y1:y2, x1:x2]
 # storing faces into gray_face
          try:
gray_face = cv2.resize(gray_face, (emotion_target_size))
# resizing for emotion detection
          except:
continue

gray_face = preprocess_input(gray_face, True)
#converting image into a float 32bit Array
gray_face = np.expand_dims(gray_face, 0)
          #adding 0 axis to the facial float 32-bit array
gray_face = np.expand_dims(gray_face, -1)
          # adding a new axis to the facial 32-bit array

emotion_prediction = emotion_classifier.predict(gray_face)
          # predicting the emotion
emotion_probability = np.max(emotion_prediction)
          # select the emotion with the maximun probability

emotion_label_arg = np.argmax(emotion_prediction)
           # return the index of the emotion with max probability

emotion_text = emotion_labels[emotion_label_arg]
```

```
            # create the emotion label with the label from the index of emotion
emotion_window.append(emotion_text)
 # add the label to emotion window

if len(emotion_window) >frame_window:           # limiting the list to 10 items only
emotion_window.pop(0)
 try:
emotion_mode = mode(emotion_window)
  # find the mode of the emotion window items
 except:
         continue

 if emotion_text == 'angry':    # giving the text colors with numpy (R,G,B) values
       color = emotion_probability * np.asarray((255, 0, 0))
elifemotion_text == 'sad':
        color = emotion_probability * np.asarray((0, 0, 255))
elifemotion_text == 'happy':
        color = emotion_probability * np.asarray((255, 255, 0))
elifemotion_text == 'surprise':
        color = emotion_probability * np.asarray((0, 255, 255))
 else:
        color = emotion_probability * np.asarray((0, 255, 0))

  color = color.astype(int)
  color = color.tolist()

draw_bounding_box(face_coordinates, rgb_image, color)
  # finally drawing the the bounding box

draw_text(face_coordinates,     rgb_image,     emotion_mode,color,     0,     -45,     1,     1)
# adding the emotion text

bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
  cv2.imshow('window_frame', bgr_image)
```

```
  if cv2.waitKey(1) & 0xFF == ord('q'):                    # exit conditions
break
 print(emotion_mode)
cap.release()
 cv2.destroyAllWindows()
```

no_capture.py (play songs with out emotion)

```
import pandas as pd
import os
import spotipy
import json
from spotipy.oauth2 import SpotifyClientCredentials

sp=spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.environ["SPOTIPY
_CLIENT_ID"],
client_secret=os.environ["SPOTIPY_CLIENT_SECRET"]))
csv_file = pd.read_csv('happy.csv')
recomnd_list_songs=[]
sorted_csv = csv_file.sort_values(by=['popularity'])
for i in range(5):
default_song_id=sorted_csv._get_value(i, 'id')
default_song_info=sp.track(default_song_id)
recomnd_list_song=[sorted_csv._get_value(i, 'name'),int(sorted_csv._get_value(i,
'year')),sorted_csv._get_value(i, 'artists'),default_song_info['external_urls']['spotify']]
recomnd_list_songs.append(recomnd_list_song)
print(json.dumps(recomnd_list_songs))
```

7.2 Output screenshots:

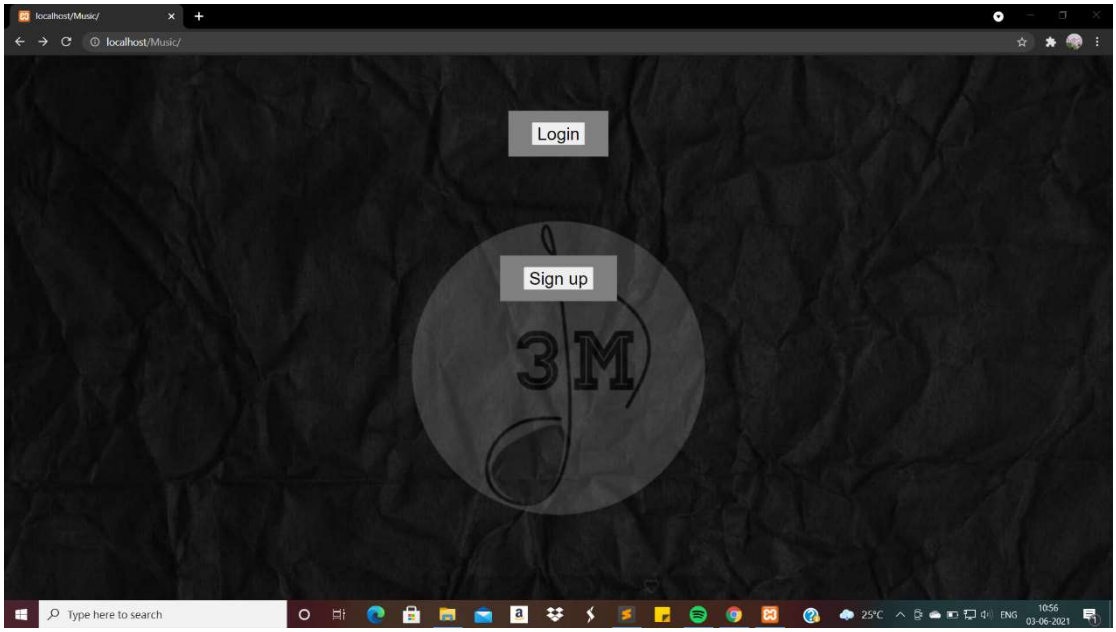

Fig. 7.1 Home Page

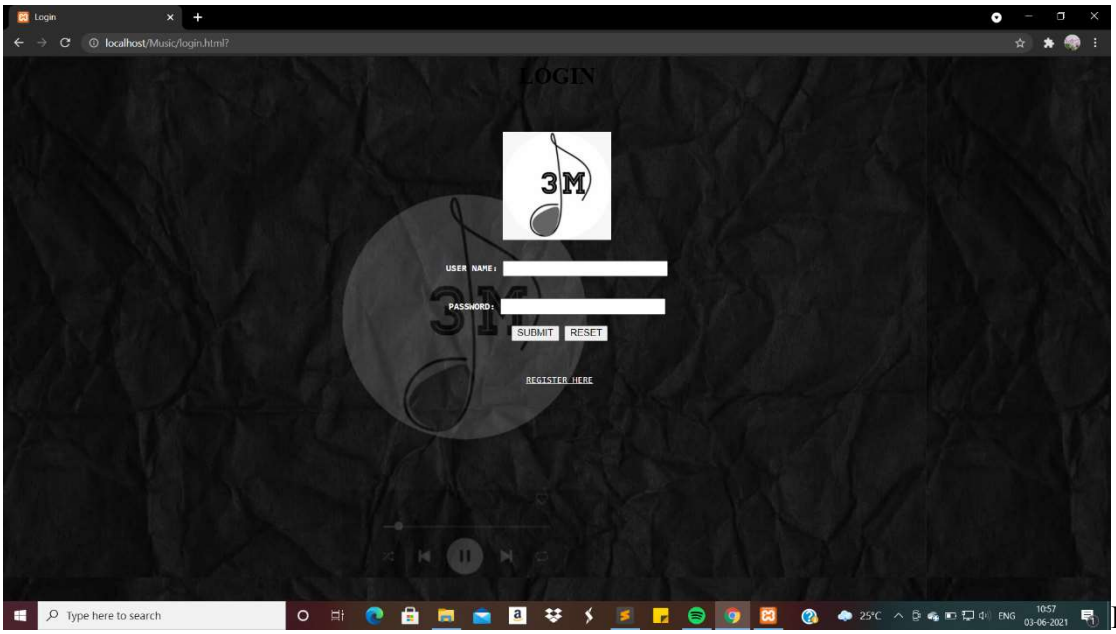

Fig. 7.2 Login page

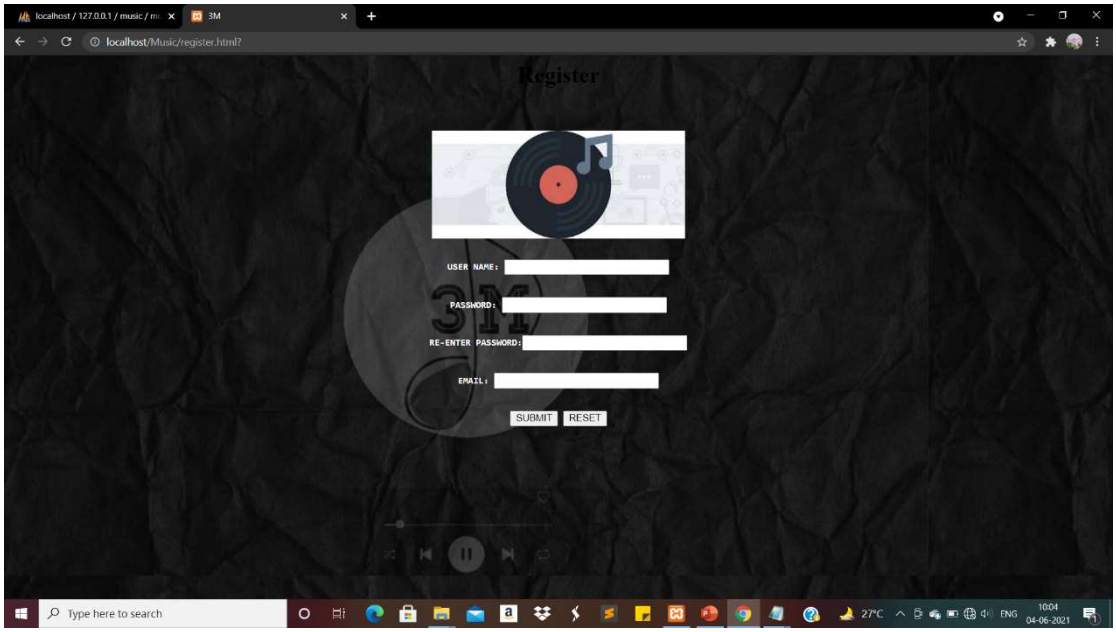

Fig. 7.3 Registration page

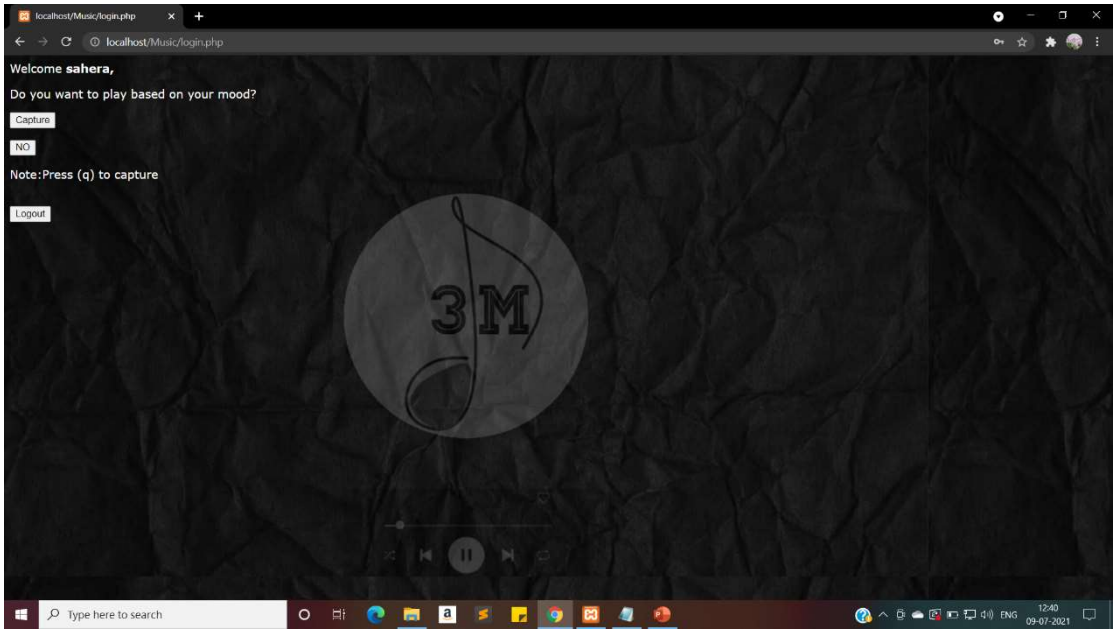

Fig.7.4 User Choice Page

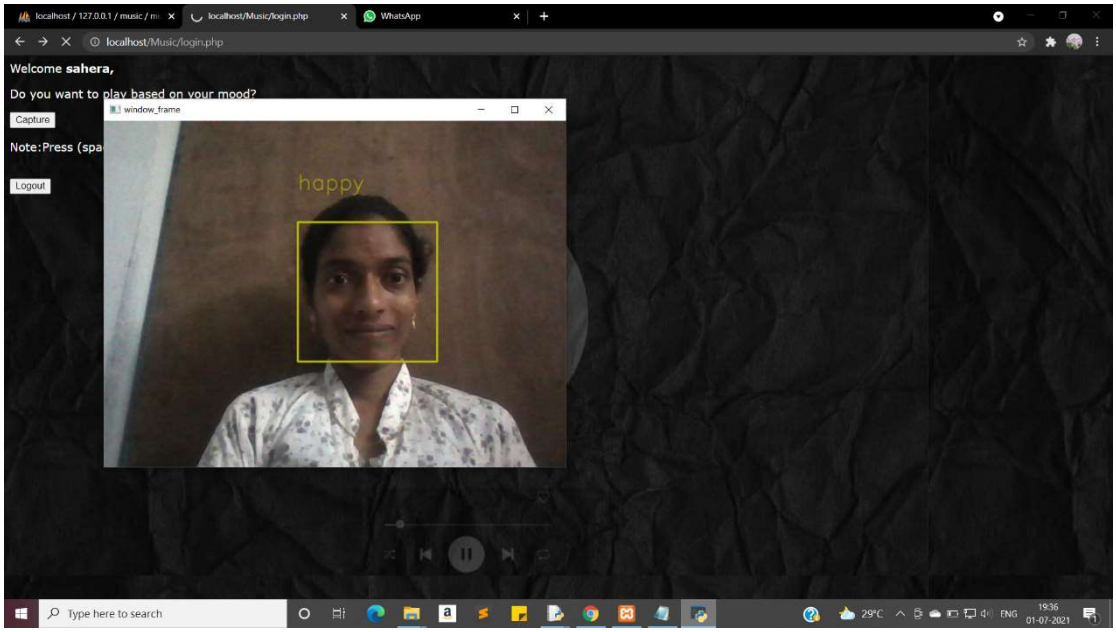

Fig. 7.5 Happy Face

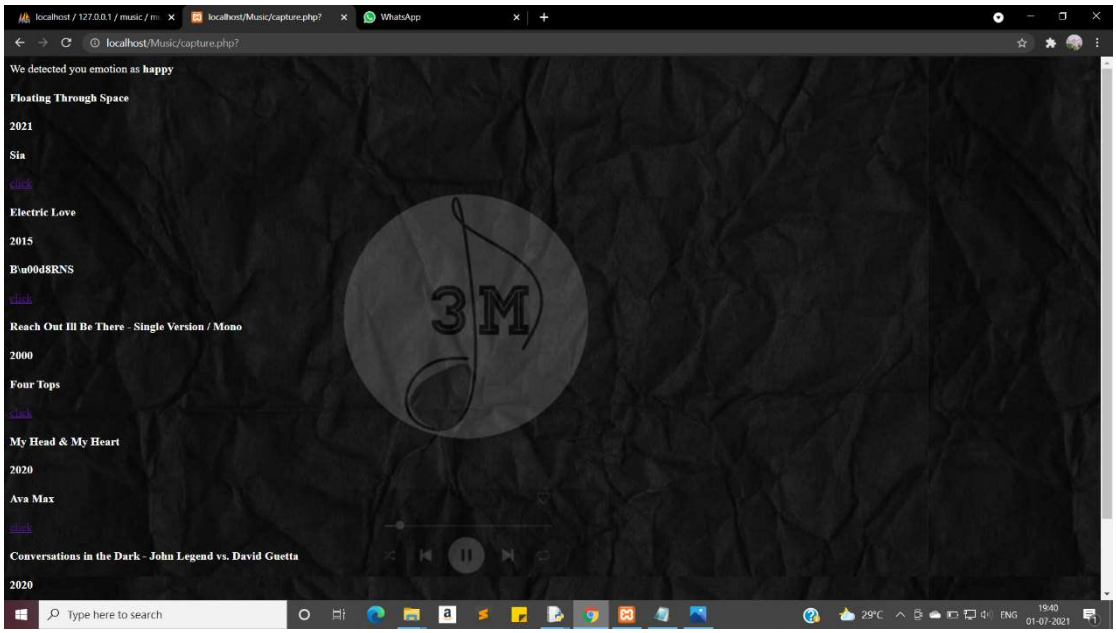

Fig .7.6 Happy songs

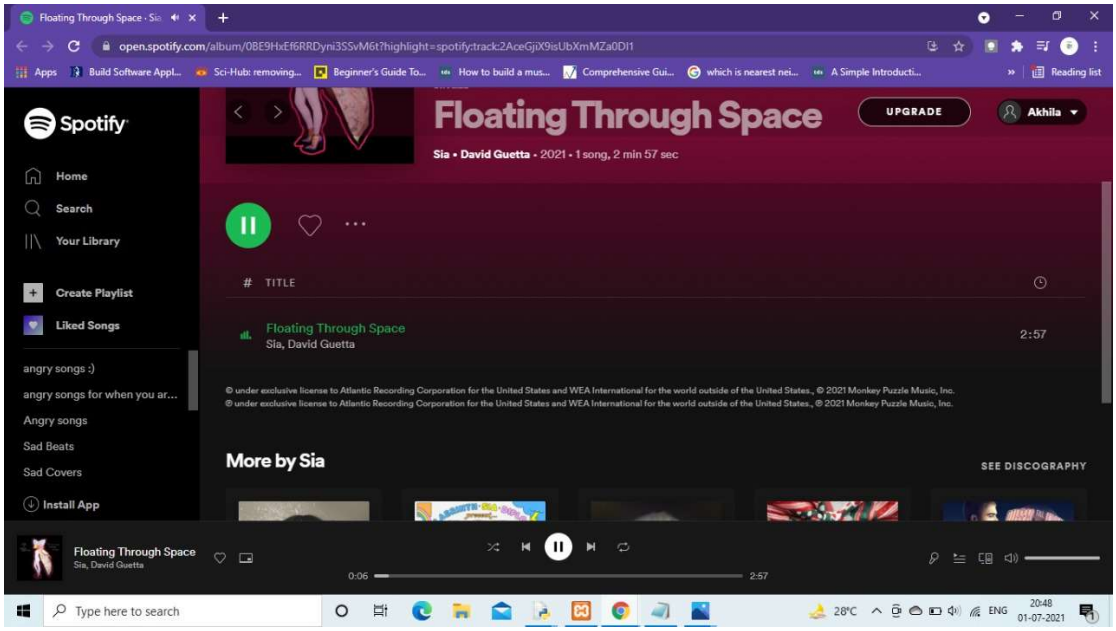

Fig.7.7 Happy songs playing

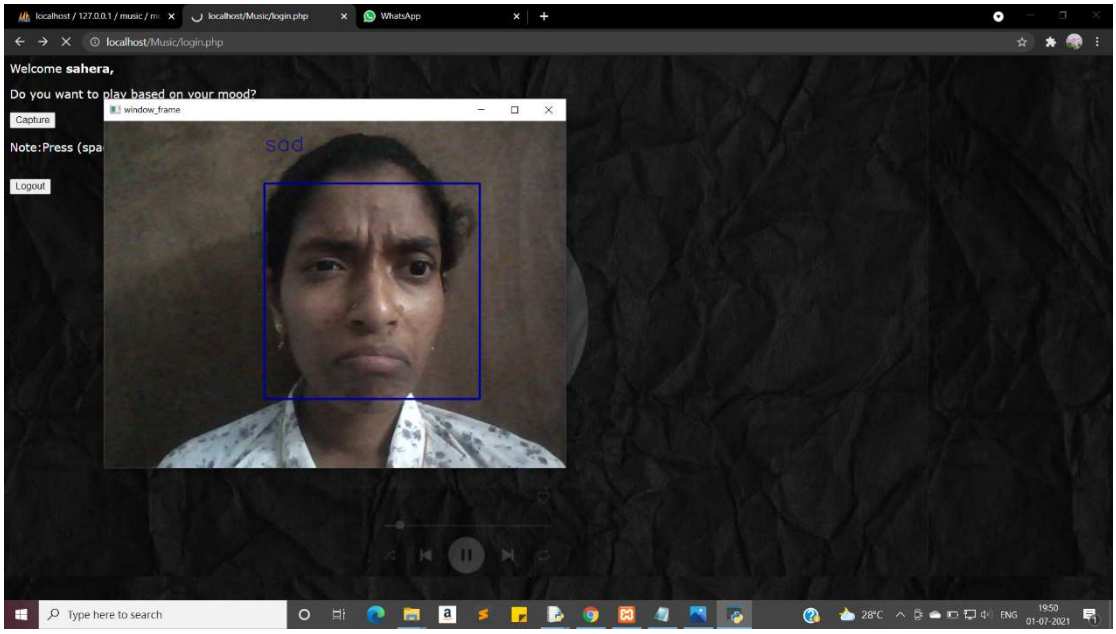

Fig.7.8 Sad Face

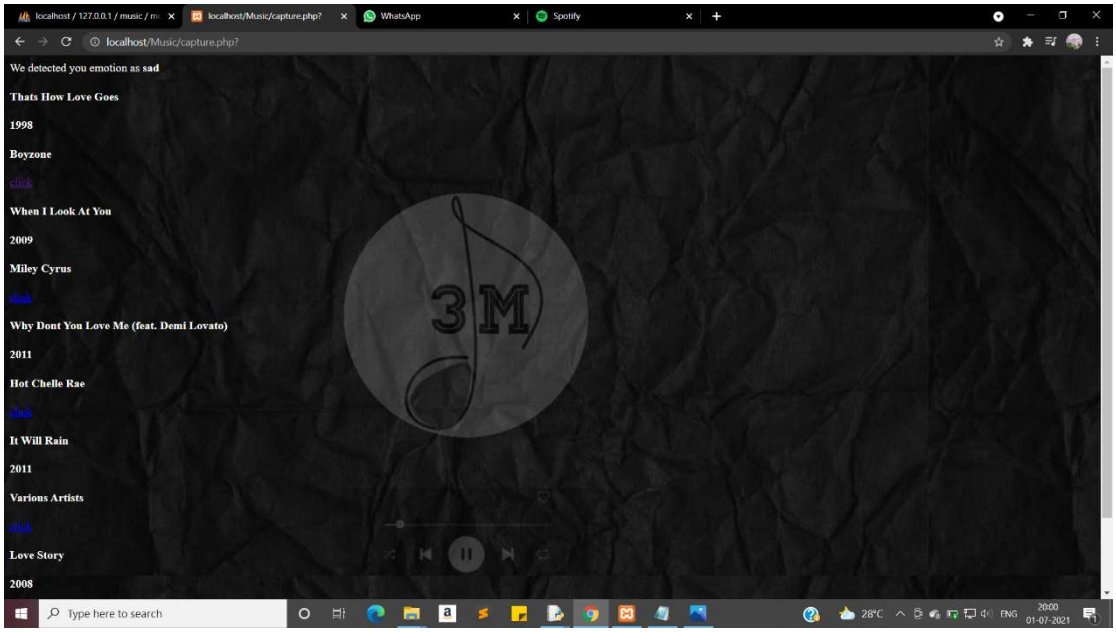

Fig 7.9 Sad songs

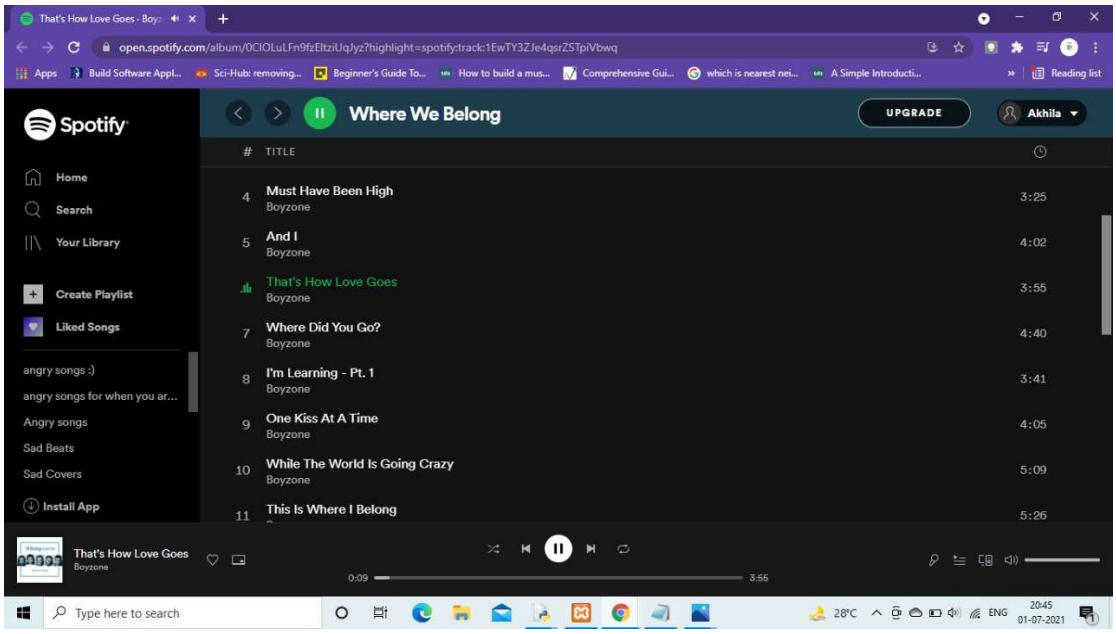

Fig.7.10 Sad songs playing

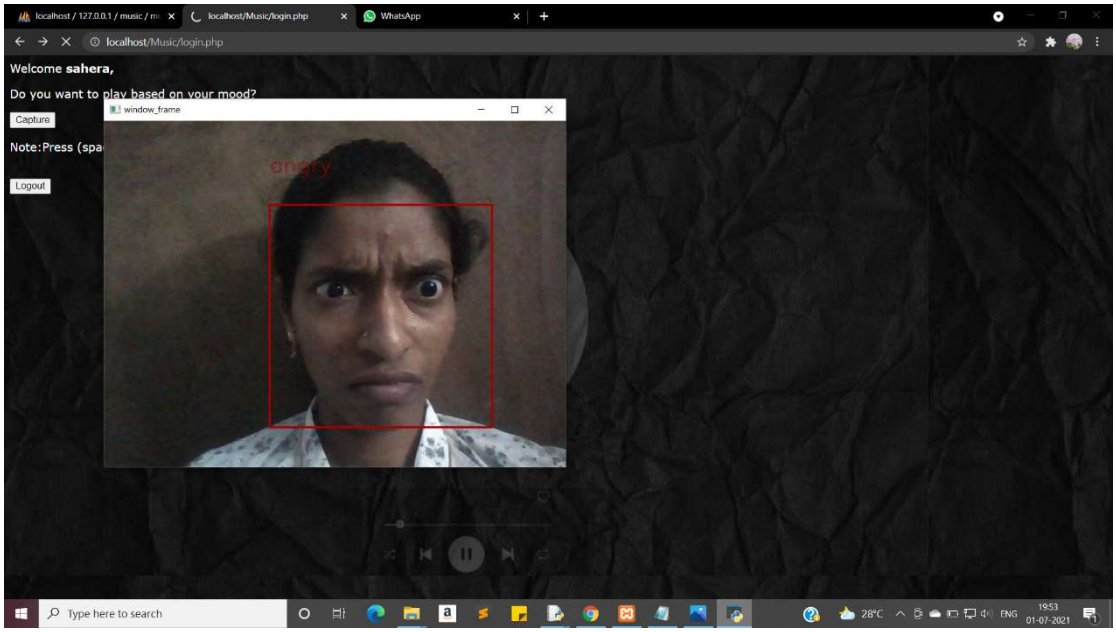

Fig 7.11 Angry face

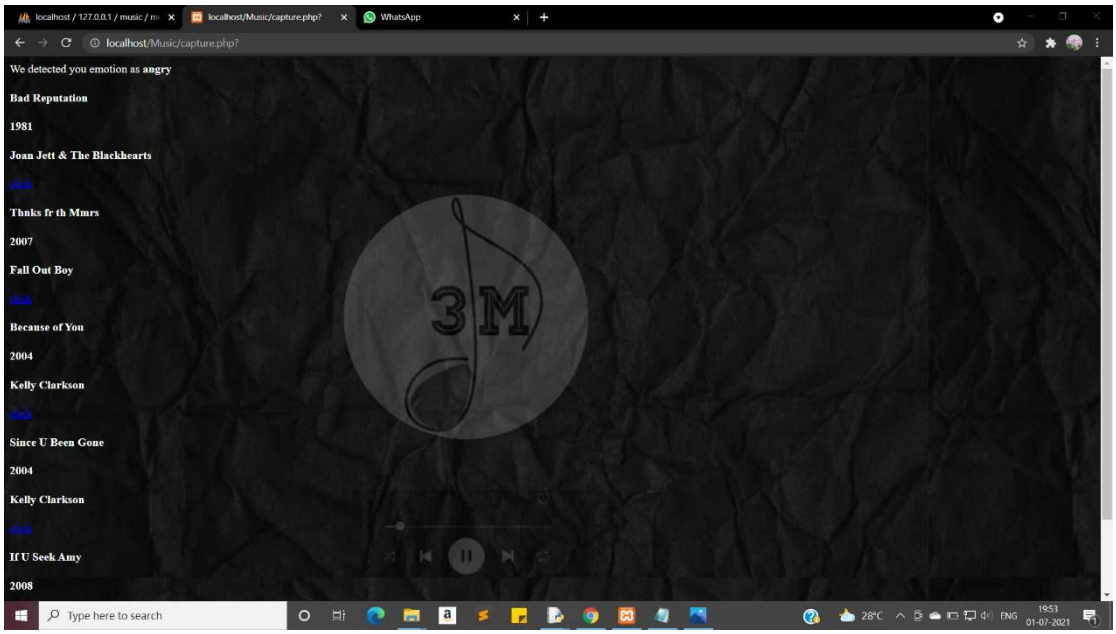

Fig 7.12 Angry songs

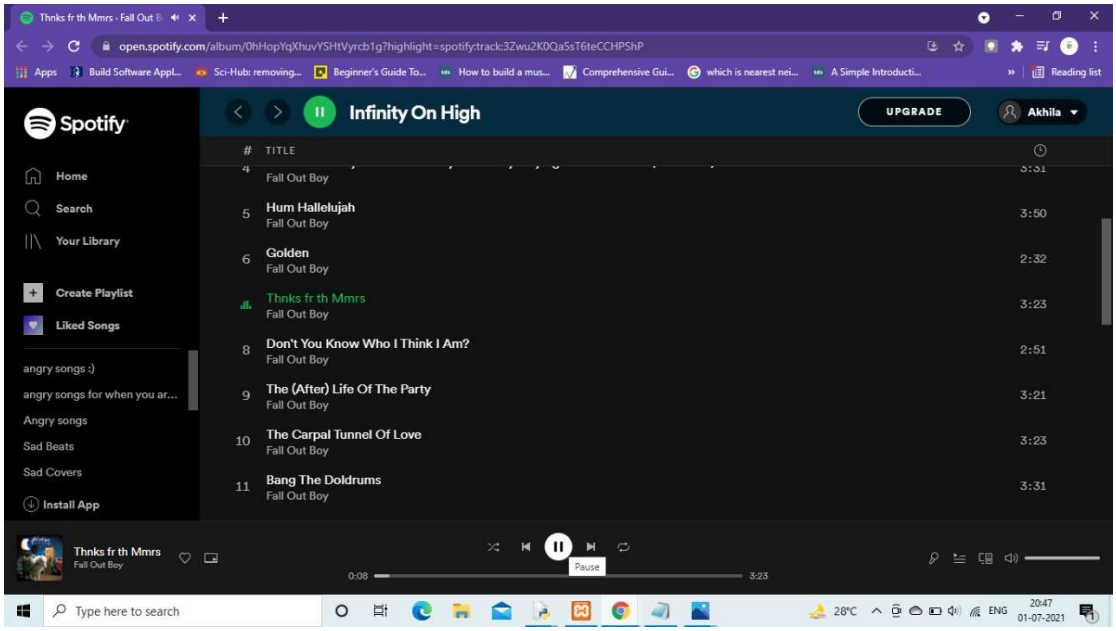

Fig 7.13 Angry songs playing

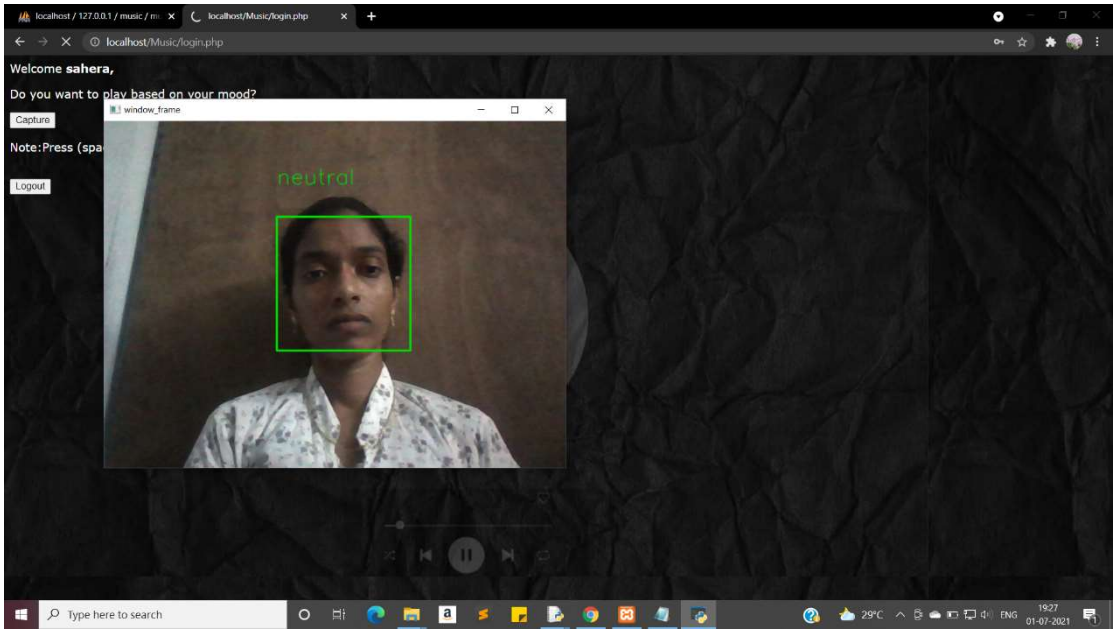

Fig.7.14 Neutral Face

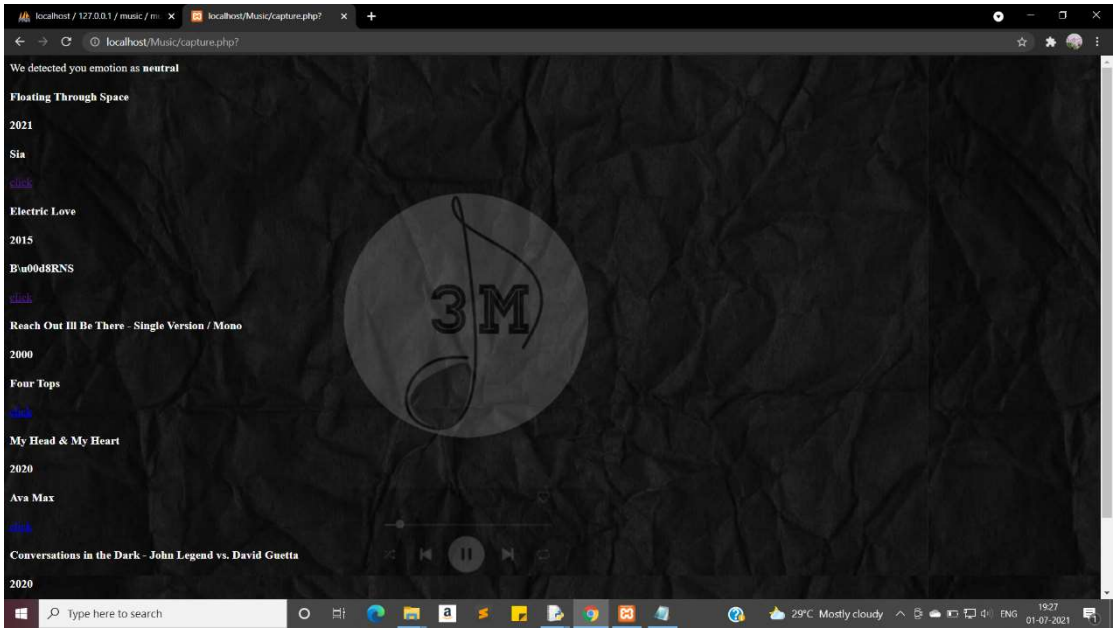

Fig.7.15 Neutral songs

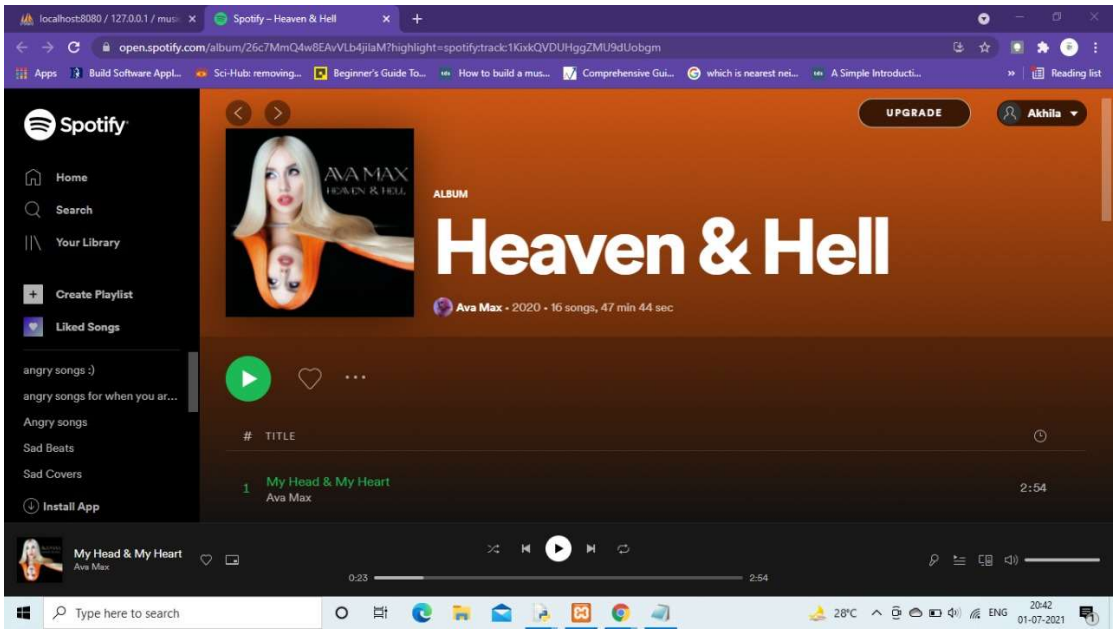

Fig 7.16 Neutral songs

CHAPTER 8

TESTING

8.1 TESTING TECHNOLOGIES

System Testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a login interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Testing Objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.
- The testes are inadequate to detect possibility present errors.
- The software more or less confirms to the quality reliable standards.

Levels of Testing

In order to uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below….

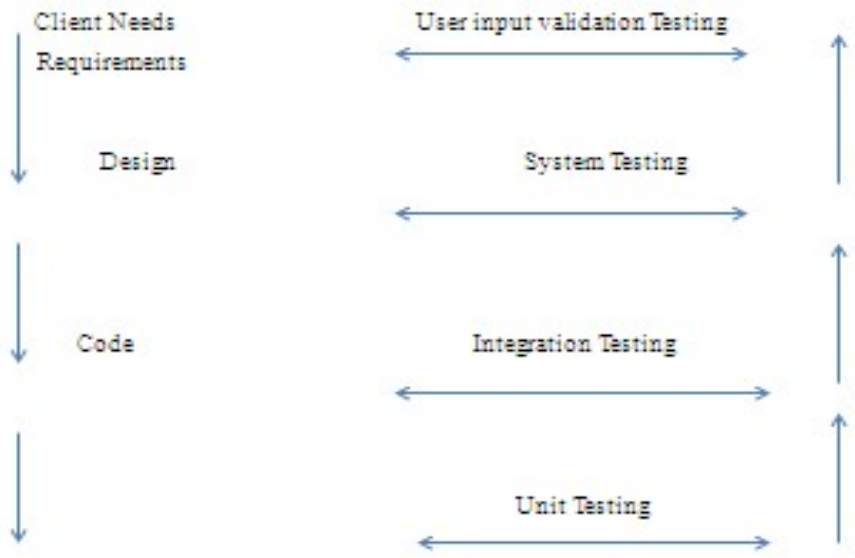

Fig 8.1 Levels of Testing

Types of Testing

- Unit testing.
- System Testing.
- User Input Validation Testing.

Unit Testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone

cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

System Testing

System testing is the first level in which **the complete application is tested as a whole**. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven't played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

User Input Validation Testing

The User input must be validated to confirm to expectedvalues.The fields should also not be empty.

8.2 TEST CASES

Testcase 1: Registration

The user needs to sign up inorder to access use this sytem. For signing up, the user has to fill all the details required.The details are Username,Pass word,mail id. Click on submit option after entering all the details. The output is a toast message as "Inserted succesfully". The user is again redirected to home page and ne needs to login there with his username and password.

Table 8.1 Testcase 1

1	**Test case ID**	**User Registration**
2	**Precondition**	Fill all the fields in Signup page.
3	**Description**	Expert has to enter details.
4	**Test Steps**	• Enter name • Enter Password • Confirm Password • Enter MailId • Click Submit
5	**Expected output**	If details entered into database it has to give the status “Inserted Successfully”.
6	**Actual output**	Toast with Inserted Successfully.
7	**Status**	Pass

Test case 2: Login Success

To start using the application, the user has to login. The login details sholud be authenticated i.e the user has to sign up before logging in. The details required for login are username and password. If the details are correct the user is redirected to user choice page.

Table 8.2 Testcase 2

1	**Test cse ID**	**Login**
2	**Precondition**	Enter login details
3	**Description**	If login details are authenticated respective pages are displayed.
4	**TestSteps**	• Enter username • Enter password • Click login
5	**Expected o/p**	Navigation to the home page of user.
6	**Actual o/p**	Navigated page is displayed.
7	**Status**	Pass

Test case 3: Login Failure

To start using the system, the user has to login. The login details sholud be authenticated i.e the user has to sign up before logging in. The details required for login are username and password. If the details are incorrect a toast message as "Invalid details" is displayed. The user is redirected to home page to login again.

Table 8.3 Testcase 3

1	**Test cse ID**	**Login Failure**
2	**Precondition**	Enter wrong login details
3	**Description**	If invalid ID or PWD error message is displayed.
4	**Test Steps**	• Enter userid • Enter password • Click login
5	**Expected o/p**	“Invalid details“ message is displayed.
6	**Actual o/p**	Toast with "invalid details" is displayed.
7	**Status**	Pass

Test case 4: Choosing capture option

After successful login, few options are given for user. Choosing capture button when user choose capture option, webcam gets activated to capture user face. Emotion detected from user's facial expressions is shown. The user has to click button "p" to continue, which as a result gives list of songs related to mood detected.

Table 8.4 Test Case 4

1	**Test case ID**	**Choose capture option**
2	**Precondition**	Choose Capture Button on user page
3	**Description**	Webcam should open up to capture image and mood should be detected.
4	**Test Steps**	Click on capture button.
5	**Expected output**	Mood of person is detected and list of songs are displayed.
6	**Actual output**	List of songs based on mood are displayed.
7	**Status**	Pass

Test case 5: Choose NO option

After successful login, few options are given for user. Choosing NO option, when user chooses no option few default songs are displayed.By default the mood os songs is happy.

Table 8.5 Test Case 5

1	**Test case ID**	**Choose No option**
2	**Precondition**	Choose No Button on user page
3	**Description**	Songs without detecting mood are displayed.
4	**Test Steps**	Click on No button.
5	**Expected output**	Happy songs are displayed by default.
6	**Actual output**	List of songs are displayed.
7	**Status**	Pass

Test case 6: Choosing Logout option

When the user no longer wants to use the system, he can simply logout by clicking logout option. After clicking logout the user is redirected to home page.

Table 8.6 Test Case 6

1	**Test case ID**	**Choose Logout option**
2	**Precondition**	Choose Logout Button on user page
3	**Description**	User gets logged out.
4	**Test Steps**	Click on Logout button.
5	**Expected output**	User is logged out and home page is displayed.
6	**Actual output**	Home page is displayed.
7	**Status**	Pass

CHAPTER 9

CONCLUSION

A simple system is proposed here for the music recommendation using face emotion recognition. It suggests music by extracting different facial emotion of a person: Happy, anger, surprise, neutral. There is a degree for further upgrades and enhancements. Progressively effective approaches to incorporate different highlights and functionalities should, in any case, be investigated due to the lopsided nature of each element set. It is additionally seen that to improve the exactness of the arrangement framework the informational collection used to construct the grouping model could be expanded further.

CHAPTER 10

SCOPE FOR FUTURE DEVELOPMENT

This project entitled MUSIC RECOMMENDATION APPLICATION BASED ON FACIAL EXPRESSIONS has been developed in such a manner, which helps for future development. In future, we want to recommend songs to user based on his interests i.e., his most played artists, most listened genre etc. We want to store create our own database for storing songs. The project is flexible to adapt the changes efficiently without affecting the present system.

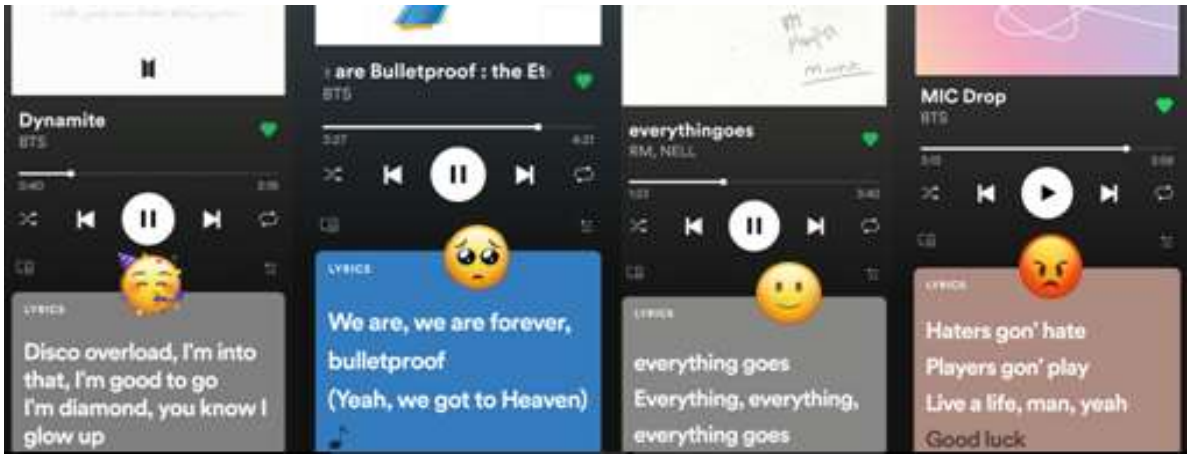

Fig 10.1 Our vision on song recommendation based on user's favourite artists and mood

BIBLIOGRAPHY

[1] Arohi Gupta, “Emotion Detection : A machine learning project”, 2019, https://towardsdatascience.com/emotion-detection-a-machine-learning-project-f7431f652b1f

[2]ArunAlvappillai, Peter Neal Barrina, “Face Recognition using Machine Learning” http://noiselab.ucsd.edu/ECE285/FinalProjects/Group7.pdf

[3]So-Hyun Park, Sun-Young Ihm, Wu-In Jang, Aziz Nasridinov, and Young-Ho Park, “A Music Recommendation Method with Emotion Recognition Using Ranked Attributes", 2020 https://sci-hub.se/https://doi.org/10.1007/978-3-662-45402-2_151

[4]Mikhail Rumiantcev, Oleksiy Khriyenko, “Emotion based Music Recommendation System”, 2020 https://fruct.org/publications/acm26/files/Rum.pdf

[5] Iyer, A.V., Pasad, V., Sankhe, S. R., Prajapati, K., “Emotion based mood enhancing music recommendation”, 2017 https://sci-hub.se/10.1109/RTEICT.2017.8256863

[6] Mustamin Anggo1 and La Arapu, “Face Recognition using Fisher Face Method” https://iopscience.iop.org/article/10.1088/1742-6596/1028/1/012119/pdf

[7] Ameya Badve1 , Atharva Deshpande2 , Hitesh Kadu3 , Prof. Mrs. B. Mahalakshmi4 . 1,2,3,4Computer Science,PCCOEPune,India E http://www.jcreview.com/fulltext/197-1594987978.pdf

[8] Dureha, A. (2014). An Accurate Algorithm for Generating a Music laylist based on Facial Expressions. International Journal of Computer Applications, 100(9). https://pdfs.semanticscholar.org/312b/2566e315dd6e65bd42cfcbe4d919159de8a1.pdf