1. Introduction

Music plays a very important role in human's daily life and in the modern advanced technologies. Usually, the user has to face the task of manually browsing through the playlist of songs to select. Here we are proposing an efficient and accurate model, that would generate a playlist based on current emotional state. Existing methods for automating the playlist generation process are computationally slow, less accurate and sometimes even require use of additional hardware like EEG or sensors. Speech is the most ancient and natural way of expressing feelings, emotions and mood and its and its processing requires high computational, time, and cost. This proposed system based on real-time extraction of facial expressions features to classify into a specific emotion that will generate a playlist automatically such that the computation cost is relatively low.

The efficiency of the applicatin depends on the algorithm used, the number of songs in the database in the application and the device on which the application is running

1.1 Problem Statement

Currently, there are no dedicated applications to suggest songs based on emotion of music listeners. There are also very few applications that focus on the user preferences and recommendations, and these are not customizable like AllMusic.

Music Listeners have to face so many problems such as :

- Create and Segregating the play-list manually when they have hundereds of songs.
- User have to select song manually every time based on interest and mood.
- User have to re-organize and play music when playstyle varies.
- Currently, there are no applications that allows users to play songs on-the-go without selecting songs manually or from a play-list
- There are no application that allow user to play song and generate play-list.

1.2 Objective

The Emotion Based Music player (MOODY) requires the user to have a profile to access the application. The user needs to grant permissions for the application to access the device's camera and media. The application allows users to upload songs and give feedback on the song. Emotion-Based Music Player saves the user profile on the device and keeps the profile logged-in until user logs out of the device manually. As soon as the user opens the application, the device's camera opens and begin capturing images. The system will determine emotions and create play-lists for the user based on emotion captured. The application also allows user's to easily customize the playlists.

It recommends songs for the user that may fit their current emotion, helping the user automate the initial song selection. The recommendations are based on the previous information about the user's preferences and usage.

1.3 Scope

Moody is a useful application for music listener with a smartphones and having internet connection. This application is accessible by anyone. This application is designed to meet some needs,

- Create an account (Login , Signup)
- Adding Songs
- Listen songs
- Remove songs
- Adding song
- Capture Emotion using camera for mood detection
- Personalized playlist

Platform Specification

1.4.1 Software Requirement

- Firebase
- Realtime DB
- Android Studio

1.4.2 Implementation Language

- Xml
- Java
- Cotlin

2. Feasibility Study

The feasibility study is a major factor which contributes to the analysis and development of the system. The decision of the system analyst whether to design a particular system or not depends on its feasibility study.

Study of requirement analysis is done through different feasibility study. Feasibility study is undertaken whenever a possibility of probability of improving the existing system or designing new system. Feasibility study helps to meet user requirements.

It enables us to determine the potential of existing system and improving it. It helps to develop a technically and economically feasible system. It helps to know what should be embedded in the system. It also helps to develop a cost-effective system. We can make better utilization of available resources.

The project concept is feasible because of the following:

3.1 Technical Feasibility

3.2 Economical Feasibility

3.3 Operational Feasibility

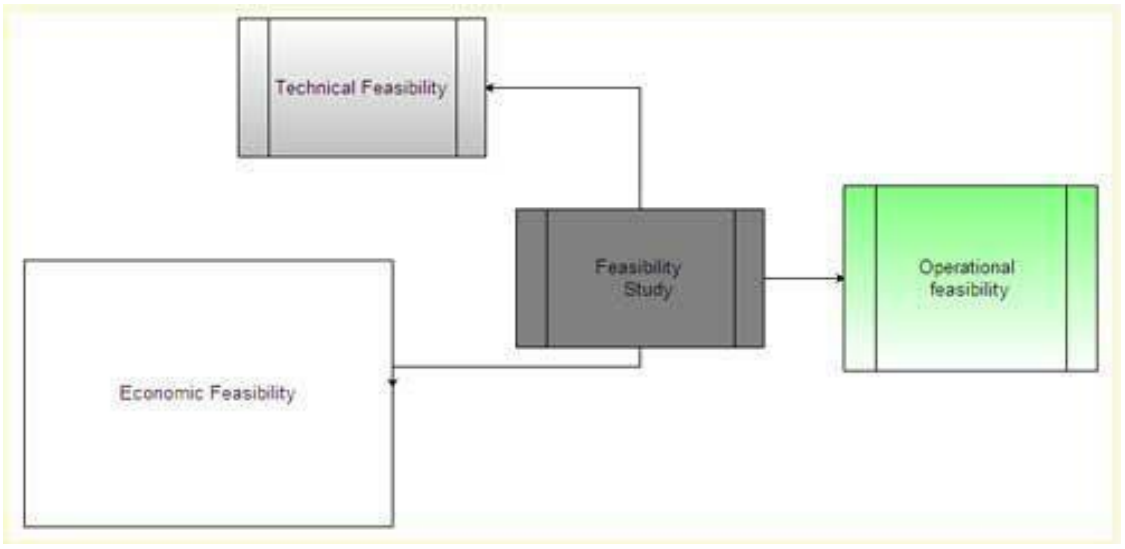


Fig : 2.1 Feasibility Study

2.1 Technical Feasibility:

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyzes the technical skills and capabilities of the software development team members.
- Determines whether the relevant technology is stable and established.
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

2.2 Economical Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
- Cost of hardware, software, development team, and training.

2.3 Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

It is a measure of how well the system will work in the organization. It is also a measure of how people feel about the system/project. In this project the user feels that the system is very user friendly. This project developed is worth and solutions to the problem will work successfully.

- Determines whether the problems anticipated in user requirements are of high priority
- Determines whether the solution suggested by the software development team is acceptable
- Analyzes whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

3. Literature Survey

Various techniques and approaches have been proposed and developed to classify human emotional state of behavior. The proposed approaches have focused only on the some of the basic emotions. For the purpose of feature recognition, facial features have been categorized into two major categories such as Appearance-based feature extraction and geometric based feature extraction by zheng . Geometric based feature extraction technique considered only the shape or major prominent points of some important facial features such as mouth and eyes. There is another scheme that is automatically segment an input image, and to recognize facial emotion using detection of color based facial feature map and classification of emotion with simple curve and distance measure is proposed and implemented. In other scheme there is automatic method for real time emotion recognition using facial expression using a new anthropometric model for facial feature extraction.

3.1 Work done by others

1.Anagha S. Dhavalikar and Dr. R.K.Kulkarni Proposed Automatic Facial Expression recognition system. In This system there are three phase :-

1.Face detection 2. Feature Extraction and 3.Expression recognition.

The First Phase Face Detection are done by YCbCr Color model, lighting compensation for getting face and morphological operations for retaining required face i.e eyes and mouth of the face. This System is also used AAM i.e Active Appearance Model Method for facial feature extraction In this method the point on the face like eye, eyebrows and mouth are located and it create a data file which gives information about model points detected and detect the face the an expression are given as input AAM Model changes according to expression.

2.Yong-Hwan Lee ,Woori Han and Youngseop Kim proposed system based on Bezier curve fitting. This system used two step for facial expression and emotion first one is detection and analysis of facial area from input original image and next phase is verification of facial emotion of characteristics feature in the region of interest. The first phase for face detection it uses color still image based on skin color pixel by initialized spatial filtering ,based on result of lighting compassion then to estimate face position and facial location of eye and mouth it used feature map

After extracting region of interest this system extract points of the feature map to apply Bezier curve on eye and mouth for understanding of emotion this system uses training and measuring the difference of Hausdorff distance With Bezier curve between entered face image and image from database.

3. Arto Lehtiniemi and Jukka Holm proposed system based on animated mood picture in music recommendation. on this system the user interact with a collection of images to receive music recommendation with respect to genre of picture. This music recommendation system is developed by Nokia researched center. This system uses textual meta tags for describing the genre and audio signal processing .

Currently, there are no dedicated applications to suggest songs based on emotion of music listeners. There are also very few applications that focus on the user preferences and recommendations, and these are not customizable, like AllMusic. Other applications suggests predefined (not user-specific) song play-lists. Application like :-

•**Saavan and Spotify** – These application gives good user accessibility featuresto play songs and recommends user with other songs of similar genre.

•**Moodfuse** - In this application , user should manually enter mood and genre that wants to be heard and moodfuse recommends the songs-list.

•**Stereomood** - User should select his mood manually by selecting the moods from the list and the application [17] plays music from YouTube.

•**Musicovery** - This application [3] has High quality songs and comprehensive music recommendations. It also suggest predefined play-list for the user.

All of these applications focus on general categorization rather than specificity to every user.

3.2 Benefits

There are several benifits of "SMART MUSIC PLAYER INTEGRATING FACIAL EMOTION RECOGNITION (MOODY)"

Accuracy of our project is 96%.

User dosen't want to select song manually.

A new feature is also added i.e "age detection" so that song is classified on the basis of emotion and age of the user.

Extreamly fast feature computation and efficient feature selection.

3.3 Proposed Solution

Numerous approaches have been designed to extract facial features and audio features from an audio signal and very few of the systems designed have the capability to generate an emotion based music playlist using human emotions and the existing designs of the systems are capable to generate an automated playlist using an additional hardware like Sensors or EEG systems thereby increasing the cost of the design proposed. Some of the drawbacks of the existing system are as follows

- Existing systems are very complex in terms of time and memory requirements for extracting facial features in real time.
- Based on the current emotional state and behavior of a user, existing systems possess a lesser accuracy in generation of a playlist.
- Some existing systems tend to employ the use of human speech or sometimes even the use of additional hardware for generation of an automated playlist, thereby increasing the total cost incurred.

The proposed system tries to provide an interactive way for the user to carry out the task of creating a playlist. the working is based on different mechanisms carrying out their function in a pre-defined order to get the desired output. The working can be stated as follows:

- The proposed System works by first providing a simple enough interface which prompts the user to scan the memory for audio files when the application is opened.
- Then after the files are detected, they are scanned for audio features and thesefeatures are extracted.
- Then the extracted feature values are subjected to classification according to the parameters provided.
- These parameters include a limited set of genre types based on which the audio feature values will be processed.

- After this, the songs are segregated into different playlists based on the feature extraction process. Hence lists of similar sounding songs or songs belonging to similar genres are generated.
- In the next step, the user camera is invoked with proper permissions and a real time graphical input(image)is provided to the system.
- The system first checks for the presence of a face in the input using the face detection process , then classifies the input and generates an output which is an emotion(mood) based on the expression extracted from the real time graphical input.
- After this the classified expression acts as an input and is used to select an appropriate playlist from the initially generated playlists and the songs from the playlists are played.

3.4 Technology Used

Frond End

- Xml

Back End

- Java
- Coltin
- Firebase

Technology

- Machine Learning
- Tensorflow
- Deep Learning - CNN

4. Requirement Analysis and Design

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore requirements analysis means to analyze, document, validate and manage software or system requirements. High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

4.1 Requirement Analysis

Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish.

Functional Requirements

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

The dataset train by support vector classifier.

Machine learns support vector classification using support vector machine.

Learn and identify image capture by web cam.

Non Functional Requirements

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

•Reliability

•Reusability

•Maintainability

•Simplicity

•Portability

•Extensibility

•Resource Utilization

4.1.1 Software Requirement Specification

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. Software requirement specification (SRS) is a technical specification of requirements for the software product. SRS represents an overview of products, features and summaries the processing environments for development operation and maintenance of the product. The goal of the requirement specification phase is to produce the software specification document also called requirement document.

Requirement Specification

This requirement specification must have the system properties. Conceptually every SRS should have the components:

- Functionality
- Performance
- Design constraints imposed on an implementation
- External interface

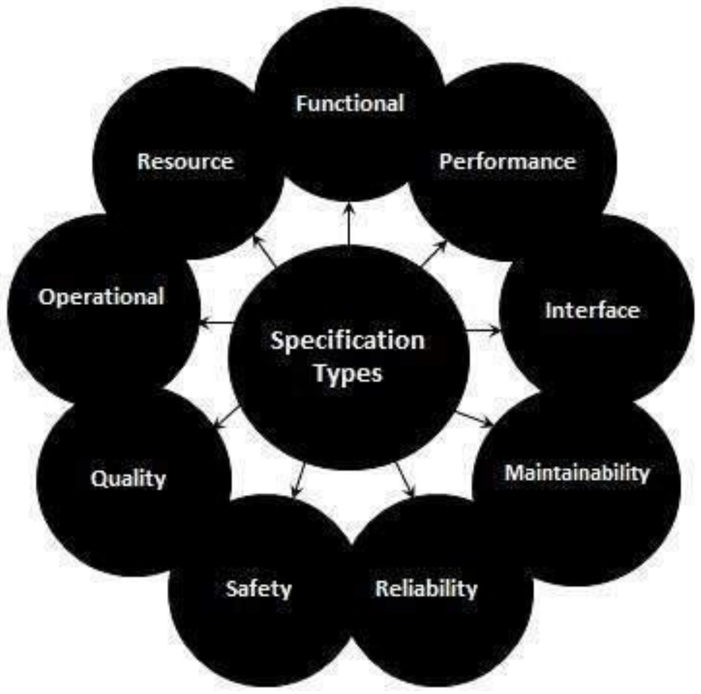


Fig : 4.1 Requirement Specification

4.1.1.1 Use Case Model

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

A use case (or set of use cases) has these characteristics:

- Organizes functional requirements
- Models the goals of system/actor (user) interactions4

- Records paths (called *scenarios*) from trigger events to goals
- Describes one main flow of events (also called a basic course of action), and possibly other ones, called *exceptional* flows of events (also called alternate courses of action)
- Is multi-level, so that one use case can use the functionality of another one.

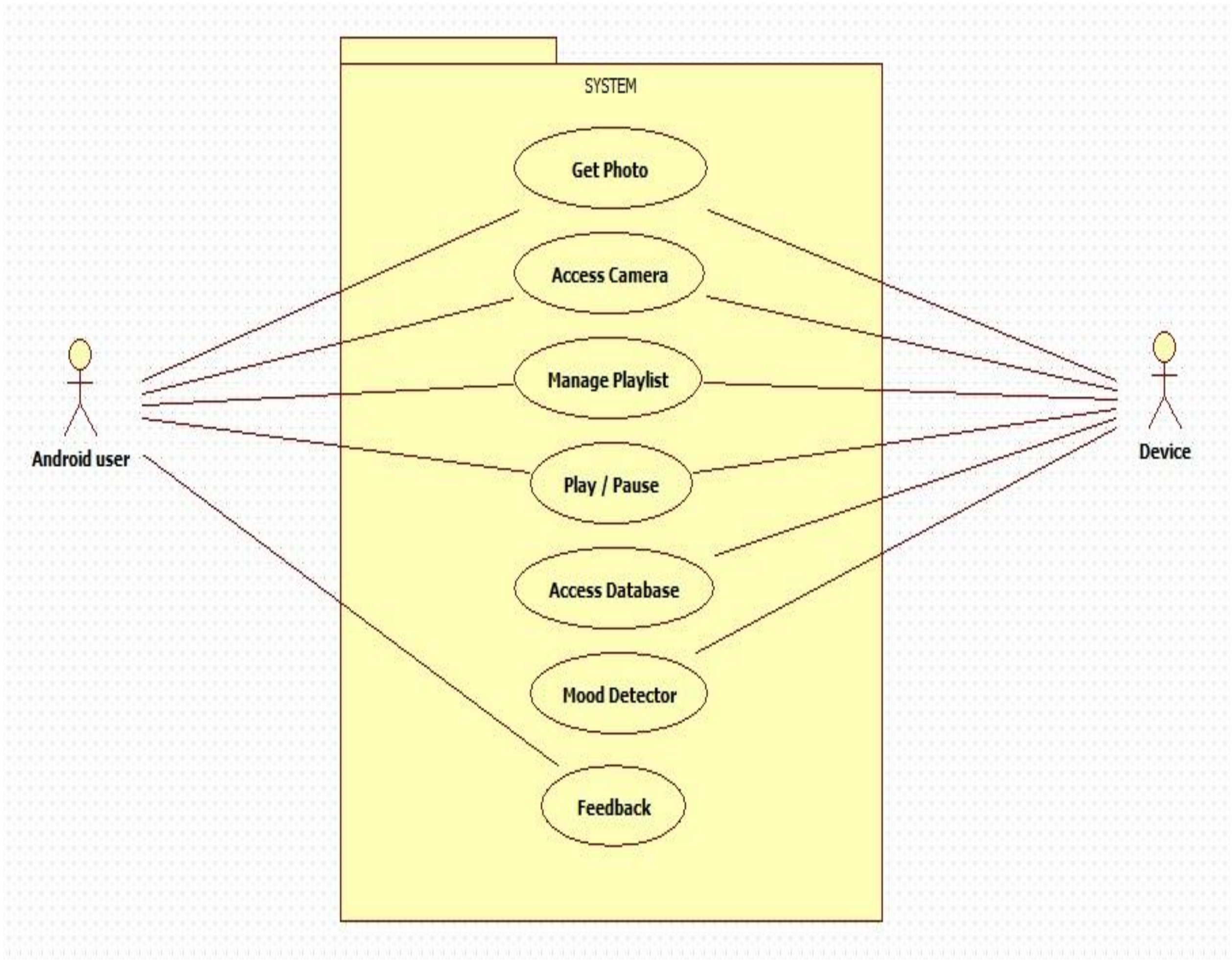


Fig 4.2 : Use Case Diagram

Use cases define interactions between external actors and the system to attain particular goals. There are three basic elements that make up a use case:

- Actors: Actors are the type of users that interact with the system.
- System: Use cases capture functional requirements that specify the intended behavior of the system.
- Goals: Use cases are typically initiated by a user to fulfill goals describing the activities and variants involved in attaining the goal.

4.1.2 Conceptual Level Activity diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc Purpose of Activity Diagrams:

The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

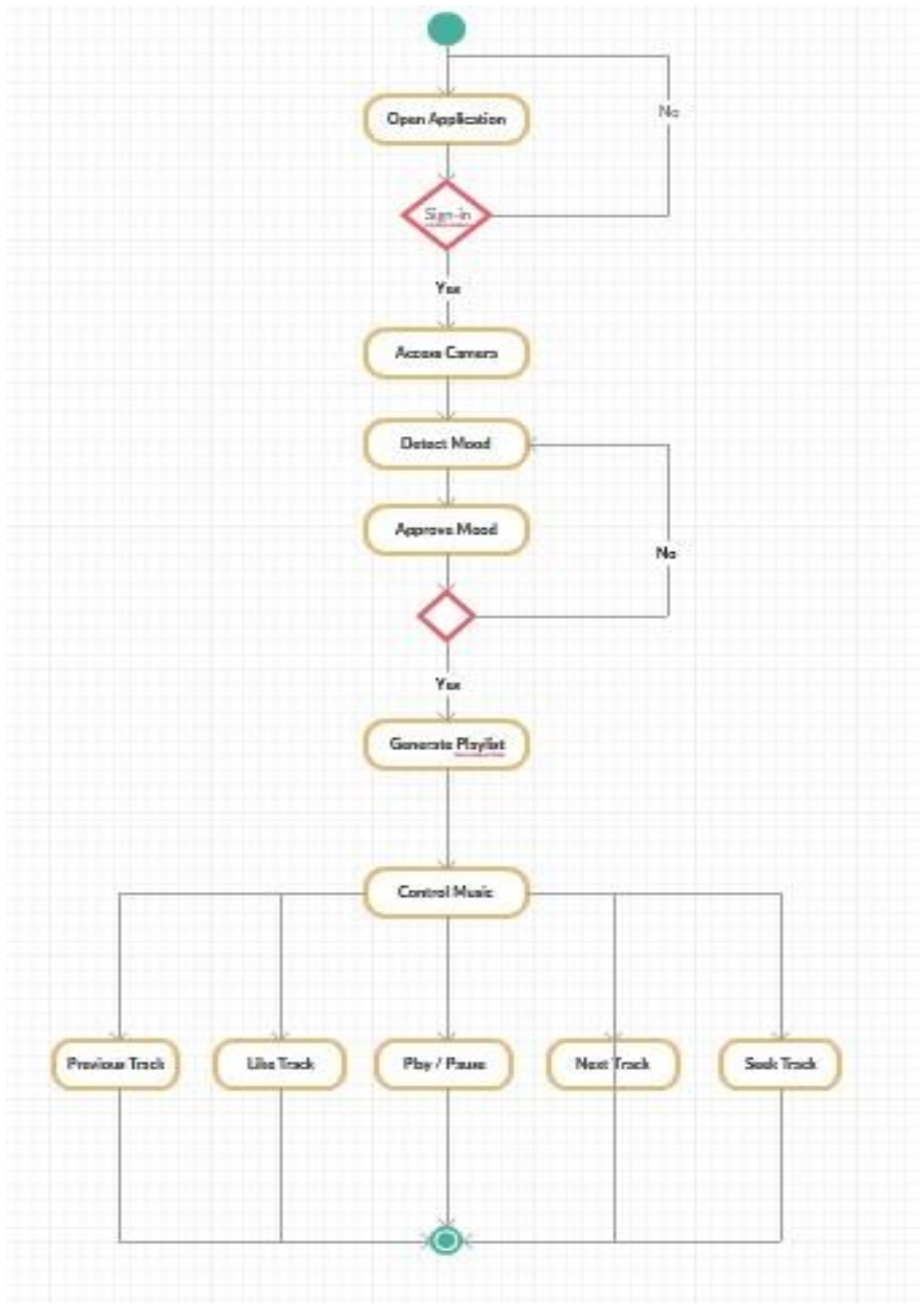


Fig 4.3 Activity Diagram

4.2 Design

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore requirements analysis means to analyze, document, validate and manage software or system requirements. High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

4.2.1 Design Concept

The purpose of the design phase is to plan a solution of the problem specified by the requirement of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us towards how to satisfy the needs. The design of system is the most critical factor affecting the quality of the software and has major impact on testing and maintenance. The output of this phase is the design document.

4.2.2 Design Technique

The design process involves developing a conceptual view of the system, establishing system structure, identifying data streams and data stores, decomposing high level functions into sub functions, establishing relationships and interconnections among components, developing concrete data representations, and specifying algorithmic details. Software design is a creative activity.

Structured analysis and design technique (SADT) is a diagrammatic notation designed specifically to help people describe and understand systems. It offers building blocks to represent entities and activities, and a variety of arrows to relate boxes. These boxes and arrows have an associated informal semantics. SADT can be used as a functional analysis tool of a given process, using successive levels of details. The SADT method not only allows one to define user needs for IT developments, which is often used in the industrial Information Systems, but also to explain and present an activity's manufacturing processes and procedures.

System Design

System design provides the understandings and procedural details necessary for implementing the system recommended in the system study. Emphasis is on the translating the performance requirements into design specifications. The design phase is a transition from a user-oriented document (System proposal) to a document oriented to the programmers or database personnel.

System Design goes through two phases of development:

- Logical design
- Physical Design

A data flow diagram shows the logical flow of the system. For a system it describes the input (source), output (destination), database (data stores) and procedures (data flows) all in a format that meets the user's requirement. When analysis prepares the logical system design, they specify the user needs at a level of detail that virtually determines the information flow into an out of the system and the required data resources. The logical design also specifies input forms and screen layouts.

The activities following logical design are the procedure followed in the physical design e.g., producing programs, software, file and a working system.

The logical design of an information system is analogous to an engineering blue print of an automobile. It shows the major features and how they are related to one another. The detailed specification for the new system was drawn on the basis of user's requirement data. The outputs inputs and databases are designed in this phase. Output design is one of the most important features of the information system. When the output is not of good quality the user will be averse to use the newly designed system and may not use the system. There are many types of output, all of which can be either highly useful or can be critical to the users, depending on the manner and degree to which they are used. Outputs from computer system are required primarily to communicate the results of processing to users, They are also used to provide a permanent hard copy of these results for later consultation. Various types of outputs required can be listed as below:

- External Outputs, whose destination is outside the organization
- Internal outputs, whose destination is with the organization
- Operational outputs, whose use is purely with in the computer department e.g., program-listing etc.
- Interactive outputs, which involve the user is communicating directly with the computer, it is particularly important to consider human factor when designing computer outputs.

End user must find outputs easy to use and useful to their jobs, without quality output, user may find the entire system unnecessary and avoid using it. The term "Output" in any information system may apply to either printer or displayed information. During the designing the output for this system, it was taken into consideration, whether the information to be presented in the form of query of report or to create documents etc.

Other important factors that were taken into consideration are:

- The End user, who will use the output.
- The actual usage of the planned information
- The information that is necessary for presentation when and how often output and their format is needed. While designing output for project based Attendance Compilation System, the following aspects of outputs designing were taken into consideration.

Detailed Design

During detailed design the internal logic of each of the modules specified in the system design is decided. In system design the focus is on identifying the modules, where as during detailed design the focus is on designing the logic for each of modules. In other words, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in the software. During this phase further details of the data structures and algorithmic design of each of the module is usually specified in a high – level design description language, which is independent of the target language in which the software will eventually be implemented. Thus a design methodology is a systematic approach to creating a design by application of a set of techniques and guidelines.

4.2.3. Modeling

Constructing and manipulating abstract (mathematical and/or graphical) representations of economic, engineering, manufacturing, social, and other types of situations and natural phenomenon, simulated with the help of a computer system. Also called computer simulation

Modeling is one of the basic methods in empirical sciences. Generally speaking, it consists of the gradual construction of a cognitively useful – though simplified and idealized – image of described phenomena. As this image often takes the form of an abstract formal description – for example, a system of equations, or a set of logical formulas – modeling re-lies significantly on formal sciences such as mathematics, logic, or computer

science.

4.2.3.1. Detailed Class Diagram

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram. The purpose of the class diagram is to model the static view of an application.

The following points should be remembered while drawing a class diagram:

- The name of the class diagram should be meaningful to describe the aspect of the system.

- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified.
- For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

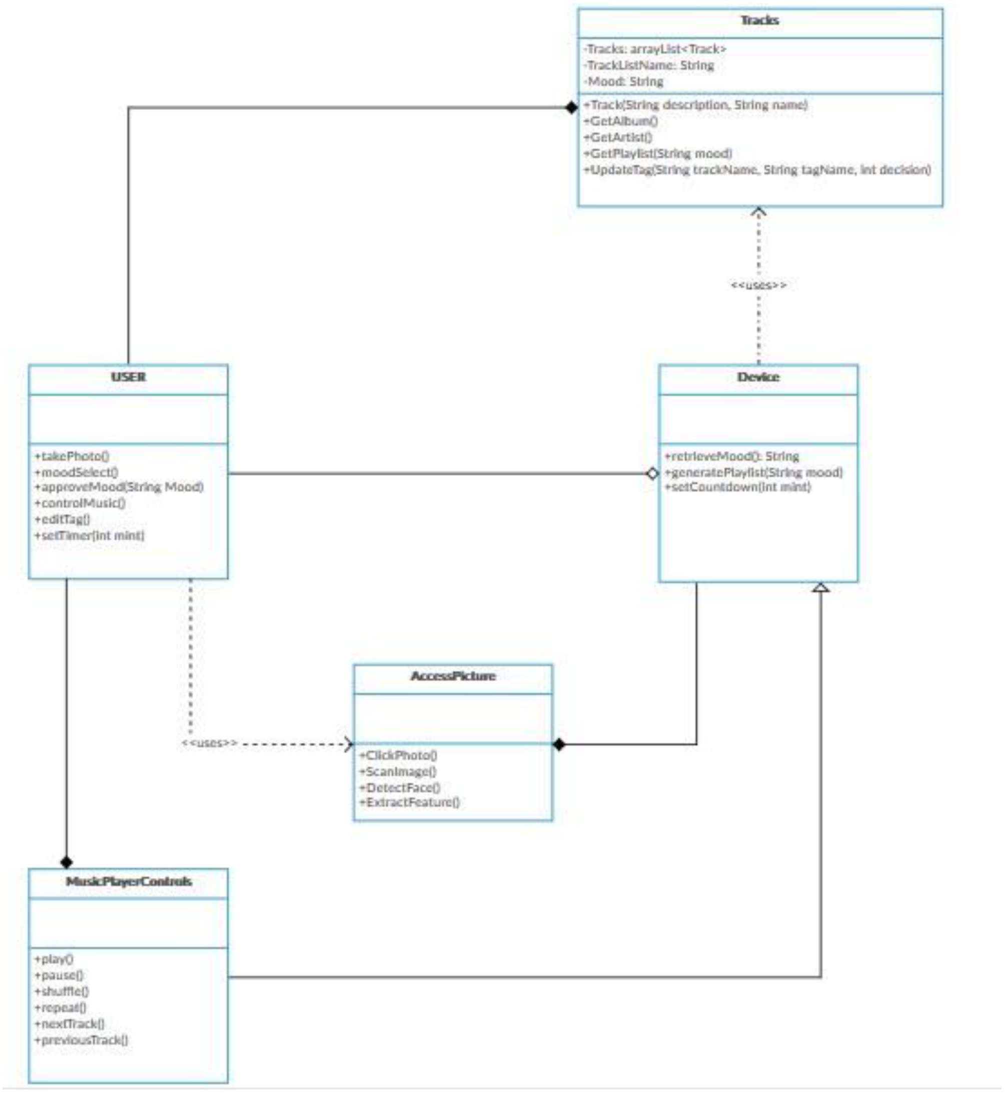


Fig 4.4 Class Diagram

4.2.3.2 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

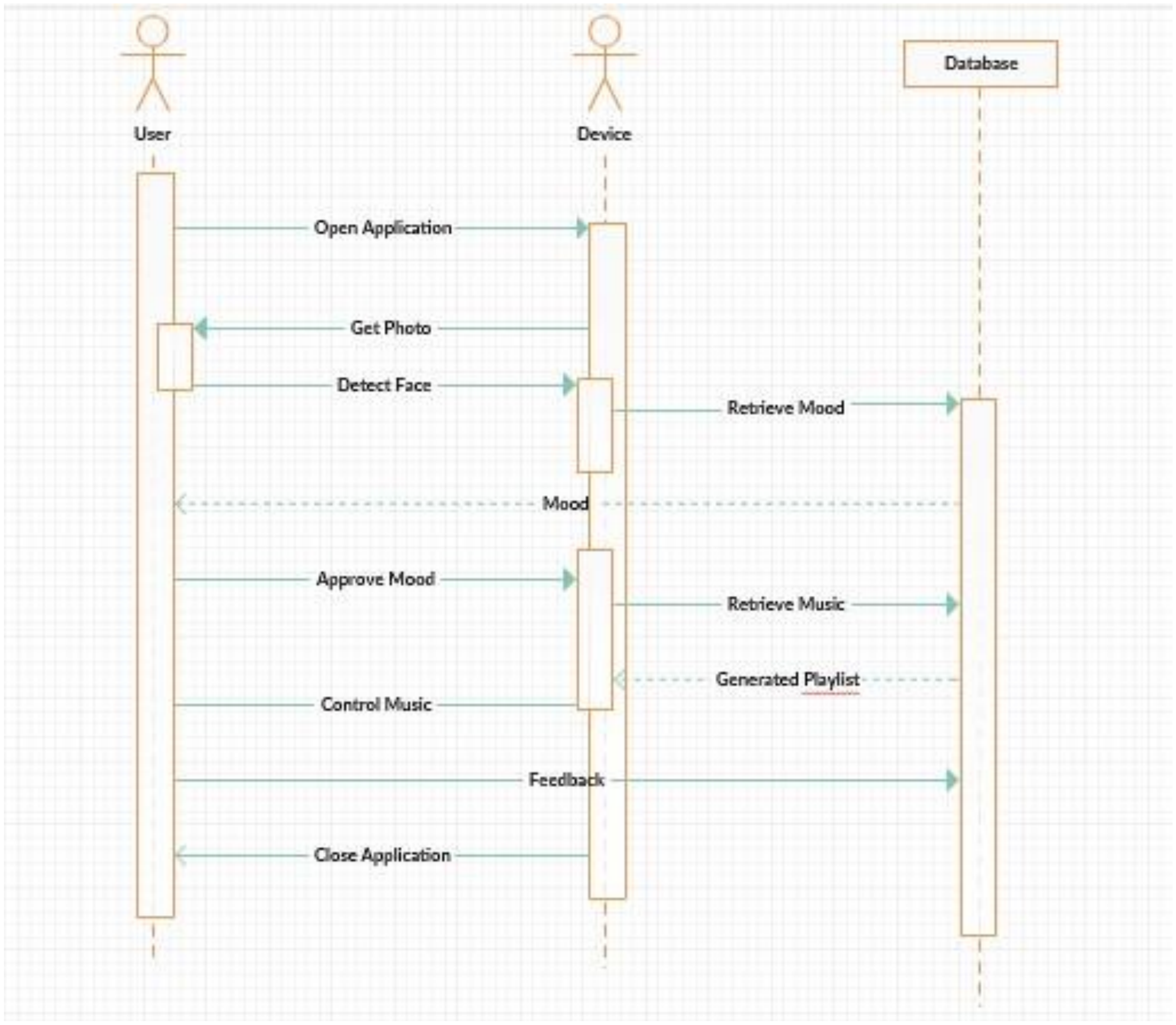


Fig 4.5 : Sequential Diagram

4.2.3.3 Activity Diagram

Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system.

Activity diagrams are used to model workflow or business processes and internal operation. The figure shows the work flow the system. The client logs in, gives the requirements, Admin views the requirements, contacts the Dealers and then finally generates the final product.

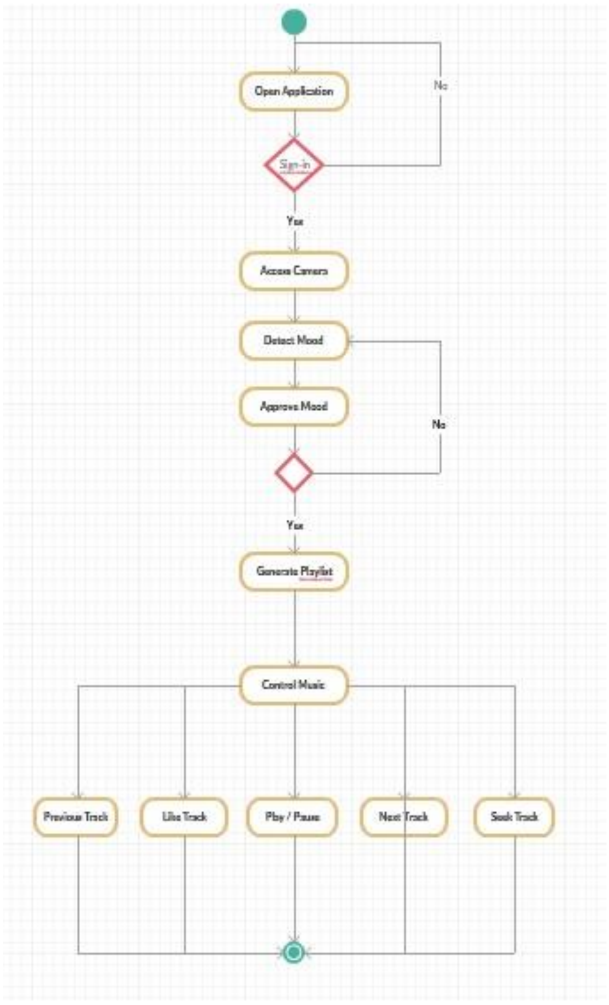


Fig 4.6 Activity Diagram

5. Methodology And Implementation Phase

The methodology is the general research strategy that outlines the way in which research is to be undertaken and, among other things, identifies the methods to be used in it. These methods, described in the methodology, define the means or modes of data collection or, sometimes, how a specific result is to be calculated. Methodology does not define specific methods, even though much attention is given to the nature and kinds of processes to be followed in a particular procedure or to attain an objective.

5.1 Methodology

Methodology is the systematic, theoretical analysis of the methods applied to a field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. Typically, it encompasses concepts such as paradigm, theoretical model, phases and quantitative or qualitative techniques.

A methodology does not set out to provide solutions—it is therefore, not the same as a method. Instead, a methodology offers the theoretical underpinning for understanding which method, set of methods, or best practices can be applied to a specific case, for example, to calculate a specific result.

A paradigm is similar to a methodology in that it is also a *constructive* framework. In theoretical work, the development of paradigms satisfies most or all of the criteria for methodology. An algorithm, like a paradigm, is also a type of constructive framework, meaning that the construction is a logical, rather than a physical, array of connected elements.

5.1.1 Description

INCREMENTAL MODEL

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.

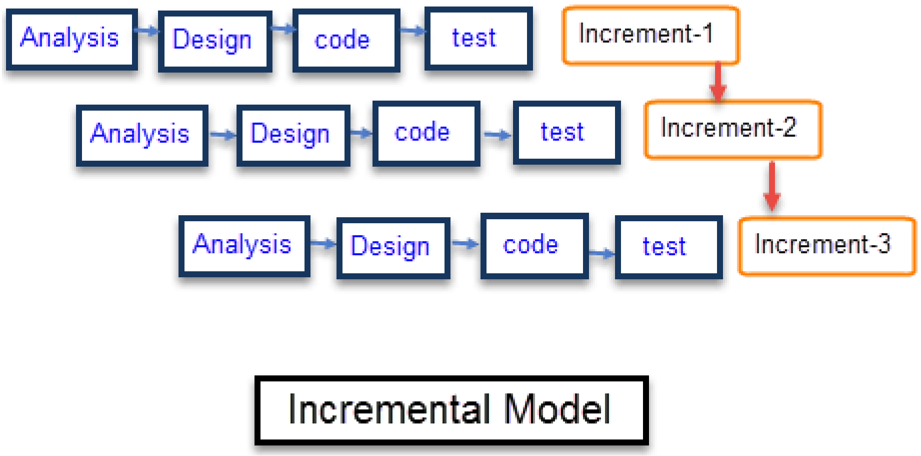


Fig 5.1 Incremental Model

The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements.

Characteristics of an Incremental model includes

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

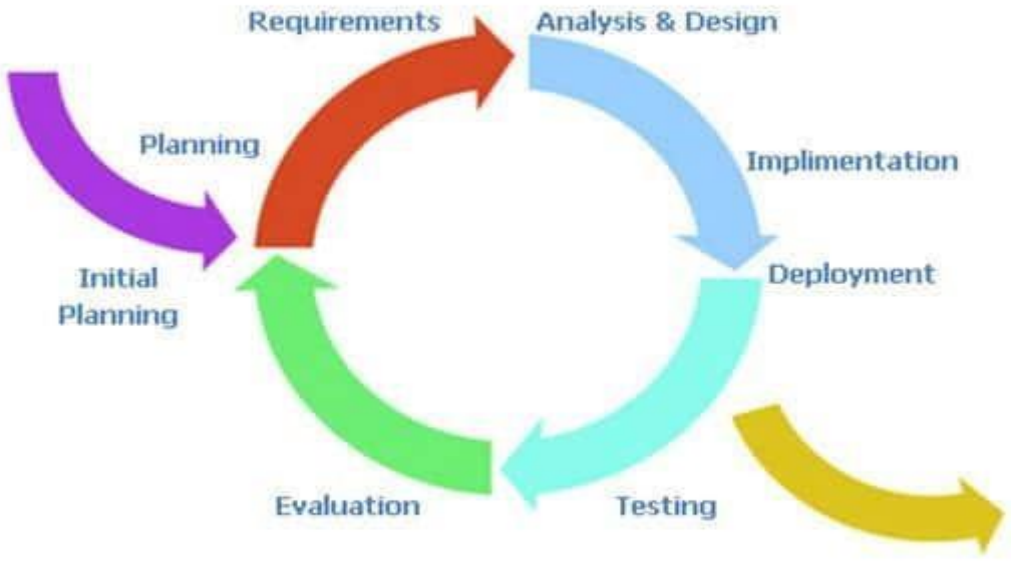


Fig 5.2 Characteristics

When to use Incremental models?

- Requirements of the system are clearly understood • When demand for an early release of a product arises.
- When high-risk features and goals are involved.
- Such methodology is more in use for web application and product based companies.
- When Projects having lengthy developments schedules.
- A new technology is being used.

5.1.2 Advantages And Disadvantages

Advantages of Incremental Model

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

- Each iteration is an easily managed milestone.
- Thought the development stages changes can be done

Disadvantages of Incremental Model

- It requires a good planning designing
- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.
- Well defined module interfaces are required.

5.1.3 Reason For Use

As we know all the requirements of our project and are clearly understood.
We have build our project in different components, Like:- **Component 1.** Sign-up and Log-in.
Component 2. Access camera for mood detection.
Component 3. Play song according to the mood or emotion.
Component 4. Playlist will be automatically generated according to the emotions.

5.2 Implementation Phase

The project takes shape during the implementation phase. This phase involves the construction of the actual project result. Programmers are occupied with encoding, designers are involved in developing graphic material, contractors are building, the actual reorganisation takes place. It is during this phase that the project becomes visible to outsiders, to whom it may appear that the project has just begun. The implementation phase is the doing phase, and it is important to maintain the momentum.

5.2.1. Language Used Characteristics

ANDROID

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below:

- Android OS basic screen provides a beautiful and intuitive user interface.
- Simple and powerful RAD tool for developing native Android applications
- Complete IDE and programming language 100% focused on Android development
- Compiles to native bytecode. No runtime libraries are required. APK files are exactly the same as APK files created with Java / Eclipse
- Performance is similar to applications written with Java
- Object oriented programming language
- Share the code with B4J - a development tool for desktop applications New!
- No need for XML programming
- Rapid debugger - supports quick deployments, hot code swapping and expressions watches New! No other native Android tool supports these features!
- Highly extensible with support for custom Java libraries
- WYSIWYG visual editor for Android. The visual editor supports multiple screens and resolutions
- Basic4android UI Cloud service. Test your layouts on a cloud of real phones and tablets

XML

XML stands for Extensible Markup Language. It is a software and hardware-independent tool for storing and transporting data.

- XML does not use predefined tags. With XML, the author must define both the tags and the document structure.
- XML is extensible. Most XML applications will work as expected even if new data is added (or removed). The way XML is constructed, older version of the application can still work.
- XML simplifies data sharing, data transport, platform changed and data availability.

XML is a W3C recommendation

5.2.2 Coding/Screenshots

5.2.2.1 Moody

As our application name is “Moody” so we have searched Moody in the search bar and the name of our application with is displayed.

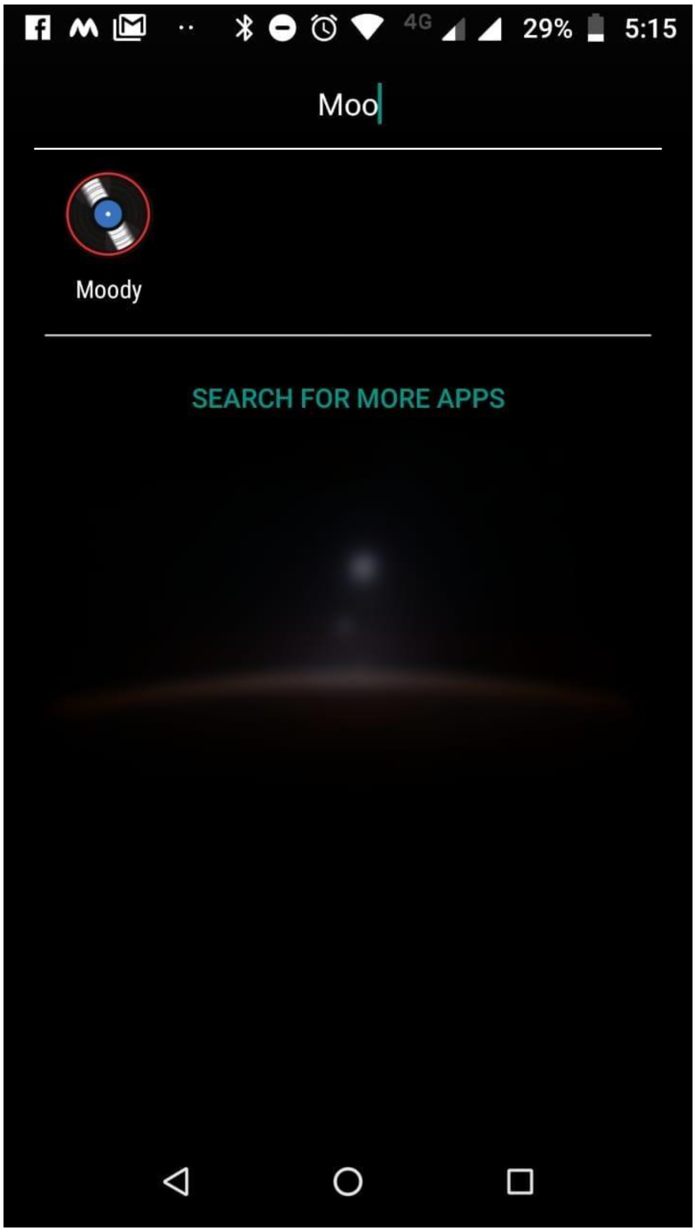


Fig 5.3 Moody

5.2.2.2 Launcher Screen

This is the launcher screen of our application. When you click on our application for open then this launcher sceen will be displayed on the user's screen.

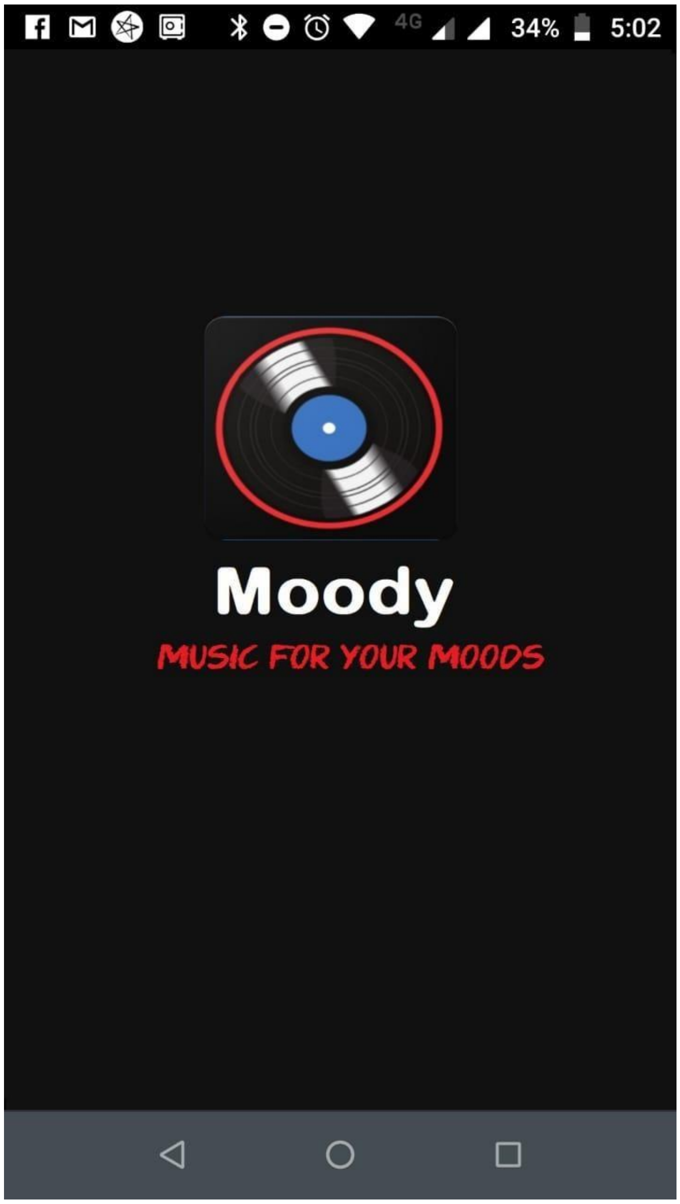


Fig 5.4 Launcher Screen

5.2.2.3 User Login and Sign-up screen

If the user is accessing the application for the first time, they need to register for the application. The user must give valid email address, name and password. If the user forgets their password an email to reset the password is sent to registered email address. All the information is validated and account creation is finished.

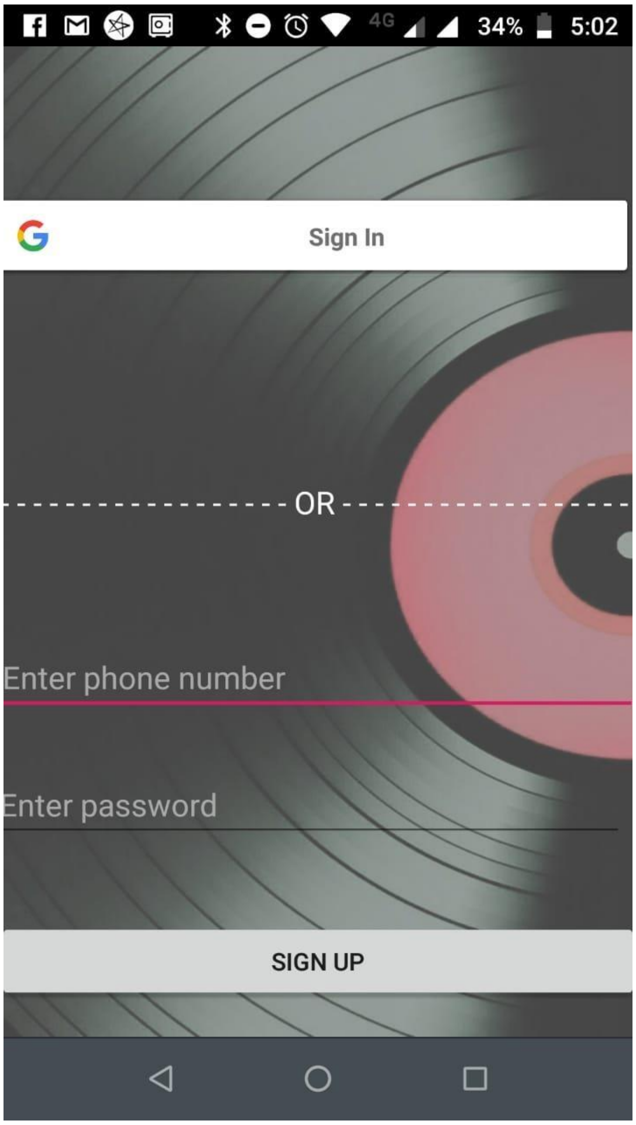
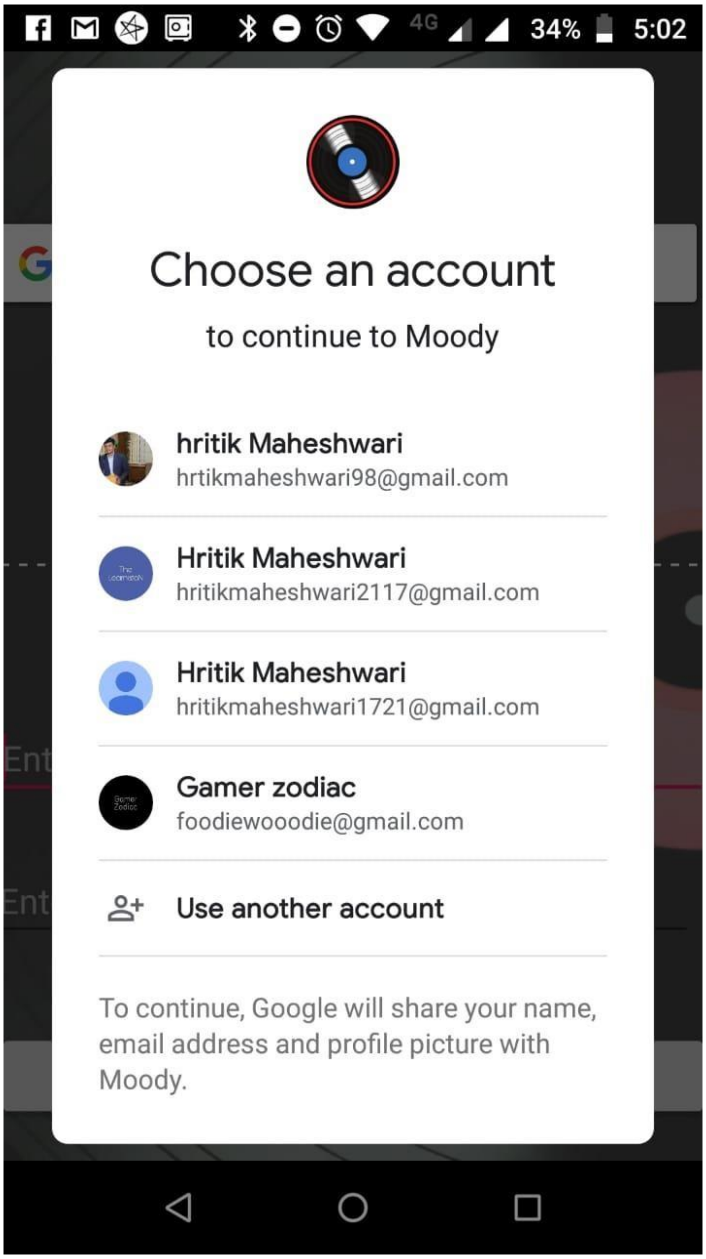


Fig 5.5 User Log-in and Sign-in Page

The user should provide a registered email address and corresponding password to log-in to the application and the in-formation will be saved on cloud. The application saves the user's authentication information as the default. The user can log-out of the application any time


Choose an account

to continue to Moody

hritik Maheshwari
hrtikmaheshwari98@gmail.com

Hritik Maheshwari
hritikmaheshwari2117@gmail.com

Hritik Maheshwari
hritikmaheshwari1721@gmail.com

Gamer zodiac
foodiewooodie@gmail.com

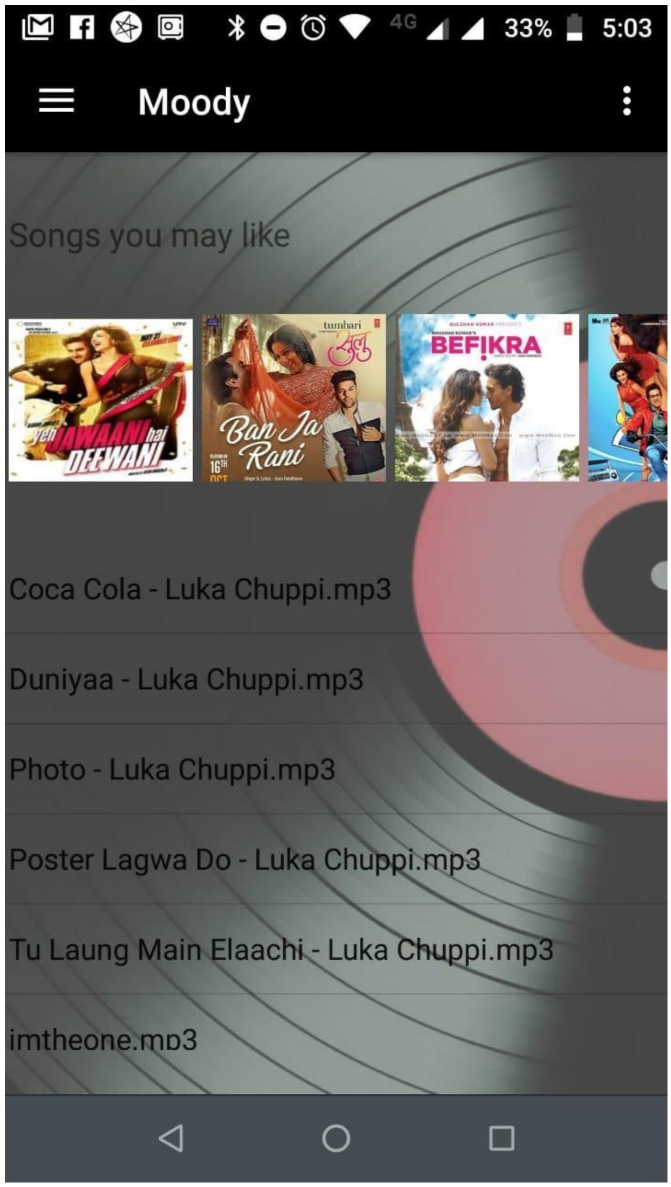
Use another account

To continue, Google will share your name, email address and profile picture with Moody.

Fig : 5.6

5.2.2.4 Main Screen

As soon as user logs-in to the application, the system loads the required components. If the user is logging in for the first time,the user's profile and preferences are created during this phase. After successfully login to the application the user will redirected to the initial page i.e Main Screen.



Songs you may like

Coca Cola - Luka Chuppi.mp3

Duniyaa - Luka Chuppi.mp3

Photo - Luka Chuppi.mp3

Poster Lagwa Do - Luka Chuppi.mp3

Tu Laung Main Elaachi - Luka Chuppi.mp3

imtheone.mp3

Fig : 5.7 Main Screen

5.2.2.5 Emotion Detector

Once the SDK is loaded, the starting detector screen appears and opens device's camera to capture the emotion. If the application is running for the first time, the application asks permissions to access media and camera.

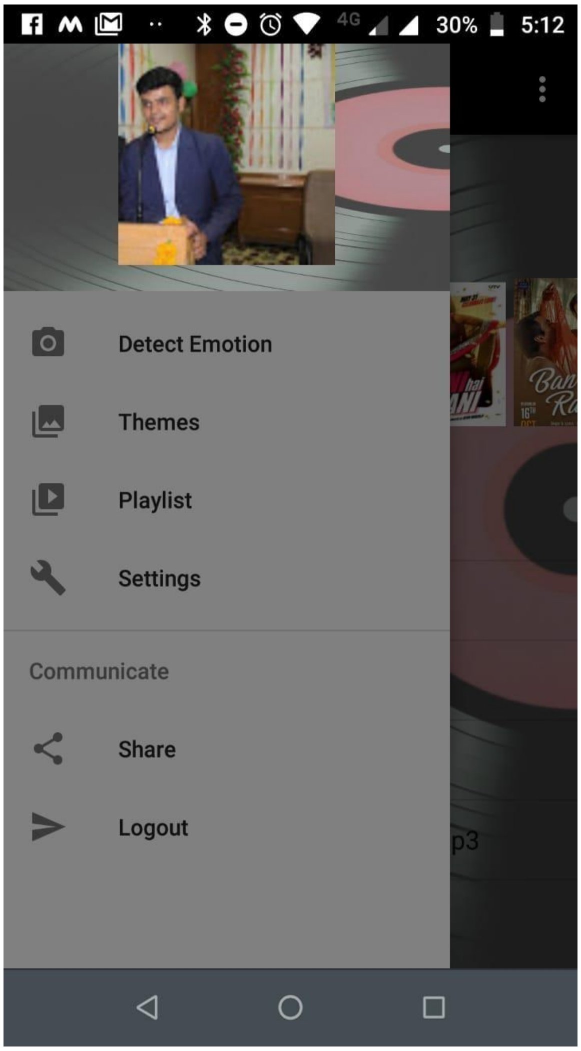


Fig : 5.8

The captured image quality depends on the user's device. Each device has its own capturing techniques, and the Emotion-Based Music Player inherits all the features of the device’s regular camera. If the user’s device has beautification features, zooming, or timer techniques included with default camera, then these features are also loaded by the detector and can be used to increase image quality. After the picture is captured, the application prompts the user to accept or retake the captured image. Then the captured image is sent to the SDK to detect the image’s emotion.

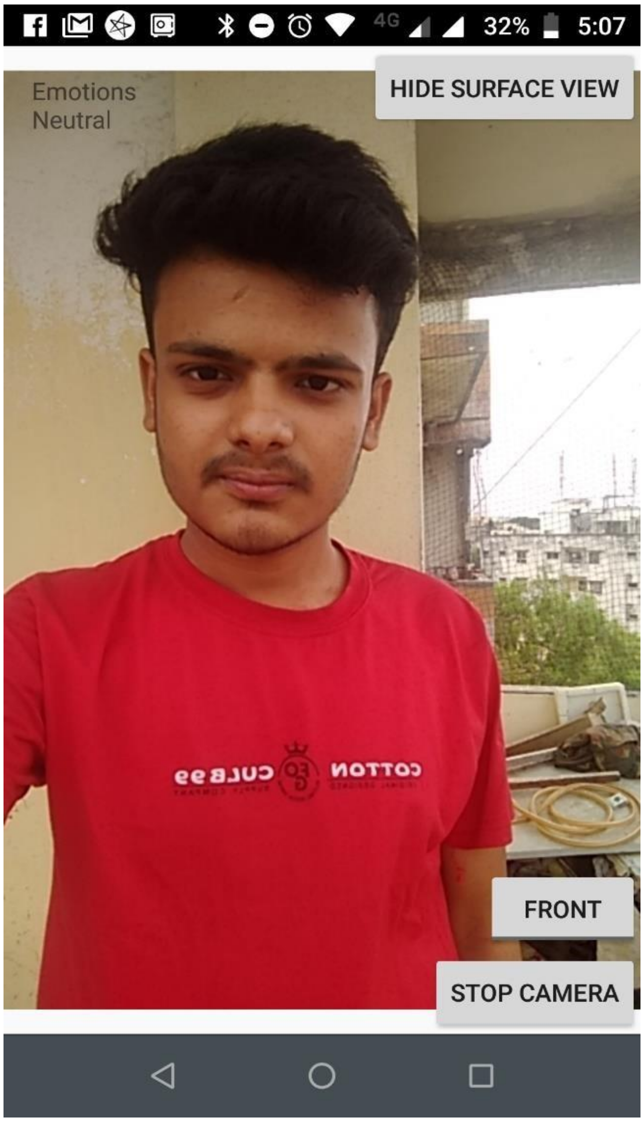
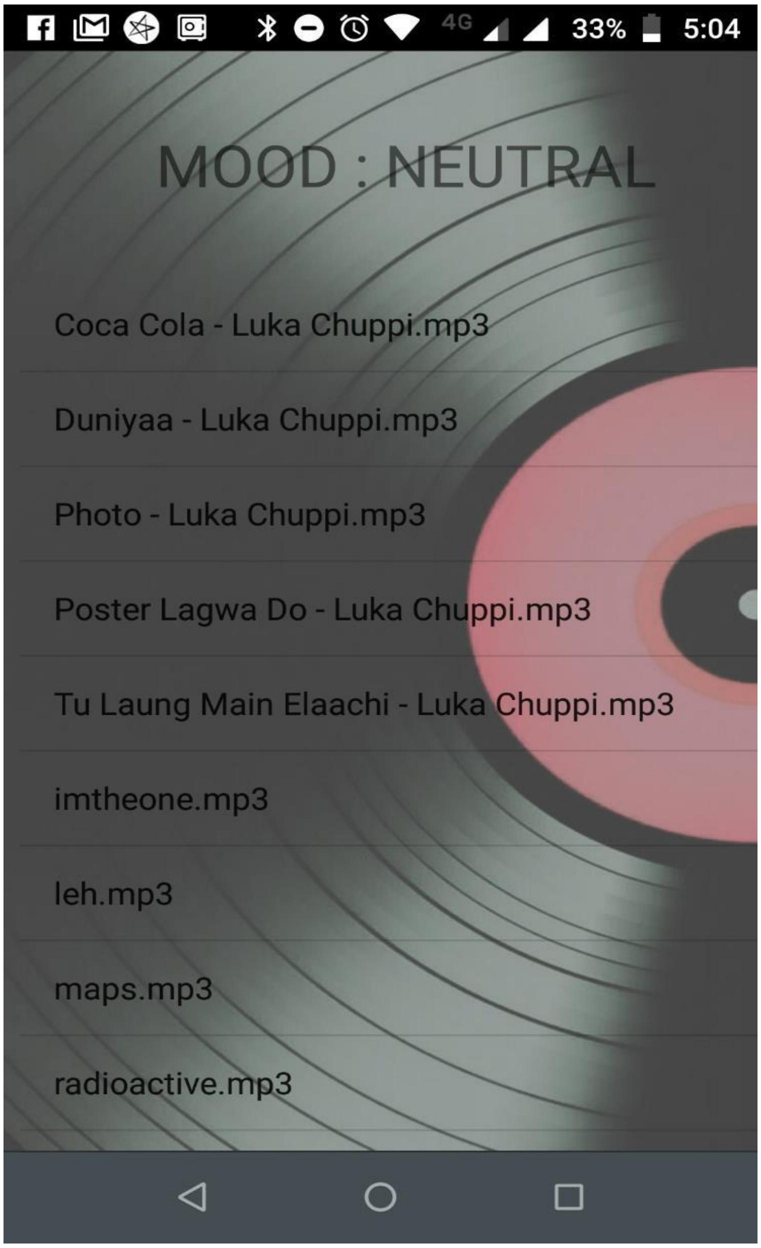


Fig : 5.9

The mood detected by our application is Neutral.

So the playlist generated according to the mood is :


4G 33% 5:04

MOOD : NEUTRAL

Coca Cola - Luka Chuppi.mp3

Duniyaa - Luka Chuppi.mp3

Photo - Luka Chuppi.mp3

Poster Lagwa Do - Luka Chuppi.mp3

Tu Laung Main Elaachi - Luka Chuppi.mp3

imtheone.mp3

leh.mp3

maps.mp3

radioactive.mp3

Fig : 5.10

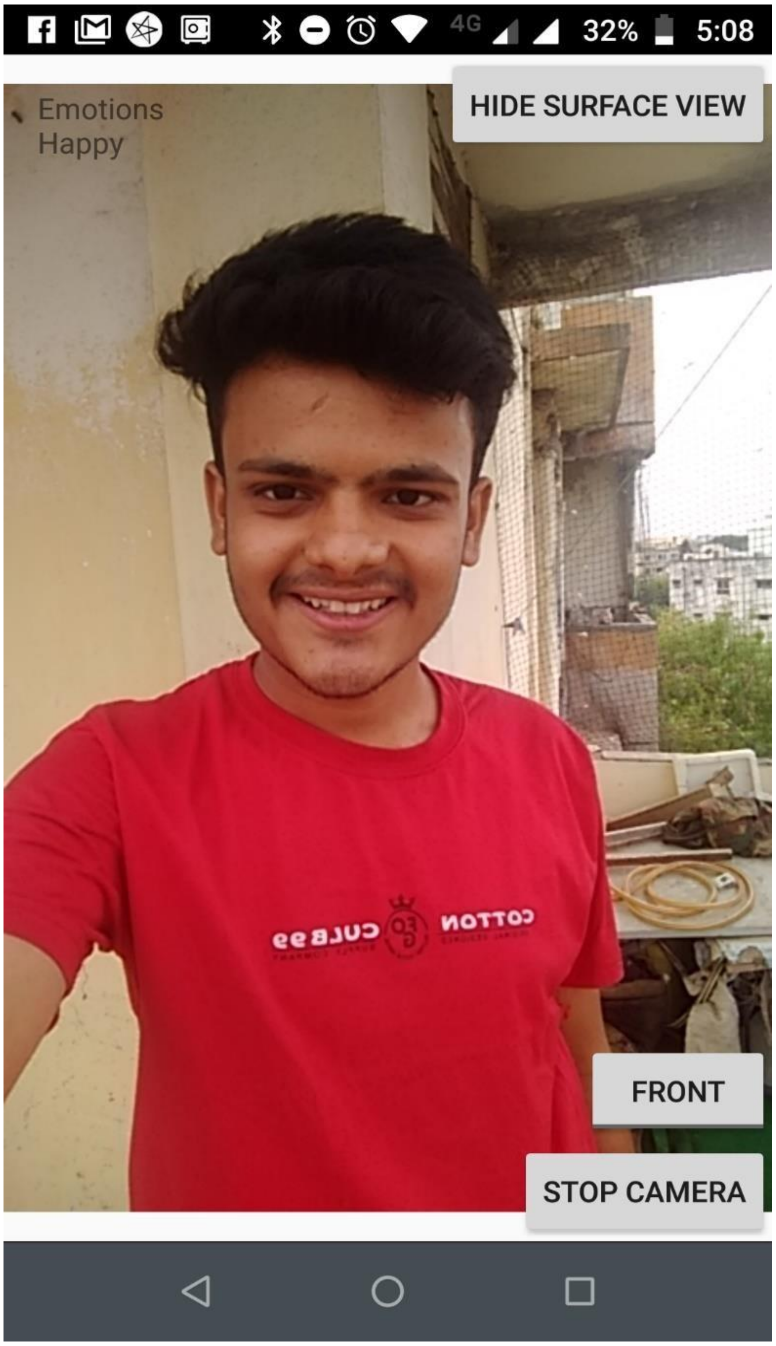
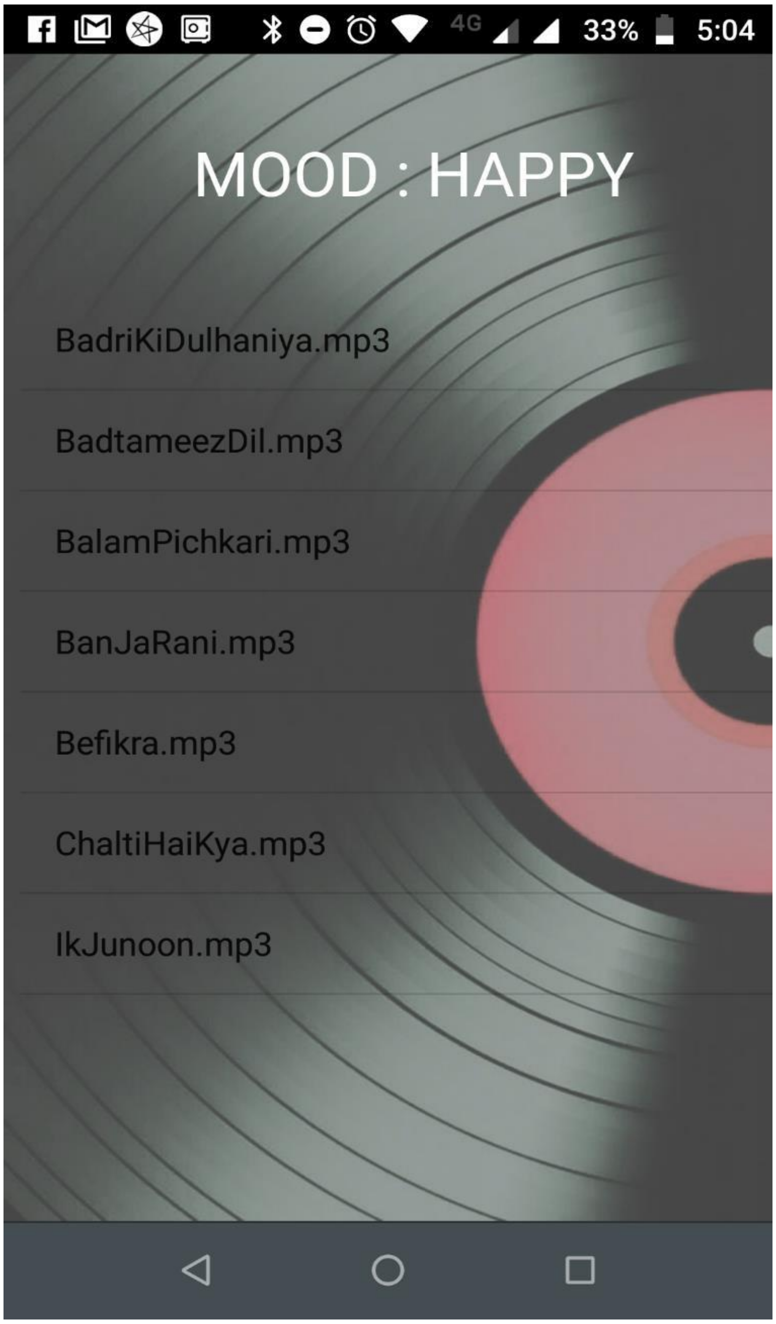


Fig : 5.11

The mood detected by our application is Happy.

So the playlist generated according to the mood is :


MOOD : HAPPY

BadriKiDulhaniya.mp3

BadtameezDil.mp3

BalamPichkari.mp3

BanJaRani.mp3

Befikra.mp3

ChaltiHaiKya.mp3

IkJunoon.mp3

Fig : 5.12

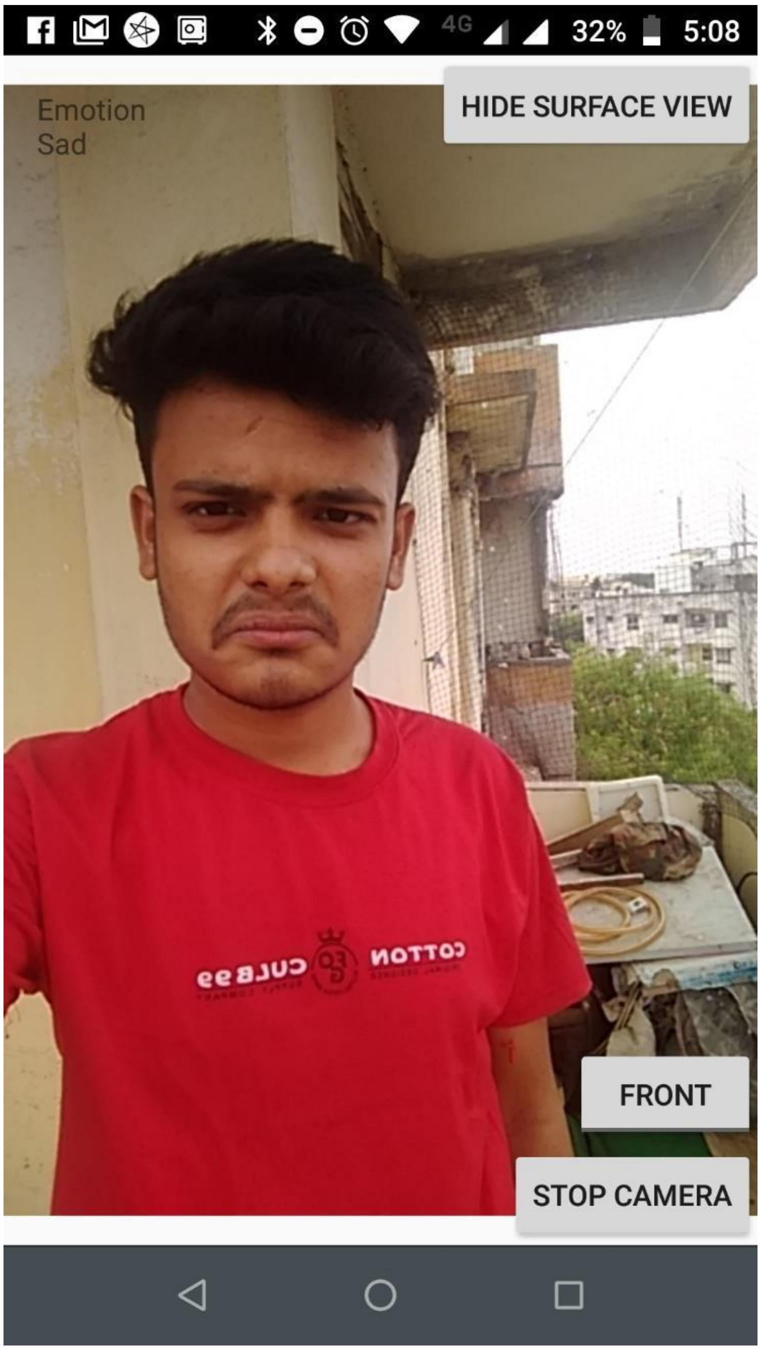
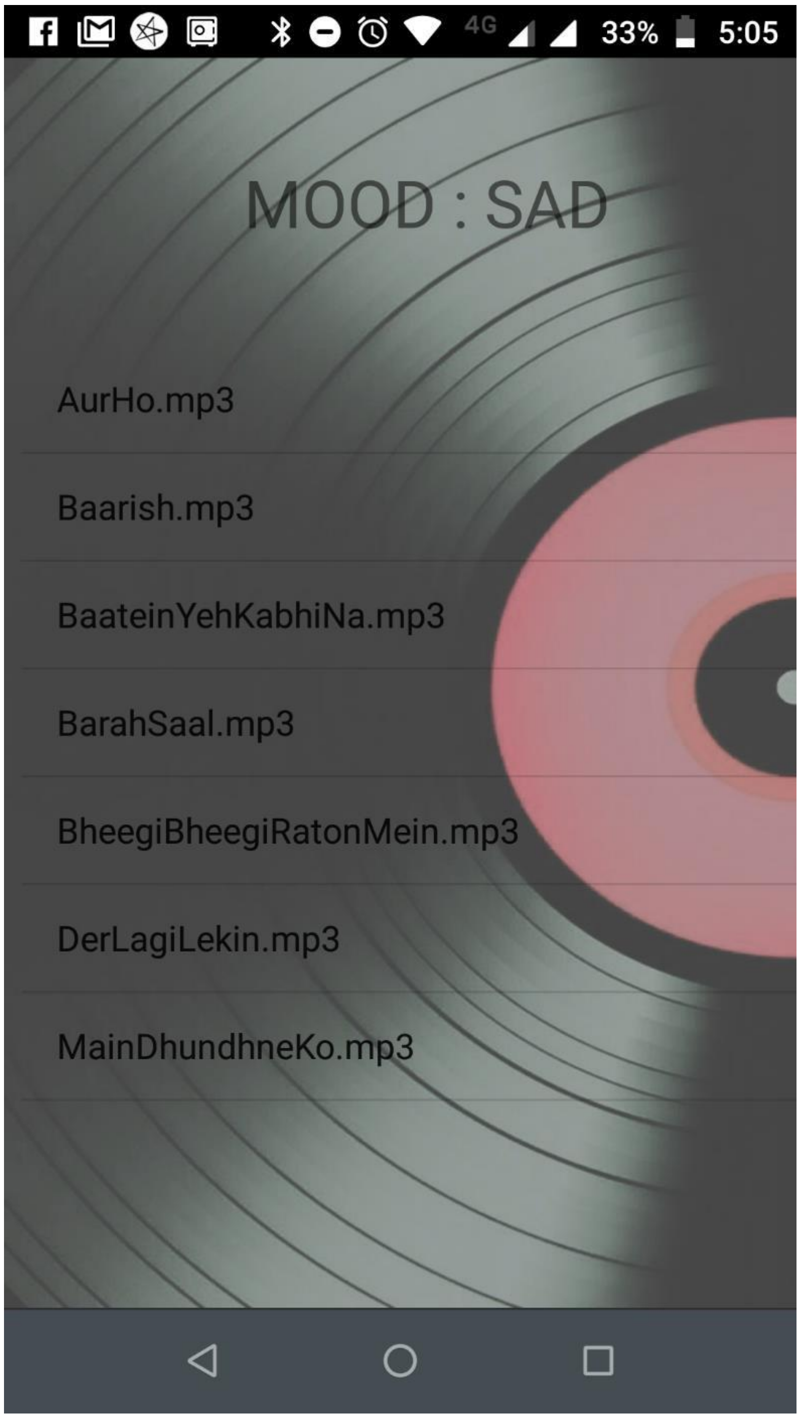


Fig 5.13

The mood detected by our application is Sad.

So the playlist generated according to the mood is :


MOOD : SAD

AurHo.mp3

Baarish.mp3

BaateinYehKabhiNa.mp3

BarahSaal.mp3

BheegiBheegiRatonMein.mp3

DerLagiLekin.mp3

MainDhundhneKo.mp3

Fig 5.14

5.2.3 Code Efficiency

Code efficiency is an important aspect for smooth working of code. There are various factors which focus on code efficiency. Basically we need to answer three questions for estimation code efficiency:

- Have functions been optimized for speed?
- Have repeatedly used block of code been formed into subroutines?
- Are there memory leaks or overflow errors?

For optimization of speed we have not imported the complete library rather we have imported individual classes of that library for fulfilling our requirements. For sort, we haven’t used math library for arithmetic calculations. Also we have taken a low quality of icons and .png images so that our project can run smoothly.

There were many occasions where we need to use same piece of code repeatedly(like JSONParser.class file). But instead of using the same code again and again, we used its object wherever required.

For preventing overflow errors and memory leaks, we have restricted the user to enter data of specific size. In database , the size of each datatype is predefined. This will not let memory leaks and overflow errors to occur.

5.2.4 Optimization Of Code

Code optimization is one of the most important aspect for efficiency measurement. Optimization of code is defined as how efficiently a code can run with fewest possible resources. Here are some of the optimization practices that we have involved in our project:

- Avoid_constant_expressions_in_loops
- Avoid_duplication_of_code
- Do_not_declare_members_accessed_by_inner_class_private
- Avoid_synchronized_modifier_in_method
- Avoid_empty_if

- Avoid_unnecessary_if
- Avoid_unnecessary_parentheses
- Avoid_unnecessary_implementing_Clonable_interface
- Remove_unnecessary_if_then_else_statement
- Avoid_instantiation_of_class_with_only_static_members
- Close_jdbc_connections
- Avoid_boolean_array
- Avoid_string_concatenation_in_loop
- Place_try_catch_out_of_loop
- Avoid_empty_try_blocks
- Avoid_empty_loops
- Avoid_unnecessary_substring
- Avoid_unnecessary_exception_throwing
- Use_PreparedStatement_instead_of_Statement
- Avoid_Extending_java_lang_Object
- Avoid_empty_catch_blocks
- Avoid_synchronized_methods_in_loop
- Avoid_synchronized_blocks_in_loop

5.2.5 Validation Check **Login:**

Sr. no.	Input values	Test cases	Condition Being checked	Result
1	User id	Empty	Please enter valid email id	Successful
2	Password	Empty	Password cannot be empty	Successful

Table : 5.1

6. Testing

Testing is the major quality control that can be used during software development. Its basic function is to detect the errors in the software. During requirement analysis and design, the output is a document that is usually textual and non-executable. After the coding phase, computer program is available that can be executed for testing purposes. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during previous phases. Thus the goal of the testing is to uncover requirement, design and coding errors in the program.

An elaborate testing of data is prepared and the system is tested using that test date. Errors noted and corrections made during the testing. The corrections are also noted for future use. The users are trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully in future. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately before live operation commences. Testing is vital to the success of any system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

6.1 Testing Objectives

- 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 undiscovered error
- A successful test is one that uncovers an as-yet undiscovered error

Testing Principles

- All tests should be traceable to customer requirements
- Tests should be planned long before testing begins
- Testing should begin "in the small" and progress toward testing "in the large"
- Exhaustive testing is not completely possible
- To be most effective, testing should be conducted by an independent third party

6.2 Testing Methods

Software Testing Strategies

A strategy for software testing integrates software test case design methods into a wellplanned series of steps that result in the successful construction of software. As important, a software testing strategy provides a road map. Testing is a set of activities that can be planned in advance and conducted systematically.

Various strategies are given below:

- Unit Testing
- Integration Testing
- Validation Testing
- User Acceptance Testing
- System Testing

Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design of module. This is also known as “Module Testing”. Acceptance of package is used for computerization of module. Machine Utilization was prepared and approved by the project leader.

In this testing step, each module is found to be working satisfactory as regards to the expected output from the module. The suggested changes were incorporated into the system. Here each module in the Machine Utilization has been tested.

Integration Testing

After the package is integrated, the user test version of the software was released. This testing consists of testing with live data and various stress tests and result were noted down. Then the corrections were made based on the users feedback. Integration testing is systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus the integration testing step, all the errors uncovered are corrected for the next steps.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been uncovered and corrected, and a final series of software tests - Validation testing - may begin.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at time of development and making changes wherever required. This is done in regard to the following points:

- Input Screen Design
- On-line Messages to guide the user
- Format of reports and other outputs

After performing all the above tests the system was found to be running successfully according to the user requirements i.e., (constraints).

System Testing

Software is only one element of a larger computer-based system.

Ultimately, software is incorporated with other system elements and a series of system integration and validation tests are conducted. The various types of system testing are:

- Recovery Testing: Many computer-based systems must recover from faults and resume processing within a pre specified time.
- Security Testing: Security testing attempts to verify that protection mechanisms

built into a system will in fact protect it from improper penetration.

- Stress Testing: Stress tests are designed to confront programs with abnormal situations.
- Performance Testing: Performance testing is designed to test run-time performance of software within the context of an integrated system.

Black Box Testing

Black box testing is carried out to check the functionality of the various modules. Although they are designed to uncover errors, black-box tests are used to demonstrate that software functions are operational; that input is properly accepted and output is correctly produced; and that the integrity of external information is maintained. A blackbox test examines some fundamental aspect of the system with little regard for the internal logical structure of the software.

White Box Testing

White-box testing of software is predicated on close examination of procedural detail providing the test cases that exercise specific sets of conditions and, loops tests logical paths through the software. White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, following test cases can be derived.

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- The errors that can be encountered while conducting white-box testing are Logic errors and incorrect assumptions.
- Typographical errors

7. CONCLUSION

The Emotion-Based Music Player(Moody) is used to automate and give a better music player experience for the end user. The application solves the basic needs of music listeners without troubling them as existing applications do: it uses technology to increase the interaction of the system with the user in many ways. It eases the work of the end-user by capturing the emotion using a camera,and suggesting a customized play-list through a more advanced and interactive system..

7.1 Limitation of Project

- Require good camera quality.
- Internet connectivity

7.2 Future Enhancement Suggestion

- The future scope in the system would to design a mechanism that would be helpful in categorizing songs on the basis of emotions as well as on the basis of age.
- Making the application run without needing internet connection.
- Including other emotions.
- Playing songs automatically.

8. Bibliography And References

1. Chang, C. Hu, R. Feris, and M. Turk, ―Manifold based analysis of facial expression,‖ Image Vision Comput ,IEEE Trans. Pattern
Anal. Mach. Intell. vol. 24, pp. 05–614, June 2006.
International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February, 2015

2. A. habibzad, ninavin, Mir kamalMirnia,‖ A new algorithm to classify face emotions through eye and lip feature by using particle swarm optimization.‖

3. Byeong-jun Han, Seungmin Rho, Roger B. Dannenberg and Eenjun Hwang,
―SMERS: music emotion recognition using support vector
regression‖, 10thISMIR , 2009.

4. Alvin I. Goldmana, b.Chandra and SekharSripadab, ―Simulationist models of facebased emotion recognition‖.

5. Michael lyon and Shigeru Akamatsu, ―Coding Facial expression with Gabor wavelets.‖, IEEE conf. on Automatic face and gesture recognition, March 2000 6. Shlok Gilda, Husain Zafar, Chintan Soni and Kshitija Waghurdekar – "Smart Music
Player Integrating Facial Emotion Recognition and Music Mood
Recommendation",IEEE WiSPNET conference 2017