CERTIFICATE

This is to certify that Mr. Hritik Maheshwari, Mr. Ishrak Khan, Mr. Vrashabh Paliwal , working in a group have satisfactorily completed the minor project titled "Facial Emotion based music player (**MOODY)**" towards the partial fulfillment of the degree in Bachelor of Engineering(Computer Science Engineering) Awarded by Shri Vaishnav Vidyapeeth Vishwavidyalaya for the academic year 2019-20.

Project Guide

Ms. Rupali Dave	Dr.Anand Rajawat
Assistant Professor	Head of Department

Internal	External

ACKNOWLEDGEMENT

It has been great honour and privilege to complete our Minor project on Smart Music Player Integrating Facial Emotion Recognition under the guidance of Ms. Rupali Dave.

We are very much thankful to Dr.Anand Singh Rajawat(Head of the Department, CSE) for providing all facilities and support to meet our project requirements.

We would like to take this opportunity to express our humble gratitude towards under whom we executed this project. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us complete its assigned tasks.

We are thankful to our project internal guide, whose invaluable guidance helped us understand this project better. Although there may be many who remain unacknowledged in this humble note of gratitude, there are none who remain unappreciated.

1. Hritik Maheshwari
2. Ishrak Khan
3. Vrashabh Paliwal

Contents

Abstract:

The human face is an important organ of an individual‘s body and it especially plays an important role in extraction of an individual‘s behavior and emotional state. Manually segregating the list of songs and generating an appropriate playlist based on an individual‘s emotional features is a very tedious, time consuming, labor intensive and upheld task. Various algorithms have been proposed and developed for automating the playlist generation process. However the proposed existing algorithms in use are computationally slow, less accurate and sometimes even require use of additional hardware like EEG or sensors. This proposed system based on facial expression extracted will generate a playlist automatically thereby reducing the effort and time involved in rendering the process manually. Thus the proposed system tends to reduce the computational time involved in obtaining the results and the overall cost of the designed system, thereby increasing the overall accuracy of the system. Testing of the system is done on both user dependent (dynamic) and user independent (static) dataset. Facial expressions are captured using an inbuilt camera. The accuracy of the emotion detection algorithm used in the system for real time images is around 85-90%, while for static images it is around 98-100%. The proposed algorithm on an average calculated estimation takes around 0.95-1.05 sec to generate an emotion based music playlist. Thus, it yields better accuracy in terms of performance and computational time and reduces the designing cost, compared to the algorithms used in the literature survey.

Extracting the required input from the human face can now be done directly using a camera. This input can then be used in many ways. One of the applications of this input can be for extracting the information to deduce the mood of an individual. This data can then be used to get a list of songs that comply with the „mood" derived from the input provided earlier. This eliminates the time-consuming and tedious task of manually segregating or grouping songs into different lists and helps in generating an appropriate playlist based on an individual's emotional features.

CHAPTER 1

INTRODUCTION

SMART MUSIC PLAYER INTEGRATING FACIAL EMOTION RECOGNITION

1. INTRODUCTION:

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 tough time creating and segregating the play-list manually when they have hundreds of songs. It is also difficult to keep track of all the songs: sometimes songs that are added and never used, wasting a lot of device memory and forcing the user to find and delete songs manually. User's have to manually select songs every time based on interest and mood. User's also have difficulty to re-organize and playing music when play-style varies. Currently in existing application, music is organized using play-list, and play-list songs cannot be modified or altered in one click. User's have to manually change or update each song in their play-list every time. The sequence of songs in a play-list might not be the same every time, and songs that a user wants to listen frequently might not be given priority or might be left out from the list. Currently, there are no applications that allows users to play songs on-the-go without selecting songs manually or from a 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 playlists 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

1.4 PLATFORM SPECIFICATION

1.4.1 SOFTWARE REQUIREMENT:

- Database-Firebase Realtime – DB
- Authentication – Firebase authentication tool
- Android Studio

1.4.2 IMPLEMENTATION LANGUAGE:

- JAVA
- XML (Extensible Markup Language)

CHAPTER 2

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS:

2.1 IDENTIFICATION OF NEED:

Music is an important entertainment medium. With advancement of technology,the optimization of manual work has gained a lot of attention. Currently, thereare many traditional music players that require songs to be manually selected andorganized. User, have to create and update play-list for each mood, which is timeconsuming. Some of the music players have advanced features like providing lyrics andrecommending similar songs based on the singer or genre . Although some of thesefeatures are enjoyable for user, there is room to improve in the field of automationwhen it comes to music players. Selecting songs automatically and organizing thesebased on the user's mood gives user's a better experience. This can be accomplishedthrough the system reacting to the user's emotion, saving time that would have beenspent entering information manually.

Emotions can be expressed through gestures, speech, facial expressions, etc. Forthe system to understand a user's mood, we use facial expression. Using the mobiledevice's camera, we can capture the user's facial expression. There are many emotion recognition systems which take captured image as input and determine the emotion..

CHAPTER 3

FEASIBILITY STUDY

3. 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.

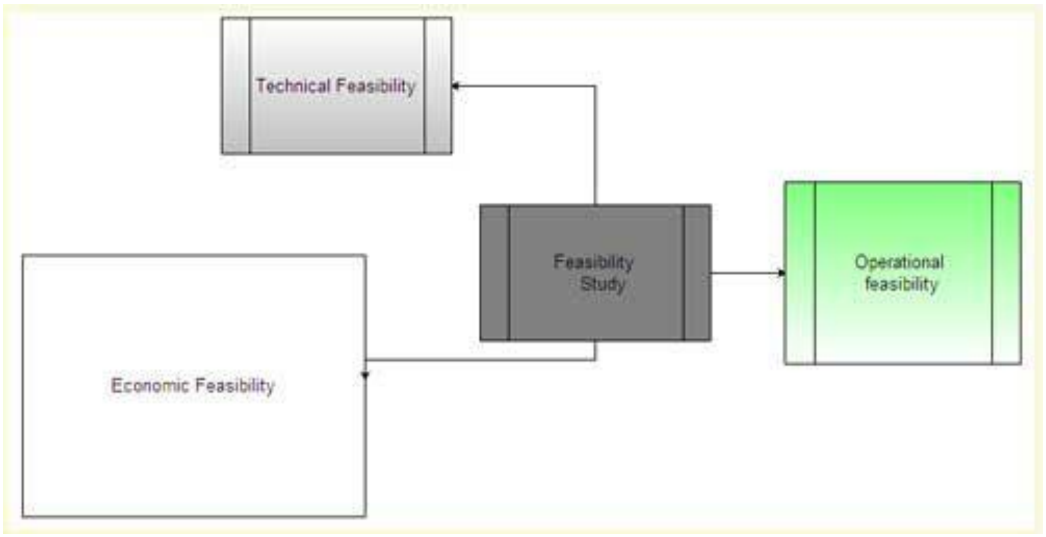
It enables us to determine the potential of existing system and improving it. It helps to develop a technically and economically feasible system. It helps to know what should be embedded in the system. It also helps to develop a cost-effective system. We can make better utilization of available resources.

The project concept is feasible because of the following:

3.1 Technical Feasibility

3.2 Economical Feasibility

3.3 Operational Feasibility



3.1 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.

3.2 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.

3.3 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.

CHAPTER 4

LITERATURE SURVEY

4. 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.

4.1 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 .

[4]. F. Abdat, C. Maaoui and A. Pruski proposed system fully automatic facial expression and recognition system based on three step face detection, facial characteristics extraction and facial expression classification. This system proposed anthropometric model to detect the face feature point combined to shi and Thomasi method. In this metod the variation of 21 distances which describe the facial feature from neutral face and the classification base on SVM (Support Vector Machine).

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.

4.2 Benefits:

There are several benefits of “SMART MUSIC PLAYER INTEGRATING FACIAL EMOTION RECOGNITION (MOODY)”

- Accuracy of our project is 94.5%.
- User doesn’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.
- Extremely fast feature computation and efficient feature selection.

4.3 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:

1.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.

2.Then after the files are detected, they are scanned for audio features and these features are extracted.

3.Then the extracted feature values are subjected to classification according to the parameters provided.

4.These parameters include a limited set of genre types based on which the audio feature values will be processed.

5.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.

6.In the next step, the user camera is invoked with proper permissions and a real time graphical input(image)is provided to the system.

7.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.

8.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.

4.4 Technology Used:

Frond End:

- XML(Extensible Markup Language)

Back End:

- JAVA
- Kotlin
- Firebase

Technology:

- Machine Learning
    - ❖ TensorflowLite
    - ❖ Deep Learning - CNN

CHAPTER 5

TECHNICAL PART

5.1 DESCRIPTION OF SOFTWARE AND IMPLEMENTATION LANGUAGES

5.1.1 SOFTWARE USED

1.ANDROID STUDIO

It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. To support application development within the Android **operating system**, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Used for development of complete application.

2.FIREBASE

Firebase is a Backend-as-a-Service—BaaS—that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform. Used for all the back-end services required by application.

FIREBASE AUTHENTICATION

Used to authenticate user(Google sign in enabled).

FIREBASE REALTIME DATABASE

Use to store users information at real time.

5.1.2 IMPLEMENTATION LANGUAGE

1. XML

Stands for Extensible Markup Language , used for designing entire GUI(Graphical User Interface) of application.

2.JAVA

All the back-end logics and algorithm designed using JAVA.

5.2 TIME ESTIMATION OF VARIOUS MODULES

MODULES	TIME TAKEN
Face detection	**0.8126**
Facial emotion extraction	**0.9216**
Playlist generation	**0.3245**

CHAPTER 6

SOFTWARE ENGINEERING APPROACH

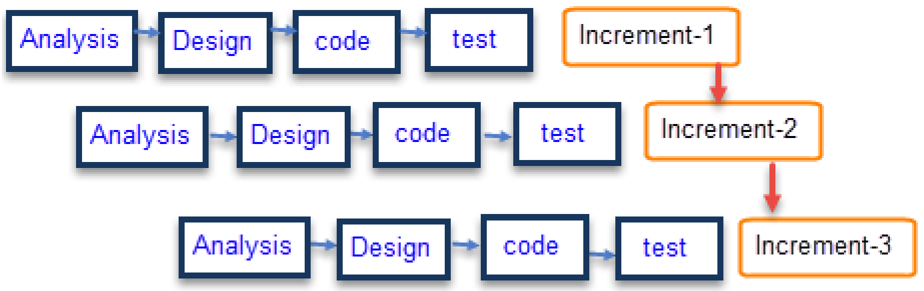
6. SOFTWARE ENGINEERING APPROACH:

6.1 SOFTWARE ENGINEERING PARADIGM APPLIED

6.1.1 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.

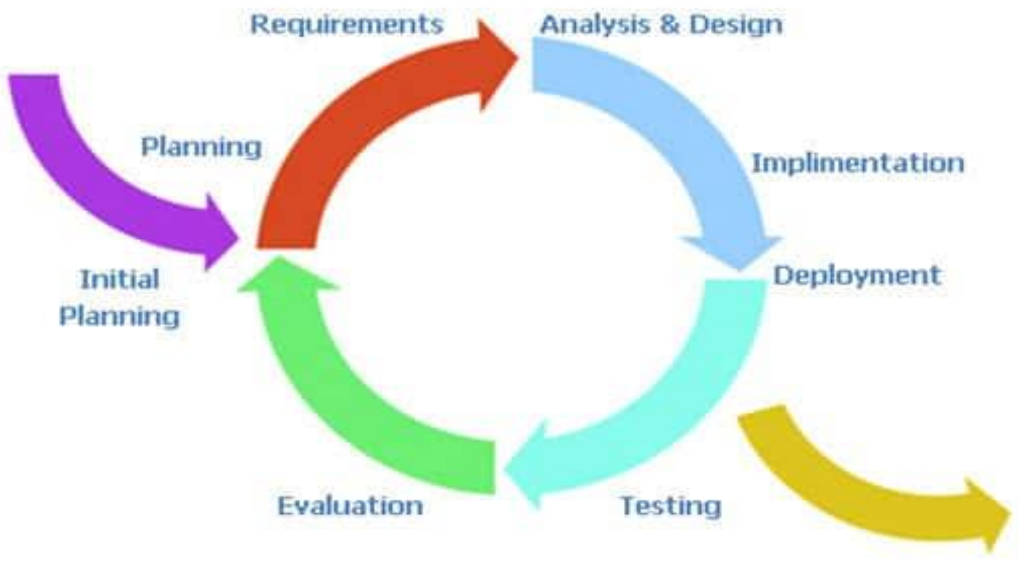


Incremental Model

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



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.

6.1.2 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.

6.1.3 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.

6.2 Requirement Analysis:

Requirement Analysis:

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.

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
- Maintainability
- Portability
- Efficiency
- Accuracy
- Performance
- Simplicity
- Extensibility

6.2.1 Software Requirement Specification:

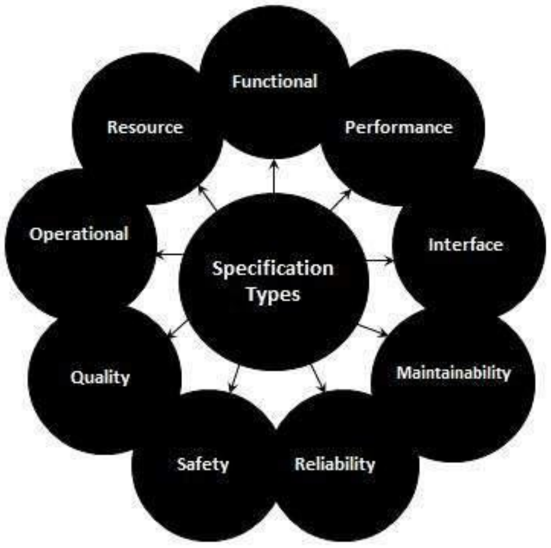
A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with 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



6.2.1.3 Use Case Model:

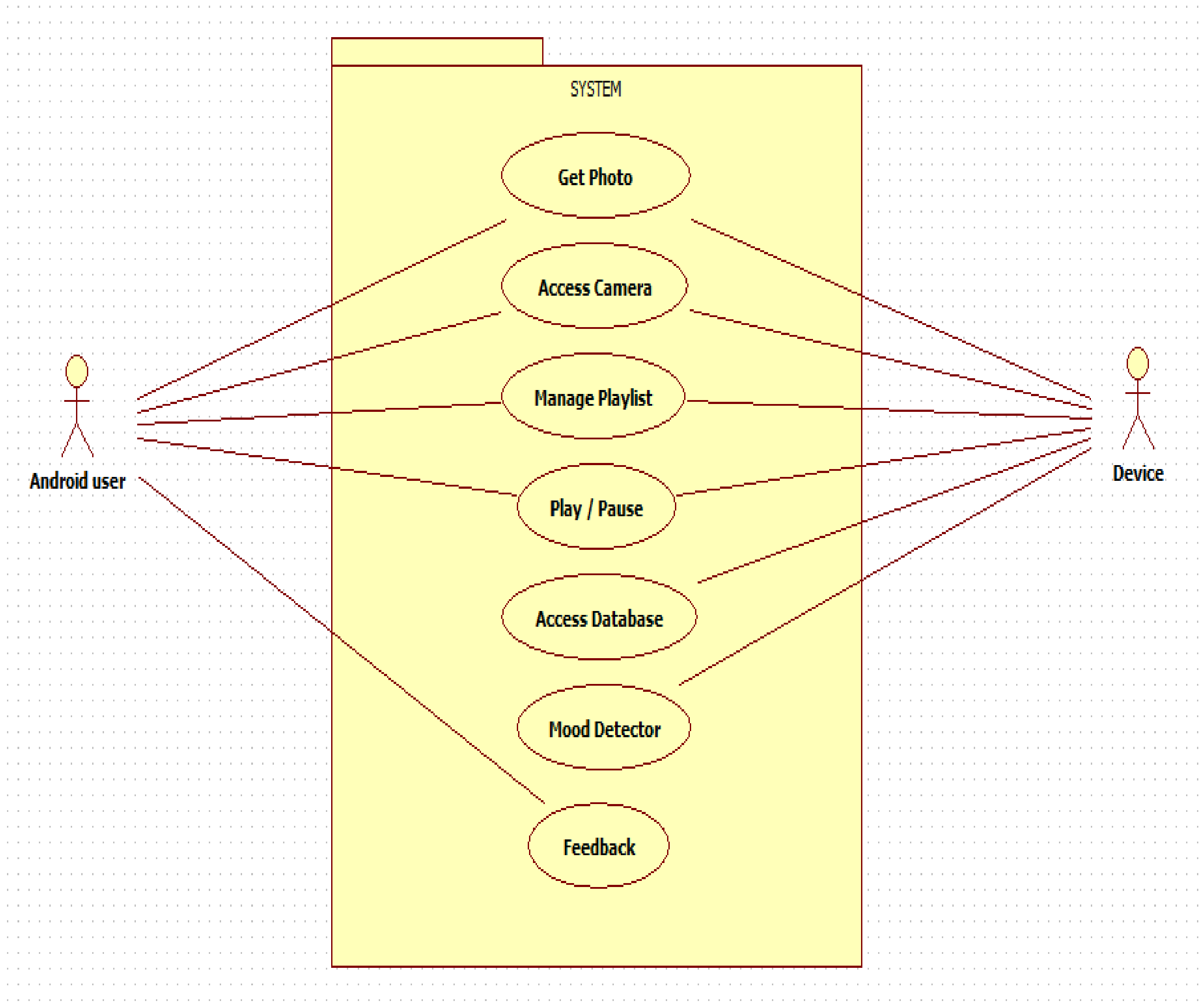
A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (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.

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.



6.2.1.3 Use case diagram

6.2.1.4 COMPARATIVE ANALYSIS DOCUMENTS:

Application features	**Gaana**	**Saavn**	**Wynk**	**Moody**
Platforms	Android Ios	Android ios	Android ios	Android
Themes	Less choices available	Can't change theme	Can't change theme	Comparatively more themes available
Songs language	12+ languages available	12+ language available	7+ language available	12+ languages
Subscription	INR - 399 for gaana+ subscription	Free for jio users only(biased)	For airtel users – INR 29 per month Non-airtel users – INR 99 /month(biased)	Free Subscription
Playlist Creation	Yes	Yes	No	Yes
Download Songs	Gaana+ subscription required	(biased)Jio user can download free	(biased)Airtel user can download free	Anyone can download free
Lyrics	Available	Not available	Not available	Available
Night mode	yes	yes	no	yes

Buffer time	Take time for hd quality	Take less time	Take time	Comparatively take less time
Recommendation on user preference	Yes	Yes	No	Yes
Recommendation by facial emotions recognition	No	No	No	Yes

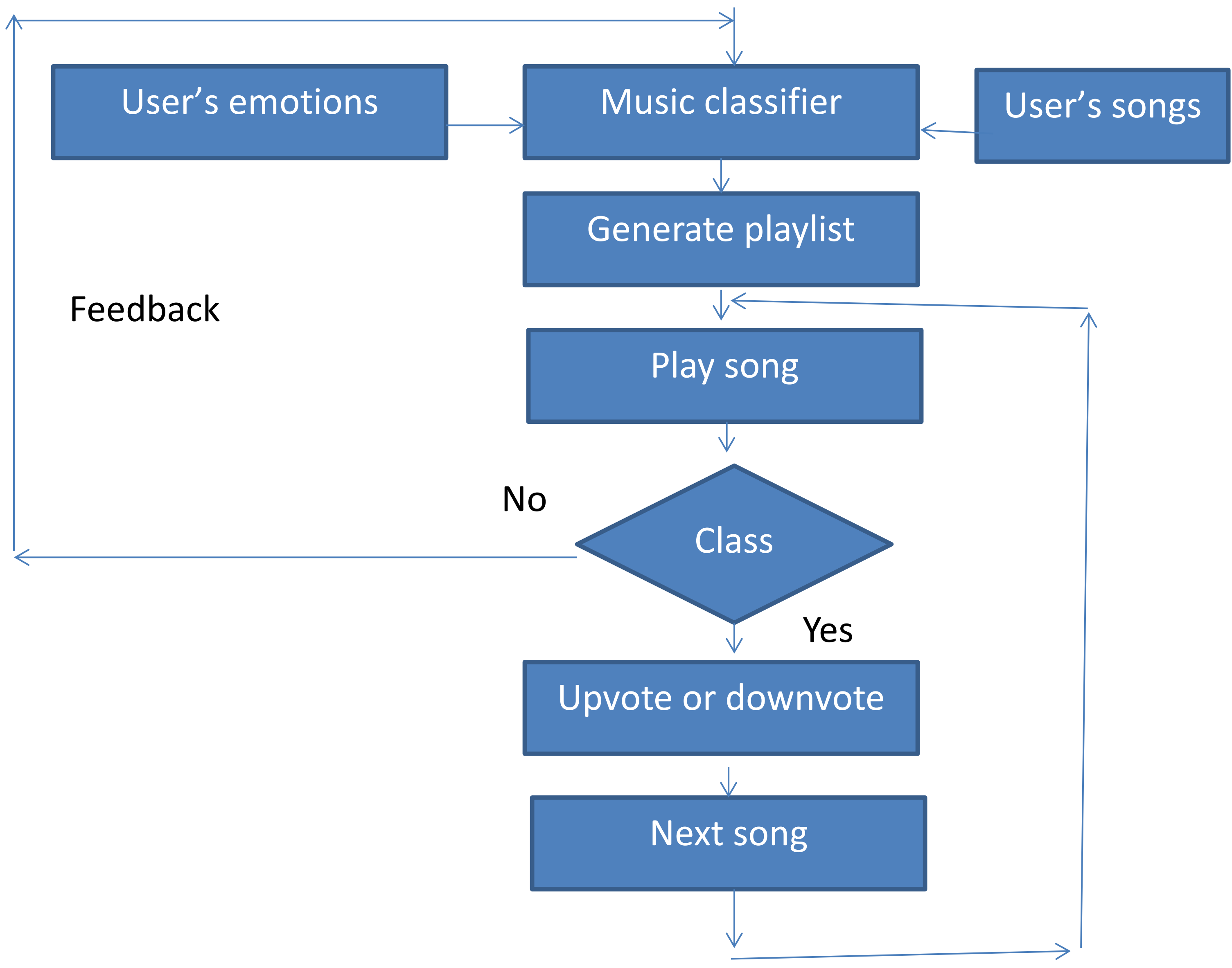
6.2.2 Conceptual Level Activity diagram:

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.


User's emotions
Music classifier
User's songs
Generate playlist
Feedback
Play song
No
Class
Yes
Upvote or downvote
Next song

6.3 PLANNING MANAGERIAL ISSUES

6.3.1 PLANNING SCOPE

The first activity in software project planning is the determination of software scope.
Software scope describes the data and control to be processed, function, performance, constraints, interfaces, and reliability. Functions described in the statement of scope are evaluated and in some cases refined to provide more detail prior to the beginning of estimation. Because both cost and schedule estimates are functionally oriented, some degree of decomposition is often useful. Performance considerations encompass processing and response time requirements. Constraints identify limits placed on the software by external hardware, available memory, or other existing systems.

6.3.2 PROJECT RESOURCES

The second software planning task is estimation of the resources required to accomplish the software development effort.
The development environment—hardware and software tools—sits at the foundation of the resources pyramid and provides the infrastructure to support the development effort. At a higher level, we encounter reusable software components—software building blocks that can dramatically reduce development costs and accelerate delivery. At the top of the pyramid is the primary resource—people.
Each resource is specified with four characteristics: description of the resource, a statement of availability, time when the resource will be required; duration of time that resource will be applied. The last two characteristics can be viewed as a time window. Availability of the resource for a specified window must be established at the earliest practical time.

Human Resources
The planner begins by evaluating scope and selecting the skills required to complete development. Both organizational positions (e.g., manager, senior software engineer) and specialty (e.g., telecommunications, database, and client/server) are specified. For relatively small projects (one

person-year or less), a single individual may perform all software engineering tasks, consulting with specialists as required. The number of people required for a software project can be determined only after an estimate of development effort (e.g., person-months) is made.

Reusable Software Resources

Component-based software engineering (CBSE) emphasizes reusability—that is, the creation and reuse of software building blocks. Such building blocks, often called components, must be cataloged for easy reference, standardized for easy application, and validated for easy integration.
Off-the-shelf components: Existing software that can be acquired from a third party or that has been developed internally for a past project.
Full-experience components: Existing specifications, designs, code, or test data developed for past projects that are similar to the software to be built for the current project. Members of the current software team have had full experience in the application area represented by these components. Therefore, modifications required for full-experience components will be relatively low-risk.
Partial-experience components: Existing specifications, designs, code, or test data developed for past projects that are related to the software to be built for the current project but will require substantial modification. Members of the current software team have only limited experience in the application area represented by these components. Therefore, modifications required for partial-experience components have a fair degree of risk.
New components: Software components that must be built by the software team specifically for the needs of the current project.

Environmental Resources

The environment that supports the software project, often called the software engineering environment (SEE), incorporates hardware and software. Hardware provides a platform that supports the tools (software) required to produce the work products that are an outcome of good software engineering practice. Because most software organizations have multiple constituencies that require access to the SEE, a project planner must prescribe the time window required for hardware and software and verify that these resources will be available.
When a computer-based system (incorporating specialized hardware and software) is to be engineered, the software team may require access to hardware elements being developed by other

engineering teams. For example, software for a numerical control (NC) used on a class of machine tools may require a specific machine tool (e.g., an NC lathe) as part of the validation test step; a software project for advanced page-layout may need a digital-typesetting system at some point during development. Each hardware element must be specified by the software project planner.

6.3.3 TEAM ORGANIZATION:

The following options are available for applying human resources to a project that will require n people working for k years:

N individuals are assigned to M different functional tasks, relatively little combined work occurs; coordination is the responsibility of a software manager who may have six other projects to be concerned with.

N individuals are assigned to m different functional tasks (m< n) so that informal "teams" are established; an ad hoc team leader may be appointed; coordination among teams is the responsibility of a software manager.

N individuals are organized into t teams; each team is assigned one or more functional tasks; each team has a specific structure that is defined for all teams working on a project; coordination is controlled by both the team and a software project manager.

Although it is possible to voice arguments for and against each of these approaches, a growing body of evidence indicates that a formal team organization (option 3) is most productive.

The “best” team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty.

How should a software team be organized?

Democratic decentralized (DD): This software engineering team has no permanent leader. Rather, "task coordinators are appointed for short durations and then replaced by others who may coordinate different tasks." Decisions on problems and approach are made by group consensus. Communication among team members is horizontal.

Controlled decentralized (CD): This software engineering team has a defined leader who coordinates specific tasks and secondary leaders that have responsibility for subtasks. Problem solving remains a group activity, but implementation of solutions is partitioned among subgroups by the team leader. Communication among subgroups and individuals is horizontal. Vertical communication along the control hierarchy also occurs.

Controlled Centralized (CC):Top-level problem solving and internal team coordination are managed by a team leader. Communication between the leader and team members is vertical.

Seven project factors that should be considered when planning the structure of software engineering teams:

- The difficulty of the problem to be solved.
- The size of the resultant program(s) in lines of code or function points
- The time that the team will stay together (team lifetime).
- The degree to which the problem can be modularized.
- The required quality and reliability of the system to be built.
- The rigidity of the delivery date.
- The degree of sociability (communication) required for the project.

Because a centralized structure completes tasks faster, it is the most adept at handling simple problems. Decentralized teams generate more and better solutions than individuals. Therefore such teams have a greater probability of success when working on difficult problems. Since the CD team is centralized for problem solving, either a CD or CC team structure can be successfully applied to simple problems. A DD structure is best for difficult problems.

Because the performance of a team is inversely proportional to the amount of communication that must be conducted, very large projects are best addressed by team with a CC or CD structures when subgrouping can be easily accommodated. It has been found that DD team structures result in high morale and job satisfaction and are therefore good for teams that will be together for a long time. The DD team structure is best applied to problems with relatively low modularity, because of the higher volume of communication needed. When high modularity is possible (and people can do their own thing), the CC or CD structure will work well.

It's often better to have a few small, well-focused teams than a single large team.

CC and CD teams have been found to produce fewer defects than DD teams, but these data have much to do with the specific quality assurance activities that are applied by the team. Decentralized teams generally require more time to complete a project than a centralized structure and at the same time are best when high sociability is required.

6.3.4 PROJECT SCHEDULING

Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

It is important to note, however, that the schedule evolves over time. During early stages of project planning, a macroscopic schedule is developed. This type of schedule identifies all major software engineering activities and the product functions to which they are applied. As the project gets under way, each entry on the macroscopic schedule is refined into a detailed schedule. Here, specific software tasks (required to accomplish an activity) are identified and scheduled.

Scheduling for software engineering projects can be viewed from two rather different perspectives. In the first, an end-date for release of a computer-based system has already (and irrevocably) been established. The software organization is constrained to distribute effort within the prescribed time frame. The second view of software scheduling assumes that rough chronological bounds have been discussed but that the end-date is set by the software engineering organization. Effort is distributed to make best use of resources and an end-date is defined after careful analysis of the software. Unfortunately, the first situation is encountered far more frequently than the second.

Basic principles for software project scheduling:

Compartmentalization: The project must be compartmentalized into a number of manageable activities and tasks. To accomplish compartmentalization, both the product and the process are decomposed

Interdependency: The interdependency of each compartmentalized activity or task must be determined. Some tasks must occur in sequence while others can occur in parallel. Some activities cannot commence until the work product produced by another is available. Other activities can occur independently.

Time allocation: Each task to be scheduled must be allocated some number of work units (e.g., person-days of effort). In addition, each task must be assigned a start date and a completion date that are a function of the interdependencies and whether work will be conducted on a full-time or part-time basis.

Effort validation: Every project has a defined number of staff members. As time allocation occurs, the project manager must ensure that no more than the allocated number of people has been scheduled at any given time.

Defined responsibilities: Every task that is scheduled should be assigned to a specific team member.

Defined outcomes: Every task that is scheduled should have a defined outcome. For software projects, the outcome is normally a work product (e.g., the design of a module) or a part of a work product. Work products are often combined in deliverables.

Defined milestones: Every task or group of tasks should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality and has been approved.

6.3.5 ESTIMATION

- The accuracy of a software project estimate is predicated on:
    - The degree to which the planner has properly estimated the size (e.g., KLOC) of the product to be built
    - The ability to translate the size estimate into human effort, calendar time, and money
    - The degree to which the project plan reflects the abilities of the software team
    - The stability of both the product requirements and the environment that supports the software engineering effort

PROJECT ESTIMATION OPTIONS

- Options for achieving reliable cost and effort estimates
    - Delay estimation until late in the project (we should be able to achieve 100% accurate estimates after the project is complete)
    - Base estimates on similar projects that have already been completed
    - Use relatively simple decomposition techniques to generate project cost and effort estimates
    - Use one or more empirical estimation models for software cost and effort estimation.

PROJECT ESTIMATION APPROACHES

- Decomposition techniques
    - These take a "divide and conquer" approach
    - Cost and effort estimation are performed in a stepwise fashion by breaking down a project into major functions and related software engineering activities
- Empirical estimation models
    - Offer a potentially valuable estimation approach if the historical data used to seed the estimate is good

6.3.6 RISK ANALYSIS

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project. There is general agreement that risk always involves two characteristics

Uncertainty—the risk may or may not happen; that is, there are no 100% probable risks.

Loss—if the risk becomes a reality, unwanted consequences or losses will occur.

When risks are analyzed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk. To accomplish this, different categories of risks are considered.

- Project risks: if project risks become real, it is likely that project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and requirements problems and their impact on a software project.

- Technical risks: It threatens the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, implementation, interface, verification, and maintenance problems. Technical risks occur because the problem is harder to solve than we thought it would be.

- Business risks: It threatens the viability of the software to be built. Business risks often jeopardize the project or the product

- Known risks are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).
- Predictable risks are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced).

- Unpredictable risks are the joker in the deck. They can and do occur, but they are extremely difficult to identify in advance.

There are a few well-known types of risk analysis that can be used [21]. In software engineering, risk analysis is used to identify the high-risk elements of a project. It provides ways of documenting the impact of risk mitigation strategies. Risk analysis has also been shown to be important in the software design phase to evaluate criticality of the system, where risks are analyzed and necessary countermeasures are introduced . The purpose of risk analysis is to understand risk better and to verify and correct attributes. A successful analysis includes essential elements like problem definition, problem formulation, data collection.

Risk Tree Analysis and Assessment Method:

In risk tree analysis method, software risks are classified at first. Then risks are identified in each group. Afterwards, primary or basic risk events, intermediate events, top event, and the necessary sub-tree are found. All these require that managers have a complete knowledge about the projects. Then the risk tree can be constructed. Likelihood and impact must be assigned to each event and failure. Then probabilities starting from primary events to the top event are calculated. The events are ordered according to their probabilities. Maximum probability indicates the importance of those events; therefore, it is necessary to attend more to them. Managers should use solutions to prevent risks from occurring or reduce undesirable incidents.

The presented classifications and risk tree structures can apply with some software tools. Fault Tree Creation and Analysis Program, Fault Tree Tool or Relax Fault Tree can be used for this analysis. These tools have facilities that help users to create tree symbols and construct the risk tree structures.

<u>6.3.7 SECURITY PLAN</u>

Security plan include these steps:

1. Identify the assets you want to protect and the value of these assets.
2. Identify the risks to each asset.
3. Determine the category of the cause of the risk (natural disaster risk, intentional risk, or unintentional risk).
4. Identify the methods, tools, or techniques the threats use.

   After assessing your risk, the next step is proactive planning. Proactive planning involves developing security policies and controls and implementing tools and techniques to aid in security.

The various types of policies that could be included are:

- Password policies
  - Administrative Responsibilities
  - User Responsibilities
- E-mail policies
- Internet policies
- Backup and restore policies

6.4 DESIGN

6.4.1. 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.

6.4.2. DESIGN TECHNIQUE

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.

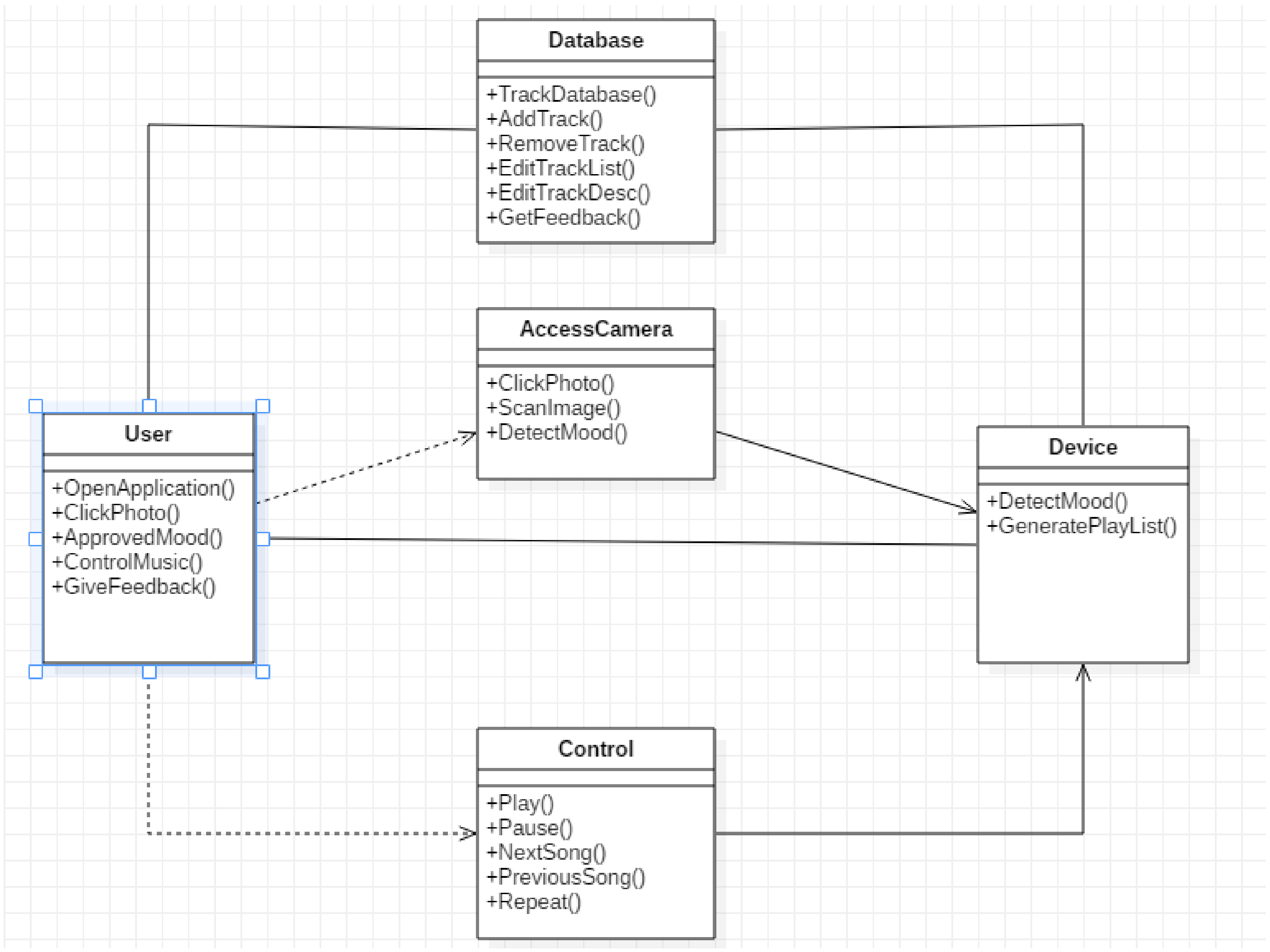
6.4.3. MODELING

6.4.3.1. 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.

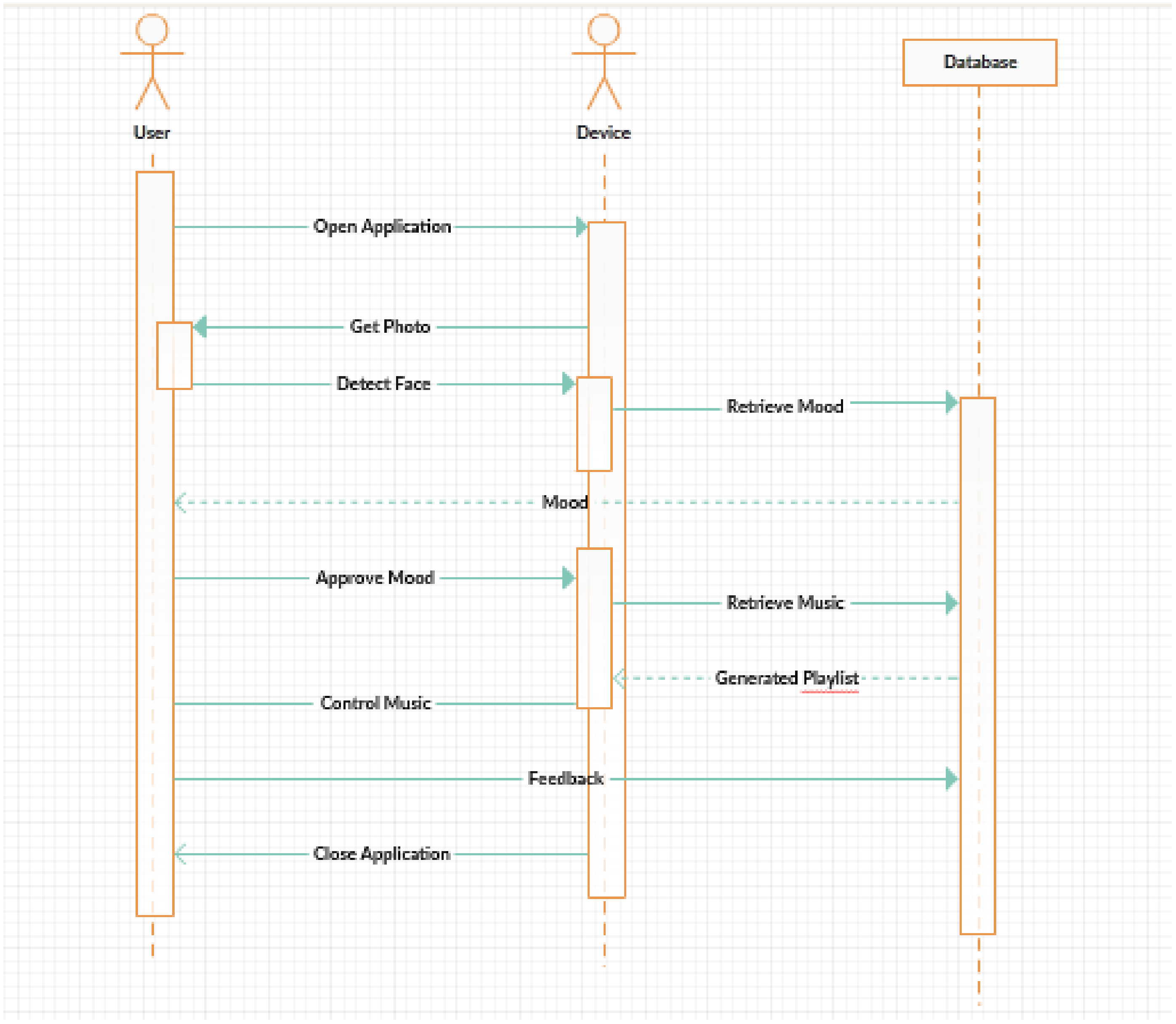
The following points should be remembered while drawing a class diagram:

- 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.


Database
+TrackDatabase()
+AddTrack()
+RemoveTrack()
+EditTrackList()
+EditTrackDesc()
+GetFeedback()
AccessCamera
+ClickPhoto()
+ScanImage()
+DetectMood()
User
+OpenApplication()
+ClickPhoto()
+ApprovedMood()
+ControlMusic()
+GiveFeedback()
Device
+DetectMood()
+GeneratePlayList()
Control
+Play()
+Pause()
+NextSong()
+PreviousSong()
+Repeat()

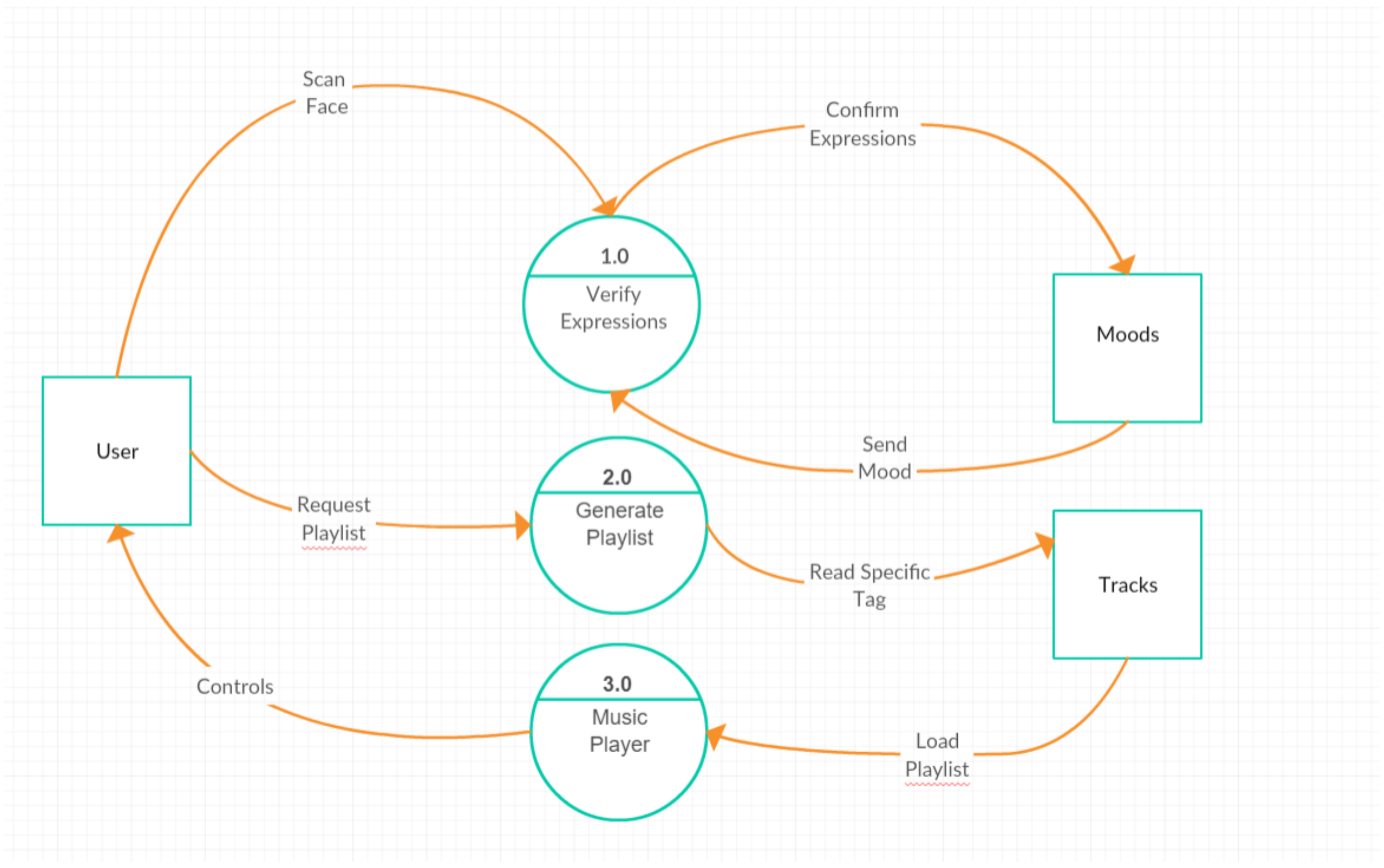
6.4.3.2. SEQUENCE DIAGRAM

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.



6.4.3.3 Data Flow Diagram:

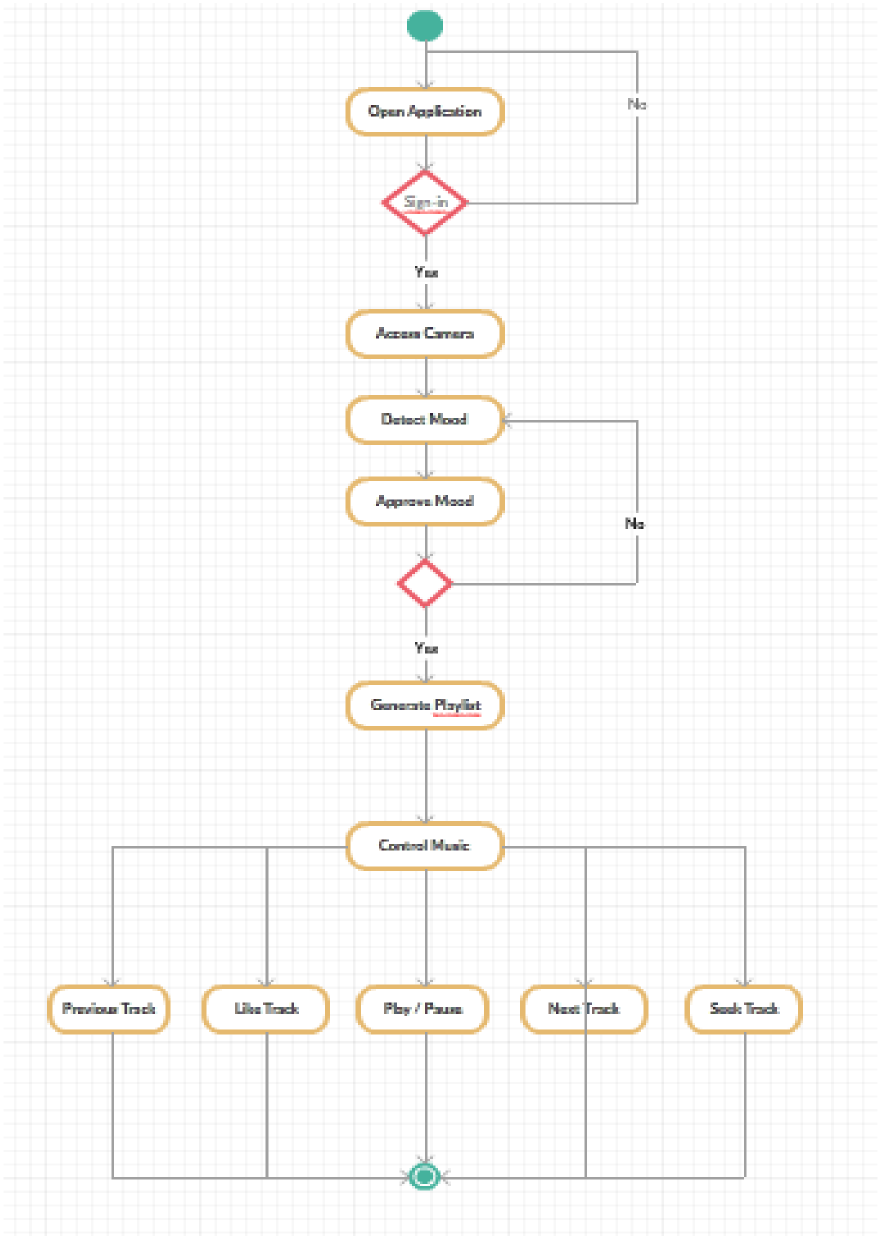
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.



6.4.3.4. 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. Typically, activity diagrams are used to model workflow or business processes and internal operation.

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.



6.5 IMPLEMENTATION PHASE:

6.5.1. 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
- Powerful designer scripts feature. Let's you easily create sophisticated layouts
- Basic4android UI Cloud service. Test your layouts on a cloud of real phones and tablets
- Supports all Android phones and tablets from Android 1.6 and up to Android 4.x
- Modern IDE with autocomplete, built-in documentation, internal index and other advanced features
- Powerful step-by-step debugger
- Large set of documentation
- Built-in code obfuscation.

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.

6.5.2. 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.

6.5.3. 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

6.5.4. 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

6.6TESTING:

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.

6.6.1 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

6.6.2TESTING METHODS

Software Testing Strategies

A strategy for software testing integrates software test case design methods into a well-planned 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 black-box 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

CHAPTER 7

Conclusion & Discussion

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 LIMITATIONS OF PROJECT:

- Require good camera quality.
- Internet connectivity

7.2 FUTURE ENHANCEMENT SUGGESTIONS:

- 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.

CHAPTER 8

Bibliography & References

8. BIBLOGRAPHY 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 face-based 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