UNIVERSITY OF PUNE

A

PROJECT REPORT

ON

“Karaoke System Application”

A Project work submitted to the University of Pune In fulfillment of the requirements for the

Bachelor of Engineering course In Computer Engineering

By

Ishan Gokhale	**B80574224**
Yogesh R. Isawe	**B80574229**
Sourabh Lakade	**B80574242**

Under the guidance of

Prof. P.S.Dhotre

DEPARTMENT OF

COMPUTER ENGINEERING

SINHGAD INSTITUTE OF TECHNOLOGY & SCIENCE, NARHE, PUNE (2012 – 2013)

SINHGAD TECHNICAL EDUCATION SOCIETY'S

SINHGAD INSTITUTE OF TECHNOLOGY AND SCIENCE

NARHE, Pune – 411 041

DEPARTMENT OF

COMPUTER ENGINEERING

CERTIFICATE

This is to certify that Project work entitled "**Karaoke System Application** "was successfully carried by

1. **Ishan Gokhale**
2. **Yogesh R. Isawe**
3. **Sourabh Lakade**

In the fulfillment of the Under Graduate Degree course in Final Year Computer Engineering, in the Academic Year 2012-2013 prescribed by the University of Pune.

Guide	**H.O.D**
Prof. P.S. Dhotre	**Prof. G.S. Navale**
Dept. of Computer Science & Engg	**Dept. of Computer Science & Engg**
SITS, Narhe	**SITS, Narhe**

Principal

Dr. S. N. MALI

SITS, Narhe

ACKNOWLEDGEMENT

With great pleasure and satisfaction we present our Project Report based on our project **Karaoke System Application.** We take this opportunity to express our sincere thanks to all those who have helped us to complete this project successfully.

We would like to thank **Prof. (Mrs) G.S.Navale** (Head, Department Of Computer Engineering, Sinhgad Institute of Technology and Science) and **Dr. S.N.Mali** (Principal, Sinhgad Institute of Technology and Science) for their support and their efforts in making sure that sufficient facilities were always available to us. We are indebted to the authors whose works have been referred for this project.

We take this opportunity to express our gratitude towards our project guide **Prof. P.S.Dhotre** for his expert guidance during the preparation of this project. We would like to thank him for the moral support, encouragement and the intelligent advice that he has imparted to us.

We offer our sincere thanks to **Prof. N.Ranjan** (Project Co-ordinator, Department Of Computer Engineering, SITS) for his guidance and support. We also thank all those who helped us with other project work.

ABSTRACT

Nowadays in the era of mobile devices, the world needs smart applications that are easy to use and very effective in the end. Android is the one of the popular mobile application platform these days which we have used. The main objective is to create an Android Application for Sound Processing and Karaoke Design, which will offer a very feature-rich experience. This project is to create an application that will take a song present on an Android Device and convert it into Karaoke. A karaoke of a song is the instrumental form of that song, in which the vocal track sung by the singer is suppressed to lower dB frequency. So, only the background beats and the orchestras remain audible. The app will also allow the device user to record his own voice, so that he can sing and record the song, sung along-side the Karaoke. Furthermore, the app will merge the Karaoke and the user's voice to produce the user's own version of the song. The user will be able to upload this song instantly on Sound Cloud through Wi-Fi connection.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO PROJECT DOMAIN

This project "*Sound Processing & Karaoke Design Application*" is a project made for Android devices like mobile phone, tablets etc. The project domain is primarily Mobile Computing. As the project includes manipulation of audio soundtracks, Sound Processing is also one of the necessary domains involved.

1.1.1 MOBILE COMPUTING

Mobile computing is a form of human computer interaction by which a computer is expected to be transported during normal usage. Mobile computing has three aspects: mobile communication, mobile hardware, and mobile software. The first aspect addresses communication issues in ad-hoc and infrastructure networks as well as communication properties, protocols, data formats and concrete technologies. The second aspect is the hardware, e.g., mobile devices or device components. The third aspect deals with the characteristics and requirements of mobile applications.

Mobile computing aims to provide a network infrastructure and corresponding terminal capability to perform all desktop-like computing functions seamlessly at any place or time, even while the terminal is moving. This means that anytime and anywhere, a user would be able to browse the web, check e-mail, play digital music, and perform all other computing activities without having to be behind a desktop at home or work. At its best, mobile computing would allow a user to have access to a consistent working environment.

Mobile Computing Hardware:
Many types of mobile computers have been introduced since the 1990s including the:

- Computer
- Personal digital assistant/enterprise digital assistant
- Smartphone
- Tablet computer
- Ultra-Mobile PC
- Wearable computer

Mobile application development is the process by which application software is developed for small low-power hand held devices such as personal digital assistants, enterprise digital assistants or mobile phones. These applications are either pre-installed on phones during manufacture, downloaded by customers from various mobile software distribution platforms, or web applications delivered over HTTP which use server-side or client-side processing (e.g. JavaScript) to providean ”application-like” experience within a Web browser.

Android, iOS, BlackBerry, HP webOS, Symbian OS, Bada from Samsung, and Windows Mobile support typical application binaries as found on personal computers with code which executes in the native machine format of the processor (the ARM architecture is a dominant design used on many current models). Windows Mobile can also be compiled to x86 executables for debugging on a PC without a processor emulator, and also supports the Portable Executable (PE) format associated with the .NET Framework. Windows Mobile, Android, HP webOS and iOS offer freeSDKs and integrated development environments to developers.

Figure 1.1 Mobile Platforms

Mobile applications are first tested within the development environment using emulators and later subjected to field testing. Emulators provide an inexpensive way to test applications on mobile phones to which developers may not have physical access. The following are examples of tools used for testing application across the most popular mobile operating systems.

- Google Android Emulator

It is Android Emulator which is patched to run on a Windows PC as a standalone app without having to download and install the complete and complex Android SDK, and can be even installed and Android compatible apps can be tested on it.
- Official Android SDK Emulator
It includes a mobile device emulator which mimics all of the hardware and software features of a typical mobile device (without the calls).

Limitations of Mobile Computing

- Insufficient bandwidth:
  Mobile Internet access is generally slower than direct cable connections, using technologies such as GPRS and EDGE, and more recently HSDPA and HSUPA 3G networks. These networks are usually available within range of commercial cell phone towers. Higher speed wireless LANs are inexpensive but have very limited range.

- Security standards
  When working mobile, one is dependent on public networks, requiring careful use of VPN. Security is a major concern while concerning the mobile computing standards on the fleet. One can easily attack the VPN through a huge number of networks interconnected through the line.

- Power consumption
  When a power outlet or portable generator is not available, mobile computers must rely entirely on battery power. Combined with the compact size of many mobile devices, this often means unusually expensive batteries must be used to obtain the necessary battery life.

- Transmission interferences
  Weather, terrain, and the range from the nearest signal point can all interfere with signal reception. Reception in tunnels, some buildings, and rural areas is often poor.

- Potential health hazards

People who use mobile devices while driving are often distracted from driving and are thus assumed more likely to be involved in traffic accidents.(While this may seem obvious, there is considerable discussion about whether banning mobile device use while driving reduces accidents or not. Cell phones may interfere with sensitive medical devices. There areallegations that cell phone signals may cause health problems.

- Human interface with device

  Screens and keyboards tend to be small, which may make them hard to use. Alternate input methods such as speech or handwriting recognition require training.

1.1.2 SOUND PROCESSING

Audio signal processing, sometimes referred to as audio processing, is the intentional alteration of auditory signals, or sound, often through an audio effect or effects unit. As audio signals may be electronically represented in either digital or analog format, signal processing may occur in either domain. Analog processors operate directly on the electrical signal, while digital processors operate mathematically on the digital representation of that signal.

Pitch shift - similar to pitch correction, this effect shifts a signal up or down in pitch. For example, a signal may be shifted an octave up or down. This is usually applied to the entire signal, and not to each note separately. One application of pitch shifting is pitch correction. Here a musical signal is tuned to the correct pitch using digital signal processing techniques. This effect is ubiquitous in Karaoke machines and is often used to assist pop singers who sing out of tune. It is also used intentionally for aesthetic effect in such pop songs.

1.2 PROBLEM STATEMENT

In Sound Processing & Karaoke Design Application, we propose an Android Smart Phone application for the 'music geeks'. Through this application the user can have a better entertainment & enjoyment in the world of music.This project is to create an application that will take a song present on an Android Device and convert it into Karaoke.The app will also allow the device user to record his own voice, so that he can sing and record the song, sung along-side the Karaoke. Furthermore, the app will merge the Karaoke and the user's voice to produce the user's own version of the song. The user will be able to upload this song instantly on Sound Cloud through Wi-Fi connection.

1.3 PROJECT OBJECTIVE

In today's world of automation and upcoming technologies people prefer luxurious and smart things around that can ease their life. And also music is something that have addicted today's generation a lot. So to satisfy such technocrats , our project is been developed.

There are various apps producing Karaoke, so why our project is better ---

- Our app will allow the user to merge his voice along with the Karaoke.
- TheGUI will bemore user friendly.

1.4 SYSTEM REQUIREMENT

The system requirement can be classified into two parts namely the hardware requirement and software requirement.

1.4.1 HARDWARE REQUIREMENT

The supported devices are Android devices running Android OS version 2.1 and above. Also, for absolute experience of the app features, a headset/mic device is recommended. (Provided the Android device has a headset/mic port itself.)

1.4.2 SOFTWARE REQUIREMENT

- The project will work with Android 2.1 and above.
- The product will support the audio compression formats and related codecs forMPEG-1, MPEG-2, MPEG-3, MPEG-4and Windows Media Audio (WMA).
- The system will interact with web application programming interfaces (API) of third party services, such as Sound Cloud, Facebook and iTunes.

CHAPTER 2

LITERATURE SURVEY

2.1 Acoustic-Phonetic Approach

The acoustic-phonetic approach is based on the theory of acoustic phonetics. The theory proposes that there exist finite, distinct phonetic units in spoken language. The phonetic units are characterized by a set of properties that are embedded in the speech signal or its spectrum. Even though the acoustic properties of phonetic units are highly variable, both with the speaker and with the neighbouring phonetic units, it is assumed that the rules governing the variability are straight forward and can be readily learned and applied in practical situations.

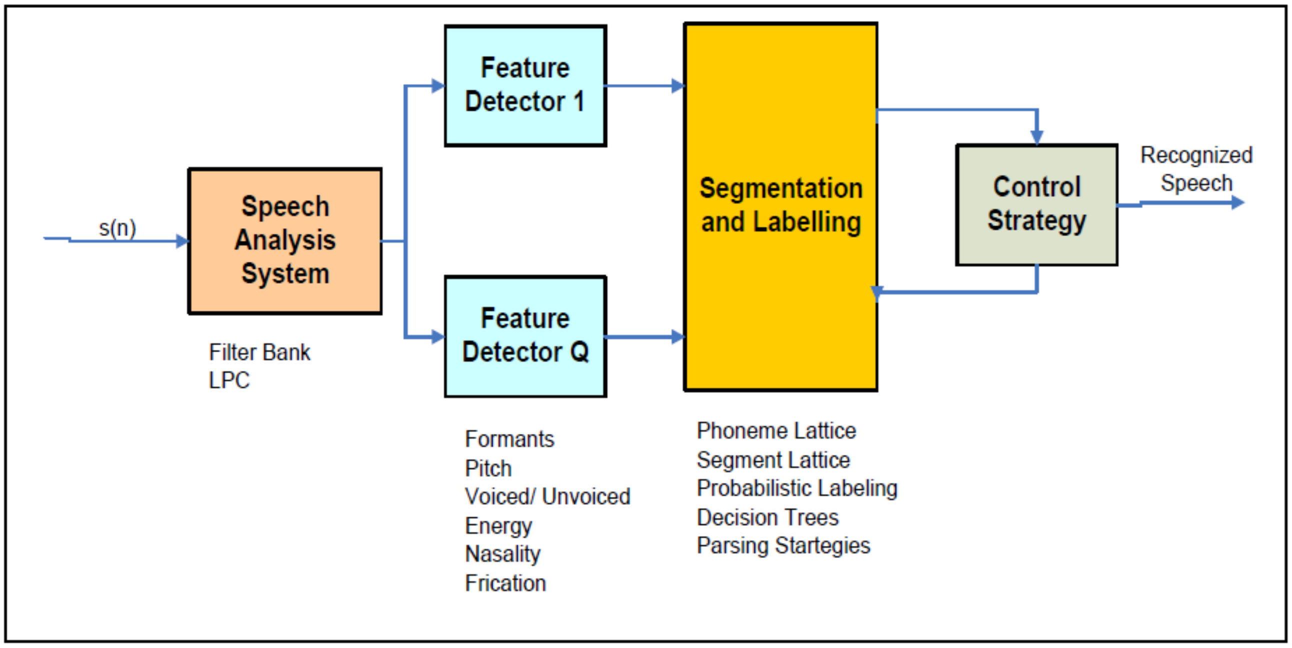


Figure 2.1: Diagram of acoustic phonetic speech recognition system

Figure 2.1 above shows the diagram of acoustic-phonetic speech recognition system. In order to recognize the speech, the first step is speech analysis or feature measurement method, which involves the filter bank processing.

Each spectral represent the characteristics of time-varying speech signal. The most common spectral analysis methods are discrete Fourier Transform (DFT), Linear Predictive Coding (LPC), or Mel Frequency Campestral Coefficients (MFCC) methods. During the feature-detection process, the spectral is converted to a set of features. Among the features are nasality (nasal resonance), frication (random excitation), locations, voiced/unvoiced classification (periodic or periodic excitation), and energy ratios.

2.2 About KARAOKE

Karaoke is the non-verbal form of a song. When a normal song is converted into Karaoke version, the voice of the singers cut off from the song & only the background beats continue to play.
This feature is used to sing along your favorite song, just as if you are the singer for that song & the Band is playing as you sing. Singing / Playing Karaoke is the most entertaining custom at nearly every restaurant in Japan.

Figure2.2: Karaoke Room

2.3 Artificial Intelligence (AI) Approach

A basic karaoke machine consists of a music player, microphone inputs, a means of altering the pitch of the played music, and an audio output. Some low-end machines attempt to provide vocal suppression so that one can feed regular songs into the machine and remove the voice of the original singer; however, this is rarely effective. Most common machines are CD+G, Laser Disc, VCD or DVD players with microphone inputs and an audio mixer built in. CD+G players use a special track called sub code to encode the lyrics and pictures displayed on the screen while other formats natively display both audio and video.

Most karaoke machines have technology that electronically changes the pitch of the music so that amateur singers can choose a key that is appropriate for their vocal range, while maintaining the original tempo of the song. (Old systems which used cassettes changed the pitch by altering playback speed, but none are still on the market, and their commercial use is virtually non-existent.)A popular game using karaoke is to type in a random number and call up a song, which participants attempt to sing.

Figure 2.3: Karaoke Stage Box

In some machines, this game is pre-programmed and may be limited to a genre so that they cannot call up an obscure national anthem that none of the participants can sing. This game has come to be called "Kamikaze Karaoke" or "Karaoke Roulette" in some parts of the United States and Canada.

Many low-end entertainment systems have a karaoke mode that attempts to remove the vocal track from regular audio CDs, using an Out Of Phase Stereo (OOPS) technique. This is done by center channel extraction, which exploits the fact that in most stereo recordings the vocals are in the center. This means that the voice, as part of the music, has equal volume on both stereo channels and no phase difference. To get the quasi-karaoke (mono) track, the left channel of the original audio is subtracted from the right channel.

The Sega Saturn also has a "mute vocals" feature that is based on the same principle and is also able to adjust the pitch of the song to match the singer's vocal range. This crude approach results in the often-poor performance of voice removal. Common effects are hearing the reverberation effects on the voice track (due to stereo reverb on the vocals not being in the center); also, other instruments (snare/bass drum, bass guitar and solo instruments) that happen to be mixed into the center get removed, degrading this approach to hardly more than a gimmick in those devices.

Recent years have seen the development of new techniques based on the Fast Fourier Transform. Although still not perfect, the results are usually much better than the old technique, because the stereo left-right comparison can be done on individual frequencies.

CHAPTER 3

ALGORITHM & FEASIBILITY

3.1 Flow of the Algorithm

Karaoke Design

Description and Priority

This feature will allow the user to listen to the Karaoke (Instrumental) form of a song present on his Android Device. This is the key part of this Android Application.

Stimulus/Response Sequences

1) The user opens the app.

2) He imports a song from his device's memory.

3) He presses the "Make Karaoke" button/functionality provided by the app.

4) He gets the Karaoke version of that song.

Functional Requirements

This feature requires the Android OS version 2.5 or above. Also the user's device must have default audio processing features.

Voice Recorder

Description and Priority

This feature will allow the user to record his own voice on his Android Device through the app. This feature will also store this recording in its own memory buffer / the device's memory.

Stimulus/Response Sequences

1) The user opens the app.

2) He presses the "Record" button/functionality provided by the app.

3) He speaks upon the vocal inlet of the device.

4) The recording is stored.

Functional Requirements

This feature requires the Android device to possess vocal inlet(like a speaker on a mobile) to allow recording function. Also, if possible, the device should have headset/mic port.

Merging two sounds

Description and Priority

This feature will allow the user to merge the Karaoke version of a song with the user's recording to produce a new song, which will have the user's vocals running parallel with theoriginal song's orchestra. This feature will also store this new song in its own memory buffer the device's memory.

Stimulus/Response Sequences

1) The user opens the app.

2) He presses the "Merge" button/functionality provided by the app.

3) The merged song is stored.

Functional Requirements

This feature requires the Android OS version 2.5 or above. Also the user's device must have default audio processing features.

Uploading on Sound Cloud

Description and Priority

This feature will allow the user to upload his own version of the song created by the Merge feature over the internet on Sound Cloud.

Stimulus/Response Sequences

1) The user opens the app.

2) He performs the Merge feature.

3) He presses the "Upload Song" button/functionality provided by the app.

4) The song is uploaded.

Functional Requirements

This feature requires the Android device to possess active internet connection through Wi-Fi having moderate speed.

3.2 FEASIBILITY STUDY

3.2.1 COMPLEXITY CLASS

S= {Q, R, C, Is, Fs, L, A}

Where

S is System,

Q is input,

M is menu driven input command,

R is result that is output,

C is set of all commands,

Is is the Input State,

Fs is the Final State,

A is set of alphabets used

<u>Initial condition</u>:

Q=NULL

R=NULL

this indicates that for no input command there would be no output.

M=NULL

R=NULL

this indicates that for no input menu driven command there would be no output.

Intermediate Condition:

Q=P1

R=R’

this indicates that for a given voice command (P1) either a valid or invalid input(Failure) would be the result.

$$\begin{tikzcd} P1 \arrow[r] & \boxed{} \arrow[r] & \text{R’ € Success} \end{tikzcd}$$

$$\begin{tikzcd} P1 \arrow[r] & \boxed{} \arrow[r] & \text{R’ € Failure} \end{tikzcd}$$

M=P2

R=R’

Thisindicates that for a given menu driven command (P2) either a valid or invalid input (Failure) would be the result.

$$\begin{tikzcd} P2 \arrow[r] & \boxed{} \arrow[r] & \text{R’ € Success} \end{tikzcd}$$

$$\begin{tikzcd} P2 \arrow[r] & \boxed{} \arrow[r] & \text{R’ € Failure} \end{tikzcd}$$

<u>Final Condition</u>:

Q=NULL

R=R1’

where R1’ is the stop command.

M=NULL

R=R1’

where R1’ is the stop command.

<u>V-set of all commands.</u>

V={v1,v2,v3……v10}

Here v1-v10 are pre saved voice command templates.

D=m*13 matrix

The database having pre saved commands.

<u>A-set of alphabet</u>

A={C}

C is to indicate giving Commands

Is : Input State

(P1=Null) ^ Vi

Vi: Indicating start

P1=NULL indicates no input

When the application begins there is no input given

There is only a button present to insert input.

(P2=Null) ^ Mi

Mi: Menu driven indicating start

P1=NULL indicates no input via menu driven commands on screen

When the application begins there is no command given

Fs : Final State

(P1=Null) ^ Vj

Vj: Conversion stop

P1=NULL indicates no input on screen

When the application exits there is no input given.

(P2=Null) ^ Mj

Vj: menu driven indicating stop

P2=NULL indicates no input via menu driven command on screen

Failure State:

P1 ^ R'(NULL)

Invalid input that is the input could not be converted by the application.

3.2.2 TECHNICAL FEASIBILITY

This feasibility study studies a few parameters:

-Is the proposed technology or solution practical ?

-Do we currently possess the necessary technology ?

To implement this project we have used Eclipse IDE for Java Developers Version: Helios Service Release 1 which is freely downloadable from http://eclipse.org/.

We also have used Android Software Development Kit 2.3 which is also freely downloadable from http://developer.android.com/sdk/android-2.3.html.

For database we have used SQLite which is embedded in android in eclipse.

-Do we possess the necessary technical expertise?

Coding for Android requires basic knowledge of Java and C languages which the team possesses. For some typical features of Android ample help is available on the net especially on the website http://developer.android.com/.

3.2.3 ECONOMIC FEASIBILITY

-Cost-benefit analysis

All the software's needed to develop this application are freely downloadable. Hence the development cost is nil.

The hardware cost is the cost of the Android Smart phone on which this software needs to be deployed; the cheapest android phone which can support this application is Spice Mi-270.It is priced for Rs 3500.

3.2.4 SCHEDULE FEASIBILITY

Can the project be completed on scheduled time?

According to the planner we can complete the project by the month of March 2013 whereas the given deadline is of April 2013.All the resources needed are available and the technology expertise has also been acquired.

CHAPTER 4

DATA FLOW DIAGRAM

4.1 DATA FLOW DIAGRAMS

Data Flow Diagrams provided below gives the overall object flow that takes place in “Karaoke System Application”. Where user select karaoke to be played and searches for lyrics. User may select record voice functionality at any instance and stop the same at any arbitrary time. The recorder file is thus stored in memory of the device.

- **DFD :1**

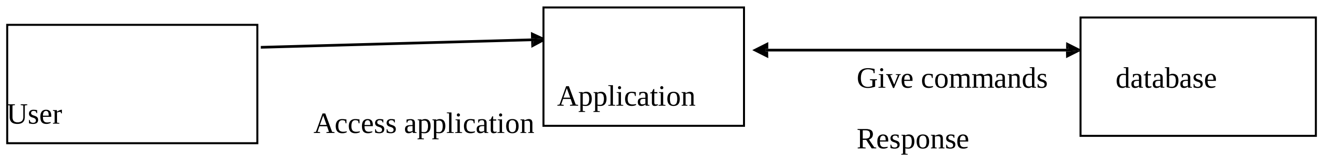


Figure 4.1: DFD :1

- **DFD :2**

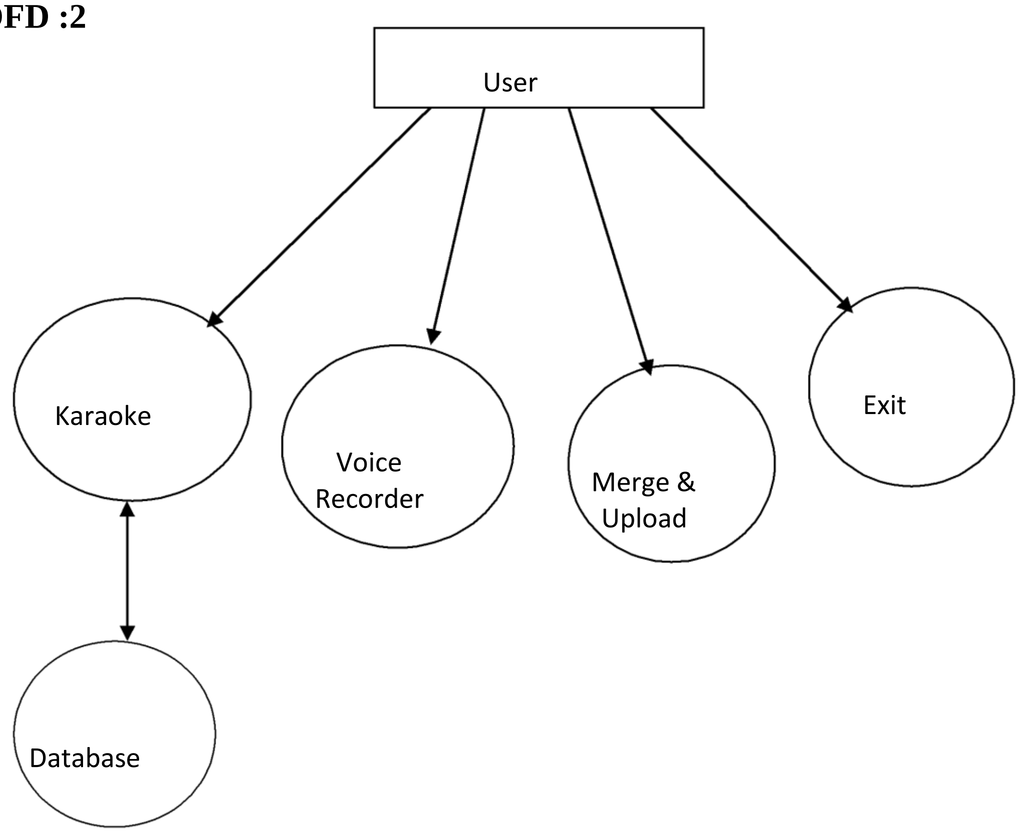


Figure 4.2: DFD :2

- **DFD :3**

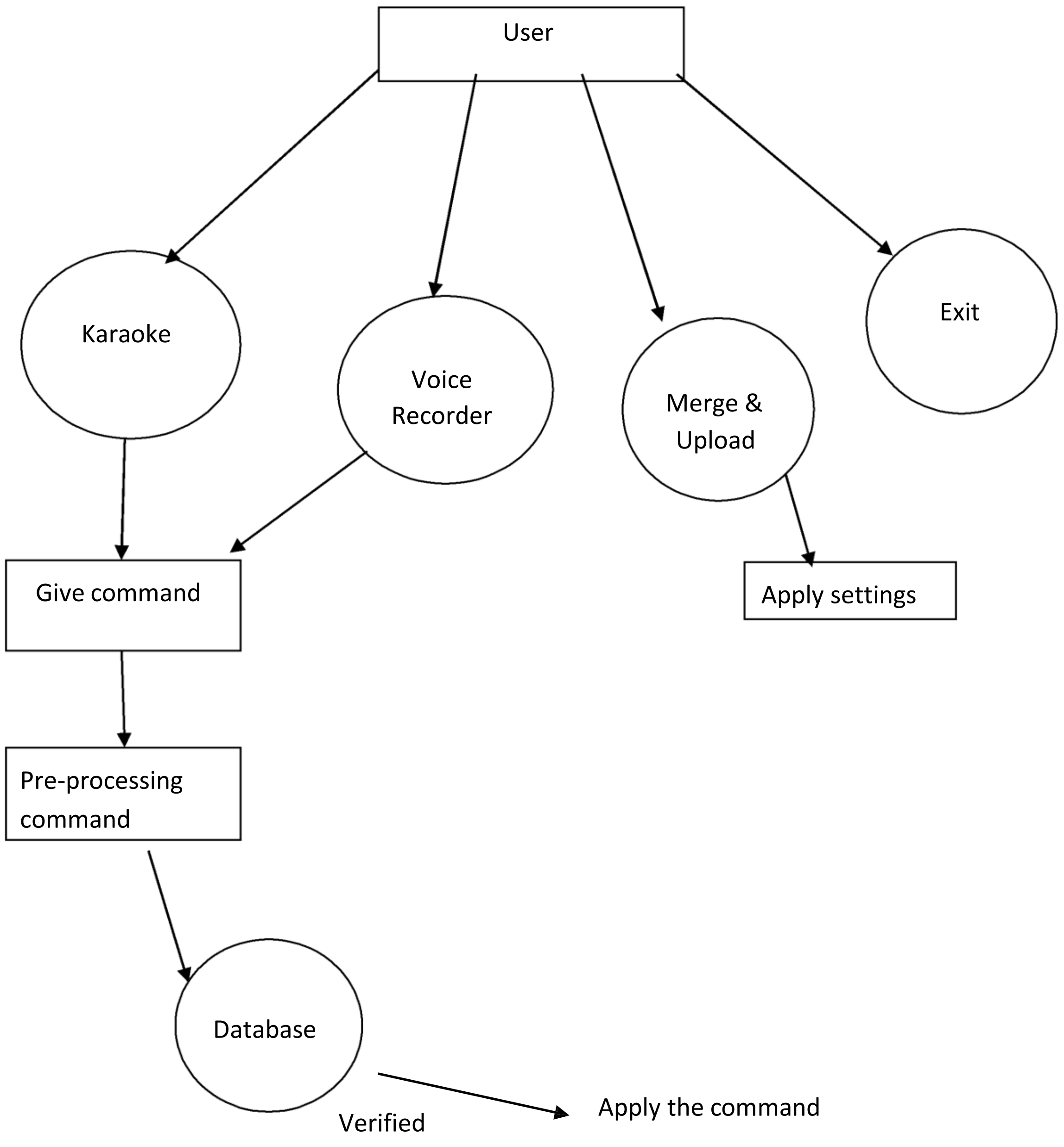


Figure 4.3: DFD :3

CHAPTER 5

UML DIAGRAM

5.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram portrays the different types of users of a system and the various ways that they interact with the system.

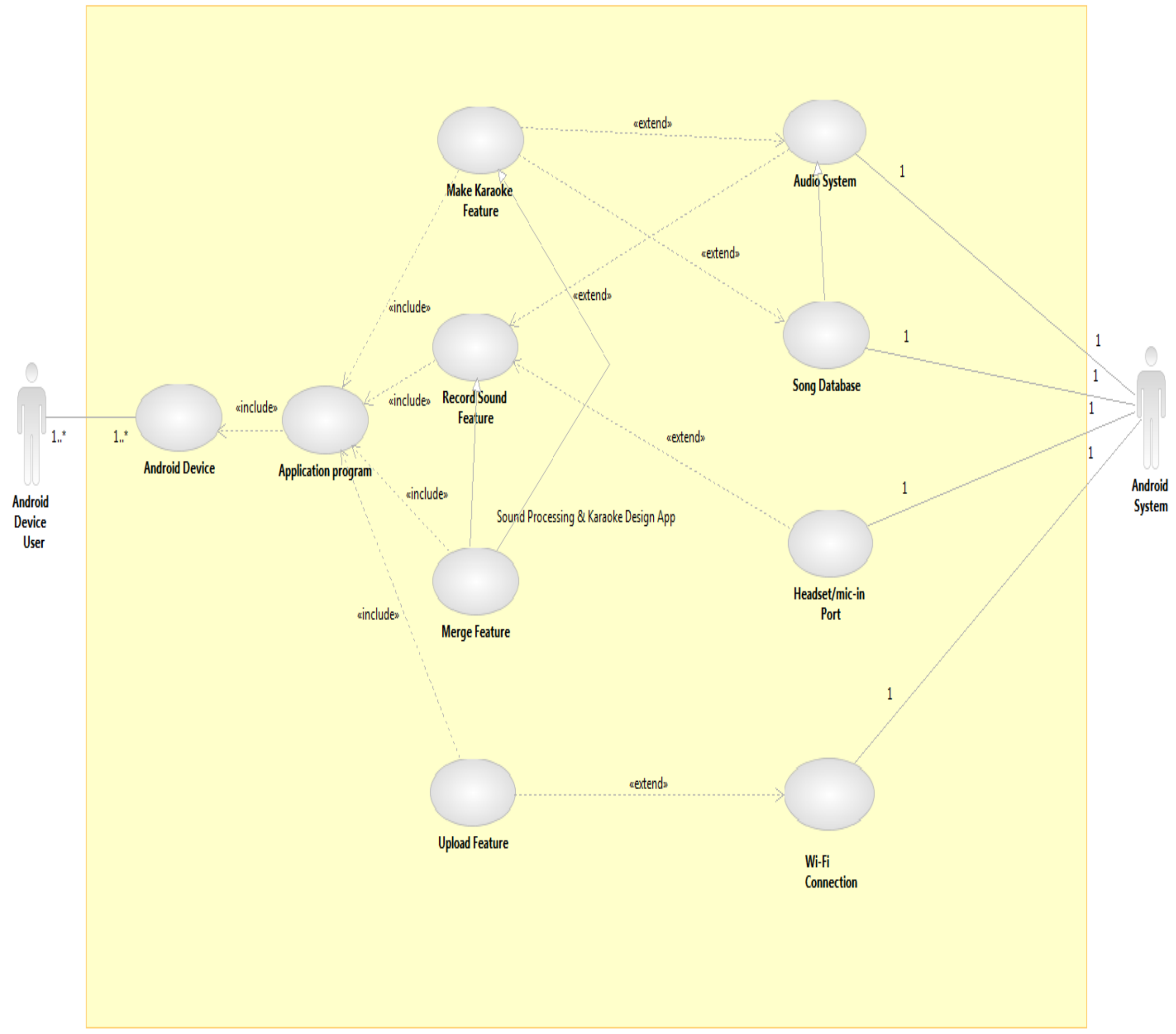


Figure 5.1: Use Case Diagram

5.2 CLASS DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

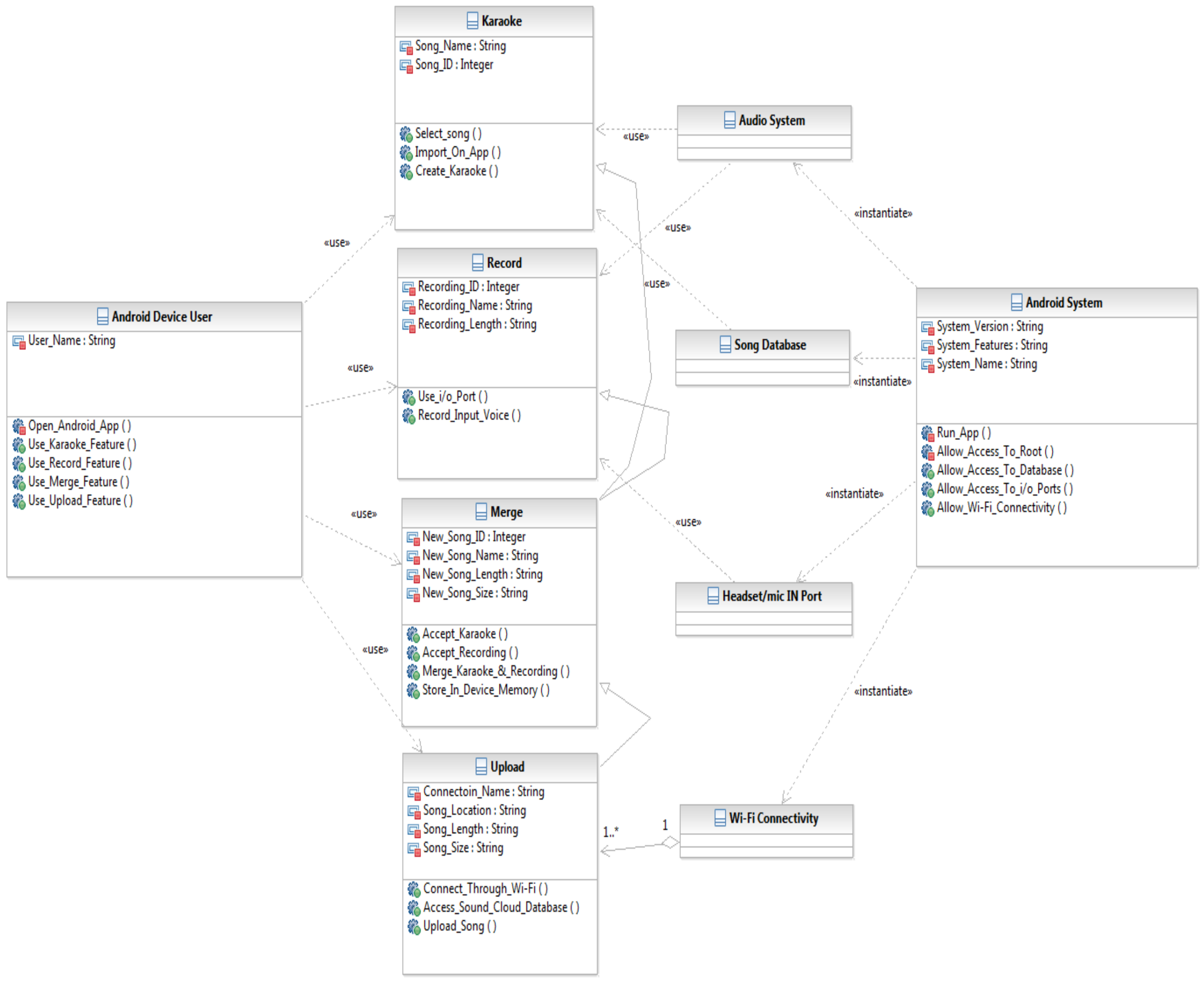


Figure 5.2: Class Diagram

5.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

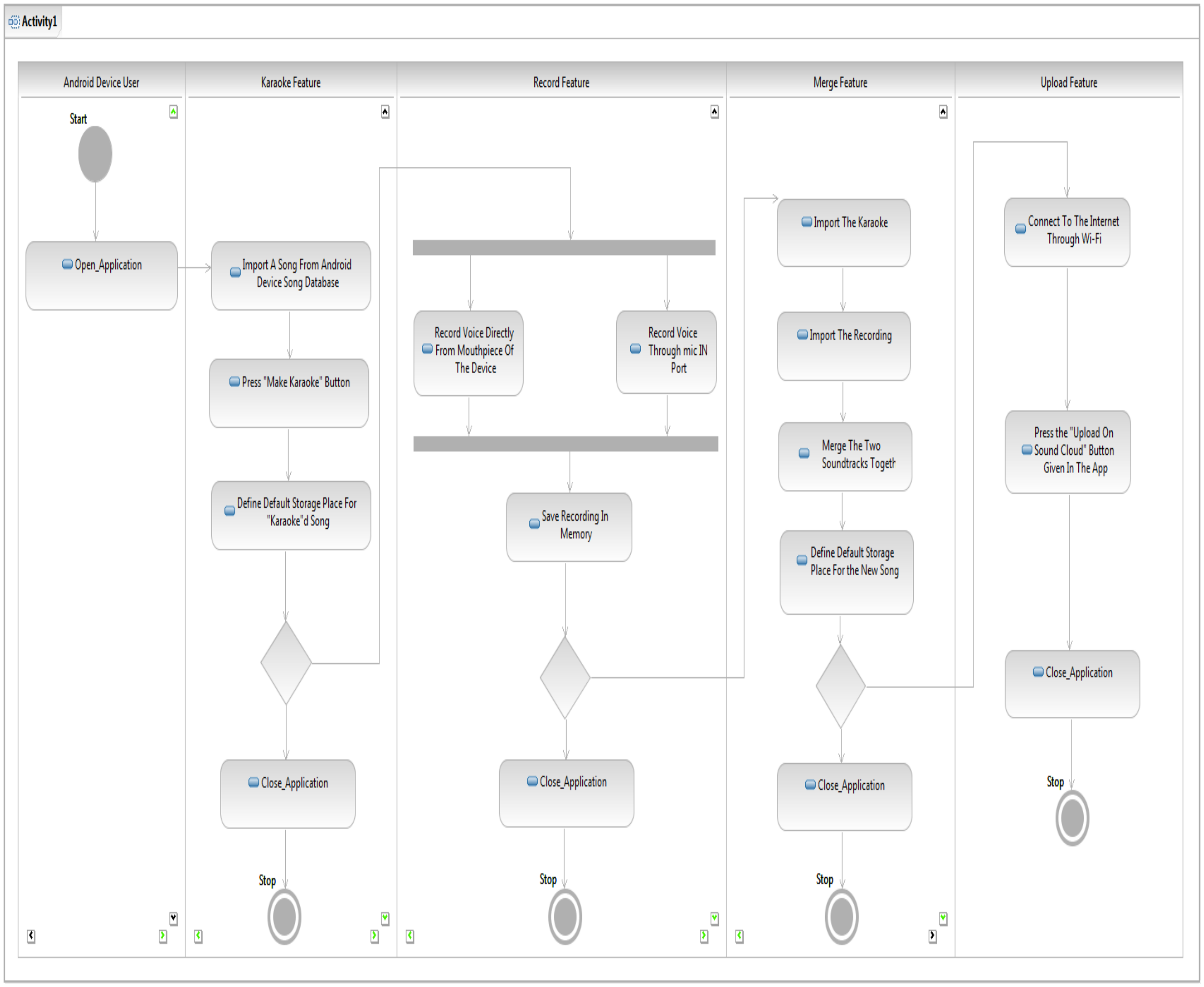

Activity1
Android Device User
Karaoke Feature
Record Feature
Merge Feature
Upload Feature
Start
Open_Application
Import A Song From Android Device Song Database
Press "Make Karaoke" Button
Define Default Storage Place For "Karaoke"d Song
Close_Application
Stop
Record Voice Directly From Mouthpiece Of The Device
Record Voice Through mic IN Port
Save Recording In Memory
Close_Application
Stop
Import The Karaoke
Import The Recording
Merge The Two Soundtracks Togeth
Define Default Storage Place For the New Song
Close_Application
Stop
Connect To The Internet Through Wi-Fi
Press the "Upload On Sound Cloud" Button Given In The App
Close_Application
Stop

Figure 5.3: Activity Diagram

CHAPTER 6

IMPLEMENTATION & TESTING

6.1 IMPLEMENTATION:

- Once the user launches Application from the home screen.
  The Music player is loaded and UI loaded for interaction.

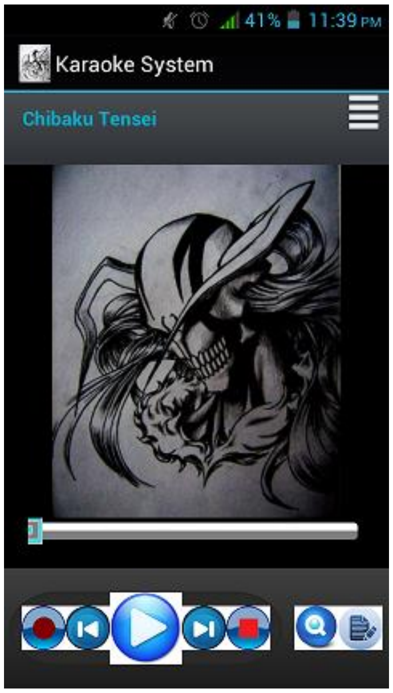
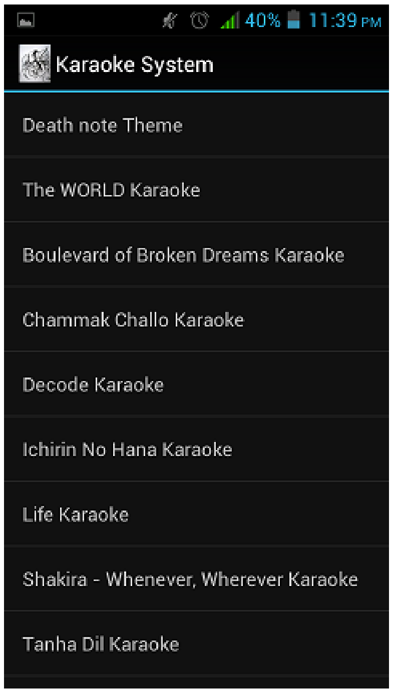


Figure 6.1: Music Player

- Here user can simple hit Play Button play previously loaded song or watch the lyrics , search lyrics record voice (without song playing, too)

- The symbol on right upper corner provides the access to the playlist.
  Playlist is Created (each time) “Dynamically” by scanning available, compatible media from various sources like System Memory or External memory i.e. Extended Card Memory.
- On the bottom right corner is the button to launch the Lyrics viewer that displays text file containing Lyrics for the current song i.e. being played.
  Done by looking up the text file with the same name as that of the song being played.



11:39 PM

Karaoke System

Death note Theme

The WORLD Karaoke

Boulevard of Broken Dreams Karaoke

Chammak Challo Karaoke

Decode Karaoke

Ichirin No Hana Karaoke

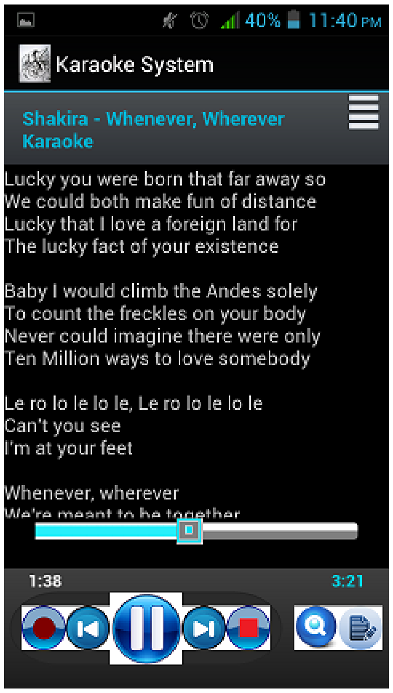
Life Karaoke

Shakira - Whenever, Wherever Karaoke

Tanha Dil Karaoke

Figure 6.2: Playlist View

- If the lyrics for current music is not available the query to find lyrics is fired on Google Search Engine ,and user is taken to browser where he can choose the correct lyrics and even copy same to memory to be stored in Edit lyrics window thus on System memory .



11:40 PM

Karaoke System

Shakira - Whenever, Wherever Karaoke

Lucky you were born that far away so
We could both make fun of distance
Lucky that I love a foreign land for
The lucky fact of your existence

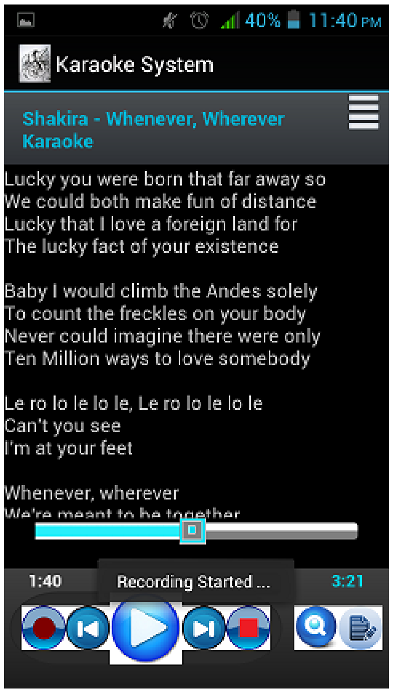
Baby I would climb the Andes solely
To count the freckles on your body
Never could imagine there were only
Ten Million ways to love somebody

Le ro lo le lo le, Le ro lo le lo le
Can't you see
I'm at your feet

Whenever, wherever
We're meant to be together

1:38 3:21

Figure 6.3: Lyrics View



Lucky you were born that far away so
We could both make fun of distance
Lucky that I love a foreign land for
The lucky fact of your existence

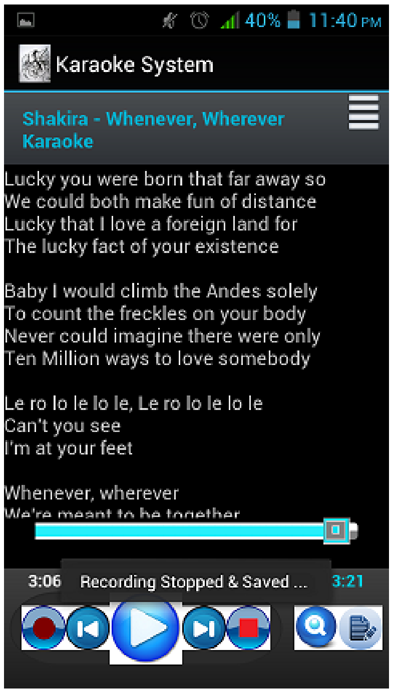
Baby I would climb the Andes solely
To count the freckles on your body
Never could imagine there were only
Ten Million ways to love somebody

Le ro lo le lo le, Le ro lo le lo le
Can't you see
I'm at your feet

Whenever, wherever
We're meant to be together

Figure 6.4: Start Recording View

- User may touch the Record Button on left bottom corner to start recording the voice any time during system interaction.
- The Stop Button is provided to halt the recording and store the recorded voice while into the system memory under the random name generated by the Device.



Lucky you were born that far away so
We could both make fun of distance
Lucky that I love a foreign land for
The lucky fact of your existence

Baby I would climb the Andes solely
To count the freckles on your body
Never could imagine there were only
Ten Million ways to love somebody

Le ro lo le lo le, Le ro lo le lo le
Can't you see
I'm at your feet

Whenever, wherever
We're meant to be together

Figure 6.5: Stop Recording View

6.2 TESTING:

6.2.1 BACKGROUND

Testing focuses primarily on the evaluation or assessment of product quality realized through a number of core practices:

1. Finding and documenting defects in software quality.
2. Advising about perceived software quality.
3. Proving the validity of the assumption made in design and requirement specification through concrete demonstration.
4. Validating the software functions as designed.
5. Validating that the requirements have been implemented appropriately.

Test challenges the assumptions and the risks. Testing of software is difficult. The different ways in which a program can behave are indefinable. A well-conceived methodology and use of state-of-art tolls can help improve the productivity and effectiveness of software testing.

1. The requirement discipline captures requirements for the software product and those requirements are one of the primary inputs for identifying what tests are to be performed.
2. The analysis and design discipline determines the appropriate design for the software product, this is another important input for identifying what tests are to be performed.
3. The implementation discipline produces builds of software product that are validated by the test discipline. Within iteration multiple builds will be tested.
4. The environment discipline develops and maintains supporting artifacts that are used during test, such as guidelines and test environment.

6.2.1 TESTING WITH DIFFERENT TEST CASES- MANUAL TESTING

- **UNIT TESTING**

During Unit Testing phase, the system is tested while it is developed. Here all the options of the system are validated. During the first phase of this testing person tests the system by entering the valid data, or by performing the appropriate function which the system requests for. This phase of testing is done to verify whether the system performs all requested functions.

Testing for Player:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC1-A	Test Player	New, Old, Diff files exist on system.	File should be Played.	After searching files. Player plays file.	Pass

Table 6.1: Unit Testing of Player

Testing for Lyrics:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC2-A	Test Lyrics	New, Old, Diff files exist on system.	File should be opened.	After searching files. Player opens file.	Pass

Table 6.2: Unit Testing of Lyrics

Testing for Record:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC3-A	Test Record	New, Old, Diff files exist on system.	Audio file should be created.	Audio File Created.	Pass

Table 6.3: Unit Testing of Record

- **INTEGRATION TESTING**

Integration testing is a phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input, the modules that have been unit tested, and groups them in large aggregate. It then applies test defined in an integration test plan to those aggregates, and delivers its output as an integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assembles (or group of units), are exercised through their interface using black box testing, success and error cases being simulated via appropriate parameter and design inputs. Simulated uses of shared data area and inter-crosses communication is tested and individual sub-systems are exercised through their input interface. Test cases are constructed to test that all components within assemblers interact correctly, for example, across procedure calls or process activation and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblers are added to a verified base which is then used to support the integration testing of further assemblers.

Testing for creating Audio File:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC4-A	Start Recording	Click on Record Button	Recording of Audio should start	Recording of audio starts at target path	Pass

Table 6.4: Integration testing for creating Audio File

Testing for Lyrics Module:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC5-A	Lyrics	Click on Lyric Button	Lyrics should Display	Lyrics displayed	Pass
TC5-B	Edit	Click on Edit	Window should open to edit	Window to edit opens	Pass

Table 6.5: Integration testing for Lyrics Module

Testing for displaying Playlist:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC6-A	Display Media Playlist	Click Playlist Icon in Main Frame	All Media should be displayed	All media are displayed in list form	Pass

Table 6.6: Integration testing for Playlist

Testing for Forward/Previous:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC7-A	Start Forward/Previous Media	Click either buttons	Process should start	Process starts	Pass

Table 6.7: Integration testing for Forward/Previous

- **BLACK BOX TESTING**

Black box testing is a method of software testing that tests the functionality of application as opposed to its internal structures or workings. Specific knowledge of the application's code/internal structure and programming knowledge in general are not required. The tester is only aware of what the software is supposed to do, but not how, i.e. when he enters a certain input he get certain output without being aware of how the output was produced in the 1st place. Test cases are built around specification and requirements, i.e. what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of object's internal structure.

Testing for creating Karaoke Audio:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC8-A	Play karaoke and click on record to create audio file	Lyrics, Karaoked Media, User Vocals, Target Path	Audio media should be created containing base Karaoke + Voacals.	Audio file is created at target path as envisioned	Pass

Table 6.8: Black Box testing for creating Karaoke Audio

Testing for Searching Media Files:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC10-A	Search for a particular media	Enter all or some of required media	If supported media exists it is returned	Searched supported media	Pass

Table 6.9: Black Box testing for searching Media Files

Testing for Editing Lyrics:

Test Case ID	Step Description	Input Data	Expected Result	Actual Result	Pass/Fail
TC10-B	Edit Lyrics	Select Lyrics of currently Playing media	Lyrics opens Edit Window	Lyrics opened in Editing Window	Pass

Table 6.10: Black Box testing for Editing Lyrics

- ## MONKEY TESTING

A monkey test is a unit test that runs with no specific test cases in mind. The monkey in these cases is a producer of an input. For example, a monkey tester can enter random strings into text boxes to ensure handling of all possible users input or provide garbage file to check for loading routines that have blind faith in their data. The test monkey is technically known to conduct stochastic testing, which is in the category of black box testing. The name 'monkey' comes from the adage that "A thousand monkeys at a thousand typewriters will eventually type out entire works of Shakespeare".

After application was completely developed we carried out our monkey testing or random testing on it. In this we kept on giving random input to see if the application responded correctly. For example, we tried to select multiple folders as source but it took only one folder at a time which was the expected output. We also tried to create a VM with existing name but it gave an error as was expected.

- **PERFORMANCE TESTING**

For performance testing we test the application with various people. The reason to do this was that all of them had different perceptions and methods. We even tested with different virtualization software and found that application is pretty reliable. The Binary diff algorithm is a good one as it only takes delta and it improved the efficiency significantly.

- **BETA TESTING**

After the application was used in various tests, the program was given to users for operating without any presence of developer/tester. It was observed after getting the result that program was simple, efficient and user-friendly to use.

CHAPTER 7

RESULT & DISCUSSION

7.1 Result:

- **Existing System Analysis:**

  The Android market store currently consists of basic applications that provide unique features for particular applications that belong to their respective domain.
  The existing system provides following functionalities individually:

  - Player System: Allows Playback and basic Media Player Functionality.
  - Lyrics System: Allows Lyrics viewing and editing functionality.
  - Recording System: Allows Voice recording Functionality.

  The System that can provide these features combined under a single application is not available. Hence the need for our application…

- **Designed System Analysis:**

  Whatever the objective of our project was mentioned in our abstract was followed in following sequence to get the specified result. The currently developed system is able to provide the combined functionalities of all the existing systems mentioned above.

  The different steps followed are as follows:

  - To Record Vocals along with Karaoke Media and store.
  - The Basic Player can perform all the possible operations.
  - The Lyrics available on the system is displayed on text viewer.
  - Edit button can be used add extra information on lyrics file
  - If the Lyrics for the current media is unavailable then user is able to search on web using browser.
  - The lyrics searched can be copied to the edit lyrics window where its stored on the system.

7.2 Discussion:

Future Scope

- Functionality to actually create Karaoke on the system.
- More accurate final recorded audio file.
- Noise removal, hiss removal.
- Cloud Sharing features using existing cloud accounts and social networking technologies .

CHAPTER 8

CONCLUSION

8.1 Conclusion:

The Sound processing & Karaoke Design app is a unique perspective of entertainment and a feature-rich experience for music lovers. With the additional record-merge-upload features, the app proves to be one of a kind in the field of android application development.

This project is to create an application that will take a song present on an Android Device and convert it into Karaoke. A karaoke of a song is the instrumental form of that song, in which the vocal track sung by the singer is suppressed to lower dB frequency. So, only the background beat and the orchestra remain audible.

The app will also allow the device user to record his own voice, so that he can sing and record the song, sung along-side the Karaoke. Furthermore, the app will merge the Karaoke and the user's voice to produce the user's own version of the son

CHAPTER 9

REFERENCES

9.1 REFERENCES:

[1] Speech and Audio Processing, IEEE Transactions on **Volume**: 13 **Issue:** 6 Publication Year: 2005 **ISSN:** 1063-6676 Published by**, IEEE Signal Processing Society**.
Persistent Link: http://ieeexplore.ieee.org/servlet/opac?punumber=89

[2] Designing Sound: By Andy Farnell: Published: 28 Sep 2010

[3] Sound Unbound: By DJ Spooky That Subliminal Kid: Published: 31 May 2008

[4] Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference.

[5] http://en.wikipedia.org/wiki/Karaoke

[6] http://developer.android.com/sdk/eclipse-adt.html

[7] http://en.wikipedia.org/wiki/Android_software_development

[8] http://developer.android.com/reference/android/database/sqlite/Package-summary.html

CHAPTER 10

APPENDIX

10.1 Pseudo Code

- **Music Player:**

```
        /**
         * Play button click event
         * plays a song and changes button to pause image
         * pauses a song and changes button to play image
         * */
        btnPlay.setOnClickListener(new View.OnClickListener() {

              @Override
              public void onClick(View arg0) {
                    // check for already playing
                    if(mp.isPlaying()){
                          if(mp!=null){
                                mp.pause();
                                // Changing button image to play button

  btnPlay.setImageResource(R.drawable.btn_play);
                          }
                    }else{
                          // Resume song
                          if(mp!=null){
                                mp.start();
                                // Changing button image to pause button

  btnPlay.setImageResource(R.drawable.btn_pause);
                          }
                    }

              }
        });

        /**
         * Forward button click event
         * Forwards song specified seconds
         * */
  /*    btnForward.setOnClickListener(new View.OnClickListener() {

              @Override
              public void onClick(View arg0) {
                    // get current song position
                    int currentPosition = mp.getCurrentPosition();
                    // check if seekForward time is lesser than song
duration
                    if(currentPosition + seekForwardTime <=
mp.getDuration()){
                          // forward song
                          mp.seekTo(currentPosition + seekForwardTime);
                    }else{
                          // forward to end position
                          mp.seekTo(mp.getDuration());
                    }
}
            });*/

            /**
             * Backward button click event
             * Backward song to specified seconds
             * */
            /*btnBackward.setOnClickListener(new View.OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        // get current song position
                        int currentPosition = mp.getCurrentPosition();
                        // check if seekBackward time is greater than 0 sec
                        if(currentPosition - seekBackwardTime >= 0){
                              // forward song
                              mp.seekTo(currentPosition - seekBackwardTime);
                        }else{
                              // backward to starting position
                              mp.seekTo(0);
                        }

                  }
            });
            */
```

- **Lyrics Module:**

```
/**
       * Lyrics Find Button click event
       */

            btnLrcFind.setOnClickListener(new View.OnClickListener() {

                  @Override
                  public void onClick(View v) {

                  String songTitle1 =
songsList.get(currentSongIndex).get("songTitle");
                        File f1= new File(songTitle1);
                        openURL("http://www.google.com/m?q=" +

      URLEncoder.encode(FileUtils.fileNameWithoutExtension(f1) +
                                                " " +
getString(R.string.search_lyrics)));

                  }
```

```
			private void openURL(String url) {
				try {
					Intent intent = new Intent();
					intent.setAction(Intent.ACTION_VIEW);
					intent.setData(Uri.parse(url));
					startActivity(intent);
				} catch(Exception e) {
					e.printStackTrace();
				}
			}

		});

		/**
		 * Lyrics Edit Button click event
		 */

		btnLrcEdit.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {

				// Copy text to edit lyrics.

	songLyricsEditText.setText(songLyricsTextView.getText());
				// Hide view lyrics and edit button.
				btnLrcEdit.setVisibility(View.GONE);
				songLyricsTextView.setVisibility(View.GONE);
				// Show edit lyrics and save button.
				btnLrcSave.setVisibility(View.VISIBLE);
				songLyricsEditText.setVisibility(View.VISIBLE);

			}
		});

		/**
		 * Lyrics Save Button click event
		 */

		btnLrcSave.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {

				// Copy text to edit lyrics.

	songLyricsTextView.setText(songLyricsEditText.getText());
				// Save lyrics to file.
				String songTitle2 = 
songsList.get(currentSongIndex).get("songTitle");
				File f2= new File(songTitle2);
				File name1=new 
File(FileUtils.fileNameWithoutExtension(f2));
				lyricsFilePath=new 
String(SongsManager.MEDIA_PATH+"/"+name1+".txt");
				try {
```

```
FileUtils.filePutContents(lyricsFilePath,
songLyricsEditText.getText().toString());
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
                        // Hide edit lyrics and save button.
                        btnLrcSave.setVisibility(View.GONE);
                        songLyricsEditText.setVisibility(View.GONE);
                        // Show view lyrics and edit button.
                        btnLrcEdit.setVisibility(View.VISIBLE);
                        songLyricsTextView.setVisibility(View.VISIBLE);

                  }
            });
```

- **Record Module:**

```
/**
             * Record Button click event
             */

            btnRecord.setOnClickListener(new View.OnClickListener() {


                  @Override
                  public void onClick(View v) {

                        btnRecord.setEnabled(false);
                        btnRecordStop.setEnabled(true);
                        btnRecord.setClickable(false);
                        btnRecord.setVisibility(1);

                         File sampleDir =
Environment.getExternalStorageDirectory();
                          try {
                              audiofile = File.createTempFile("sound", ".3gp",
sampleDir);
                          } catch (IOException e) {
                              Log.e(TAG, "sdcard access error");
                              return;
                          }
                          recorder = new MediaRecorder();

recorder.setAudioSource(MediaRecorder.AudioSource.MIC);

recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);

recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
                          recorder.setOutputFile(audiofile.getAbsolutePath());
                          try {
                                    recorder.prepare();
                              } catch (IllegalStateException e) {
                                    e.printStackTrace();
                              } catch (IOException e) {
                                    e.printStackTrace();
                  }
```

```
 recorder.start();
                         Toast.makeText(getApplicationContext(), "Recording
Started, Tendou....", Toast.LENGTH_LONG).show();

                 }
            });

            /**
             * Record_Stop Button click event
             */


            btnRecordStop.setOnClickListener(new View.OnClickListener() {

                  @Override
                  public void onClick(View v) {

                        btnRecord.setEnabled(true);
                        btnRecordStop.setEnabled(false);
                        btnRecordStop.setClickable(false);
                        btnRecordStop.setVisibility(1);

                         recorder.stop();
                       recorder.release();

                       ContentValues values = new ContentValues(4);
                          long current = System.currentTimeMillis();
                          values.put(MediaStore.Audio.Media.TITLE, "audio" +
audiofile.getName());
                          values.put(MediaStore.Audio.Media.DATE_ADDED, (int)
(current / 1000));
                          values.put(MediaStore.Audio.Media.MIME_TYPE,
"audio/3gpp");
                          values.put(MediaStore.Audio.Media.DATA,
audiofile.getAbsolutePath());
                          ContentResolver contentResolver =
getContentResolver();

                          Uri base =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
                          Uri newUri = contentResolver.insert(base, values);

                          sendBroadcast(new
Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
                          Toast.makeText(getApplicationContext(), "Recording
Saved, Tendou....", Toast.LENGTH_LONG).show();

                  }

            });
```

10.2 Paper:

Sound Processing & Karaoke Design Application

ISHAN VASANT GOKHALE **YOGESH R. ISAWE** **SAURABH LAKADE**
ishan.zangetsu@gmail.com yisawe@yahoo.com

Sinhagad Institute of Technology and Science, Narhe, Pune 411041, INDIA
University of Pune, INDIA

***Abstract*—**
Nowadays in the era of mobile devices, the world needs smart applications that are easy to use and very effective in the end. Android is the one of the popular mobile application platform these days which we have used. The main objective is to create an Android Application for Sound Processing and Karaoke Design, which will offer a very feature-rich experience. This project is to create an application that will take a song present on an Android Device and convert it into Karaoke. A karaoke of a song is the instrumental form of that song, in which the vocal track sung by the singer is suppressed to lower dB frequency. So, only the background beats and the orchestras remain audible. The app will also allow the device user to record his own voice, so that he can sing and record the song, sung along-side the Karaoke. Furthermore, the app will merge the Karaoke and the user's voice to produce the user's own version of the song. The user will be able to upload this song instantly on Sound Cloud through Wi-Fi connection.

Keywords-Android, Karaoke, Record, Merge, Upload

I. INTRODUCTION

This project is to create an application that will take a song present on an Android Device and convert it into Karaoke. A karaoke of a song is the instrumental form of that song, in which the vocal track sung by the singer is suppressed to lower dB frequency. So, only the background beat and the orchestra remain audible. The app will also allow the device user to record his own voice, so that he can sing and record the song, sung along-side the Karaoke. Furthermore, the app will merge the Karaoke and the user's voice to produce the user's own version of the song. The user will be able to upload this song instantly on Sound Cloud through Wi-Fi connection.

The software being used for development is the Android development kit which is a plug-in to the Eclipse IDE.

The Sound Processing / Karaoke Design Application can be operated on android version 2.3 and above. It provides quite good features and excellent pass-time experience. It also provides very simple GUI (Graphical User Interface) which decreases the complexity of the application to use it.

This application, as mentioned earlier, also provides the facility of real time uploading through the Android device.Nowadays each Android device comes with a secure Wi-Fi connection that provides instant internet establishment with possible 3-G facility. This feature makes the application more unique and entertaining.

II. Flow of the Application

On entering the application the welcome screen would be displayed. This screen would simply have the application name and logo. Then user can perform the features as being shown in the UML model-

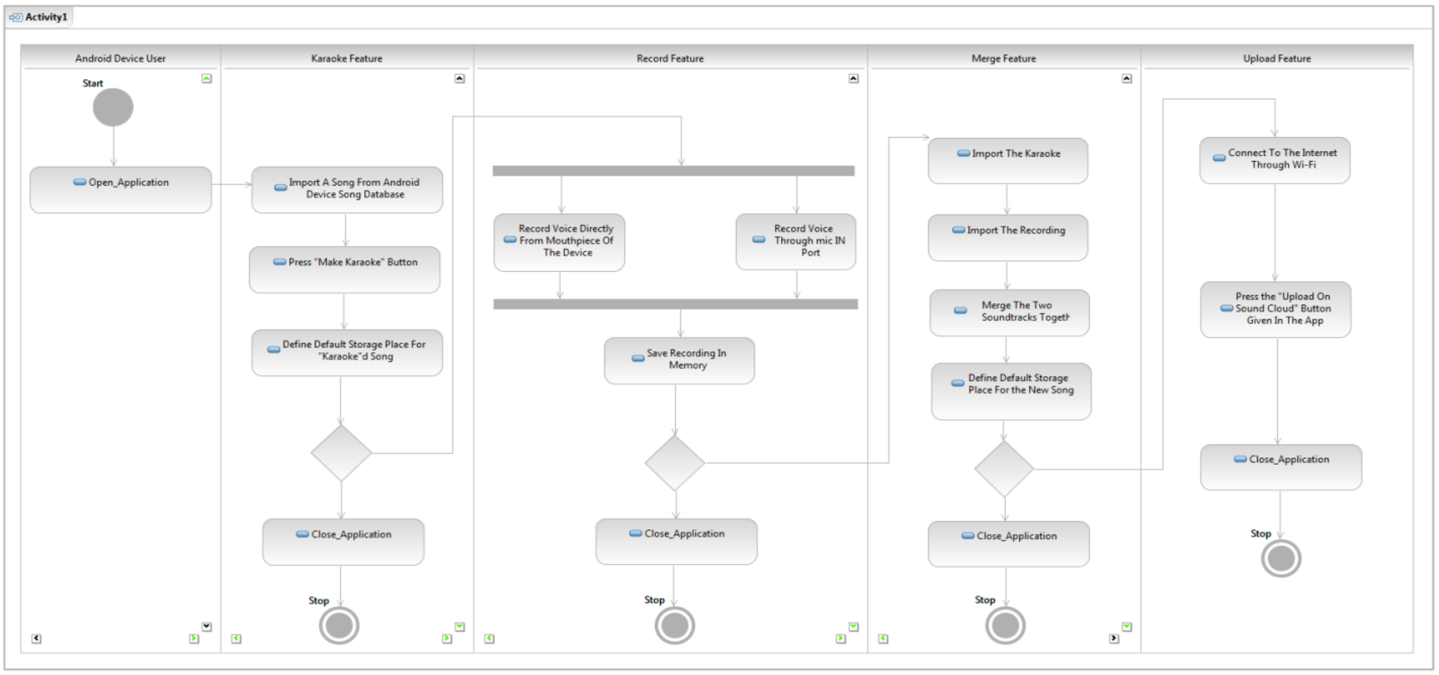

Activity1
Android Device User
Karaoke Feature
Record Feature
Merge Feature
Upload Feature
Start
Open_Application
Import A Song From Android Device Song Database
Press "Make Karaoke" Button
Define Default Storage Place For "Karaoke"d Song
Close_Application
Stop
Record Voice Directly From Mouthpiece Of The Device
Record Voice Through mic IN Port
Save Recording In Memory
Close_Application
Stop
Import The Karaoke
Import The Recording
Merge The Two Soundtracks Togeth
Define Default Storage Place For the New Song
Close_Application
Stop
Connect To The Internet Through Wi-Fi
Press the "Upload On Sound Cloud" Button Given In The App
Close_Application
Stop

FIGURE: Flow of Sound Processing & Karaoke Design Application

III. FEASIBILITY

<u>COMPLEX CLASSES</u> :

S= {Q, R, C, Is, Fs, L, A}

Where
S is System
Q is input
M is menu driven input command
R is result that is output
C is set of all commands.
Is is the Input State
Fs is the Final State
A is set of alphabets used

<u>Initial condition</u>:

Q=NULL
R=NULL

this indicates that for no input command there would be no output.

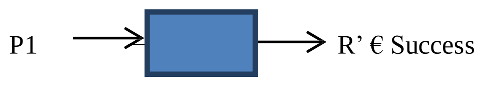
M=NULL
R=NULL

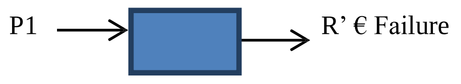
this indicates that for no input menu driven command there would be no output.

<u>Intermediate Condition</u>:

Q=P1
R=R’

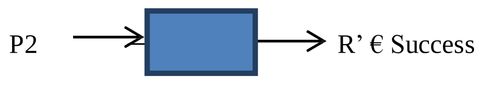
this indicates that for a given voice command (P1) either a valid or invalid input(Failure) would be the result.

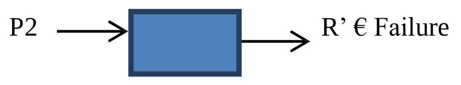




M=P2
R=R’

this indicates that for a given command (P1) either a valid or invalid input(Failure) would be the result.





Q=NULL
R=R1’

where R1’ is the stop command.
M=NULL
R=R1’

where R1’ is the stop command.

<u>V-set of all commands</u>

V={v1,v2,v3……v10}

Here v1-v10 are pre saved voice command templates.

<u>D=m*13 matrix</u>

The database having pre saved commands.

A-set of alphabets
A={C}
C is to indicate giving Commands

<u>Is : Input State</u>

(P1=Null) ^ Vi
Vi: Indicating start
P1=NULL indicates no input

When the application begins there is no input given
There is only a button present to insert input.

(P2=Null) ^ Mi
Mi: Menu driven indicating start
P1=NULL indicates no input via menu driven
commands on screen.

When the application begins there is no command given

<u>Fs : Final State</u>

(P1=Null) ^ Vj
Vj: Conversion stop
P1=NULL indicates no input on screen

When the application exits there is no input given.

(P2=Null) ^ Mj
Vj: menu driven indicating stop
P2=NULL indicates no input via menu driven
command on screen

<u>Failure State</u>:

P1 ^ R’(NULL)

Invalid input that is the input could not be converted by the application.

IV. FUTURE WORK

The future enhancements in the application include processing & manipulating other sound elements like Amplitude, Phase etc. This app can be uploaded to Google play store for download.

V. CONCLUSION

The Sound processing & Karaoke Design app is an unique perspective of entertainment and a feature-rich experience for music lovers.
With the additional record-merge-upload features, the app proves to be one of a kind in the field of android application development.

ACKNOWLEDGMENT

We would like to offer our sincere thanks to our guide Professor Mr.PrashantDhotre, Sinhagad Institute of Technology and Science, Narhe.
We would also like to thank Cue-logic Technology Pvt Ltd. for their sponsorship and guidance.

REFERENCES

- *Google Android Homepage:*

http://code.google.com/android/

- *Google Android SDK:*

http://code.google.com/android/download.html

- *Karaoke Wikipedia*

http://en.wikipedia.org/wiki/Karaoke

- *IEEE Reference*

Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference

10.3 Certificates:

IFRSA's INTERNATIONAL JOURNAL OF COMPUTING (IIJC)

www.ifrsa.org

ISSN (Print):2231:2153, ISSN (Online):2230:9039

(A quarterly refereed Journal of International Forum Of Researchers Students and Academician)

Published by: International Forum of Researchers Students and Academician (IFRSA)

Dear IshanVasantGokhale
Paper Title: **Sound Processing & Karaoke Design Application**

Herewith, we are pleased to inform you that your draft paper mentioned above has been accepted by the IFRSA's International Journal Of Computing (IIJC), and the revised final paper (in . doc and .pdf) after revision and formatting will be published in the IIJC. Kindly note that this is subject to receipt of the **Final Paper, filled Copyright Form, Filled Information Form and Scanned Payment** till October 23, 2012.

Your paper will be charged for publishing, and the detailed payment information can be found in attached Information Form.

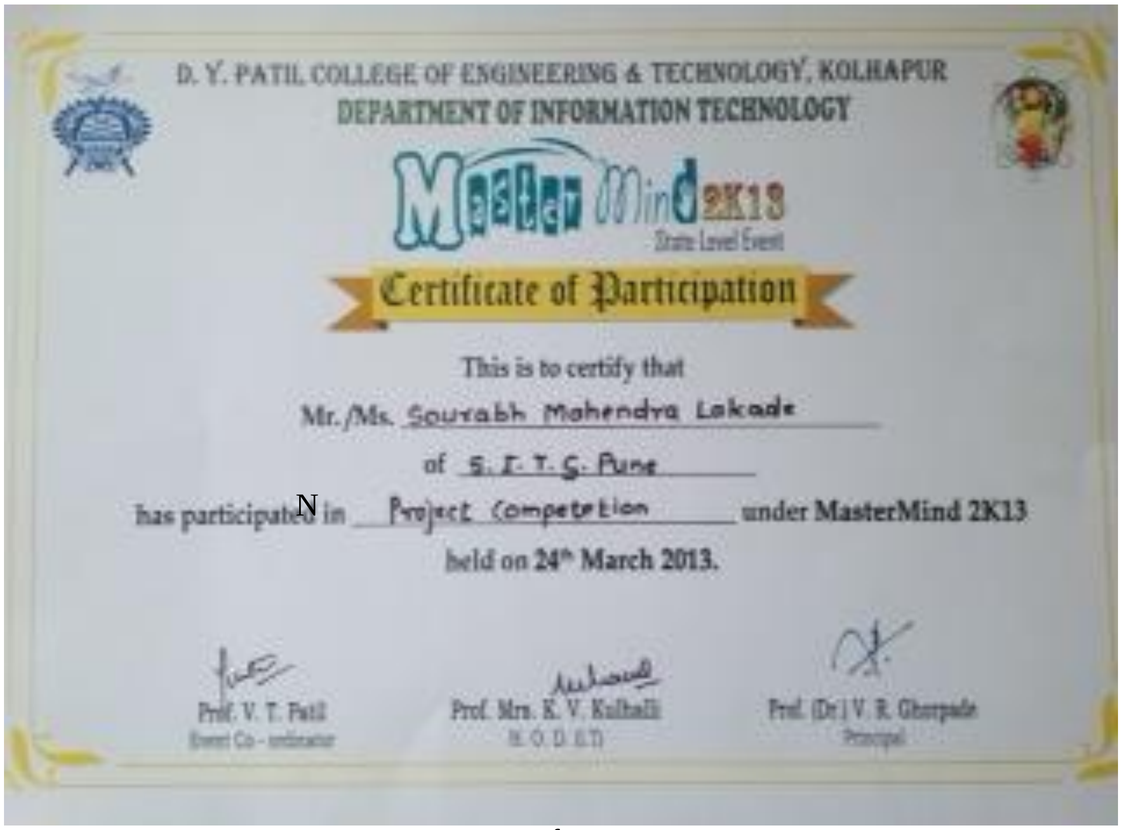
Enclosures for Registration and Paper Composition:
1. Scanned copyright form (download from ifrsa.org link :instructionfor author
2. Complete Information form (Attached with mail)
3. Final Paper (.doc and .pdf) format

Paper Publication in Proceedings:
1. All the registered papers in the journal are available freely with online full-text content and permanent worldwide web.
2. **Printed copy of journal will also be provided to first author**.

Yours sincerely,
IIJC Editorial Boards
http://www.ifrsa.org

- For Any query you can contact our publisher DharaEdutech on Mobile : 09268659101

- National Paper Presentation Conference



D. Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY, KOLHAPUR
DEPARTMENT OF INFORMATION TECHNOLOGY
MasterMind 2K13
State Level Event
Certificate of Participation

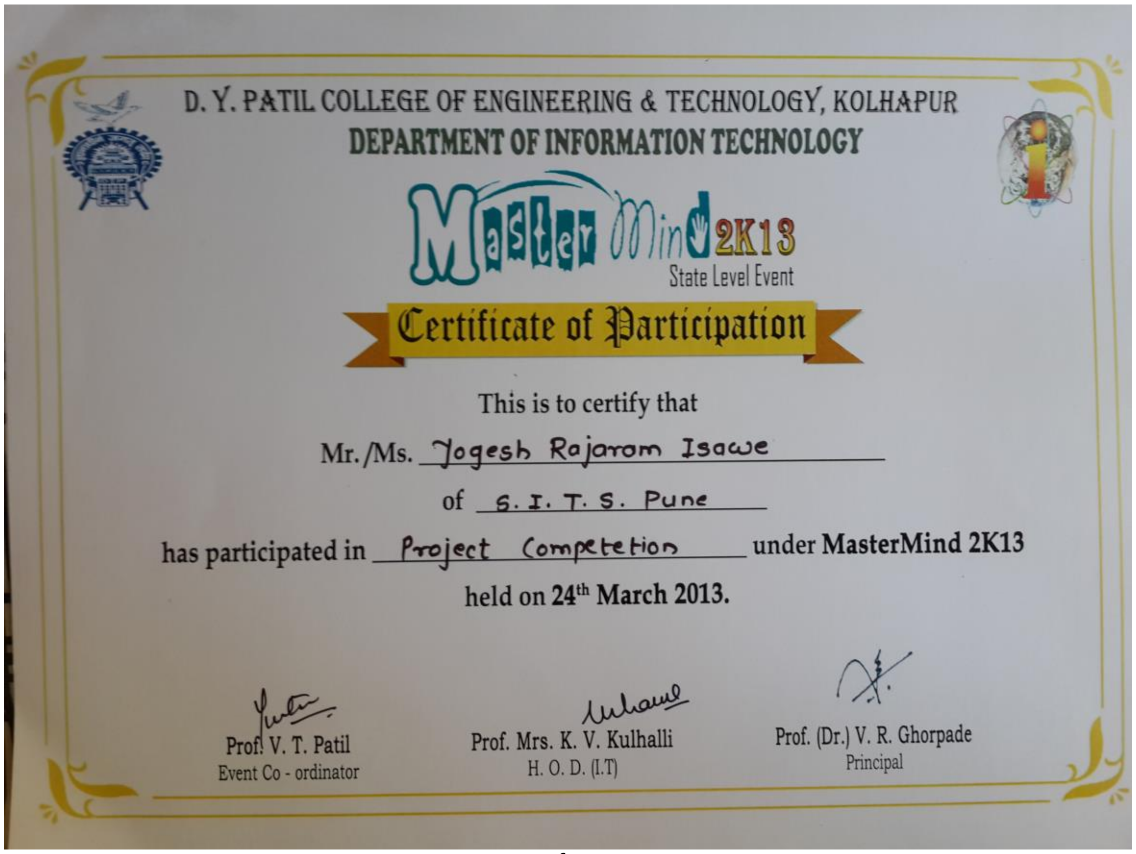
This is to certify that
Mr./Ms. Sourabh Mahendra Lokade
of S. I. T. S. Pune
has participated in Project Competition under MasterMind 2K13
held on 24th March 2013.

Prof. V. T. Patil
Event Co - ordinator

Prof. Mrs. K. V. Kulhalli
H. O. D. (I.T)

Prof. (Dr.) V. R. Ghorpade
Principal

Certificate 1



D. Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY, KOLHAPUR
DEPARTMENT OF INFORMATION TECHNOLOGY
MasterMind 2K13
State Level Event
Certificate of Participation

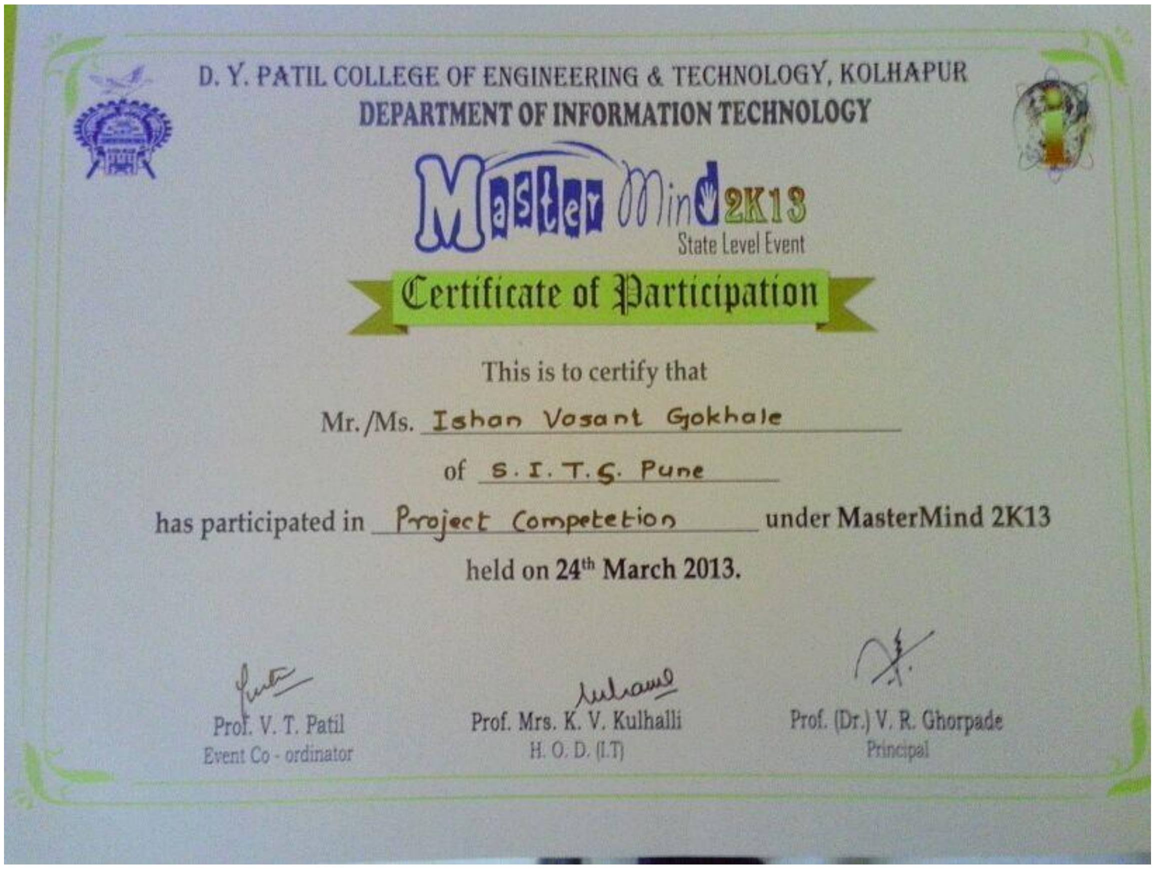
This is to certify that
Mr./Ms. Yogesh Rajaram Isawe
of S. I. T. S. Pune
has participated in Project Competition under MasterMind 2K13
held on 24th March 2013.

Prof. V. T. Patil
Event Co - ordinator

Prof. Mrs. K. V. Kulhalli
H. O. D. (I.T)

Prof. (Dr.) V. R. Ghorpade
Principal

Certificate 2



D. Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY, KOLHAPUR
DEPARTMENT OF INFORMATION TECHNOLOGY

MasterMind 2K13
State Level Event

Certificate of Participation

This is to certify that

Mr./Ms. Ishan Vasant Gokhale

of S.I.T.S. Pune

has participated in Project Competetion under MasterMind 2K13

held on 24th March 2013.

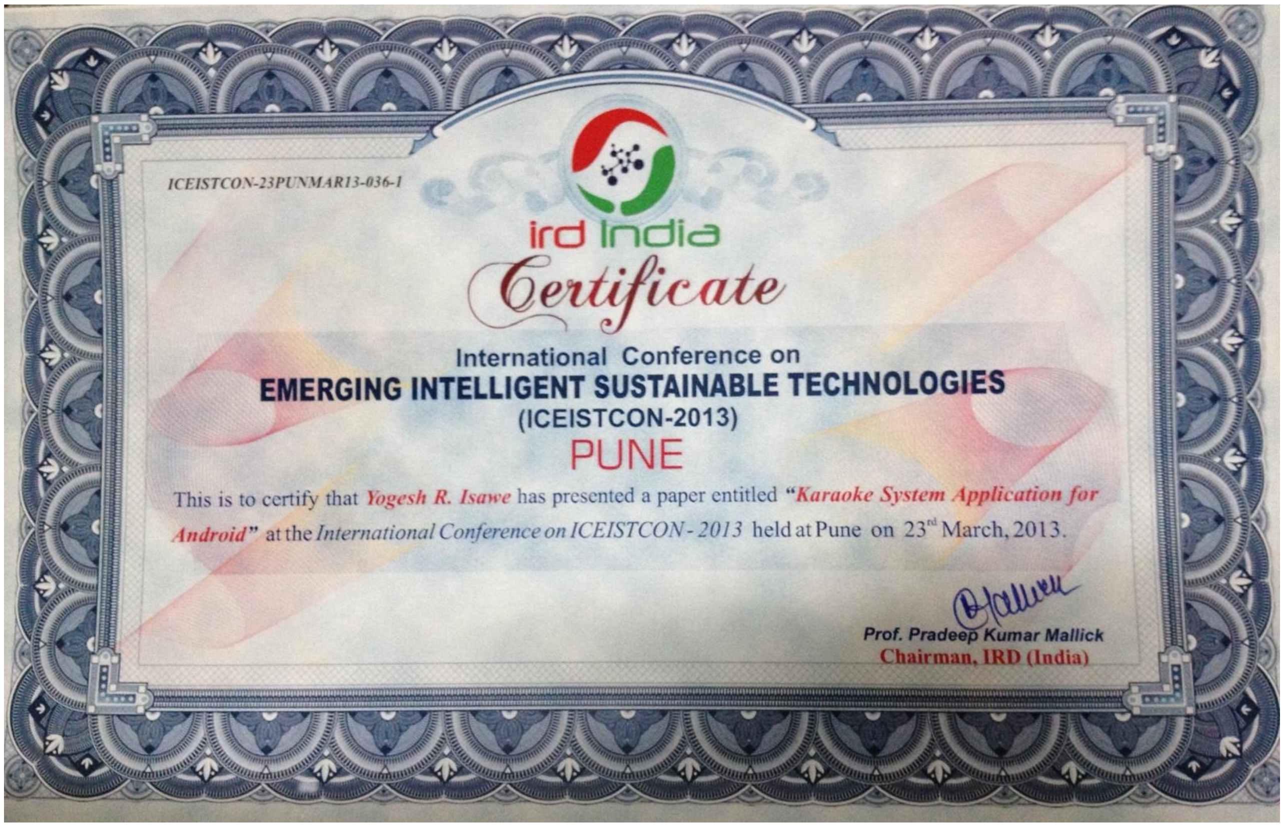
Prof. V. T. Patil
Event Co - ordinator

Prof. Mrs. K. V. Kulhalli
H. O. D. (I.T)

Prof. (Dr.) V. R. Ghorpade
Principal

Certificate 3

- International Paper Presentation Conference



ICEISTCON-23PUNMAR13-036-1

ird India

Certificate

International Conference on

EMERGING INTELLIGENT SUSTAINABLE TECHNOLOGIES

(ICEISTCON-2013)

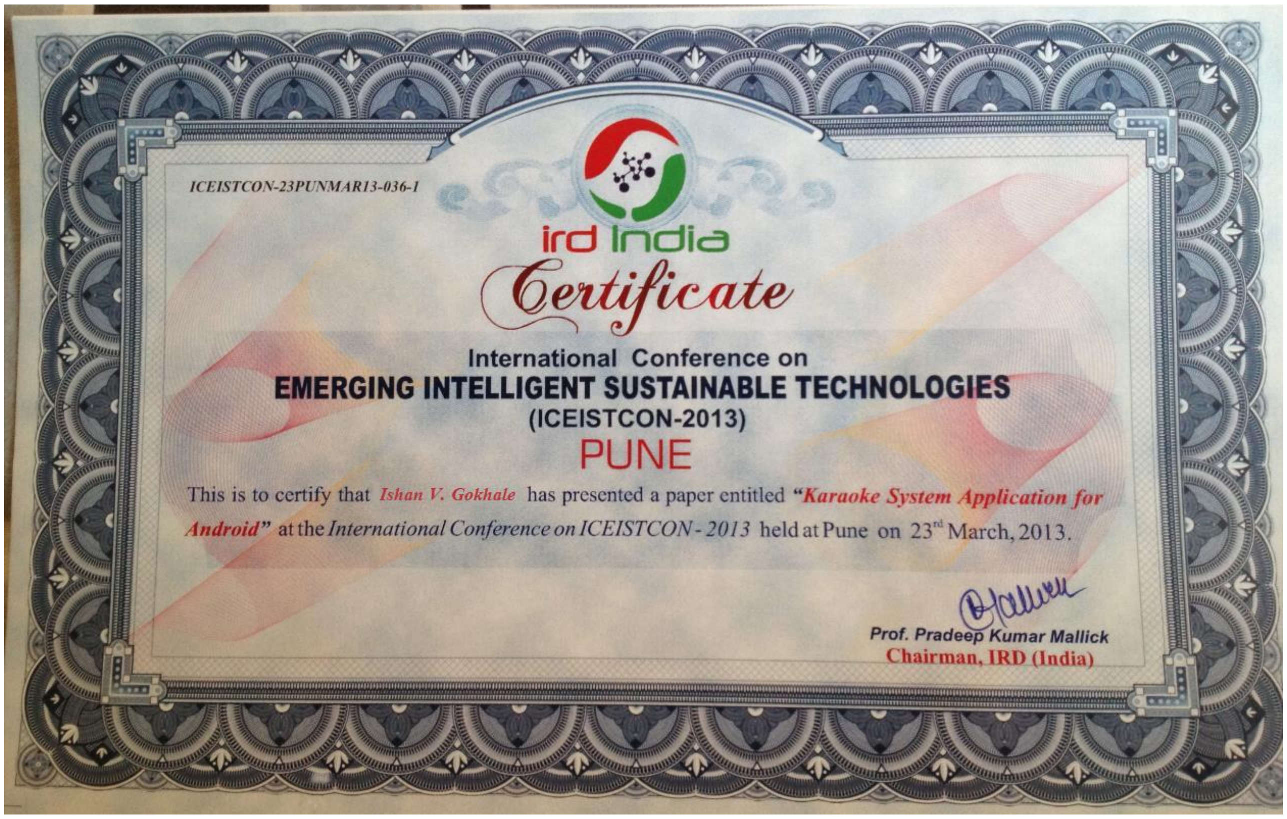
PUNE

This is to certify that *Yogesh R. Isawe* has presented a paper entitled ***"Karaoke System Application for Android"*** at the *International Conference on ICEISTCON - 2013* held at Pune on 23rd March, 2013.

[illegible]

Prof. Pradeep Kumar Mallick
Chairman, IRD (India)

Certificate 1



ICEISTCON-23PUNMAR13-036-1

ird India

Certificate

International Conference on

EMERGING INTELLIGENT SUSTAINABLE TECHNOLOGIES

(ICEISTCON-2013)

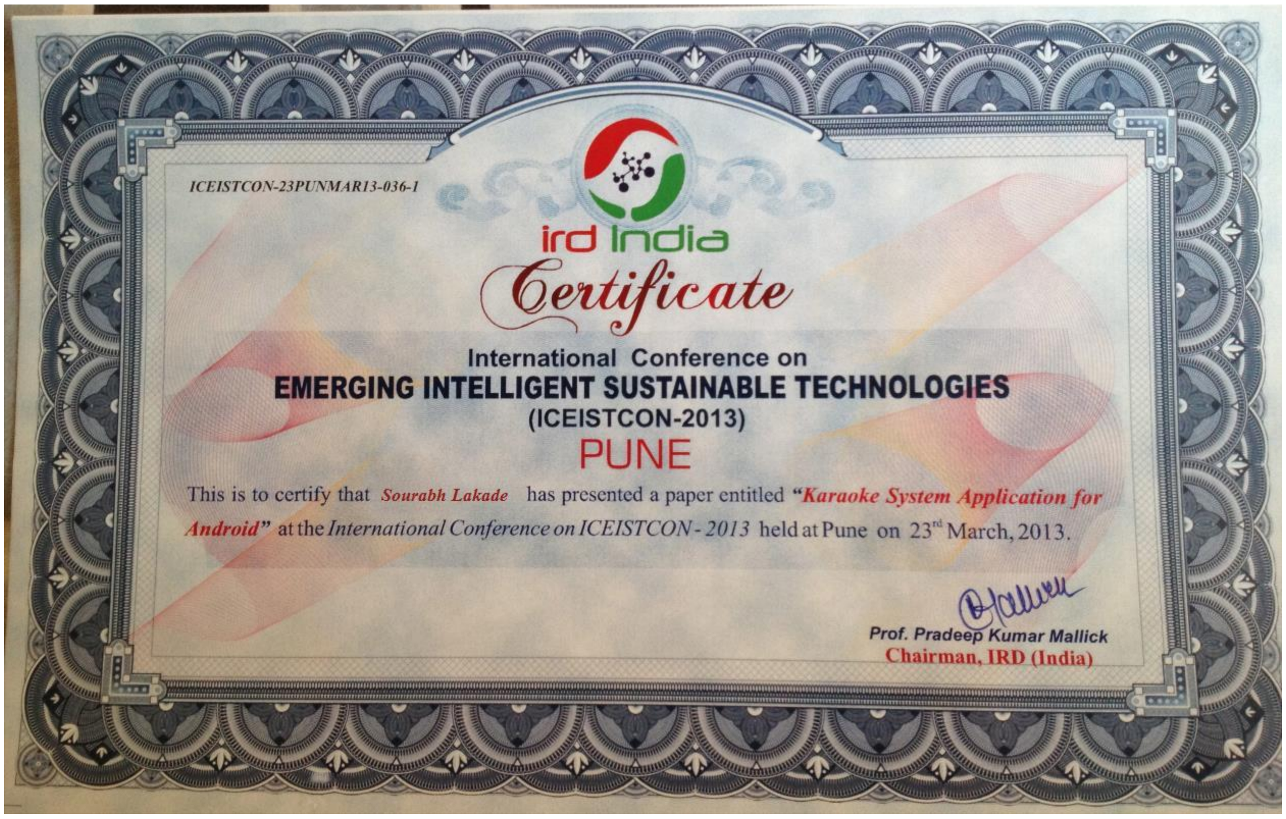
PUNE

This is to certify that *Ishan V. Gokhale* has presented a paper entitled ***"Karaoke System Application for Android"*** at the *International Conference on ICEISTCON - 2013* held at Pune on 23rd March, 2013.

[illegible]

Prof. Pradeep Kumar Mallick
Chairman, IRD (India)

Certificate 2



ICEISTCON-23PUNMAR13-036-1

ird India

Certificate

International Conference on

EMERGING INTELLIGENT SUSTAINABLE TECHNOLOGIES

(ICEISTCON-2013)

PUNE

This is to certify that *Sourabh Lakade* has presented a paper entitled ***"Karaoke System Application for Android"*** at the *International Conference on ICEISTCON - 2013* held at Pune on 23rd March, 2013.

[illegible]

Prof. Pradeep Kumar Mallick

Chairman, IRD (India)

Certificate 3