MUSIC RECOMMENDATION SYSTEM

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering

under the supervision of

Dr.Yashwant Singh

By

Prabhjot Kaur(091201)

Muskan Maheshwary(091220)

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that the work titled "**Music Recommendation System**" submitted by Prabhjot Kaur , Muskan Maheshwary in partial fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision.

This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Date: **Supervisor's Name: Dr. Yashwant Singh**

Designation: Assistant Professor

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to the people who have helped & supported us throughout my project. I am grateful to my mentor **Dr. Yashwant Singh** for his continuous support for the project, from initial advice & contacts in the early stages of conceptual inception & through ongoing advice & encouragement to this day.

A special thank of us to our group members who helped each other in completing the project & exchanged their interesting ideas, thoughts & made this project easy and accurate.

Date:

Prabhjot Kaur
(091201)

Muskan Maheshwary
(091220)

Table of Contents

S.No	Topics	Page no
1	**Chapter 1:Introduction**	**1**
	1.1 Introduction	1
	1.2 Genesis	3
	1.3 Problem Statement	4
	1.4 Objectives	5
	1.5 Approach Used	5
	1.6 Organization of Thesis	6
2	**Chapter 2:Literature Survey**	**8**
	2.1 Existing Music Recommendation Systems	8
	2.1.1 MyStrands	8
	2.1.2 iTunes	9
	2.1.3 iLike	9
	2.1.4 Last.fm	11
	2.1.5 Musicovery	12
	2.1.6 tastekid	13
	2.1.7 Flokoon	13
	2.1.8 Xiggy	13
	2.1.9 Pandora Radio	14
	2.1.10 Songza	14
	2.2 Content Based System	15
	2.3 Metadata Based System	17
	2.4 ID3v1	18
	2.4.1 Ext ended Tags	19

	2.5 ID3v2	21
3	**Chapter 3: Music Recommendation Systems**	**23**
	3.1 Architecture	23
	3.2 Interface of MRS	25
	3.3 MYSQL Database	26
	3.4 Algorithm used in Recommendation	30
4	**Chapter 4:Implementation**	**31**
	4.1 Feasibility Study	31
	4.2 Requirement Analysis	32
	4.2.1 Windows Presentation Foundation	32
	4.2.2 Xaml	33
	4.3 Designs of MRS	35
	4.3.1 Use Case Diagram of MRS	35
	4.3.2 Data Flow Diagram of MRS	36
	4.3.3 Sequence Diagram of MRS	37
	4.3.4 Flow Chart of MRS	38
	4.4Simulation	39
	4.4.1 mainwindow.xaml	39
	4.4.2 mainwindow.xaml.cs	45
	4.5 Output	66
	4.6 Testing	77
5	**Chapter 5:Conclusion**	**78**
6	**References**	**79**

Table of Figures

S.No	Figures	Page No
1	2.1 ID3v1	18
2	3.1 Architecture of MRS	23
3	3.2 Another Architecture of MRS	24
4	3.3 MRS	26
5	3.4 Songentry Table	27
6	3.5 Songentry Table’s entry	27
7	3.6 Songdetails table	28
8	3.7 Songdetails table’s entry	29
9	4.1 . net 3.0 Stack	33
10	4.2 WPF	34
11	4.3 Use case of MRS	35
12	4.4 Data Flow Diagram of MRS	36
13	4.5 Sequence Diagram of MRS	37
14	4.6 Flow Chart of MRS	38
15	4.7 Output	66
16	4.8 Open button functioning	67
17	4.9.1 Window in Small size	68
18	4.9.2 Window in adjustable size	68
19	4.10 Add button -1	69
20	4.11 Add button -2	70
21	4.12 Add button -3	70
22	4.13 Delete button	71
23	4.14 Button functionality	72
24	4.15 Datagrid working	73
25	4.16 Playing recommended playlist	74

26	4.17 Selecting new song without tags	75
27	4.18 New playlist generated	76

ABSTRACT

The growing use of various softwares, has led to the proliferation of technologies to deploy rich applications. Among these applications are present the music service providers. These systems allow users to listen to music which they like without making them involved at end of every song.hence these softwares use recommendations techniques to improve the user experience.The objective of this project is to develop a music recommendation system. The system will determine the musical preferences of the users based on the analysis of their interaction during use. This way the system is able to estimate what artist or group would match user preferences to the user at a given time. It has been taken into account the fact that we do not always want to hear the same artists or genres, we do have favorite bands, but sometimes we appreciates a surprise, a new discovery. The system uses music information collected from database. This sytem helps users discover new artists, albums or songs making the musical catalog available for listening. The dynamic characteristics of the interface allows the user to browse music collections while listening to a song or playing a video. The user will receive information related to her interaction patterns in form of recommendations of items. These items will probably match user preferences and they are shown as the user interacts with the system and only when it has enough information about user preferences

Chapter 1: Introduction

1.1 Introduction

Musicians are competing for an audience among millions of others trying just as hard. And it's not the listener's fault if they miss out on something that will change their lives these days, anyone can gain access to a library of over 15 million songs on demand for free. To a musician turned computer scientist (as I and so many of my colleagues are) this is the ultimate hidden variable problem. If there was something "intelligent" that could predict a song or artist to a person, both sides (musician and listener) win, music is amazing, there's a ton of data, and it's very far from solved.

But anyone in the entire field of music technology has to treat music discovery with respect: it's not about the revenue of the content owner, it's not about the technology, it's not about click through rates, listening hours or conversion. The past few years have shown us over and over that filters and guides are invaluable for music itself to coexist with the new ways of getting at it. We track over 2 million artists now – I estimate there are truly 50 million, most of them currently active. Every single one of them deserves a chance to get their art heard. And while we can laugh when Amazon suggests you put a Norah Jones CD in your cart after you buy a leaf blower, the millions of people that idly put on Pandora at work and get excited about a new band they've never heard deserve a careful look. Recommendation technology is powering the new radio and we have a chance to make it valuable for more than just the top 5 percent of musicians.

When people talk about "music recommendation" or "music discovery" they usually mean one of a few things:

- **Artist or song similarity**: an anonymous list of similar items to your query. You can see this on almost any music service. Without any context, this is just a suggestion of what other artists or songs are similar to the one you are looking at. Formally, this is not truly a recommendation as there is no user model involved (although since a query took the user to the list, I still call these a recommendation. It's a recommendation in the sense that a web search result is.)

- **Personalized recommendation**: Given a "user model" (your activity on a service – plays, skips, ratings, purchases) a list of songs or artists that the service does not think you know about yet that fits your profile.
- **Playlist generation**: Most consumers of music discovery are using some form of playlist generation. This is different from the above two in that they receive a list of items in some order (usually meant to be listened to at the time.) The playlist can be personalized (from a user model) or not, and it can be within catalog (your own music, ala iTunes Genius's or Google's Instant Mix playlists) or not (Pandora, Spotify's or Rdio's radio, iHeartRadio.) The playlist should vary artists and types of songs as it progresses, and many rely on some form of steering or feedback (thumbs up, skips, etc.)
- Where popular services sit in discovery

	Personalized	**Anonymous**
Playlist	Pandora	Radio
Suggestions	Amazon	All Music Guide

These are three very different ways of doing music discovery, but for every technology and approach I know of, they are simply applications on top of the core data presented in different ways. For example, at the Echo Nest we do quite a bit to make our playlists "radio-like" using our observed statistics, acoustic features and a lot of QA but that sort of work is outside the scope of this article – all three of our similarity API, taste profiles (personalized recommendation) API and playlist API start with the same knowledge base culled from acoustic and text analysis of music.

However, the application means a lot to the listener. People seem to love playlists and radio-style experiences, even if the data driving both that and the boring list of songs to check out are the same. One of the great things about working at the Echo Nest is seeing the amazing user interfaces and experiences and people put on top of our data. Listeners want to hear music, and they want to trust the service and have fun doing it. And conversely, a Pandora completely powered by Echo Nest data would feel the same to users but would have far better scale and results and thus add to the experience. Because of this very welcome sharding of discovery applications, it’s helpful less to talk about these applications directly and more to talk about “what the services know” about music – how they got to the result that Kreayshawn and Uffie are similar, no matter where it appeared in the radio station or suggestion or what user model led them there. We can leave the application and experience layer to another lengthy blog post.

My (highly educated, but please know I have no direct inside information except for Echo Nest of course) guesses on the data sources are:

How popular services know about music

Service	**Source of data**
Pandora	Musicologists take surveys
Songza	Editors or music fans make playlists
Last.fm	Activity data, tags on artists and songs, acoustic analysis
All music guide	Music editors & writers
Amazon	Purchase & browsing history
iTunes Genius	Purchase data, activity data from iTune
Echo Nest	Acoustic analysis, text analysis

There are many other discovery platforms but this list covers the widest swath of approaches. Many services you interact with use either these platforms directly (Last.fm, Echo Nest, AMG all license data or give away data through APIs) or use similar enough approaches that it’d be not worth going into in detail.

1.2 Genesis Of Problem

In this modern digital age there is an overflow of music content in every format possible. There are online radios, e-music stores, and specialized music players that help to look for new music available online. It is highly likely that the modern day music listener would want to spend less time looking for music than actually listening it. This is where music recommenders or playlist generators play a big role. We used already conducted research about different types of recommendation system available and to see the working of the above mentioned systems. We will also have a look at the properties of a music file that the recommendation system needs to access. For this we will be limiting ourselves to Mp3 files and ID3 tags. We looked on the two different kinds of recommendation systems (content based & metadata based). We also explore the various existing recommendation systems available in the market today. Since the ID3 tags are an integral part of the metadata based recommendation systems, we will be introducing the different versions of ID3 tags.

1.3 Problem Statement

As users accumulate digital music in their digital devices, the problem arises for them to manage the large number of tracks in their devices. If a device contains thousands of tracks, it is difficult, painful, and even impractical for a user to pick suitable tracks to listen to without using pre-determined organization such as playlists. The topic of this thesis is computationally generated recommendations. Music recommendation is significantly different from other types of recommendations, such as those for movies, books and electronics. Because a same song can be recommended to a same users many times if we successfully keep from him/her bored with it. A main purpose of a music recommendation system is to minimize user’s effort to provide feedback and simultaneously to maximize the user’s satisfaction by playing appropriate song at the right time. Reducing the amount of feedback is an important point in designing recommendation systems, since users are in general lazy. Using this idea we propose a method to automatically recommend music in a user’s device as the next song to be played. In order to keep small the computation time for calculating recommendation, the method is based on user behavior and high-level features but not on content analysis. Which song should be played next can be determined based on various factors.

1.4 Objectives

The main objective of our project are:

- To make a music player

  Player should be created where the functions such as stop , pause, next ,previous Sliding, play should work efficiently and interface should be user friendly

- To make a recommendation system

  Song can be recommended as per song selected by the user . And the Recommended song should be linked to the selected song some context Like their tags and metadata.

- To make a playlist generator

  Another main objective is to get these recommended songs in a playlist and displayed on interface with its titles as extracted from the file.

- To make a music library manager

  Displaying is not only the task even playing all the songs from playlist should be also done efficiently .

1.5 Approach Followed

We have used both metadata and user demand based approach where the user as well song's metadata are used to recommend the song .Then we have used database for storage purpose that as back end .Visual Studio 2010 wpf is used in front end where datagrid is used to link data from tables plus connection are build by using inbuilt commands.hence both user data and metadata is used.

In the recommendation we used Sql queries to fetch the songs similar as input by the user. We had put the condition of rating and user demand which are user specific as per the behavior of the user noted by database and the content of the song which is the meta data for recommendation.Another important thing is kept in mind that while giving an input to the song we can have the same song in database so to ignore the second time playing of the same song status is used by us and query is run for the same to make it function. Duplicate addition of the songs is also prohibited with the help of queries. Not only recommendation even the player working is seen as forwarding of the songs etc.The next and previous button coding is done seeing the condition of the playist as what to give source when next button is pressed when last song of the list is playing
So the approach used is both on user demand and metadata of the songs

1.6 Organization of thesis

In Chapter 2, we have discussed the literature survey. We have looked upon the various existing recommendation systems like last.fm, Pandora.com, Itunes Genius, Musicovery, My Strands, Tastekid, Flookon, Xiggy etc. Some of them are software applications and others are websites. Each one of them has its own unique features which distinguishes it from the rest. We have introduced two types of recommendation systems namely the content-based and metadata-based. Then comes ID3 tags which are the basis of recommendation system. ID3 tags are the metadata and contain information about mp3 files such as Name of the song, Artist, Genre, Year the song was recorded, Album from which the song came, Lyrics, Comments etc. There are two unrelated versions of ID3: ID3v1 and ID3v2.thes two versions are discussed in detail.

In Chapter 3 we have discussed the music recommendation system we have designed, the approach, methodology, algorithm and architecture.

In Chapter 4 we have done feasibility study and requirement analysis. We have discussed the various designs like Data Flow Diagrams, Flow Chart, Use Case etc. Next part includes coding which discusses the basic functioning, the working of various buttons, etc. Next includes the implementation which introduces the software tools like Windows Presentation Foundation using C#, .Net 3.5, MY SQL, Visual Studio 2010. Then testing is done on the music recommendation system made.

In Chapter 5 we have thereby concluded our report and discussed the future scope of our project.

Chapter 2: Literature Survey

2.1 Existing Music Recommendation Systems

Some of the existing recommendation systems in the market are:

2.1.1MyStrands

The MyStrands has the recommendation engine of top notch. Based on songs or artists you either upload from your iTunes playlists or add as favorites on the site, MyStrands will recommend similar songs, albums, and artists that you can add. MyStrands desktop app really rocks. The application works similar to Last.fm'sscrobbling software, reworking your iTunes experience and synchronizing with the MyStrands recommendation engine. With the app, you can start managing your library of music with tags, keep track of the music your friends are listening to, and get multiple recommendations per song played. So MyStrands, a music discovery and social networking site that covers the PC, mobile and physical worlds (see our profile in January), has released an interesting new recommendations feature. It uses the MyStrands Public APIs (called OpenStrands) to link their social music recommendations to Wikipedia information. Essentially it's a mashup of MyStrands music recommendations with artist information from Wikipedia. It's not a huge feature, but it's a neat example of the innovation that is happening with music and the Web.

2.1.2 iTunes Genius

This is the best way to remaster your library into the perfect gym mix and keep you from entering into a music coma. Some key features that make Genius rock our world include the ability to select any song in your library and have Genius create a custom playlist based on the musical elements of that song (click the icon in the lower right hand corner to get started). You can then save that playlist, adjust the number of songs displayed, or start all over with a new song. What Genius is great at is not just finding songs you'll like, but playing them in an order that is complimentary to each other, hence the ability to create workout worthy playlists. Of course the other component to Genius is the sidebar, which suggests relevant songs and artists, based on your iTunes behavior (rating, play count etc), that you can then purchase from the iTunes store. And if sharing your Genius playlists with Twitter is a must, there are scripts available to help you do just that.

2.1.3 iLike

iLike was an online service that allowed users to download and share music. The website made use of a sidebar that is used with Apple's iTunes or Microsoft's Windows Media Player. The program and sidebar are not required in order to use the site but allow for ease in discovering new artists. Although the website is still in a somewhat beta version, it is open to anyone. The site attracted around half a million users in the first four months after it was launched. According to the latest statements by the company, over 60 million consumers registered to use iLike either directly on iLike.com or using the apps built by iLike for third-party social networks such as Facebook. iLike also built a "post-once publish-everywhere" dashboard for artists – major label artists as well as independent artists. It now receives an average of 150,000 visitors per day.iLike had a free Facebook application which allowed users to play clips of music they like on their profile, show concerts they are going to and play a music trivia quiz. The application had great success after its release, making it one of the most popular applications on the Facebook Platform. As of November 2007, iLike had more than 15 million users.With the launch of Facebook Pages, iLike created pages for bands. A similar feature was also available for the Bebo network.In April 2009, iLike renamed this application to simply "Music" to maintain consistency with other Facebook apps.The website and Facebook app no longer exist

2.1.4 Last.fm

Last.fm is the on-demand listening options,one can create and manage playlists, view related artists and songs, plus love, share, and tag tracks. Playlist creation is not quite as perfect as MyStrands or iTunes Genius, but when combined with the mobile app, website features, and social sharing options, Last.fm is an awesome way to find and share new music. **Last.fm** is a music website, founded in the United Kingdom in 2002. It claimed 30 million active users in March 2009.Using a music recommender system called

"Audioscrobbler", Last.fm builds a detailed profile of each user's musical taste by recording details of the tracks the user listens to, either from Internet radio stations, or the user's computer or many portable music devices. This information is transferred ("scrobbled") to Last.fm's database either via the music player itself (Rdio, Spotify, Clementine, Amarok) or via a plugin installed into the user's music player. The data is then displayed on the user's profile page and also compiled to create reference pages for individual artists.The site offers numerous social networking features and can recommend and play artists similar to the user's favourites.The current Last.fm website was developed from two separate sources: *Last.fm* and *Autoscrobbler*.The Audioscrobbler (the term *scrobbling* is defined as: to find, process and distribute information involving people, music and other data) and Last.fm teams began to work closely together, both teams moving into the same offices in Whitechapel, London, and by 2003 Last.fm was fully integrated with Audioscrobbler profiles. Input could come through an Audioscrobbler plugin or a Last.fm station. The sites also shared many community forums, although a few were unique to each site.Last.fm user can build a musical profile using any or all of several methods: by listening to their personal music collection on a music player application on a computer or an iPod with an Audioscrobbler plugin, or by listening to the Last.fm internet radio service, either with the Last.fm client, or with the embedded player. All songs played are added to a log from which personal top artist/track bar charts and musical recommendations are calculated. This automatic track logging is called **scrobbling**.Last.fm automatically generates a profile page for every user which includes basic information such as their user name, avatar, date of registration and total number of tracks played. This can be customized with additional information or photographs if desired but the fundamental layout cannot be changed. There is also a Shoutbox for public messages. Profile pages are visible to all, together with a list of top artists and tracks, and the 10 most recently played tracks (can be expanded). Each user's profile has a 'Taste-o-Meter' which gives a rating of how compatible your music taste is.Profile pages can also include lists of friends, weekly musical "neighbours", favourite tags, groups and events. Specifically, musical neighbours show users other people on the site who have the most similar musical tastes to them. The neighbours section notes favorited genres, artists, the last track the neighbour listened to, and allow individuals the ability to examine that user's profile. Finally, a neighbourhood radio is available for users to listen to, and is customized according to the interests of the neighbourhood.The most recent expanded service on Last.fm is a personal recommendations page known as "The Dashboard". This is only visible to the user concerned and lists suggested new music, events, journal entries and other people with similar tastes, all tailored to the user's own preferences.

Recommendations are calculated using a collaborative filtering algorithm so users can browse and hear previews of a list of artists not listed on their own profile but which appear on those of others with similar musical tastes. The page also lists music that has been directly recommended to the user and groups the user belongs to, journals written by users about artists the user listens to, and other users who have listened to similar music recently. There is also a 'recommendation radio' station which will play music specifically filtered based on the user's last week of listening. Last.fm also permits users to manually recommend specific artists, songs or albums to other users on their friends list or groups they belong to, providing the recommendation in question is included in the Last.fm database.Last.fm supports end-user tagging or labeling of artists, albums, and tracks to create a site-wide folksonomy of music. Users can browse via tags,

2.1.5 Musicovery

Musicovery is an interactive and customised webradio service. Listeners rate songs, resulting in a personalized programme.The webradio service is accessible on mobile phone (on 3G/Symbian Nokia Devices) iPhone and iPod Touch.Music files provided by the service are streamed, not downloaded, and the listener can buy all the songs played or tagged as favorite from major online music retailers iTunes, Amazon, and eBay.Musicovery relies on proprietary technology developed by its founders. The "mood pad" technology: a music description methodology that enables to position any song on a 2 dimension continuous space (the “mood pad”); songs are described with 40 musical parameters; The technology is the result of 3 year research on music description and human acoustic perception.The streaming music is at low quality 32kb/s on the free platform, and at good quality 128kb/s on the paid platform.Musicovery, by Frederic Vavrille, is an interactive webRadio, or if you want, an hybrid of two of the most compelling music services on the Internet, MusicPlasma and Pandora. Frederic Vavrille, creator of the discovery engine MusicPlasma, has just materialized my wish. Being an enthusiast of both services, I had acknowledged that Pandora was missing a visual component in order to map all the relationships and influences between the artists, while on the other hand, MusicPlasma was lacking a key feature of a music engine, sound. Frederic made all come together in Musicovery. The user can start by selecting a mood in the mood/energy matrix or choose a radio acording to their mood by clicking in the colored matrix. After that the interaction is very similar to MusicPlasma, where one can easily navigate between different artists, only this time with sound.

It is a visual approach to recommending new music, offers a really fun and unique way to listen and interact with you your custom playlists. It’s pretty basic to get started, just select your mood or tempo, eliminate genres that you’re not interested in, and you’re off. Musicovery populates a musical road map with songs that match your interests. You can add favorites, block songs, and narrow in on specific songs to repopulate the map. And you can also purchase songs from eBay, Amazon, and iTunes.

2.1.6 Tastekid

Tastekid provides music, movies, shows, books and authors recommendations, based on your existing preferences. You choose the Music tab and enter your query and Tastekid brings up a list of related bands. Click on the search result of your choice and a pop-up window displays a short introductory text to the band in question along with a video that plays right in the window and links to the Wikipedia article and a Google search. There are options to like, dislike or save an artist (and even a Meh button if you aren’t certain). The same box contains tabs that let you share your find on Twitter and Facebook or find it on Amazon and iTunes store. It works even for little known bands.

2.1.7 Flokoon

Flokoon is an innovative search engine that applies its technology on data from YouTube, Fotolia and Last.fm. You click on radio buttons to choose to search for an artist (default) or a tag. When you do your search, a kind of map or web appears: The band you searched for is presented in the middle, surrounded by related bands. To expand the web, click on the artist of your choice among the search results. Related artists appear and in this way you can continue to explore a genre, artist by artist. There is an x in the top right corner of each item that lets you remove items from your search as well. You can learn more about the suggested artists or bands by pointing your mouse to an item. Three options pop up: More Details, Popular Tags and Top Fans. More details brings up bio and a list of albums, though this doesn't work for more obscure bands. In this box, you can also click "Add to my discoveries" to save your find in a tab at the bottom left of the screen .The option Popular Tags lets you explore tags associated with the artist in question while Top Fans shows Last.fm users that like the artist. There is no way to listen, though, so you have to have iTunes store or a similar source of music available.

2.1.8Xiggy

In Xiggy enter band or artist names than song titles and click the search button. The results that return do not contain info on the band you searched for or the usual 10 items of a text snippet and a link. Instead you are presented with a list of 11 band or artist names (and links to lots more). As you point your mouse to one of them, a short text appears, accompanied with an image. But the search continues. Drag the results you like to the search box to refine your search. Each time you add an item, Xiggy changes the search results accordingly. The data are provided by Last.fm, so when you click on the search results, you arrive on the relevant Last.fm page, where you can listen to music.

2.1.9 Pandora Radio

Pandora Internet Radio (also known as Pandora Radio or simply Pandora) is an automated music recommendation service and "custodian" of the Music Genome Project.

The service, operated by Pandora Media, Inc., is fully available in the United States, Australia and New Zealand. The service plays musical selections of a certain genre based on the user's artist selection. The user then provides positive or negative feedback for songs chosen by the service, which are taken into account when Pandora selects future songs.

While listening, users are offered the ability to buy the songs or albums at various online retailers. Over 400 different musical attributes are considered when selecting the next song. These 400 attributes are combined into larger groups called focus traits. There are 2,000 focus traits. Examples of these are rhythm syncopation, key tonality, vocal harmonies, and displayed instrumental proficiency.

2.1.10 Songza

Songza is a free music streaming and recommendation service. Stating that its playlists are made by music experts, the service recommends various playlists based on time of day and mood or activity. Songza offers playlists for activities such as waking up, working out, commuting, concentrating, unwinding, entertaining, and sleeping.Users can vote songs up or down, and the service will adapt to the user's personal music preferences.Users can find playlists not just based on artists, songs, or genres, but also based on themes, interests, and eras.If it’s one thing that you really don’t have to worry about too much on the Android platform, it’s not being able to find a decent music streaming application. To go along with the big two, Spotify and Pandora, you also have the likes of Slacker or even TuneIn Radio if you have pangs for listening to your old radio stations from back home. While it is getting to be a crowded market, there is always room for more and the folks over at Songza Media hope that you will be willing to give their product, aptly named Songza, a try as well.if it’s one thing that Songza does have going for it, it’s the fact that there are zero advertisements that you have to listen to. You can also skip songs to your hearts content instead of being limited to so many every day. That isn’t all that’s good with Songza, but I figured that it would good to go ahead and let you know what differentiates it from some of the other options out there. Upon logging in you are presented with a menu of choices to help you figure out what kind of music you would like to listen to: are you curing road rage, making out, unwinding after work or waking up on the right side of the bed. Those are just a small sample of the “Activities” section, there are also areas for “Moods”, “Decades”, “Culture”, “Record-store Clerk”, “Genres”, or you can listen to tracks that have been curated “Just For You” by way of your thumbs up/down voting individual tracks.Along the top of application is where you find quick access to other functionality provided: “Collection”, “Discover”, “People”, “Currently Playing”. Collection is where you will find the previously mentioned music shelf and is where you can create your own lists of music. They aren’t really “playlists” per-se, as you cannot keep track and save individual songs, instead think of each music shelf as holding many “Best of” playlists. Discover is the application’s main home screen which is where you select your Mood or Activity. People is an interesting tab; while Songza isn’t a social network (that I can tell) it is possible to follow what other people have listened to and lists that they have curated.

Music recommender systems can be broadly categorized into two different approaches:

2.2 Content-Based Systems

Content-based systems “listen” to the audio content of the music and build playlists by finding songs that sound similar or that have similar semantic descriptions. The online radio station Pandora.com hires musicologists to listen to each of the 1 million songs in its “music genome” database and objectively characterize their acoustic content using 400 semantic descriptors. The purpose of the CB method is to recommend the music objects that belong to the music groups the user is recently interested in. To capture the recent interests of the user, we analyze the latest transactions in the access history as follows. In the following example, we only use the latest five transactions for simplicity. Each transaction is assigned a different weight, where the latest transaction has the highest weight. Moreover, the music group containing more accessed music objects in a transaction has a higher weight than other groups in the same transaction. The weight *GWi*of music group *Gi*is computed as follows:

$$GW_i = \sum_{j=1}^{n} TW_j \times MO_{j,i} \quad (4)$$

Where TW_j is the weight of transaction T_j; n is the number of latest transactions used for analysis, and $MO_{j,i}$ is the number of music objects which belong to music group G_i in transaction T_j. These weights will be recorded in a *preference table* for the user.

Recommender systems are a special type of information filtering systems. Information filtering deals with the delivery of items selected from a large collection that the user is likely to find interesting or useful and can be seen as a classification task. Based on training data a user model is induced that enables the filtering system to classify unseen items into a positive class c (relevant to the user) or a negative class c (irrelevant to the user). The training set consists of the items that the user found interesting. These items form training instances that all have an attribute. This attribute specifies the class of the item based on either the rating of the user or on

implicit evidence. Formally, an item is described as a vector (n) X x , x ,..., x 1 2 = of n components. The components can have binary, nominal or numerical attributes and are derived from either the content of the items or from information about the users' preferences. The task of the learning method is to select a function based on a training set of m input vectors that can classify any item in the collection. The function h(X) will either be able to classify an unseen item as positive or negative at once by returning a binary value or return a numerical value. In that case a threshold can be used to determine if the item is relevant or irrelevant to the user.

- Content Page
- Navigation Page
- Hybrid Page
- Hypertext Link

A content-based filtering system selects items based on the correlation between the content of the items and the user's preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences. PRES is a content-based filtering system. It makes recommendations by comparing a user profile with the content of each document in the collection. The content of a document can be represented with a set of terms. Terms are extracted from documents by running through a number of parsing steps. First all HTML tags and stop words (words that occur very often and cannot be used as discriminators) are removed. The remaining words are reduced to their stem by removing prefixes and suffixes [Porter 1980]. For instance the words “computer”, “computers” and “computing” could all be reduced to “comput”. The user profile is represented with the same terms and built up by analyzing the content of documents that the user found interesting. Which documents the user found interesting can be determined by using either explicit or implicit feedback. Explicit feedback requires the user to evaluate examined documents on a scale. In implicit feedback the user’s interests are inferred by observing the user’s actions, which is more convenient for the user but more difficult to implement. There are several ways in which terms can be represented in order to be used as a basis for the learning component

2.3 Metadata Based Systems

Metadata-based systems use information associated with the music that is not directly related to the acoustic content, such as ‘artist name’, ‘genre’ or any other tag information. Based on the collaborative approach, the purpose of the COL method is to provide surprising findings due to the information sharing between *relevant users*. To refer to the information from other users, we group the users first. In the COL method, we apply the technique proposed in Wu et al. (2001) for user grouping. The idea of the technique is to derive the profiles of user interests and behaviors from transactions in the access histories. Users with similar profiles of interests and behaviors will be identified as relevant users.

2.4 ID3v1 Tags

Most people think that the mp3 file just holds the code for playing back an audio file and nothing else. Most mp3 files also store ID3 tags. These ID3 tags are the metadata and contain information such as the following about mp3 files:

Name of the song

Artist

Genre

Year the song was recorded

Album from which the song came

Lyrics

Comments

Most modern MP3 players can read, display and react to this included data when playing a file. There are two unrelated versions of ID3: ID3v1 and ID3v2. Although ID3 is sometimes referred to as a *standard*, the term applies only in the *de facto* sense, as no standardization body was involved in its creation nor has such an organization given it a formal approval status.

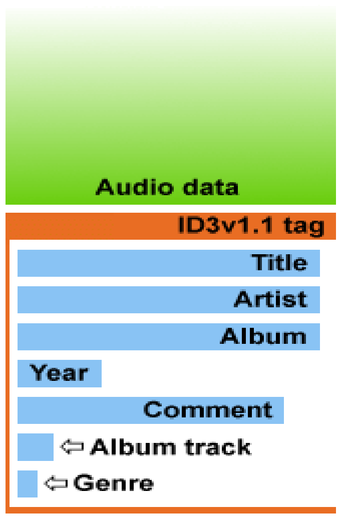


Fig.2.1 ID3v1

Internal layout of an ID3v1.1 tagged file

After the creation of the MP3 standard, there appeared a problem with storing data about the file. Standalone MP3s didn't have any special method of doing this. In 1996 Eric Kemp had the idea to add a small chunk of data to the audio file, thus solving the problem. The method, now known as ID3v1, quickly became the *de facto* standard for storing metadata in MP3s.[2] The format was released by Damaged Cybernetics, an underground group that specialized in cracking console gaming systems. There was no identifying information for any of the ROMS, thus an ID tagging system was created to make tracking easier. Eric and associates carried this over into mp3 files. This format was used for a number of file formats unknown at that time.

The ID3v1 tag occupies 128 bytes, beginning with the string *TAG*. The tag was placed at the end of the file to maintain compatibility with older media players. Some players would play a small burst of static when they read the tag, but most ignored it, and almost all modern players will correctly skip it. This tag allows 30 bytes each for the title, artist, album, and a "comment", four bytes for the year, and a byte to identify the genre of the song from a predefined list of 80 values (Winamp later extended this list to 148 values).

One improvement to ID3v1 was made by Michael Mutschler in 1997. Since the comment field was too small to write anything useful, he decided to trim it by two bytes and use those two bytes to store the track number. Such tags are referred to as ID3v1.1.[2]

2.4.1 Extended Tag:

The extended tag is an extra data block before an ID3v1 tag, which extends the title, artist and album fields by 60 bytes each, offers a freetext genre, a one-byte (values 0-5) speed and the start and stop time of the music in the MP3 file, e.g., for fading in. If none of the fields are used, it will be automatically omitted.

Some programs supporting ID3v1 tags can read the extended tag, but writing may leave stale values in the extended block. The extended block is not an official standard, and is only supported by few programs, not including XMMS or Winamp. The extended tag is sometimes referred to as the “enhanced” tag.

Layout:

Strings are either space- or zero-padded. Unset string entries are filled using an empty string.

ID3v1: 128 bytes

Field	Length	Description
Header	3	"TAG"
Title	30	30 characters of the title
Artist	30	30 characters of the artist name
Album	30	30 characters of the album name
Year	4	A four-digit year
Comment	28 or 30	The comment.
zero-byte	1	If a track number is stored, this byte contains a binary 0.
Track	1	The number of the track on the album, or 0. Invalid, if previous byte is not a binary 0.
Genre	1	Index in a list of genres, or 255

Table 2.1 ID3V1

Extended tag (placed before the ID3v1 tag): 227 bytes

Field	Length	Description
Header	4	"TAG+"
Title	60	Next 60 characters of the title (90 characters total)
Artist	60	Next 60 characters of the artist name

Album	60	Next 60 characters of the album name
Speed	1	0=unset, 1=slow, 2= medium, 3=fast, 4=hardcore
Genre	30	A free-text field for the genre
start-time	6	the start of the music as mmm:ss
end-time	6	the end of the music as mmm:ss

Table 2.2 ID3V1-Extended Tags

2.5 ID3V2

In 1998, a new specification called ID3v2 was created by multiple contributors.[4] Although it bears the name ID3, it has little to no relation to ID3v1.

ID3v2 tags are of variable size, and usually occur at the start of the file, to aid streaming media. They consist of a number of *frames*, each of which contains a piece of metadata. For example, the*TIT2* frame contains the title, and the *WOAR* frame contains the URL of the artist's website. Frames can be up to 16MB in length, while total tag size is limited to 256MB. The internationalization problem was solved by allowing the encoding of strings not only with ISO-8859-1, but also with UTF-16.

Textual frames are marked with an encoding byte.[5]

$00 – ISO-8859-1 (ASCII).
$01 – UCS-2 (UTF-16 encoded Unicode with BOM), in ID3v2.2 and ID3v2.3.
$02 – UTF-16BE encoded Unicode without BOM, in ID3v2.4.
$03 – UTF-8 encoded Unicode, in ID3v2.4.

In the latest ID3v2 specification there are 84 types of frame, and applications can also define their own types. There are standard frames for containing cover art, BPM, copyright and license, lyrics, and arbitrary text and URL data, as well as other things.

There are three versions of ID3v2:

- ID3v2.2 was the first public version of ID3v2. It used three character frame identifiers rather than four (*TT2* for the title instead of *TIT2*). Most of the common v2.3 and v2.4 frames have direct analogues in v2.2. Now this standard is considered obsolete.[6]

- ID3v2.3 expanded the frame identifier to four characters, and added a number of frames. A frame could contain multiple values, separated with a */* character. This is the most widely used version of ID3v2 tags.

- ID3v2.4 is the latest version published, dated November 1, 2000. Notably, it allows textual data to be encoded in UTF-8, which was a common practice in earlier tags (despite the standard, since it was not supported yet) because it has several noticeable advantages over UTF-16. It uses a null byte to separate multiple values, so the character "/" can appear in text data again. Another new feature allows the addition of a tag to the end of the file before other tags (like ID3v1).

Chapter 3: Music Recommendation System

3.1 Architecture

User interacts with the application and select the song plus rate the song if he has to add any song. Then the application using recommendation techniques and metadata reflect the similar type of song plus increments the userdemand by one when song get played. User has interaction with the application and database where song are stored .Rating is another criteria and rating above 5 is not allowed .System too interact with user and database and recommend the song related to input by the user.

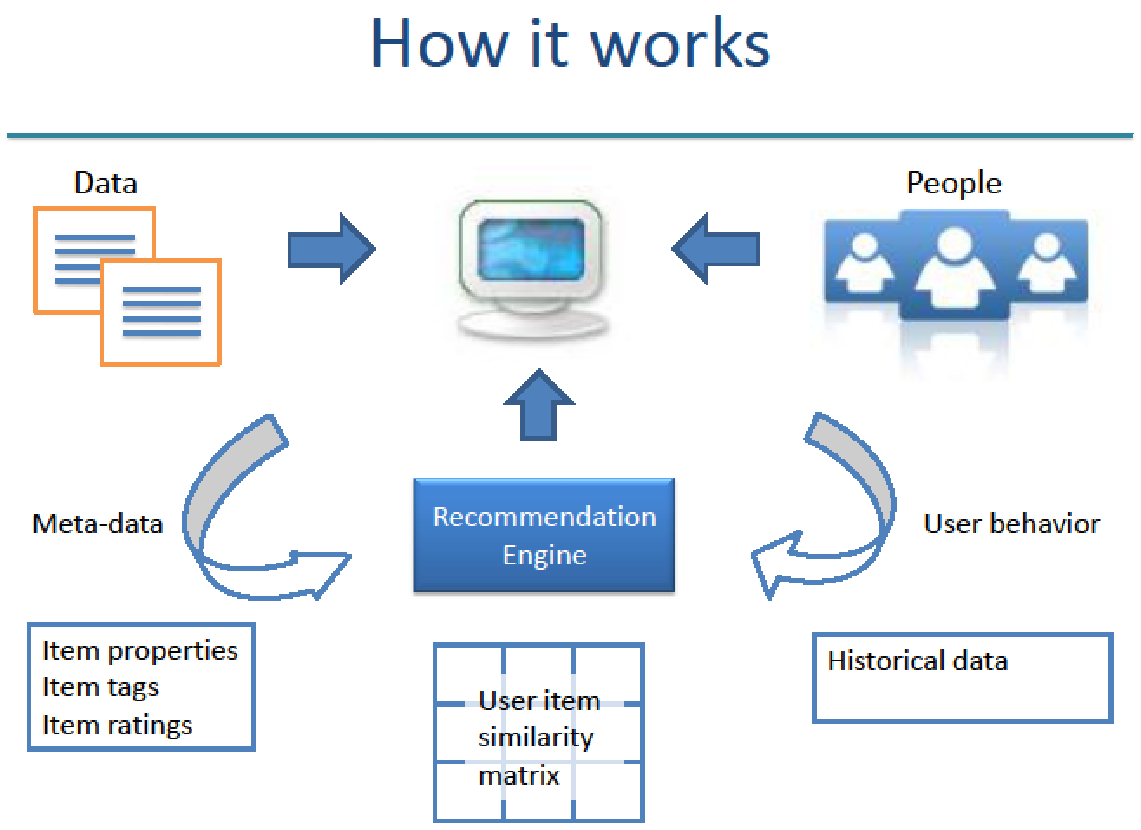


Fig 3.1 Architecture of MRS

Database interacts with the system as the song when it play its demand is incremented ,tags with extraction arestored in tables and to remove duplicate records etc in every case Database query running on the system helps in maintaining the playlist of recommended songs ,addition of songs and deletion .

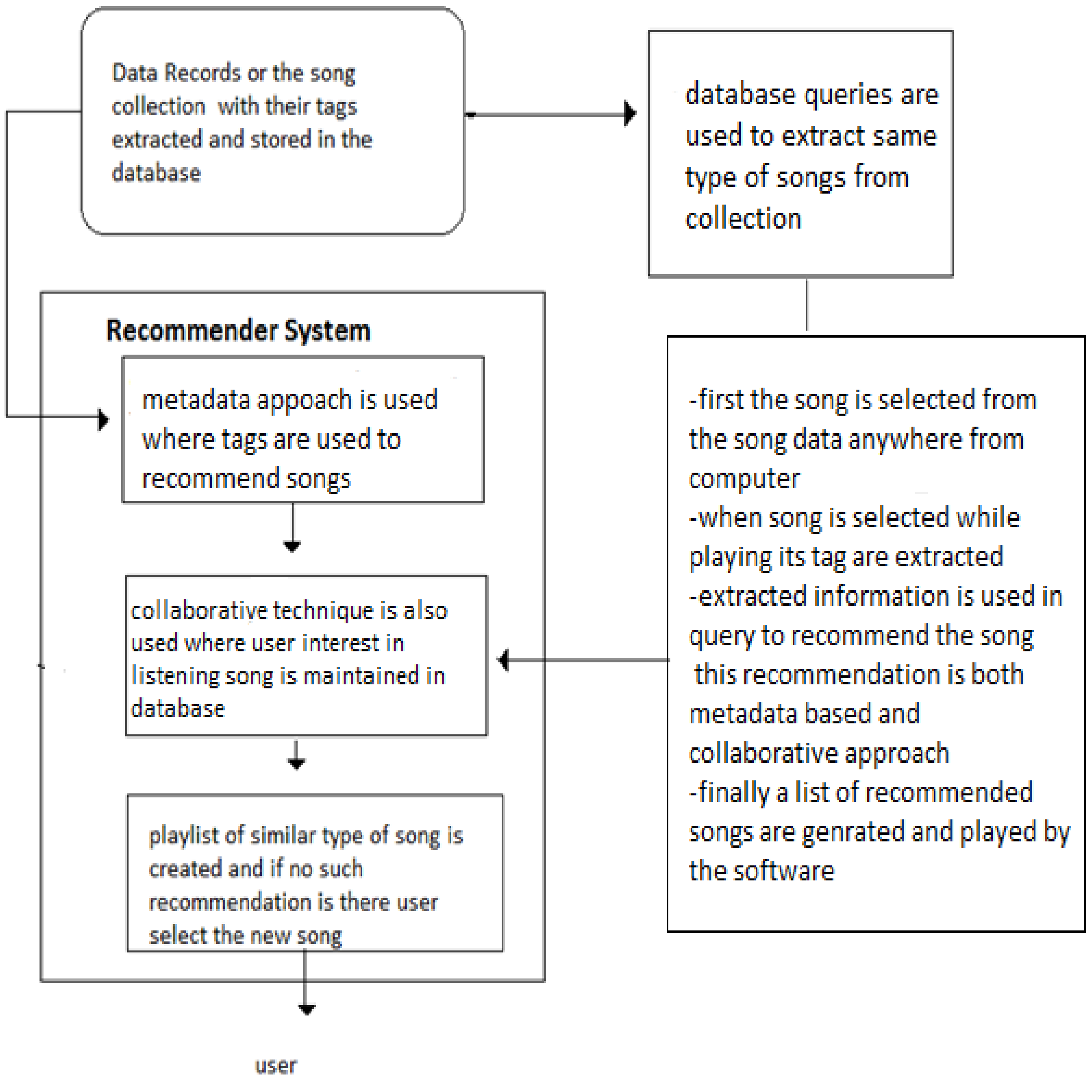

Data Records or the song collection with their tags extracted and stored in the database
database queries are used to extract same type of songs from collection
Recommender System
metadata appoach is used where tags are used to recommend songs
collaborative technique is also used where user interest in listening song is maintained in database
playlist of similar type of song is created and if no such recommendation is there user select the new song
-first the song is selected from the song data anywhere from computer
-when song is selected while playing its tag are extracted
-extracted information is used in query to recommend the song this recommendation is both metadata based and collaborative approach
-finally a list of recommended songs are genrated and played by the software
user

Fig 3.2 Another architecture of MRS

3.2 Interface of Music Recommendation System

We have created a music recommendation system where we had an application ability to play a song ,pause it, stop it switch to the next and previous songs. The size of application is adjustable to any size .We have set The open button at the top left corner which help in randomly selecting the song from the computer. On the right corner we have minimize to minimize ,resize adjust the window size and close button which perform the function respective to their name.

Then we have used three Datagrid one for deletion second for song files and third for song titles named song name in the interface as shown. These datagrids helps in connecting the data from the database .In the Delete grid a delete button is in front of every song and if we want to delete that particular song than we press that button.Similarly for songname datagrid only one column of the table that is the title is displayed and the song file in the another datagrid which is hidden and these two datagrid are made consistence in every case by order then on the basis of id.

At the middle of the application we have music element in which the song file plays. This Music element has open and end media functions which plays at the open and ending of the files and this media ended function is used to put the source of the next song to the media element plus also to increment the user demand when song plays and check that same song does not play again.

The slider and the progress bar is at below of the media element which dynamically fuctions when a song is playing .the various buttons we can see such as play pause astop next and previous to perform their function .on thr right we designed two botton add and delete to add and delete the song .for addditon we anquire the rating of the song by the user so that user preference is also seen.

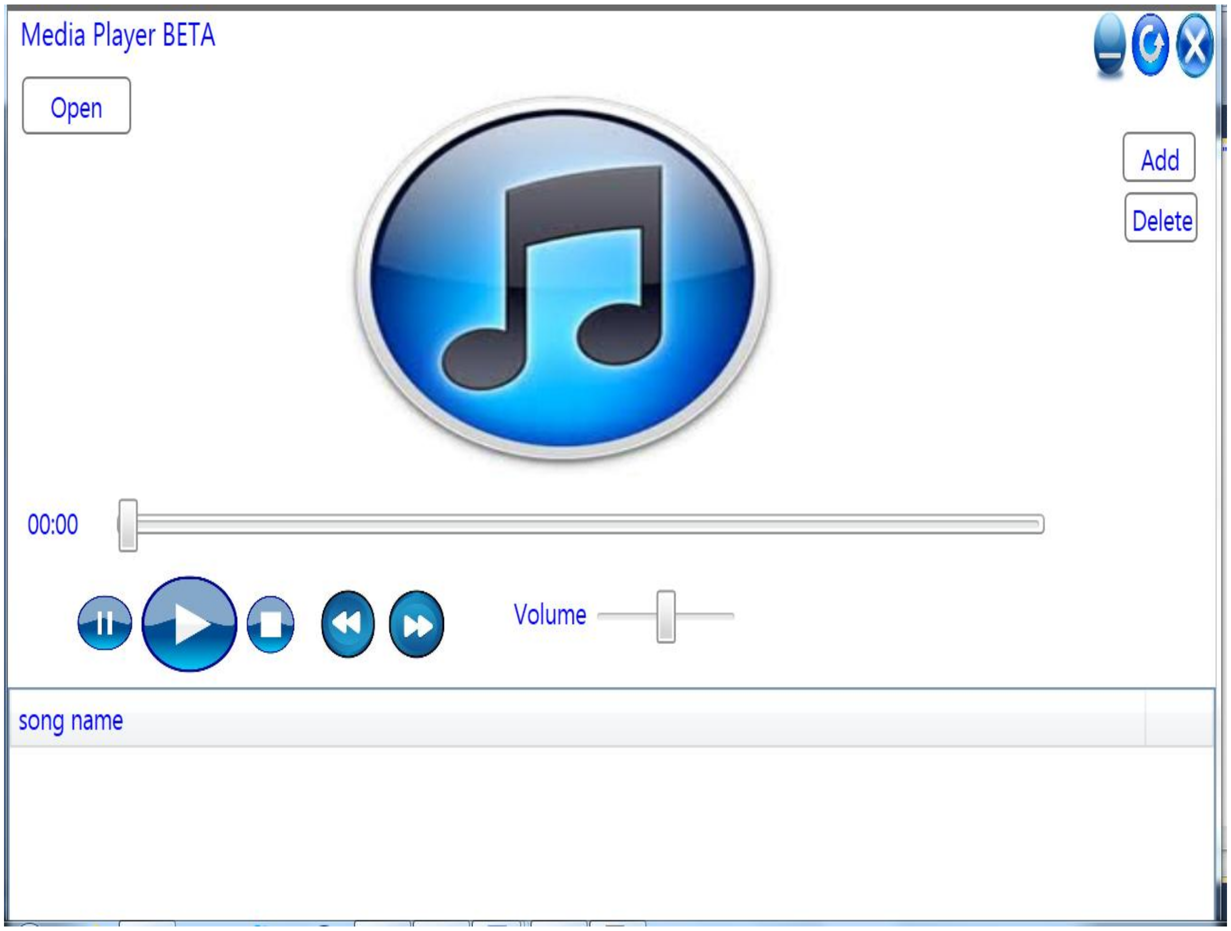
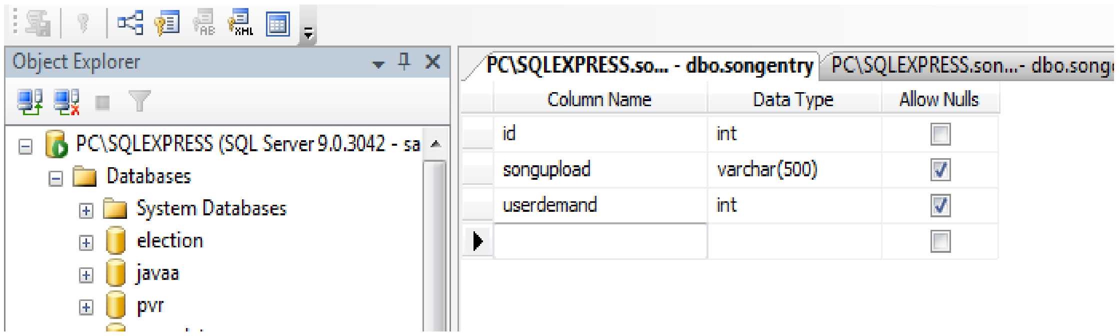


Fig.3.3 Music Recommendation System

3.3 MY SQL Database

We have used database to store the data such as song file and its metadata .We used two table where all these data are stored . We have database name Songdata in which we have two table are created.The Database is linked and with the Query the table is modified with data. Simple and complex queries are used to retrieve and add information to the table. Even subqueries are also used.

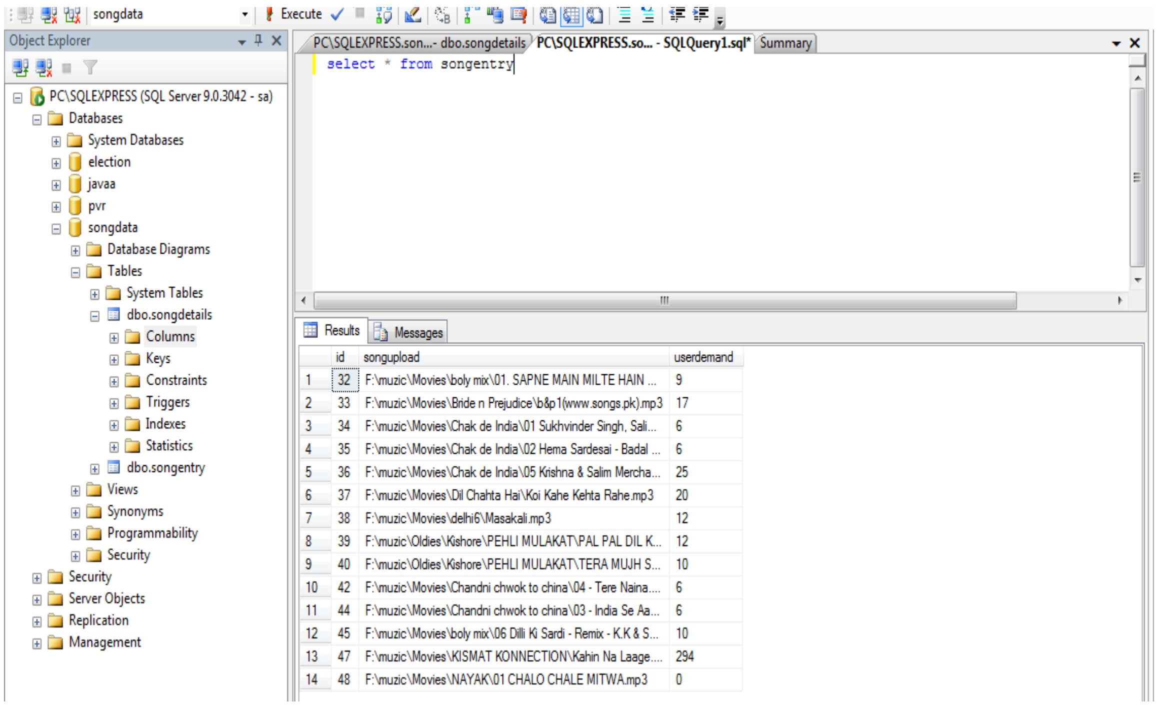
Songentry is one of the table where the mp3/music files are stored and userdemand column is their which is incremented as the song is played in application and this userdemand is used in recommendation as the song which is heard almost every time by the user.


Object Explorer
PC\SQLEXPRESS (SQL Server 9.0.3042 - sa
Databases
System Databases
election
javaa
pvr
PC\SQLEXPRESS.so... - dbo.songentry
PC\SQLEXPRESS.son...- dbo.song

Column Name	Data Type	Allow Nulls
id	int	☐
songupload	varchar(500)	✓
userdemand	int	✓
		☐

Fig.3.4 Songentry Table

In this table we have three attributes id which is the serial number of the records and is like a primary key of the table songupload is the file which is to be stored in the for running songs and userdemand to list the user opnion for the song

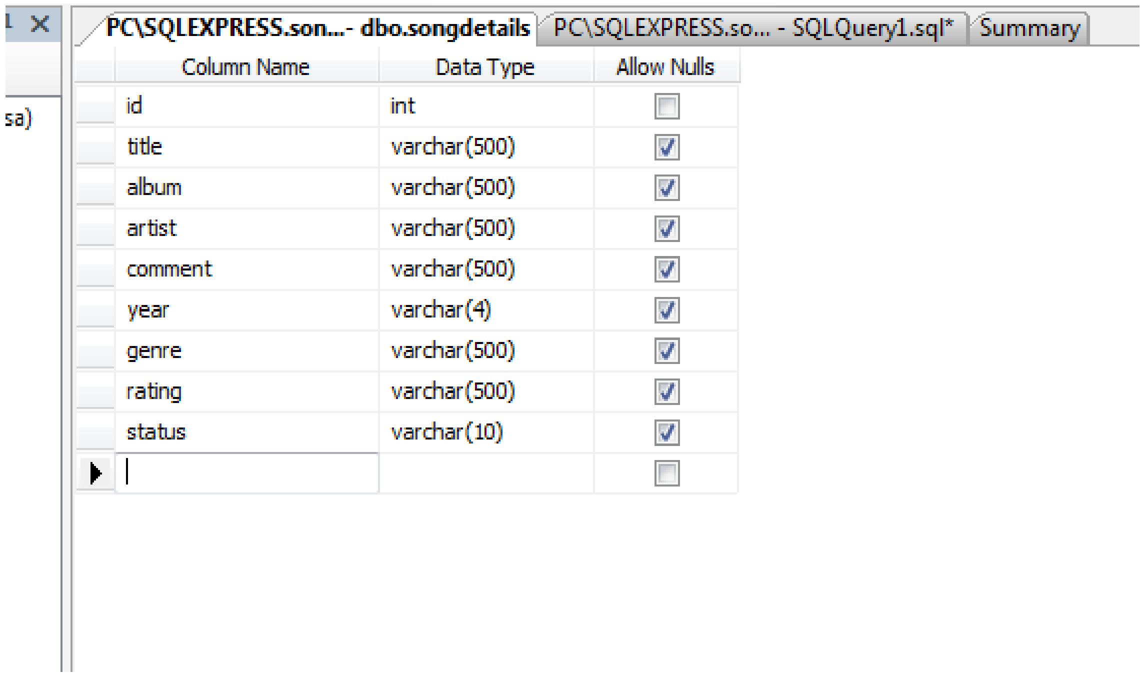


```
select * from songentry
```

	id	songupload	userdemand
1	32	F:\muzic\Movies\boly mix\01. SAPNE MAIN MILTE HAIN ...	9
2	33	F:\muzic\Movies\Bride n Prejudice\b&p1(www.songs.pk).mp3	17
3	34	F:\muzic\Movies\Chak de India\01 Sukhvinder Singh, Sali...	6
4	35	F:\muzic\Movies\Chak de India\02 Hema Sardesai - Badal ...	6
5	36	F:\muzic\Movies\Chak de India\05 Krishna & Salim Mercha...	25
6	37	F:\muzic\Movies\Dil Chahta Hai\Koi Kahe Kehta Rahe.mp3	20
7	38	F:\muzic\Movies\delhi6\Masakali.mp3	12
8	39	F:\muzic\Oldies\Kishore\PEHLI MULAKAT\PAL PAL DIL K...	12
9	40	F:\muzic\Oldies\Kishore\PEHLI MULAKAT\TERA MUJH S...	10
10	42	F:\muzic\Movies\Chandni chwok to china\04 - Tere Naina....	6
11	44	F:\muzic\Movies\Chandni chwok to china\03 - India Se Aa...	6
12	45	F:\muzic\Movies\boly mix\06 Dilli Ki Sardi - Remix - K.K & S...	10
13	47	F:\muzic\Movies\KISMAT KONNECTION\Kahin Na Laage....	294
14	48	F:\muzic\Movies\NAYAK\01 CHALO CHALE MITWA.mp3	0

Fig.3.5 Songentry Table's Entry

Songdetails is another table where the details of songs which are extracted from the mp3/music file are stored and a column of status is there which help in repeated playing of the song which is randomly selected as it may be there in database too. So if staus is 1 then that song is not played. Here we can see the table and how data is stored in it.

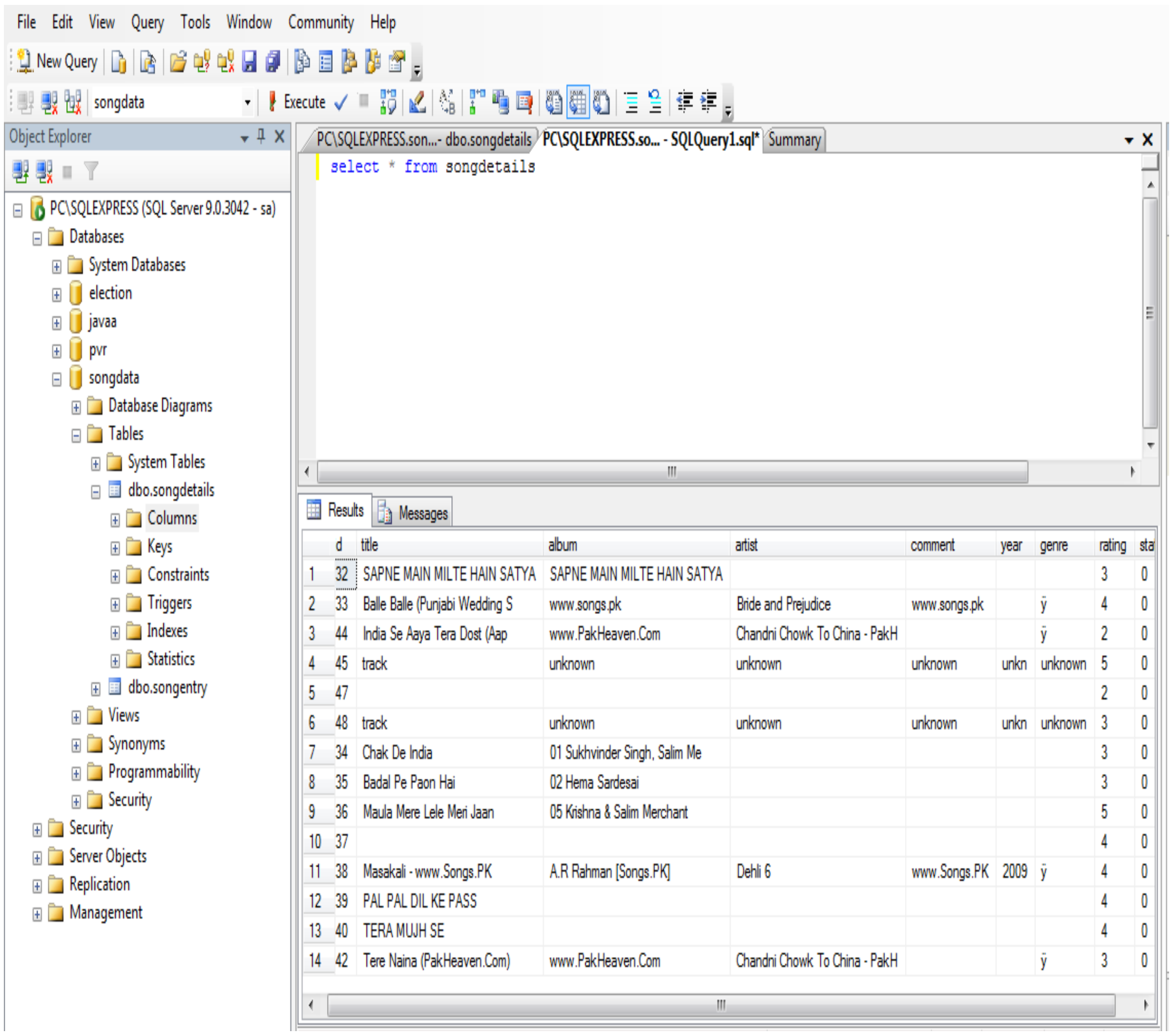


PC\SQLEXPRESS.son...- dbo.songdetails
PC\SQLEXPRESS.so... - SQLQuery1.sql*
Summary
sa)

Column Name	Data Type	Allow Nulls
id	int	☐
title	varchar(500)	✓
album	varchar(500)	✓
artist	varchar(500)	✓
comment	varchar(500)	✓
year	varchar(4)	✓
genre	varchar(500)	✓
rating	varchar(500)	✓
status	varchar(10)	✓
		☐

Fig.3.6Songdetails Table

In this table we nine attributes id for serial number which in this table also act as primary key and id of both the tables are same as extracted tags are stored in this table which are stored in ID3v1 tags.Status and Rating are the field which are not extracted but are provider to make the recommendation better.



```
select * from songdetails
```

	d	title	album	artist	comment	year	genre	rating	sta
1	32	SAPNE MAIN MILTE HAIN SATYA	SAPNE MAIN MILTE HAIN SATYA					3	0
2	33	Balle Balle (Punjabi Wedding S	www.songs.pk	Bride and Prejudice	www.songs.pk		ÿ	4	0
3	44	India Se Aaya Tera Dost (Aap	www.PakHeaven.Com	Chandni Chowk To China - PakH			ÿ	2	0
4	45	track	unknown	unknown	unknown	unkn	unknown	5	0
5	47							2	0
6	48	track	unknown	unknown	unknown	unkn	unknown	3	0
7	34	Chak De India	01 Sukhvinder Singh, Salim Me					3	0
8	35	Badal Pe Paon Hai	02 Hema Sardesai					3	0
9	36	Maula Mere Lele Meri Jaan	05 Krishna & Salim Merchant					5	0
10	37							4	0
11	38	Masakali - www.Songs.PK	A.R Rahman [Songs.PK]	Dehli 6	www.Songs.PK	2009	ÿ	4	0
12	39	PAL PAL DIL KE PASS						4	0
13	40	TERA MUJH SE						4	0
14	42	Tere Naina (PakHeaven.Com)	www.PakHeaven.Com	Chandni Chowk To China - PakH			ÿ	3	0

Fig.3.7 Songdetail Table's Entry

The above table shows the data which is stored with the metadata of the songs and used in recommendation of songs and checking double entry of the song in the table

3.4 Algorithm used in recommendation

```
-we have status , userdemand in database which is used in recommendation
-variable colcount(count total song recommended),i
-song is selected randomly from computer
-selected song file is checked whether it has tag stores in it or not
   File has 128 bytes for data
   If first three bytes are 'TAG'
     Then it contains the metadata in it
   Else no metadata
-when song is selected its recommendation is done accordingly
   If it contain tags then metadata is used  with userdemand and status and rating to recommend
      the Song
   Else only rating and userdemand and status are used
-playing of datagrid of songs
    the recommended songs are acquired by query of  database and it source is set to datagrid
    colcount counts total songs
    media_element_end is used for putting source in media element as it's the part which is in
    action when song is ending
    if(i<colcount-1)
      source to media element of next song is given
      userdemand is incremented
      status is set back to 0 if it is 1 in any case     //queries are used for all this
-songs from the datagrid is played one by one and at last the last song is played repeatedly
```

Chapter 4 Implementation

4.1 Feasibility Study

Music recommendation System is like a boom for the music lover .Not only music lover even for the refreshment Music recommendation system is very helpful. This is a highly feasible project where music is the strength of the system as every person in world has love for music secondly nowadays every music file have metadata stored in it which help in this type of systems.Many software application like itunes etc are very much in the market and even websites like musicovery is high in demand .ID3 tags are used in this implementation which can easily extracted from the file which contain data .Reaseaarch paper available provide knowledge both of content and metadata based system plus even collaborative system can also be developed.In the system we used both user demand and metadata for recommendation so it feasibility gets incremented .ID3V1 is used for extraction where artist ,genre, year, album, comment ,title are stored plus rate is used which user can provide .On having songs in thousand number recommendation help in selection of few favorite songs which make it sustain in market even more strongly.There various source on internet for any enquiry in developing the software.the only weakness is that it need to use two softwares Visual Studio 2010 and MYSQL .High company are demanding for such innovative application and public is seeing forward for these recommendation systems.Hence resource are available for every type of problem as per the refrences we have mentioned and these project have scope in future too. Even the software used are user friendly and application developed too is userfriendly.

4.2 Requirement Anlalysis

To make a music recommendation system we need to have

- Window with windows7 as OS
- Windows Presentation Foundation using C#
- .Net 3.5
- MY SQL
- Visual Studio 2010

4.2.1 Windows Presentation Foundation:

Developed by Microsoft, the Windows Presentation Foundation (or WPF) is a computer-software graphical subsystem for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Rather than relying on the older GDI subsystem, WPF utilizes DirectX. WPF attempts to provide a consistent programming model for building applications and provides a separation between the user interface and the business logic. It resembles similar XML-oriented object models, such as those implemented in XUL and SVG.

WPF employs XAML, an XML-based language, to define and link various UI elements. WPF applications can also be deployed as standalone desktop programs, or hosted as an embedded object in a website. WPF aims to unify a number of common user interface elements, such as 2D/3D rendering, fixed and adaptive documents, typography, vector graphics, runtime animation, and pre-rendered media. These elements can then be linked and manipulated based on various events, user interactions, and data bindings.

WPF runtime libraries are included with all versions of Microsoft Windows since Windows Vista and Windows Server 2008. Users of Windows XP SP2/SP3 and Windows Server 2003 can optionally install the necessary libraries.

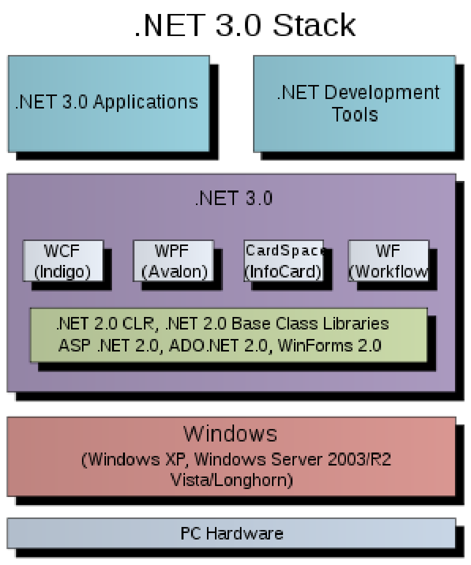


Fig.4.1 .NET 3.0 Stack

4.2.2 XAML:

Following the success of markup languages for web development, WPF introduces a new language known as eXtensible Application Markup Language (XAML; /ˈzæməl/), which is based on XML. XAML is designed as a more efficient method of developing application user interfaces.

The specific advantage that XAML brings to WPF is that XAML is a completely declarative language, allowing the developer (or designer) to describe the behavior and integration of components without the use of procedural programming. Although it is rare that an entire application will be built completely in XAML, the introduction of XAML allows application designers to more effectively contribute to the application development cycle. Using XAML to develop user interfaces also allows for separation of model and view, which is considered a good architectural principle. In XAML, elements and attributes map to classes and properties in the underlying APIs.

As in web development, both layouts and specific themes are well suited to markup, but XAML is not required for either. Indeed, all elements of WPF may be coded in a .NET language (C#, VB.NET). The XAML code can ultimately be compiled into a managed assembly in the same way all .NET languages are.

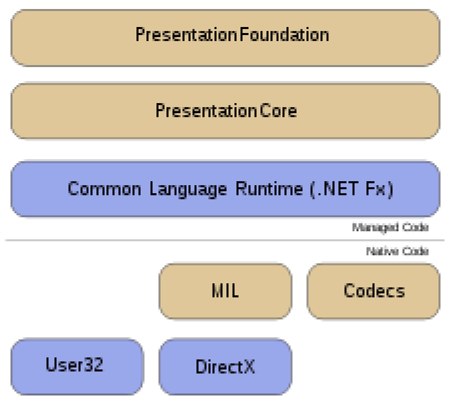


Fig.4.2 WPF

The WPF architecture: Blue elements are Windows components; brown ones are WPF components.

4.3 Designs Of Music Recommendation Systems

4.3.1 Use Case Diagram

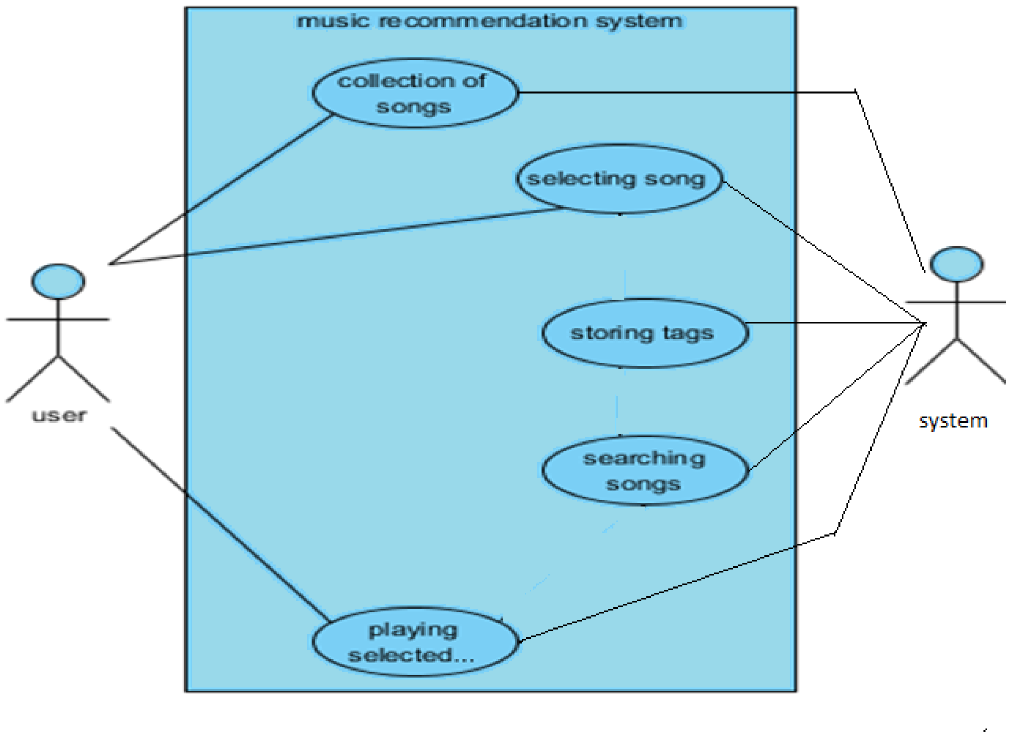


Fig 4.3 Use case of Music Recommendation System

Use Case describes the way in which user interacts with the system and functions. There are various functions like collection of songs, selecting of song, storing tags, etc. user interacts with collection of songs and thereby selects the song, whereas the system listens to the selected song and store tags. Recommendation of a new song is done on the basis of tags extracted. Finally user listens to the song recommended by the system.

4.3.2 Data Flow Diagram

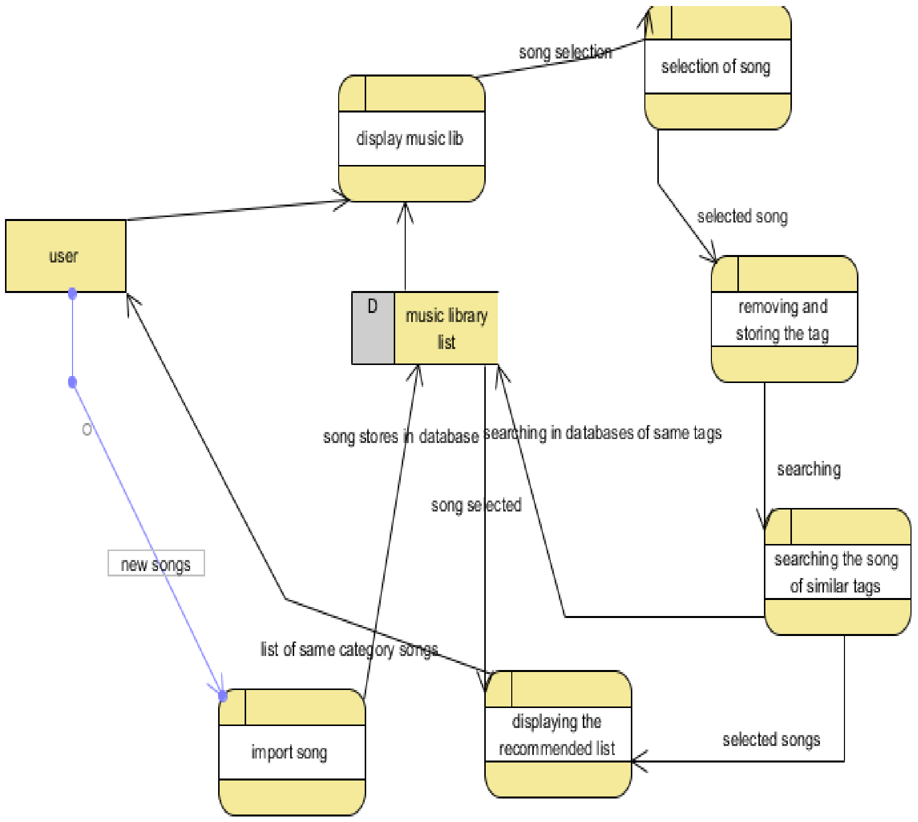


Fig.4.4. Data flow diagram of MRS

As in the diagram the process will be as the user enter any random song in the interface and the system will search the similar type of the songs by first extracting the tags from the song and then the playlist of similar type of songs is created by the system

4.3.3 Sequence Diagram

rary

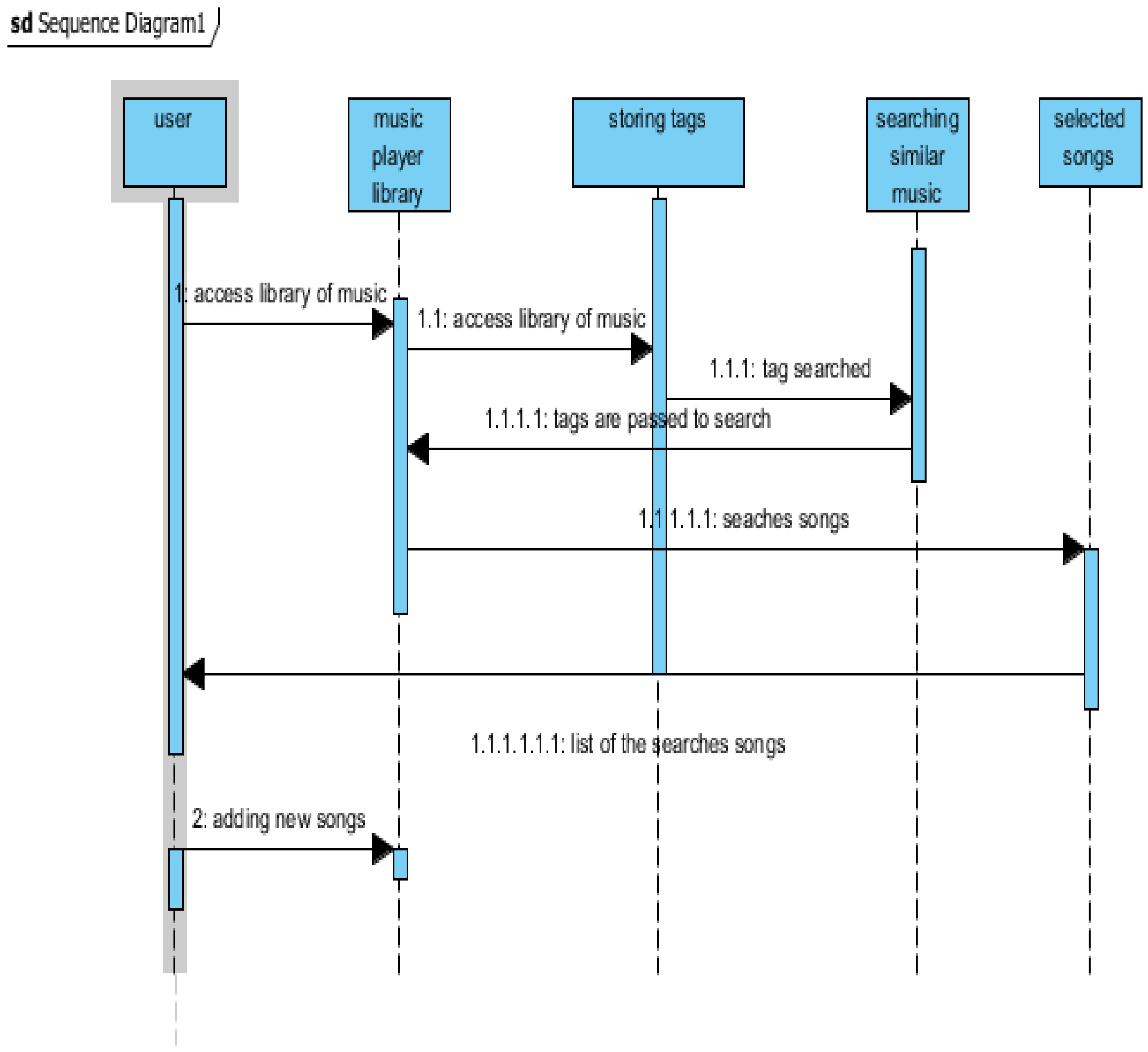


Fig.4.5. Sequence diagram of MRS

In the above figure, we can see that user performs two actions on the system. One is selection of the song to get recommended music and second is addition and deletion of the songs. For the first step user selects song randomly by accessing the computer’s music library. Then the tags of the song are extracted and are used further for recommendation. Tags may or may not be present. Accordingly recommendation of the songs is done.

4.3.4 Flow Chart

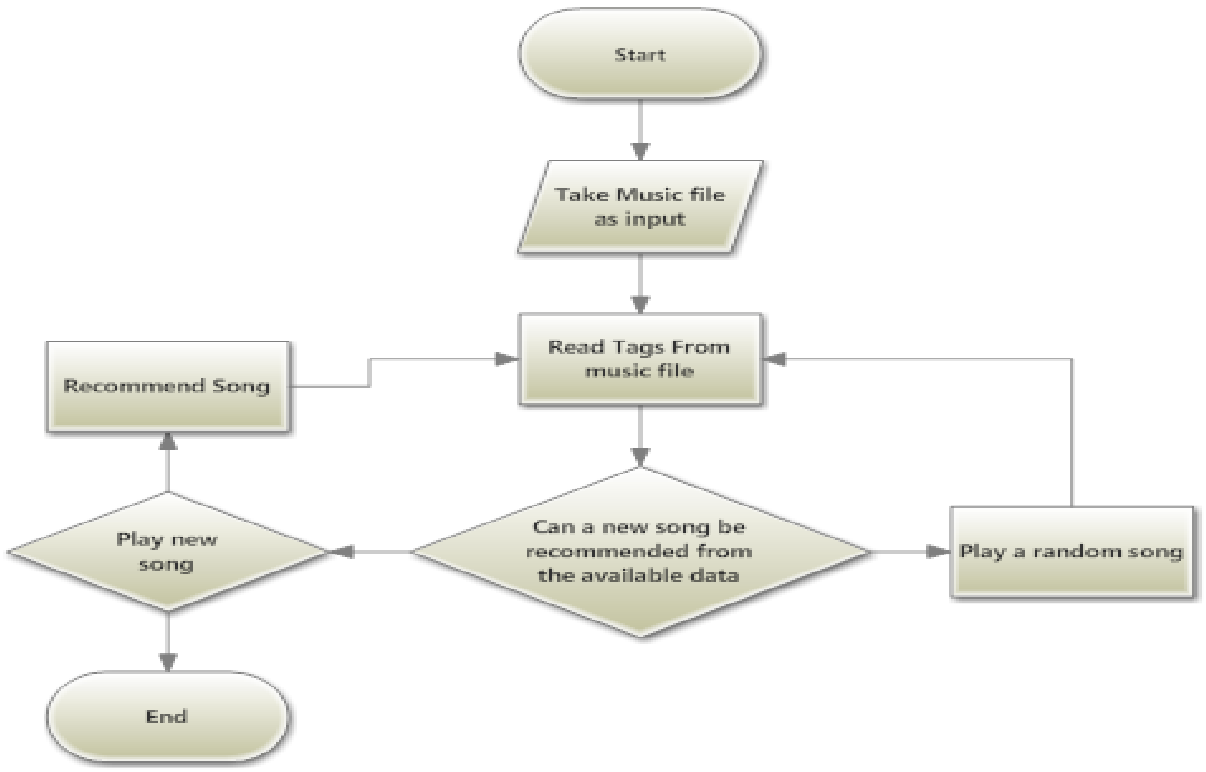

Start
Take Music file as input
Read Tags From music file
Recommend Song
Can a new song be recommended from the available data
Play a random song
Play new song
End

Fig.4.6 Flow Chart of MRS

Flowchart describes the steps by which recommendation is done. Firstly the user takes a music file as input. Tags are read from the music file and song is recommended on the basis of tags extracted. If further a new song can be recommended from the available data then it is done otherwise recommended song is played again.

4.4 Simulation

4.4.1 mainwindow.xaml

```
<Viewbox Stretch="Uniform" >

<MediaElement MediaEnded="mediaElement_MediaEnded"
MediaOpened="mediaElement_MediaOpened" LoadedBehavior="Manual"
Height="112" HorizontalAlignment="Right" Margin="0,0,242,213"
Name="mediaElement" VerticalAlignment="Bottom" Width="196" Opacity="1"
AllowDrop="True" Stretch="Uniform" IsMuted="False" Grid.ColumnSpan="2" >
</MediaElement>

<Button Content="Open" Height="24" HorizontalAlignment="Left" Margin="8,27,0,0"
Name="openFileButton" VerticalAlignment="Top" Width="61"
Click="openFileButton_Click" Background="White" Foreground="Blue" />

<Slider Height="30" HorizontalAlignment="Right" Margin="0,239,265,0"
Name="volumeSlider" VerticalAlignment="Top" Width="87" Value="0.5"
Maximum="1" SmallChange="0.01" LargeChange="0.1" Background="White"
Foreground="Blue" IsMoveToPointEnabled="True" Grid.Column="1">
</Slider>

<TextBlock Height="23" HorizontalAlignment="Left" Margin="10,241,0,0"
Foreground="Blue" Name="textBlock1" Text="Volume" VerticalAlignment="Top"
Grid.Column="1" />
<TextBlock Height="25" HorizontalAlignment="Left" Margin="11,204,0,0"
Name="currentTimeTextBlock" Text="00:00" VerticalAlignment="Top" Width="45"
Background="White" Foreground="Blue"></TextBlock>
```

```
<ProgressBar Height="9" Margin="61,208,0,0" Name="progressBar" VerticalAlignment="Top" Background="white" Foreground="DarkBlue" HorizontalAlignment="Left" Width="521" Grid.ColumnSpan="2" />

<Slider PreviewMouseLeftButtonUp="seekSlider_PreviewMouseLeftButtonUp" PreviewMouseLeftButtonDown="seekSlider_PreviewMouseLeftButtonDown" MouseLeftButtonUp="seekSlider_MouseLeftButtonUp" Margin="62,202,0,173" Name="seekSlider" Foreground="Blue" HorizontalAlignment="Left" Width="522" IsMoveToPointEnabled="True" TickPlacement="None" SnapsToDevicePixels="False" IsManipulationEnabled="True" IsSnapToTickEnabled="False" IsSelectionRangeEnabled="False" IsTabStop="False" AutoToolTipPlacement="None" AllowDrop="False" Grid.ColumnSpan="2" ValueChanged="seekSlider_ValueChanged" Height="22"></Slider>

<DataGrid AutoGenerateColumns="False" Height="108" HorizontalAlignment="Left" Background="White" Margin="0,280,0,0" Name="dataGrid1" VerticalAlignment="Top" Width="678" ItemsSource="{Binding}" Grid.ColumnSpan="2" RowBackground="#00F5ECEC" Foreground="Blue" HorizontalGridLinesBrush="White" VerticalGridLinesBrush="White" RowHeight="17" FontSize="12">
<DataGrid.Columns >
<DataGridTextColumn Header="Id" Binding="{Binding id}" Width="40" Visibility="Hidden" Foreground="Blue">
 </DataGridTextColumn>
<DataGridTextColumn Header="song name" Binding="{Binding title}" Width="638" Foreground="Blue" >
</DataGridTextColumn>
</DataGrid.Columns>
</DataGrid>
<DataGrid.Columns>
```

```
<DataGridTextColumn Header="Id" Binding="{Binding id}" Width="40"
></DataGridTextColumn>
<DataGridTextColumn Header="song name" Binding="{Binding songupload}"
Width="110"></DataGridTextColumn>
</DataGrid.Columns>
</DataGrid>

<Button Content="Button" Grid.ColumnSpan="2" Height="167"
HorizontalAlignment="Left" Margin="178,33,0,0" Name="button2"
VerticalAlignment="Top" Width="273" Background="White">
<Button.Template>
<ControlTemplate>
<Image Source="C:\Users\Prabjot\Documents\Visual Studio
2010\Projects\MediaPlayer_Beta\MediaPlayer_Beta\images (23).jpg"
Stretch="Fill"></Image>
</ControlTemplate>
</Button.Template>
</Button>

<Grid Grid.Column="1" Height="121" HorizontalAlignment="Left"
Margin="182,104,0,0" Name="gridadd" Visibility="Hidden" VerticalAlignment="Top"
Width="206" Background="SkyBlue" >

<Grid.ColumnDefinitions>
<ColumnDefinition Width="229*" />
<ColumnDefinition Width="0*" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="31*" />
<RowDefinition Height="*" />
<RowDefinition Height="55*" />
```

```
<RowDefinition Height="2*" />
</Grid.RowDefinitions>

<Button Content="open" Height="27" HorizontalAlignment="Right" Margin="0,20,6,0" Name="getsong" VerticalAlignment="Top" Width="59" Background="White" Foreground="Blue" Click="getsong_Click" Grid.RowSpan="3" />

<Button Content="addsong" Height="18" HorizontalAlignment="Left" Margin="72,46,0,0" Name="AddSong" VerticalAlignment="Top" Width="48" Background="White" Foreground="Blue" Click="AddSong_Click" Grid.Row="2" />



<TextBox Height="23" HorizontalAlignment="Left" Margin="83,9,0,0" Background="White" Foreground="Blue" Name="rate" VerticalAlignment="Top" Width="54" Grid.Row="2" />

<Button Content="Button" Name="button1" Click="button1_Click_1" Background="White" Foreground="Blue" Margin="193,0,2,26">
<Button.Template>
<ControlTemplate>
<Image Source="C:\Users\Prabjot\Documents\Visual Studio 2010\Projects\MediaPlayer_Beta\MediaPlayer_Beta\download (5).jpg" Stretch="Fill"></Image>
</ControlTemplate>
</Button.Template>

</Button>
</Grid>
```

```
<Grid Grid.Column="1" Height="166" HorizontalAlignment="Left" Margin="184,231,0,0" Name="grid1" VerticalAlignment="Top" Width="220" Visibility="Hidden" Background="SkyBlue" >

<DataGrid AutoGenerateColumns="False" HorizontalAlignment="Left" Background="White" Margin="6,17,0,0" Name="deletegrid1" VerticalAlignment="Top" Visibility="Hidden" DataContext="{Binding}" Height="133" Width="206" ItemsSource="{Binding}" HorizontalGridLinesBrush="White" VerticalGridLinesBrush="White" Foreground="Blue">

<DataGrid.Columns>
<DataGridTextColumn Header="Id" Binding="{Binding id}" Width="40"></DataGridTextColumn>
<DataGridTextColumn Header="song name" Binding="{Binding songupload}" Width="110"></DataGridTextColumn>
<DataGridTemplateColumn>
<DataGridTemplateColumn.CellTemplate>
<DataTemplate>
<Button Content="del" Click="delete1" Name="del" Command="Delete"></Button>
</DataTemplate>
</DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
</DataGrid.Columns>
</DataGrid>

<Button Content="Button" Height="21" HorizontalAlignment="Left" Margin="192,18,0,0" Name="button3" VerticalAlignment="Top" Width="20" Click="button3_Click_1">
<Button.Template>
<ControlTemplate>
```

```
<Image Source="C:\Users\Prabjot\Documents\Visual Studio
2010\Projects\MediaPlayer_Beta\MediaPlayer_Beta\download (5).jpg"
Stretch="Fill"></Image>
</ControlTemplate>
</Button.Template>
</Button>
</Grid>

</Viewbox>
</Window>
```

4.4.2 mainwindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Configuration;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Win32;
using System.IO;
using System.Windows.Threading;
using System.Threading;
using System.Data;
using System.Data.SqlClient;



namespace MediaPlayer_Beta
{
  /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
```

```
 public partial class MainWindow : Window
    {
//  class for tad storage
        class musicID3tag
        {
            public byte[] TAGID = new byte[3];
            public byte[] title = new byte[30];
            public byte[] artist = new byte[30];
            public byte[] album = new byte[30];
            public byte[] year = new byte[4];
            public byte[] comment = new byte[30];
            public byte[] genre = new byte[1];
        }

      DispatcherTimer timer;

        public delegate void timerTick();
        timerTick tick;
        bool val = false;
        bool isDragging = false;
        bool fileIsPlaying = false;
        string sec, min, hours;
        int colcount, i = 0,s=0,p=0;

        public MainWindow()
        {
            InitializeComponent();

            timer = new DispatcherTimer();
            timer.Interval = TimeSpan.FromSeconds(1);
            timer.Tick += new EventHandler(timer_Tick);
```

```
           tick = new timerTick(changeStatus);

        }

        void timer_Tick(object sender, EventArgs e)
        {
           Dispatcher.Invoke(tick);
        }



//Recommending of the songs
        public void playgrid()
        {
           string songupload = currentsong.Text;
           string title, artist, album, year, genre, comment;
           string cmdstring = string.Empty;
           string cmdstring4 = string.Empty;
           string cmdstring3 = string.Empty;
           string cmdstring2 = string.Empty;
           string cmdstring1 = string.Empty;
           //using (SqlConnection con = new SqlConnection(ConnectionString))

            SqlConnection ConnectionString1 = new
SqlConnection("server=PC\\SQLEXPRESS;uid=sa;pwd=student;database=songdata");

           using (FileStream fs = File.OpenRead(songupload))


              if (fs.Length >= 128)
              {
```

```
musicID3tag tag = new musicID3tag();
fs.Seek(-128, SeekOrigin.End);
fs.Read(tag.TAGID, 0, tag.TAGID.Length);
fs.Read(tag.title, 0, tag.title.Length);
fs.Read(tag.artist, 0, tag.artist.Length);
fs.Read(tag.album, 0, tag.album.Length);
fs.Read(tag.year, 0, tag.year.Length);
fs.Read(tag.comment, 0, tag.comment.Length);
fs.Read(tag.genre, 0, tag.genre.Length);
string theTAGID = Encoding.Default.GetString(tag.TAGID);


if (theTAGID.Equals("TAG"))
{
   title = Encoding.Default.GetString(tag.title);
   artist = Encoding.Default.GetString(tag.album);
   album = Encoding.Default.GetString(tag.artist);
   year = Encoding.Default.GetString(tag.year);
   comment = Encoding.Default.GetString(tag.comment);
   genre = Encoding.Default.GetString(tag.genre);


   cmdstring2 = "update songdetails set status=1 where(artist=@artist and
album=@album and genre=@genre and title=@title and year=@year and
comment=@comment and id=(select id from songentry where
songupload=@songupload))";
  SqlCommand cmd2 = new SqlCommand(cmdstring2, ConnectionString1);
  cmd2.Parameters.Add("@title", SqlDbType.VarChar, 500).Value = title;
  cmd2.Parameters.Add("@album", SqlDbType.VarChar, 500).Value =
album;
  cmd2.Parameters.Add("@artist", SqlDbType.VarChar, 500).Value = artist;
```

```
                    cmd2.Parameters.Add("@genre",   SqlDbType.VarChar,   500).Value  =
genre;
                    cmd2.Parameters.Add("@year", SqlDbType.VarChar, 4).Value = year;
                    cmd2.Parameters.Add("@comment", SqlDbType.VarChar, 500).Value =
                    comment;
                    cmd2.Parameters.Add("@songupload",  SqlDbType.VarChar,  500).Value
=
                   songupload;
                    ConnectionString1.Open();
                    cmd2.ExecuteNonQuery();
                    ConnectionString1.Close();

                    cmdstring  =  "select  *  from  songdetails  where  ((artist=@artist  and
album=@album   and   genre=@genre)or   artist=@artist   or   rating='4'   or   rating='5'or
id=(select id from songentry where userdemand>200)) and status=0 order by id  ";
                    SqlCommand cmd = new SqlCommand(cmdstring, ConnectionString1);

                    cmd.Parameters.Add("@album",   SqlDbType.VarChar,   500).Value   =
album;
                    cmd.Parameters.Add("@artist", SqlDbType.VarChar, 500).Value = artist;
                    cmd.Parameters.Add("@genre",   SqlDbType.VarChar,   500).Value   =
genre;

                    ConnectionString1.Open();
                    // cmd2.ExecuteNonQuery();
                    cmd.ExecuteNonQuery();

                    SqlDataAdapter adt = new SqlDataAdapter(cmd);
                    DataTable dt = new DataTable("songdetails");
                    adt.Fill(dt);
                    dataGrid1.DataContext = dt;
```

```
            dataGrid1.ItemsSource = dt.DefaultView;
            ConnectionString1.Close();

            ConnectionString1.Open();
            dataGrid3.DataContext = null;
            cmdstring1 = "select s.id,songupload from songdetails s,songentry where
(s.id=songentry.id) and (((artist=@artist and album=@album and genre=@genre)or
artist=@artist or rating='4' or rating='5'or songentry.userdemand>200) )and status=0
order by id ";
            SqlCommand cmd1 = new SqlCommand(cmdstring1, ConnectionString1);
            cmd1.Parameters.Add("@album", SqlDbType.VarChar, 500).Value =
album;
            cmd1.Parameters.Add("@artist", SqlDbType.VarChar, 500).Value =
artist;
            cmd1.Parameters.Add("@genre", SqlDbType.VarChar, 500).Value =
genre;
            cmd1.ExecuteNonQuery();
            SqlDataAdapter adt1 = new SqlDataAdapter(cmd1);
            DataTable dt1 = new DataTable("songentry");
            adt1.Fill(dt1);
            dataGrid3.DataContext = dt1;
            dataGrid3.ItemsSource = dt1.DefaultView;
            ConnectionString1.Close();



          }
          else
          {
            cmdstring3 = "update songdetails set status=1 where id=(select id from
songentry where songupload=@songupload)";
            SqlCommand cmd3 = new SqlCommand(cmdstring3, ConnectionString1);
        cmd3.Parameters.Add("@songupload", SqlDbType.VarChar, 500).Value = songupload;

        cmdstring = "select * from songdetails where (rating='4' or rating='5') and status=0  ";

            SqlCommand cmd = new SqlCommand(cmdstring, ConnectionString1);
            ConnectionString1.Open();
            cmd3.ExecuteNonQuery();
            cmd.ExecuteNonQuery();
            SqlDataAdapter adt = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable("songdetails");
            adt.Fill(dt);
            dataGrid1.DataContext = dt;
            dataGrid1.ItemsSource = dt.DefaultView;
            dataGrid3.ItemsSource = null;
            cmdstring1 = "select s.id,songupload from songdetails s,songentry where songentry.id=s.id and (((rating='4' or rating='5') or songentry.userdemand>200)) and status=0 ";
            SqlCommand cmd1 = new SqlCommand(cmdstring1, ConnectionString1);
            cmd1.ExecuteNonQuery();
            SqlDataAdapter adt1 = new SqlDataAdapter(cmd1);
            DataTable dt1 = new DataTable("songentry");
            adt1.Fill(dt1);
            dataGrid3.DataContext = dt1;
            dataGrid3.ItemsSource = dt1.DefaultView;
            ConnectionString1.Close();
          }
        }
        cmdstring4 = "update songdetails set status=0 where status=1";
        SqlCommand cmd4 = new SqlCommand(cmdstring4, ConnectionString1);
        ConnectionString1.Open();
        cmd4.ExecuteNonQuery();
        ConnectionString1.Close();
        colcount = dataGrid3.Items.Count;
        //demo.Text =Convert.ToString(colcount);
    }

    //open the file
    private void openFileButton_Click(object sender, RoutedEventArgs e)
    {
        Stream checkStream = null;
        OpenFileDialog dlg = new OpenFileDialog();
        dlg.Filter = "All Supported File
Types(*.mp3,*.wav,*.mpeg,*.wmv,*.avi)|*.mp3;*.wav;*.mpeg;*.wmv;*.avi";
        // Show open file dialog box
        if ((bool)dlg.ShowDialog())
        {
            try
            {
                if ((checkStream = dlg.OpenFile()) != null)
                {
                    //currentsong.Text = dlg.FileName;
                    mediaElement.Source = new Uri(dlg.FileName);
                    currentsong.Text = dlg.FileName;
                }
                Thread.Sleep(50);
                mediaElement.Close();
                mediaElement.Play();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
            }
        }
        playgrid();
    }

    //occurs when the file is opened
    public void mediaElement_MediaOpened(object sender, RoutedEventArgs e)
    {
        timer.Start();
        fileIsPlaying = true;

        openMedia();

    }

    //occurs when the file is done playing
    private void mediaElement_MediaEnded(object sender, RoutedEventArgs e)
    {

        mediaElement.Stop();
        volumeSlider.ValueChanged -= new
RoutedPropertyChangedEventHandler<double>(volumeSlider_ValueChanged);

        try
        {

            if (fileIsPlaying == true)
                {
                   if (i < (colcount - 1))
                   {
                      TextBlock x = dataGrid3.Columns[1].GetCellContent(dataGrid3.Items[i])
as TextBlock;
                      mediaElement.Source = new System.Uri(x.Text);
                      //                    TextBlock                    y                    =
dataGrid3.Columns[2].GetCellContent(dataGrid3.Items[i]) as TextBlock;



                      // int userdemand = Convert.ToInt32(y.Text);
                      //if(userdemand>2500)
                      //    userdemand = userdemand + 1;



                      TextBlock z = dataGrid3.Columns[0].GetCellContent(dataGrid3.Items[i])
as TextBlock;
                      int id = Convert.ToInt32(z.Text);
                      string cmdstring = string.Empty;
                      string cmdstring1 = string.Empty;
                      SqlConnection            ConnectionString1            =            new
SqlConnection("server=PC\\SQLEXPRESS;uid=sa;pwd=student;database=songdata");
                      cmdstring = "update songentry Set userdemand=userdemand+1 where
(id=@id)";

                      SqlCommand cmd = new SqlCommand(cmdstring, ConnectionString1);

                      //    cmd.Parameters.Add("@userdemand",    SqlDbType.Int,    16).Value
=userdemand;

                      cmd.Parameters.Add("@id", SqlDbType.Int, 4).Value = id;
                ConnectionString1.Open();
                cmd.ExecuteNonQuery();

                ConnectionString1.Close();

              }
              //Thread.Sleep(50);
              mediaElement.Close();

              mediaElement.Play();
              timer.Start();


            }
          }
          catch (Exception ex)
          {
            MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
          }
          i++;
        }


        //initialize properties of file
        void InitializePropertyValues()
        {
          mediaElement.Volume = (double)volumeSlider.Value;

        }
        private void getsong_Click(object sender, RoutedEventArgs e)
```

```
    {
      string filepath;
      Stream checkStream = null;
      OpenFileDialog dlg = new OpenFileDialog();
      dlg.Filter = "All Supported File
Types(*.mp3,*.wav,*.mpeg,*.wmv,*.avi)|*.mp3;*.wav;*.mpeg;*.wmv;*.avi";
      // Show open file dialog box
      if ((bool)dlg.ShowDialog())
      {
        try
        {
          if ((checkStream = dlg.OpenFile()) != null)
          {

            song.Text = dlg.FileName;
            filepath = dlg.FileName;
          }
        }
        catch (Exception ex)
        {
          MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
        }
      }
    }

    private void delete_Click(object sender, RoutedEventArgs e)
    {

      try
      {
```

```
                grid1.Visibility = Visibility.Visible;
                deletegrid1.Visibility = Visibility.Visible;
                string cmdstring = string.Empty;

                SqlConnection              ConnectionString1              =              new
SqlConnection("server=PC\\SQLEXPRESS;uid=sa;pwd=student;database=songdata");

                cmdstring = "select * from songentry";
                SqlCommand cmd = new SqlCommand(cmdstring, ConnectionString1);



                ConnectionString1.Open();
                cmd.ExecuteNonQuery();

                SqlDataAdapter adt = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable("songentry");

                adt.Fill(dt);
                deletegrid1.DataContext = dt;
                //deletegrid1.ItemsSource = dt.DefaultView;
                ConnectionString1.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
            }
        }


      //Addition of the songs
```

```
    private void AddSong_Click(object sender, RoutedEventArgs e)
     {

        string songupload = song.Text, title, album, artist, genre, year, comment;



        int rated = Convert.ToInt32(rate.Text);
        if (rated < 6)
        {
            int userdemand = 0;
            string cmdstring = string.Empty;

            SqlConnection ConnectionString1 = new
SqlConnection("server=PC\\SQLEXPRESS;uid=sa;pwd=student;database=songdata");

            cmdstring = "insert into songentry(songupload,userdemand)
values(@songupload,@userdemand)";
            SqlCommand cmd = new SqlCommand(cmdstring, ConnectionString1);
            cmd.Parameters.Add("@songupload", SqlDbType.VarChar, 500).Value =
songupload;
            cmd.Parameters.Add("@userdemand", SqlDbType.Int, 16).Value =
userdemand;
            ConnectionString1.Open();
            cmd.ExecuteNonQuery();
            ConnectionString1.Close();

            string ss = song.Text;

            using (FileStream fs = File.OpenRead(ss))
```

```
if (fs.Length >= 128)
{
    musicID3tag tag = new musicID3tag();
    fs.Seek(-128, SeekOrigin.End);
    fs.Read(tag.TAGID, 0, tag.TAGID.Length);
    fs.Read(tag.title, 0, tag.title.Length);
    fs.Read(tag.artist, 0, tag.artist.Length);
    fs.Read(tag.album, 0, tag.album.Length);
    fs.Read(tag.year, 0, tag.year.Length);
    fs.Read(tag.comment, 0, tag.comment.Length);
    fs.Read(tag.genre, 0, tag.genre.Length);
    string theTAGID = Encoding.Default.GetString(tag.TAGID);


    if (theTAGID.Equals("TAG"))
    {
        title = Encoding.Default.GetString(tag.title);
        artist = Encoding.Default.GetString(tag.album);
        album = Encoding.Default.GetString(tag.artist);
        year = Encoding.Default.GetString(tag.year);
        comment = Encoding.Default.GetString(tag.comment);
        genre = Encoding.Default.GetString(tag.genre);

        // int rated = Convert.ToInt32(rate.Text);
    }
    else
    {
        //  int i = 1;
        title = "track";
        artist = "unknown";
        album = "unknown";
        year = "unknown";
        comment = "unknown";
        genre = "unknown";

    }
    string ratedd = rate.Text;

    string cmdstring1 = string.Empty;

    cmdstring1 = "insert into
songdetails(title,artist,album,year,comment,genre,rating,status)
values(@title,@artist,@album,@year,@comment,@genre,@ratedd,'0')";
    SqlCommand cmd1 = new SqlCommand(cmdstring1, ConnectionString1);

    cmd1.Parameters.Add("@title", SqlDbType.VarChar, 500).Value = title;
    cmd1.Parameters.Add("@album", SqlDbType.VarChar, 500).Value =
album;
    cmd1.Parameters.Add("@artist", SqlDbType.VarChar, 500).Value =
artist;
    cmd1.Parameters.Add("@genre", SqlDbType.VarChar, 500).Value =
genre;
    cmd1.Parameters.Add("@year", SqlDbType.VarChar, 4).Value = year;
    cmd1.Parameters.Add("@comment", SqlDbType.VarChar, 500).Value =
comment;
    cmd1.Parameters.Add("@ratedd", SqlDbType.VarChar, 500).Value =
ratedd;
    ConnectionString1.Open();
    cmd1.ExecuteNonQuery();
    ConnectionString1.Close();
```

```
                }

            MessageBox.Show("song added");
            rate.Text = "";
            song.Text = "";

        }
        else
        {

            MessageBox.Show("rate should be less than 6");
            rate.Text = "";
            song.Text = "";

        }
    }

//deletion of the song
    private void delete1(object sender, RoutedEventArgs e)
    {
        int row, id;
        row = deletegrid1.SelectedIndex;

        TextBlock  x  =  deletegrid1.Columns[0].GetCellContent(deletegrid1.Items[row])
as TextBlock;
        //if (x != null)
        // MessageBox.Show(x.Text);

        id = Convert.ToInt32(x.Text);
```

```
            string cmdstring1 = string.Empty;
            string cmdstring2 = string.Empty;
            SqlConnection ConnectionString1 = new
SqlConnection("server=PC\\SQLEXPRESS;uid=sa;pwd=student;database=songdata");

            cmdstring1 = "delete from songentry where id=@id";
            cmdstring2 = "delete from songdetails where id=@id";
            SqlCommand cmd1 = new SqlCommand(cmdstring1, ConnectionString1);
            SqlCommand cmd2 = new SqlCommand(cmdstring2, ConnectionString1);
            cmd1.Parameters.Add("@id", SqlDbType.Int, 4).Value = id;
            cmd2.Parameters.Add("@id", SqlDbType.Int, 4).Value = id;
            ConnectionString1.Open();
            cmd1.ExecuteNonQuery();
            cmd2.ExecuteNonQuery();
            ConnectionString1.Close();
        }


//To play the next song
        private void next_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (i == 0 && s==0)
                {
                    i = 0;
                    s=1;
                }

                else
                {
```

```
                if ((i == 0 && s == 1) || (i > 0 && i < colcount - 2))
                    i = i + 1;
                else
                {
                    i = 0;
                    s = 0;
                }
            }


            TextBlock x = dataGrid3.Columns[1].GetCellContent(dataGrid3.Items[i]) as
TextBlock;
            mediaElement.Source = new System.Uri(x.Text);



            mediaElement.Close();


            mediaElement.Play();
            timer.Start();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
        }


    }
//To play previous song
    private void button4_Click(object sender, RoutedEventArgs e)
    {
        try
```

```
            {
                if (i == 0 && p == 0)
                {
                    i = colcount-2;
                    p = 1;
                }

                else
                {
                    if (i > 1 && i < colcount - 1)
                        i = i - 1;
                    else
                    {
                        i = 0;
                        p = 0;
                    }
                }



                TextBlock  x  =  dataGrid3.Columns[1].GetCellContent(dataGrid3.Items[i])  as
        TextBlock;
                mediaElement.Source = new System.Uri(x.Text);



                mediaElement.Close();


                mediaElement.Play();
                timer.Start();
            }
            catch (Exception ex)
            {
```

```
            MessageBox.Show("Error: Could not read file from disk. Original error: " +
ex.Message);
        }
}
```

4.5 Output

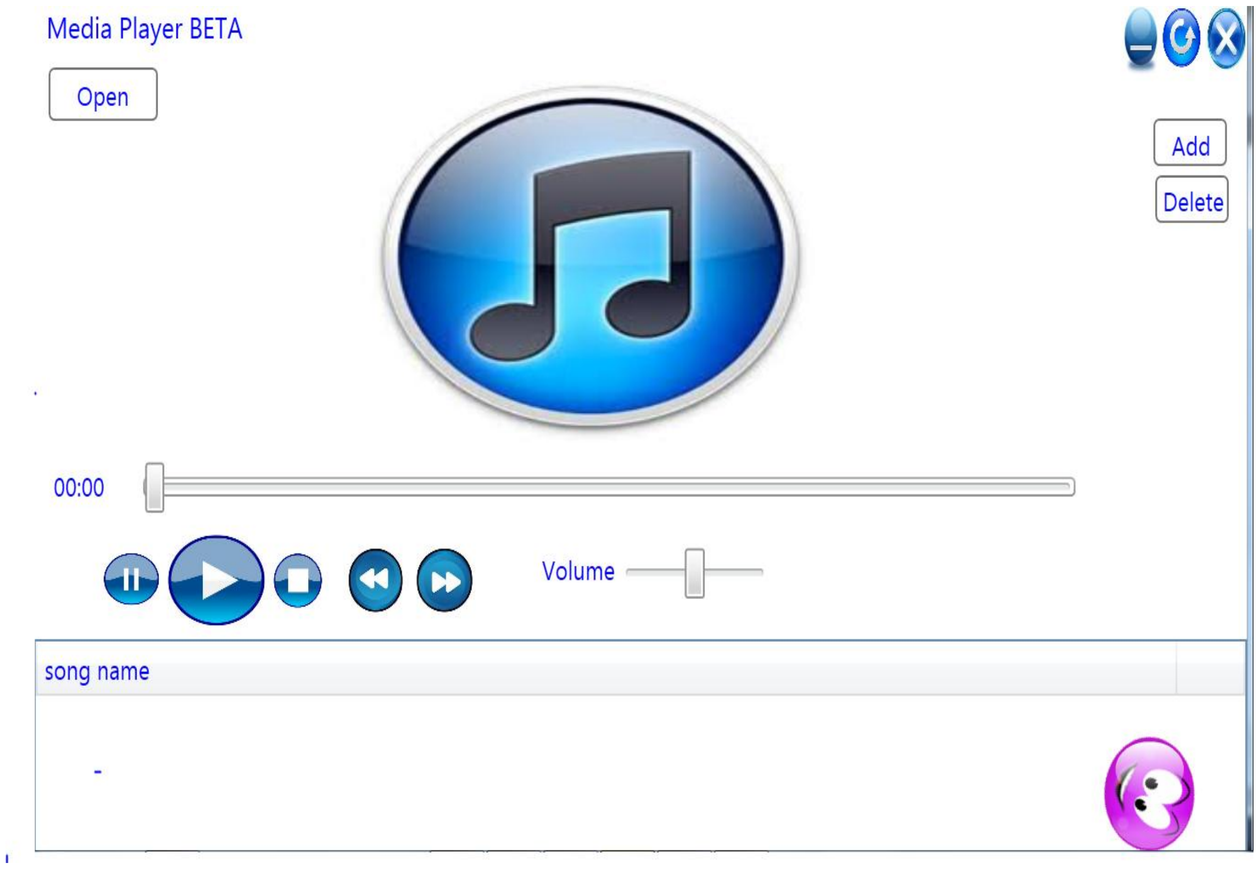


Fig.4.7 Output

Open button

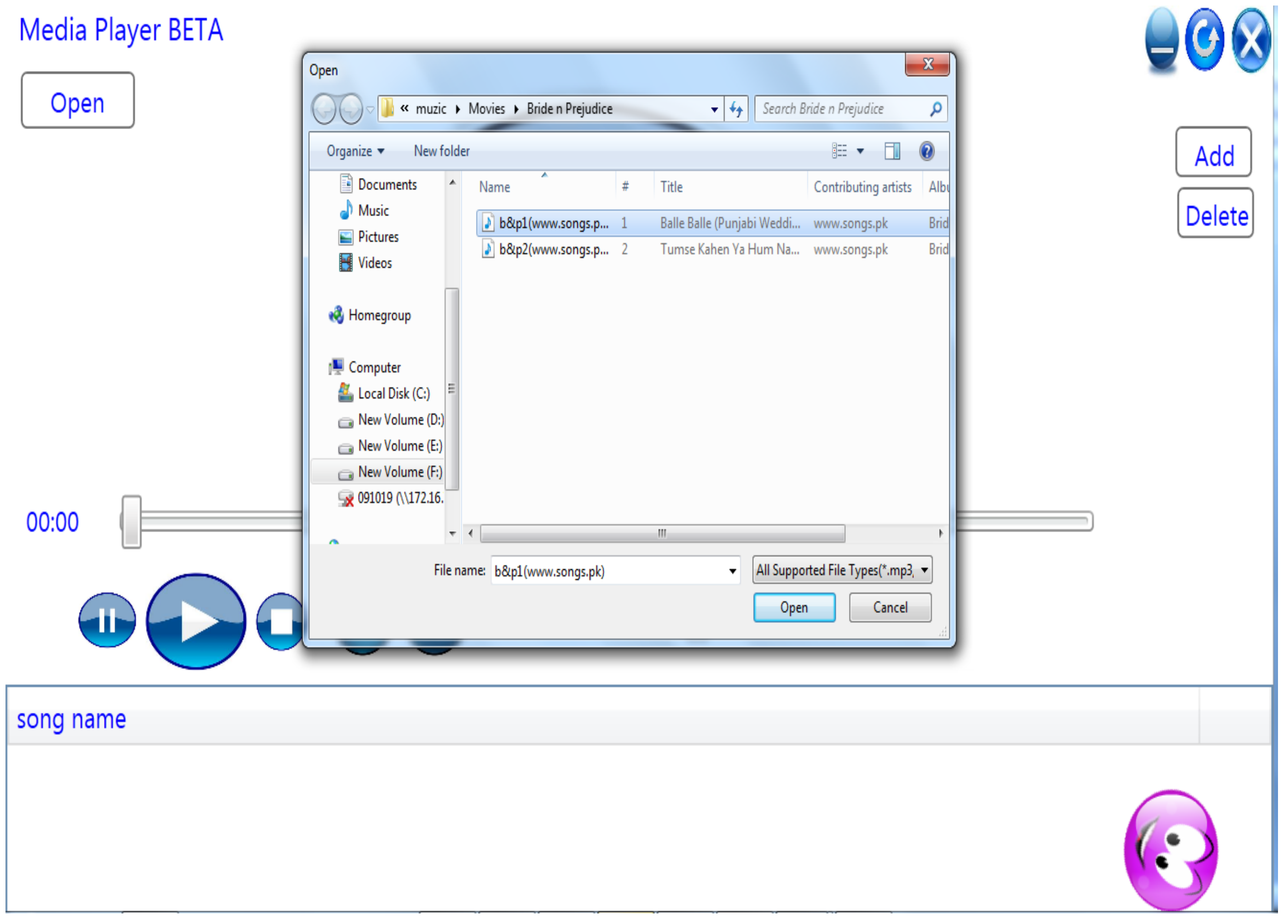


Fig.4.8 Open Button Functioning

Open button is used to select the song from computer and that song is put into the text box named the current song where the searching process start in database tables and similar types of song are recommended and if we select another song then the list of recommended songs is refreshed according to the new song selected with open button

Window can be resize to any size

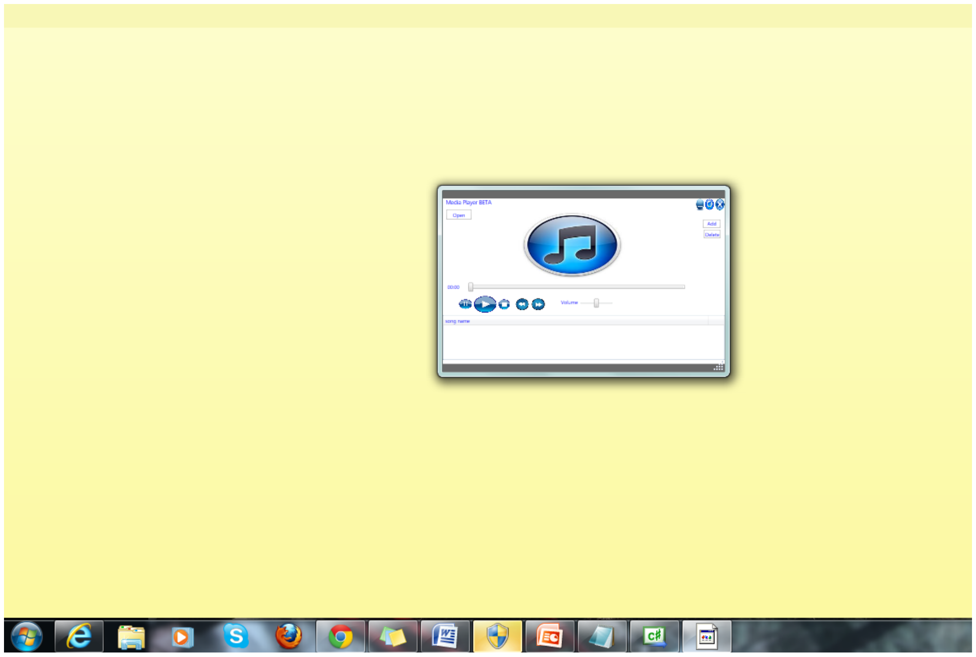


Fig 4.9.1 window in small size

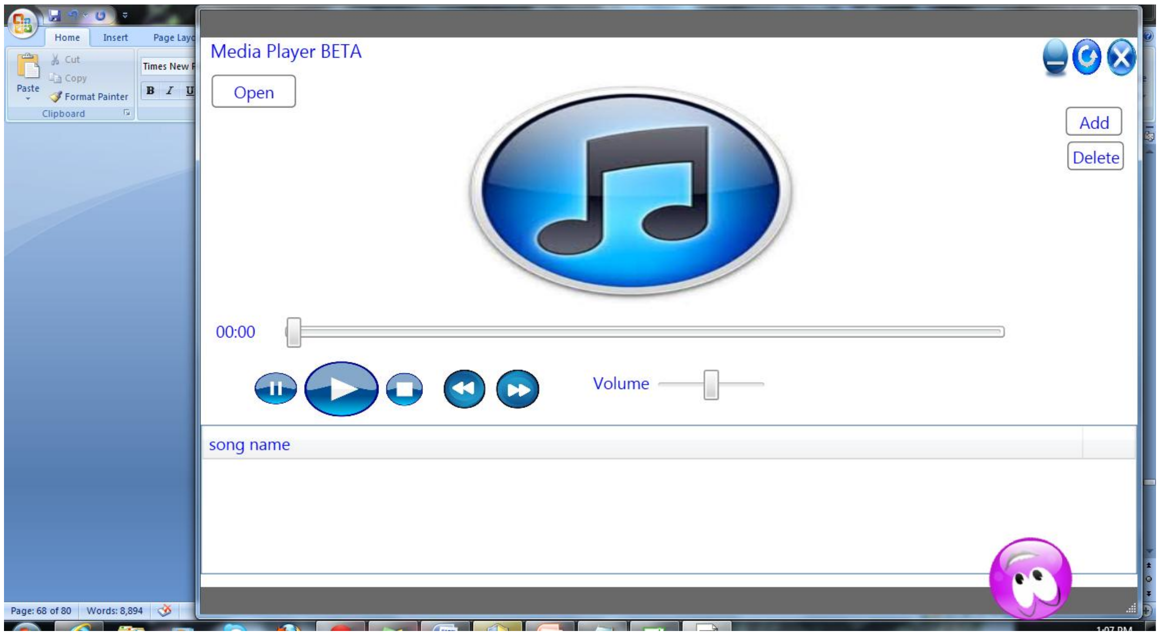


Fig4.9.2 window in adjustable size

To add a song two condition is checked firstly rating and second duplicacy and to extract the tag or know whether it has tag following condition is checked

```
        using (FileStream fs = File.OpenRead(ss))


          if (fs.Length >= 128)
          {
           ……
           ……
             string theTAGID = Encoding.Default.GetString(tag.TAGID);
             if (theTAGID.Equals("TAG"))
```

if first three byte are ‘TAG’ then they contain tags otherwise not and addition of song made according

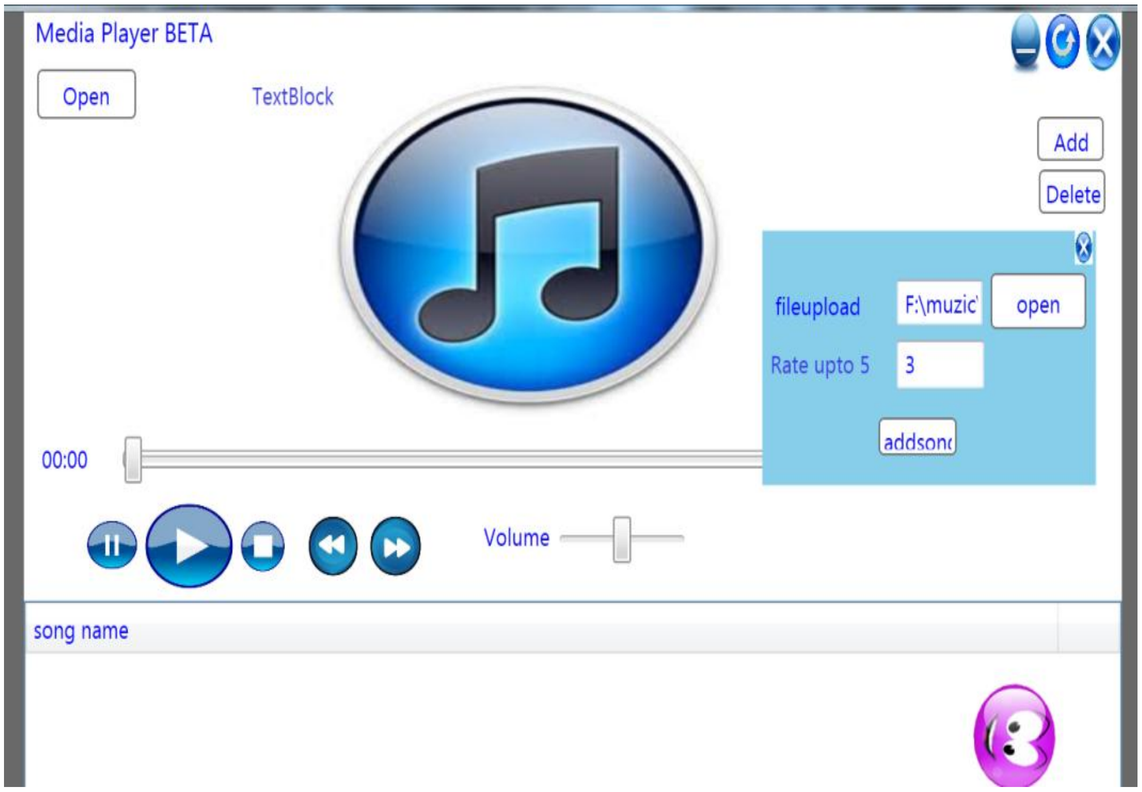
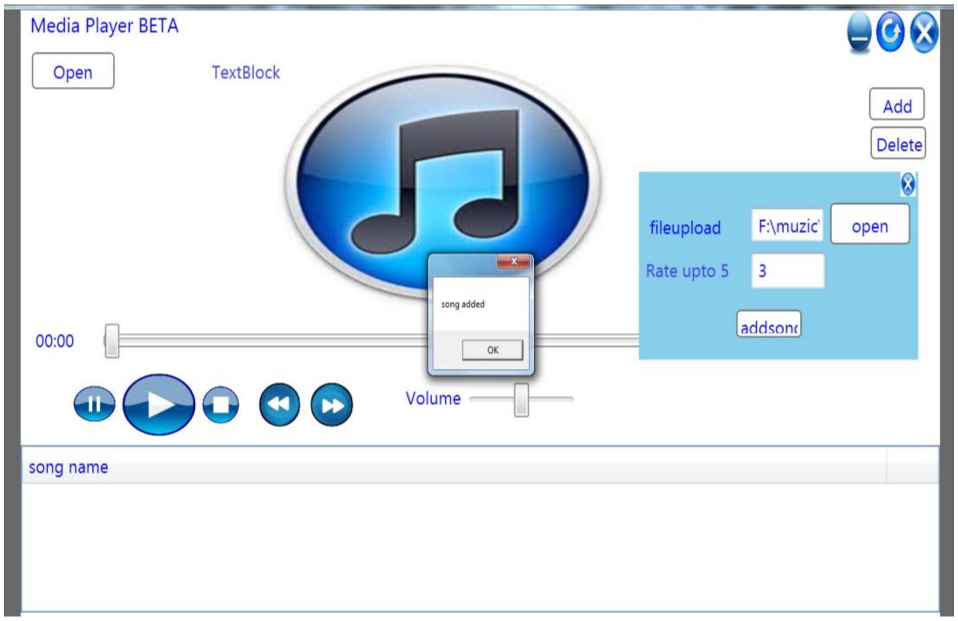


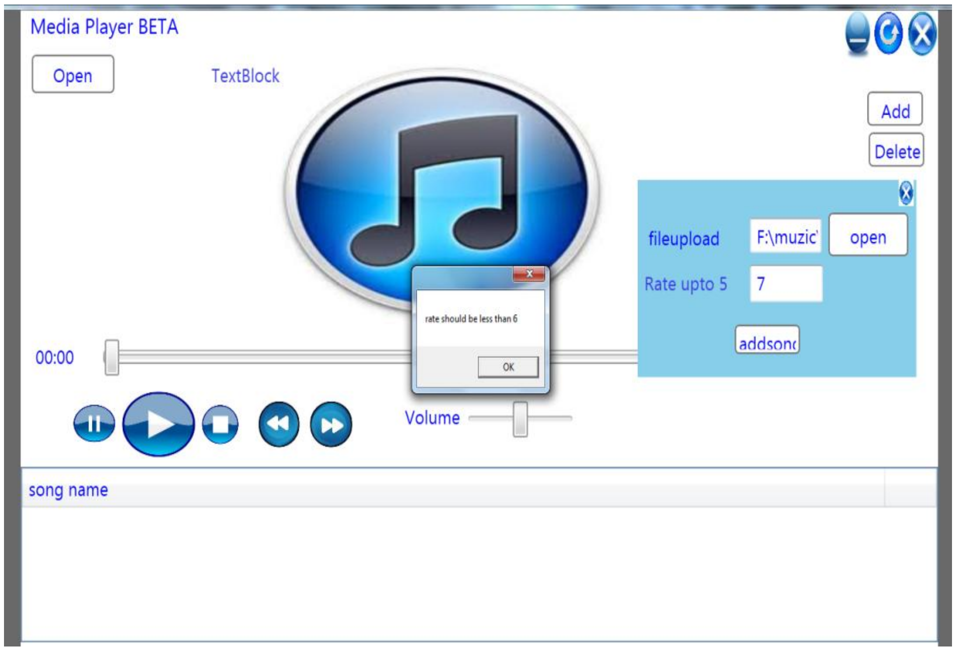
Fig.4.10 Add Button-1



song added

Fig.4.11. Add Button-2

Add button used to add new song which are not present in the database moreover other attributes in database like user demand at the entry is set to 0 and status also 0 and a message box pop up showing song added and if rating is greater than 5 again a message box pop up showing message than rate should less than 5 as in next figure we can see it.



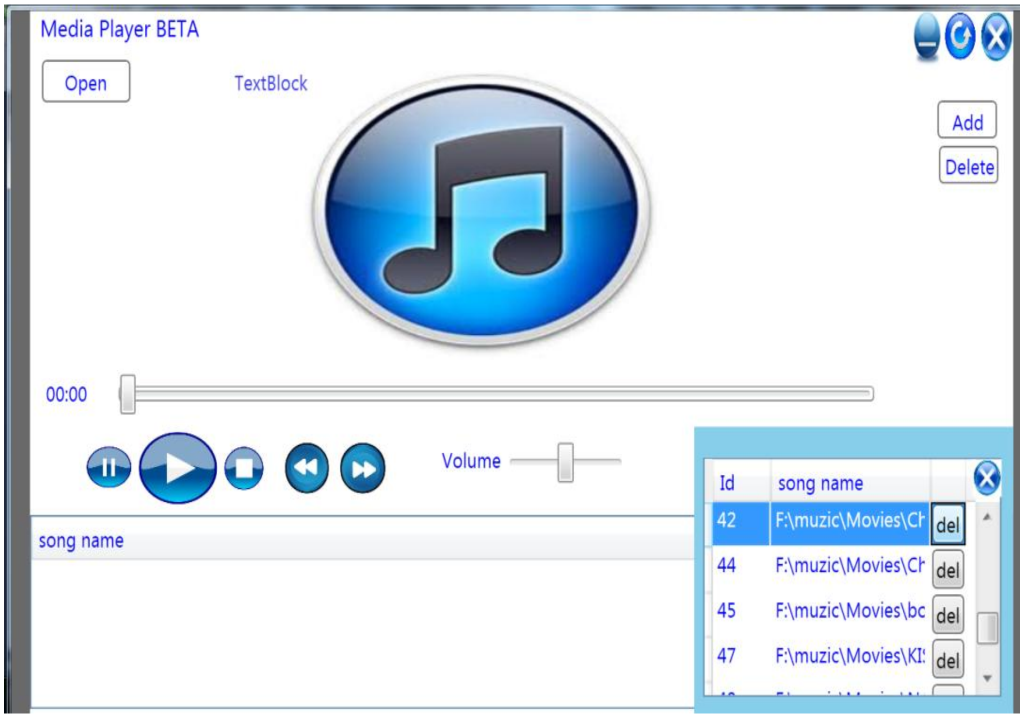
rate should be less than 6

Fig.4.12. Add Button-3

Similarly whern we to delete a particular song it id is extracted and stored and used to delete the song from both the databases

```
row = deletegrid1.SelectedIndex;
 TextBlock x = deletegrid1.Columns[0].GetCellContent(deletegrid1.Items[row]) as
TextBlock;
 id = Convert.ToInt32(x.Text);
 cmdstring1 = "delete from songentry where id=@id";
 cmdstring2 = "delete from songdetails where id=@id";
```

id of the selectd song is extracted as per line 1 and 2 and ten query is run for deletion



Id	song name	
42	F:\muzic\Movies\Ch	del
44	F:\muzic\Movies\Ch	del
45	F:\muzic\Movies\bc	del
47	F:\muzic\Movies\KIS	del

Fig4.13 Delete Button

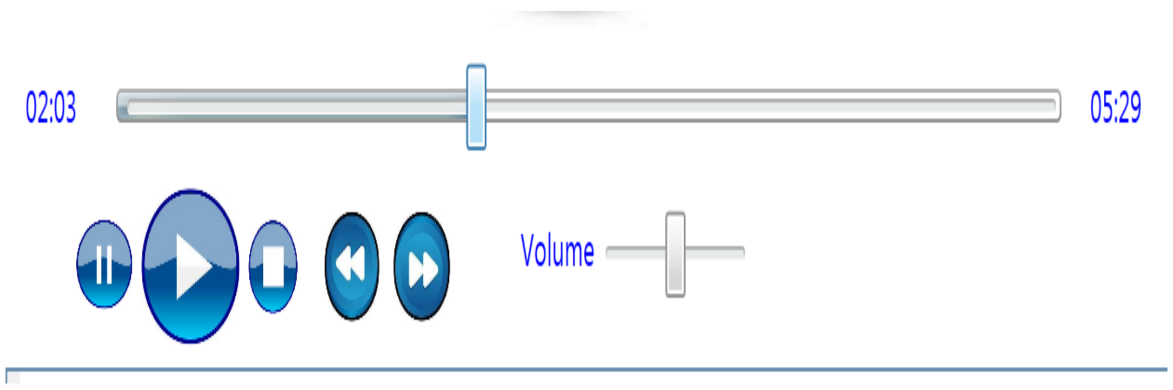


Fig.4.14 Button Functionality

Seek slider increments as the song is playing and maintain the consistency between the current time box and end timeS box

Progress bar gives the shading in corresponding to seek slider accordingly while song is playing

Stop button will make the same song to play from starting when play button is pressed

Play button to play the pause and stop song

Pause button to pause the reatian the time and status of the song

Volume button is to increment or decrement the volume of the song

Next and previous button to play next and previous song

Songname datagrid functions

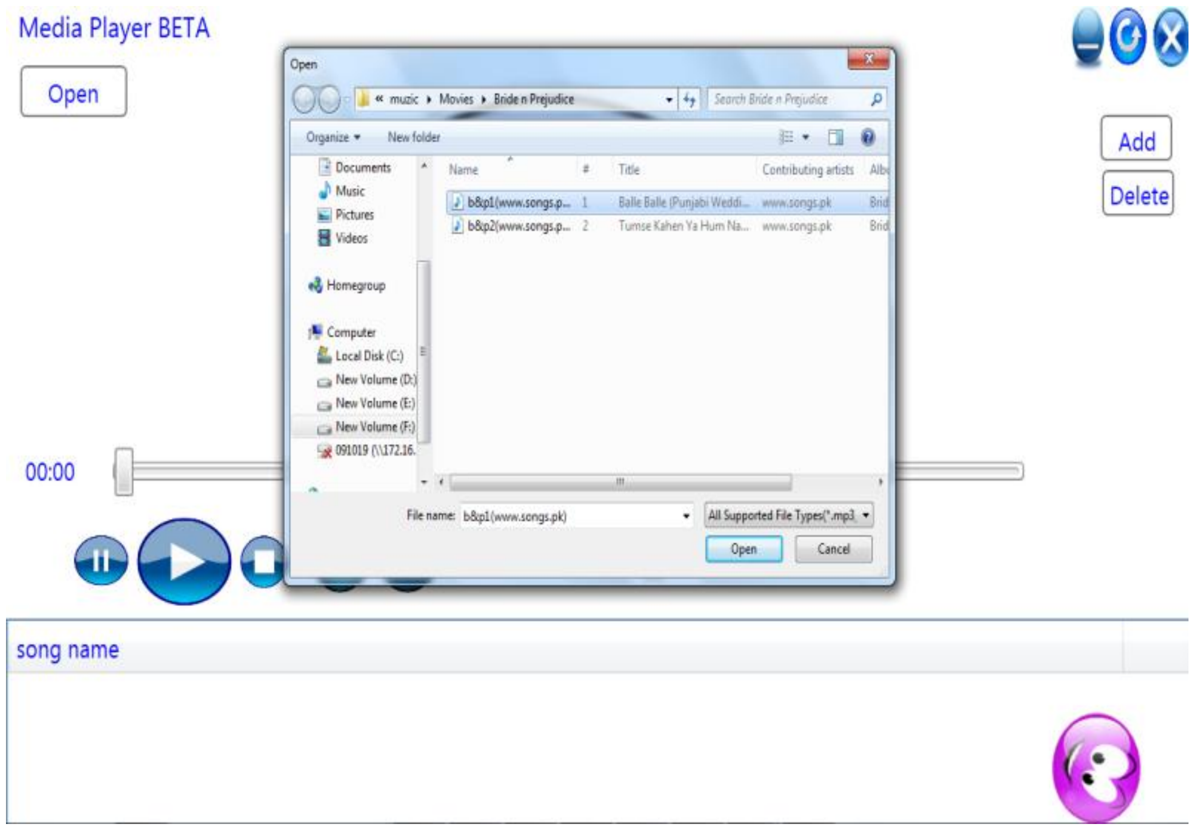


Fig.4.15 Data Grid working

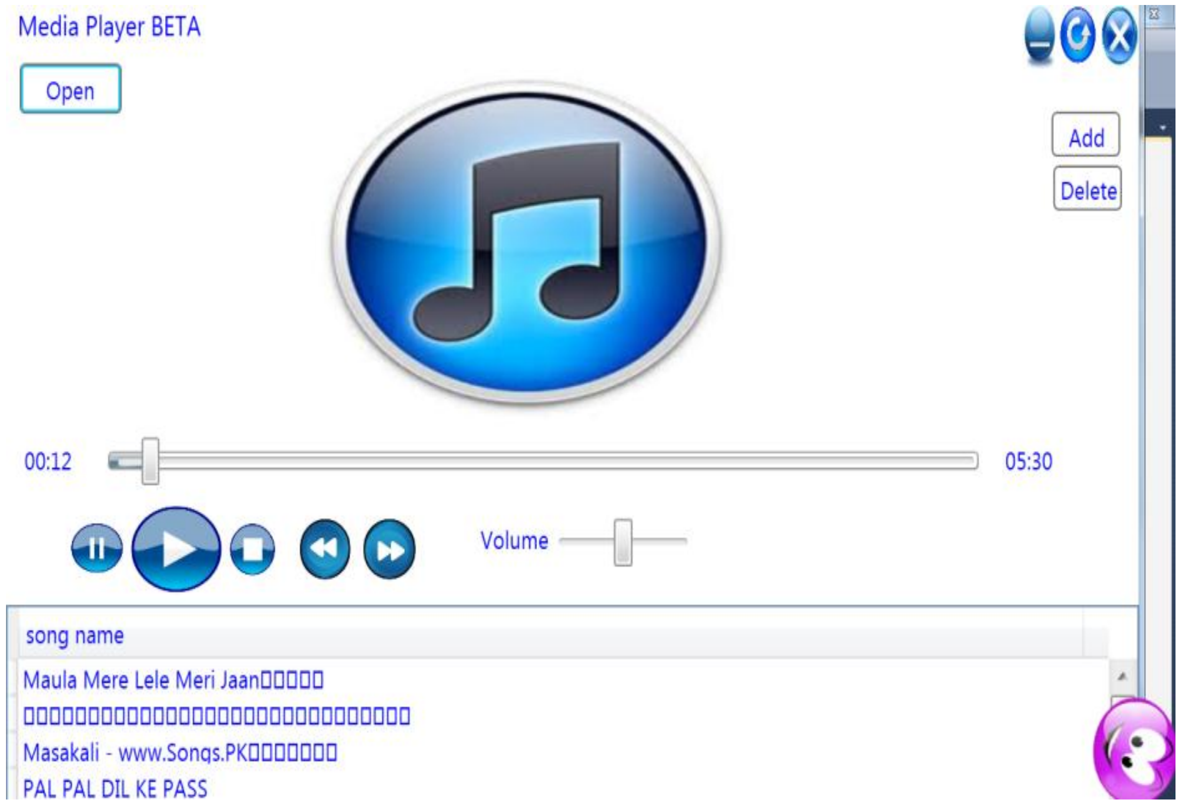
Song which is selected with tags and the query is run on that bases since we have two tables one has the mp3 file while the other has metadata so we linked both the tables in query .

cmdstring = "select * from songdetails where ((artist=@artist and album=@album and genre=@genre)or artist=@artist or rating='4' or rating='5'or id=(select id from songentry where userdemand>200)) and status=0 order by id ";

cmdstring1 = "select s.id,songupload from songdetails s,songentry where (s.id=songentry.id) and (((artist=@artist and album=@album and genre=@genre)or artist=@artist or rating='4' or rating='5'or songentry.userdemand>200))and status=0 order by id ";

first query is used to get the titles of the songs which are recommended since they are stored in a file.

Second query provide the source to media element of the application.

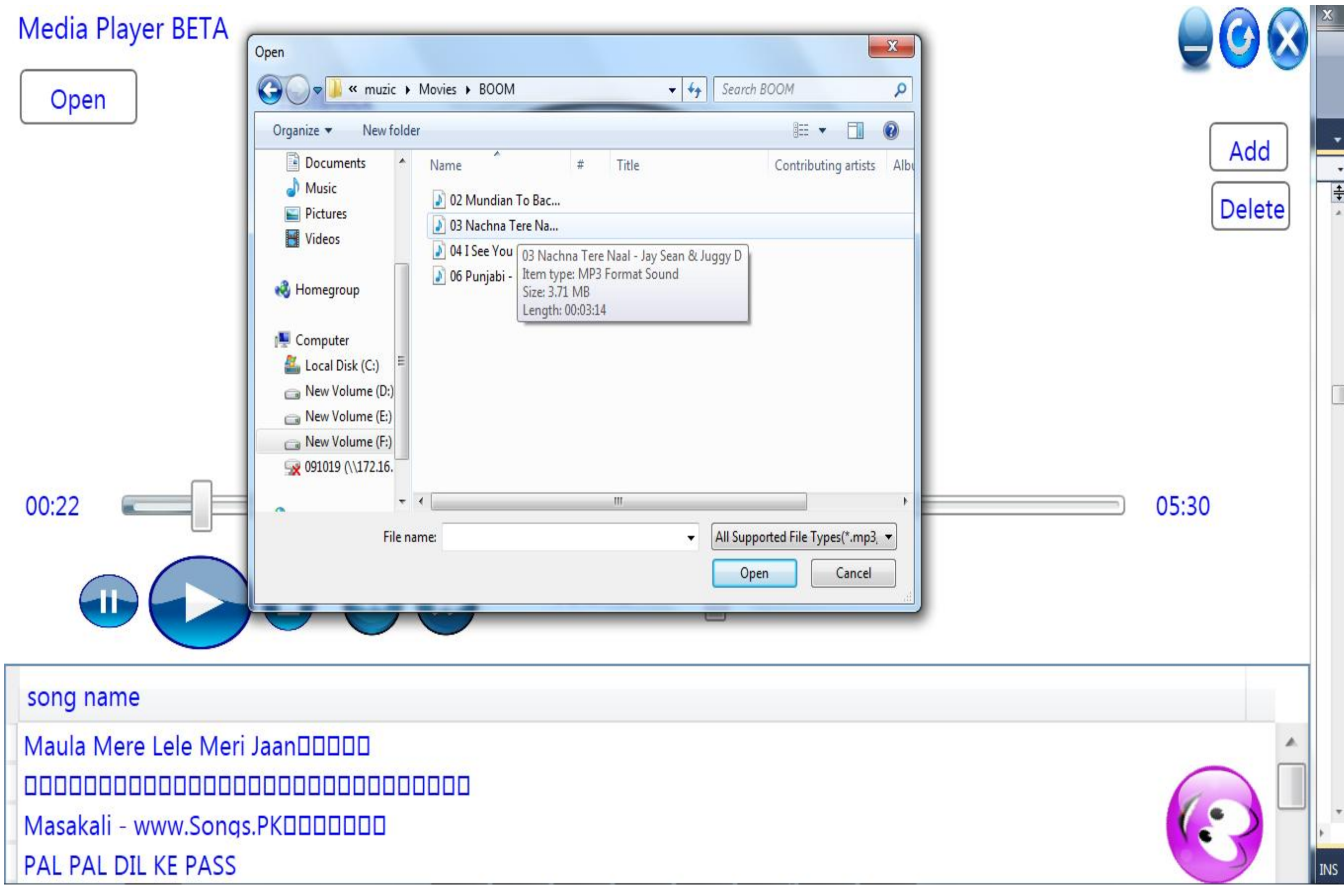


Maula Mere Lele Meri Jaan□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
Masakali - www.Songs.PK□□□□□□□
PAL PAL DIL KE PASS

Fig.4.16 Playing Recommended Playlist

Like if a song which we select from the randomly exist in database of the application then to check it status is used .if the song already exist then its status is set to 1 but as query we had used it gets the song which has status 0

Moreover if playlist according to one song is playing and if select the other than it refreshes the list and put new recommended song in the datagrid as shown in next figure where without tag song is selected while the songs are playing

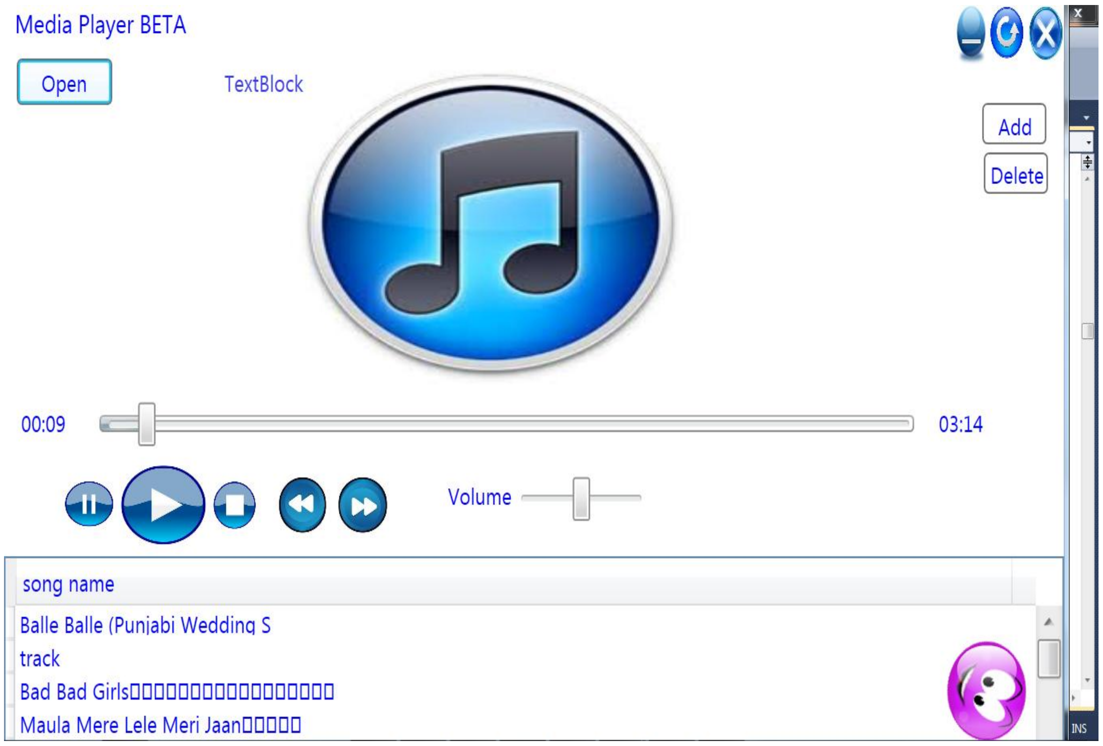


03 Nachna Tere Naal - Jay Sean & Juggy D
Item type: MP3 Format Sound
Size: 3.71 MB
Length: 00:03:14

Maula Mere Lele Meri Jaan□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
Masakali - www.Songs.PK□□□□□□□
PAL PAL DIL KE PASS

Fig.4.17. Selecting new songs without tags

Song without tag is selected while the other playlist is playing and new list is generated.



song name
Balle Balle (Punjabi Wedding S
track
Bad Bad Girls□□□□□□□□□□□□□□□□□□□
Maula Mere Lele Meri Jaan□□□□□

Fig.4.18 New playlist generated

New playlist with new recommended song is generated as in shown figure

4.6 Testing

For testing we have given our softwares to our friends and tooh their feed back to make it more efficient.Black Box testing is done on the project where we give an input and check the output whether it is upto mark or not and based on that some error like the song which user selects and present in database plays twice similarly the code of the code of the play grid is made more efficient with the help of this testing.

Review of our friends were satisfactory as the recommendation they get for music was good and they like it.

Chapter5 Conclusion

We have seen the various types of recommendation systems and their basic working. We have seen the one negative point in the market leader iTunes Genius, i.e. it can recommend some "*bad songs*" when it recommends songs just based on the artist name. Our work was focused more on the metadata based collaborative filtering on a big music database. It was observed that the development of a recommendation system with commercial features actually requires an extensive relational database to store the music previously cataloged. Creating a good relational database of music, making relations between artists, albums, musical genres and époques, could greatly expand the capabilities of a recommender algorithm will help users to discover new music. A working music player has been made that can play music files of Mp3 and we worked upon a method to extract the ID3 tags from Mp3 files. The system development has been completed successfully. Its usefulness for discovering new music has been tested meeting the goals stated at project's objectives.

References

1. http://id3.org/id3v2-00
2. Practical Common Lisp, Peter Seibil,2005.
3. ID3v2 Contributors
4. Nilsson, Martin. "ID3 Developer Information". *ID3.org.*. Retrieved 9 April 2011.
5. J.-J. Aucouturier and F. Pachet. Music similarity measures:What's the use? InISMIR, 2002.
6. A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. InISMIR, 2008
7. L. Xiao, L. Liu, F. Seide, and J. Zhou. Learning a music similarity measure on automatic annotations with application to playlist generation. In ICASSP, 2009.
8. Luke Barrington, Reid Oda, GertLanckriet, Smarter than Genius? Human Evaluation of Music Recommender Systems.
9. http://www.readwriteweb.com/archives/10_recommendation_engines.php
10. http://www.maniactools.com/soft/mp3tag/id3-tags.shtml
11. http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
12. http://publib.boulder.ibm.com/infocenter/spssstat/v20r0m0/index.jsp?topic=%2Fcom.ibm.spss.statistics.help%2Falg_tree-cart.htm

13 p://www.nerdparadise.com/tech/csharp/mediaplayerreplace/1/

14 http://stackoverflow.com/questions/6255084/fill-datagrid-from-mysql-database-in-c-sharp-wpf

15 http://www.codeproject.com/Questions/51918/wpf-datagrid-connecting-with database

16 http://www.codeproject.com/Tips/362436/Data-binding-in-WPF-DataGrid-control

17 http://www.youtube.com/watch?v=ejZSWpl3omI
18 http://www.codeproject.com/Articles/235083/WPF-Audio-Player
19 http://www.codeproject.com/Articles/20478/A-simple-WPF-media-player-withmedia-item-list
20 http://www.video-tutorial.org/4836/wpf-tutorial-1-how-to-make-a-basic-videomusic-player/
21 http://www.id3tageditor.com/
22 http://www.codeproject.com/Articles/4235/MP3FileInfo-Extract-Header-and-ID3-Tags-of-an-MP3
23 http://stackoverflow.com/questions/5028894/how-do-i-make-xaml-datagrid-columns-fill-the-entire-datagrid