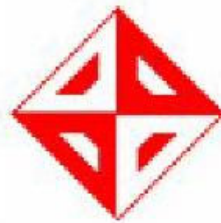


MIDDLE EAST TECHNICAL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING



SE 560
DESIGN REPORT

WEB-VR

PREPARED BY:

EMRE AKYILMAZ - 1599703

ATILLA ACAR - 1202647

TABLE OF CONTENTS

1	INTRODUCTION.....	4
1.1	Project Title	4
1.2	Project Definition	4
1.3	Project Goals and Scope.....	5
1.4	Design Constraints	6
1.4.3	Timing Constraints	6
1.4.1	Software Constraints	6
1.4.2	Hardware Constraints	6
1.5	Design Objectives	6
1.5.1	Maintenance	6
1.5.2	Reusability.....	7
1.5.3	Consistent and Fault Tolerance	7
1.6	Future Works.....	7
2	REQUIREMENTS	7
3.1	Functional Requirements.....	7
3.2	Nonfunctional Requirements.....	10
3.2.1	Efficiency:	10
3.2.2	Correctness:.....	10
3.2.3	Reliability:.....	11
3.2.4	Usability:	11
3.2.5	Understandability:	11
3.2.6	Functionality:	11
3.2.7	Performance:	11
3.2.8	Supportability:.....	12
3.2.9	Fault Tolerance.....	12
3.3	Hardware Requirements	12
3.4	Software Requirements	12
3	OVERALL ARCHITECTURE.....	13
4	DATA DESIGN	14
4.1	ER Diagrams	14
4.2	Data Dictionary & Diagram	17
4.2.1	Web VR Schema	18
4.2.2	ALBUMS	20
4.2.3	COMMENTS	20
4.2.4	GENRES	22
4.2.5	GENRE_RELATIONS	22
4.2.6	PLAYLISTS.....	24
4.2.7	RATINGS.....	24
4.2.8	SINGERS	26
4.2.9	SONGS.....	27
4.2.10	TAGS	28

4.2.11 TAG_RELATIONS	28
4.2.12 USERS	30
5 DESIGN DIAGRAMS	31
5.1 Class Diagram	31
5.2 Sequence.....	33
5.2.1 Listen Songs	33
5.2.2 Add Song to Playlist.....	34
5.2.3 Upload Song to Database	35
5.2.4 Create Playlist	36
5.3 Data Flow Diagrams.....	37
5.4 Use Case.....	41
6 USER INTERFACE DESIGN	42
7 UPDATED PROJECT PLAN	44
8 CONCLUSION	46

1 INTRODUCTION

This report includes the design descriptions of Web Based application WEB-VR. After the requirement analysis, we made some preparations for the project design and implementation. Our preparations includes overall functional and non functional search on existing products, more deeper market search, deeper IDE and programming tools search and common web based application functionality search. There are two important benefits that we gain from our preparations. First, this preparation provides us to see the lack of functionalities in our requirements. Although we did not change our requirements, we expanded the some of them. We revised and updated our ER diagrams. Second, we examine the current programming languages and development tools, which shows us to advantages and disadvantages of the integrated development environments while considering our functional and non-functional requirements. We made a proper decision to choose one of the languages and developments tools.

1.1 Project Title

Our project title is WEB-VR since, our project is a web-based application, which provides Video and Radio/Audio functionalities, and we decided to mention these functionalities in our project title. We used the shorthand version of WEB based – Video/Radio/Audio entertainment application which is WEB-VR.

1.2 Project Definition

WEB-VR is a web-based application, which combines the technology and the real life entertainment in efficient software.

This application provides opportunity to users to listen different kinds of songs and watch videos through the their web browser while sharing these media files with each other.

The WEB-VR can be summarized as a web application that provides powerful database, which users can;

- Listen, share and download songs and videos.
- Create and store playlist using repository which can be performed in one of two ways: uploading a preferred songs and videos or using songs and videos from different shared accounts;
- Search for songs and videos according to user defined criteria such as genre, name and playlist.

- Rate their favorite songs and videos
- Give comment on songs and videos.

1.3 Project Goals and Scope

As we mentioned before, since WEB-VR is an web based application, our ultimate goal is to implement simple and effective web based application. WEB-VR can be used by any user, so our application will be implemented based on an average user. An average user must know how to use a web browser. No other computer skill is necessary.

Our other goal is to design and implement an efficient database system without including too much complexity. We tried to keep our database system simple and efficient. In order to do that, we did not include unnecessary information to our system.

Each user will listen, upload and download songs easily on our application and on the other hand they will manage their account and their profile.

We can summarize our other goals as listed below:

- Ease of use: Any user can use WEB-VR without having difficulties.
- Robust and reliable: Our application always shows robust and reliable performance.
- Platform Independent: Since WEB-VR is a web-based application it must be platform independent.
- Secure: In our application user account and database system must be secure.
- Friendly user interfaces: Users will access WEB-VR through friendly graphical and simple user interfaces.
- No computer skills needed: No prior computer knowledge is necessary to use our application.
- Error Handling: All the possible error conditions will be minimized.
- Consistent: All the search, load, log in and other database operations provides persistent and stable results.
- Help Menu: All the necessary information provided to the users through the help menus. WEB-VR users can access help menus whenever they needed.
- Customizable user interface: User interfaces will be customizable as possible without dealing with too much complexity.
- Consistent database connection: Our database connection will be secure, stable and provides consistent results.

1.4 Design Constraints

1.4.3 Timing Constraints

The timing constraint is the biggest and the most important constraint from the design constraints. As all the programmers and software engineers, we have too do much work in a short period of time. Since our project group is consist of two people, and each of us working in a full time job, we have heavy work load. In order to complete our project we made calendar, which includes strict deadlines.

1.4.1 Software Constraints

The second biggest problem is the choice of software tools that can both provide the functionalities that we seek and doing this without considering the development platform. In order to overcome this issue we explore current web browser's properties, advantages and disadvantages. By doing so we made a proper decision about development platform where we can produce platform independent program

1.4.2 Hardware Constraints

WEB-VR is a web-based application, so it must be efficient, platform independent applications. One of the biggest problems while implementing web-based application is to make that application platform independent and make it efficient on every computer. So we tried to make our program to run on every platform which is the beyond our capabilities since we don't have that wide of computer options. So we tried to make it platform independent as much as possible.

1.5 Design Objectives

1.5.1 Maintenance

Our program must be providing maintenance since it is not fully implemented yet. While finalizing implementation we can come up with some changes in design in order to eliminate possible error conditions. Also some updates may be necessary in the future. In order to these our program must provide maintainability.

1.5.2 Reusability

In the design phase we also spend too much effort on making reusable classes, tables and database structures. Since we have time constraints, we decided that the best way to overcome this constraint is to make program elements and structures reusable as possible.

1.5.3 Consistent and Fault Tolerance

WEB-VR is a web based application. It can be used by many users at the same time. They can listen songs, watch videos while sharing these media objects with each other. In order provide accurate and efficient web based application, WEB-VR must be fault tolerance. It must be continue operating properly in the event of the failure of some of its components. Also its database operations must be consistent. In the design phase we spend effort to make our system fault tolerance as possible.

1.6 Future Works

The submission of design reports will finalize the project design and requirement phase. After that we will implement fully functional prototype of WEB-VR before the project presentations and demo. We will briefly examine the advantages, disadvantages and the conditions that may cause some errors. If we like our application than we will finish it with adding the possible GUI extensions such as pictures, images etc. If we found an error than we will further try to determine and eliminate the possible error condition as possible before the project presentations and demo.

2 REQUIREMENTS

3.1 Functional Requirements

SRS_1: WEB-VR provides a web service where users can directly listen songs and watch videos.

It is our ultimate goal to make accurate web based application where users can easily listen desired songs or watch the videos. In order provide effective application, the data base system of the application must be accurate and very well implemented.

SRS_2: WEB-VR provides unique accessible user accounts for users where user profiles and interface options are kept.

Users will access our application through the internet. They create their playlist which contains their favorite songs, they can categorize songs, upload and download videos and song from WEB-VR. In order to satisfy our user need, we decided to implement user accounts where users can access their accounts in order to take benefit from WEB-VR features.

Also this feature is necessary for customizable user interfaces. The users can customize user interfaces according to their needs and this information kept in their accounts.

SRS_3: Users can manage their accounts and their profiles.

Users can log in their profiles and they can change their profile information such as name, age, sex etc. Also they can manage user interface of the WEB-VR by selecting hide or unhide options for the desired tabs such as: online peoples, other peoples who are currently listening the same music etc.

With the help of hide or unhide options, this requirement provides the customizable user interfaces.

SRS_4: WEB-VR has ability that user can upload their songs database.

WEB-VR's main goal is to provide effective web based application where users listen song through the internet. User can share their favorite songs with each other with the detailed search system. One feature of WEB-VR is the databases are created dynamically. That means WEB-VR does not same number of songs. New songs can be added by users to the system by uploading the desired songs.

SRS_5: WEB-VR has ability that users can upload their videos to database.

WEB-VR also provide effective solution the users where they can watch videos through the internet. Like songs, videos are not limited, their database is updated dynamically. That means WEB-VR does not same number of videos. New videos can be added by users to the system by uploading the desired videos.

SRS_6: User will upload desired songs to their precreated playlist.

WEB-VR provides create playlist option to the users where each user can categorize their favorite songs. Each user can upload desired songs to the playlist by giving playlist name to the songs.

SRS_7: Users will search other user's songs and videos from database.

“Listen, watch and share” is the main paradigm of WEB-VR. In order to provide video and song sharing between users, WEB-VR provides detailed search system where users can search through media files by their types, names, genre, albums, rating etc..

SRS_8: Users will create new playlist.

Each users can create their playlist in order to categorize desired songs under the same topic. WEB-VR also provides detailed search through playlist such as playlist name, rating etc..

Also playlists can be categorized through their genres which provides accurate and more efficient search and store options.

SRS_9: Users will define the information of songs.

WEB-VR provides flexibility which user can define and change song information such as changing genre, changing ratings, changing types etc.. (This options provides user based genres, types etc. For example, changing genre of the song from “slow” to “depression” provides new genres.)

SRS_10: The song information describes the songs.

Song information's contains:

- Name of the song
- Format of the song
- Recording Date of the Song
- Recording User of the Song
- Album
- Singer
- Genre

SRS_11: Users will give permission to their songs so that others users can listen their songs.

Sharing song is based on each user will. User will share their song with other users by giving public permission on songs or they will keep their songs private by setting private permission on songs. This permission process also applies on playlist.

SRS_12: Users will give permission to their videos so that others users can watch their videos.

Sharing song is based on each user will. User will share their videos with other users by giving public permission on videos or they will not keep their videos private by setting private permission on songs.

SRS_13: Users will rate their songs.

Each user rates their songs so that other users can share their songs through their rate number. (Users can search songs according to their rates.)

SRS_14: Users will rate their videos.

Each user rates their video so that other users can share their video through their rate number. (Users can search video according to their rates.)

SRS_15: Users will write comment on their videos.

Users can write comment on their videos so they can remark their videos where they can add a piece of information related to that video.

SRS_16: Users will write comment on their songs.

Users can write comment on their songs so they can remark their songs where they can add a piece of information related to that song.

3.2 Nonfunctional Requirements

3.2.1 Efficiency:

The system's usage is based on datas. Therefore, it should be efficient and fast enough while working on datas. In thousands of songs and video information, one of the most important points is getting true data in a short time.

3.2.2 Correctness:

In daily life, the application area of this system is very huge. Thousands of users will be use the system through their web browsers. In order to work correctly, this web based application must be implemented carefully that should include the whole correct information, data and their relationships in database.

3.2.3 Reliability:

The thousands of users will use the results of data searches. They will listen, watch and entertain on these data's. Therefore there will be no chance for making mistakes. The system will be designed carefully and each detail will be thought. We will try to make system without any failure or need of recovery.

3.2.4 Usability:

In most menu item, there will be several help documentation for users. Therefore users can use this system efficiently and easily. Whenever the user has trouble with the system, he can reach detailed information about his trouble and the usage of each menu item. We take novice users into consideration as well. Documentations will be easy to manage and detailed enough.

3.2.5 Understandability:

Especially for novice users, the system must be used easily. The menu items will include simple states for reaching the desired data quickly. The graphics will be user friendly as well. For example is web layout is must be simple. The suitable and predictable items will be put under a menu. WEB-VR designed for everybody who has an internet connection. Therefore we will be very careful in design and implementation stages of database.

3.2.6 Functionality:

The algorithm of this system can be applied to various other entertainment systems that will use the information about certain song or video. The greatest challenge of this project is to implement an proper database that includes accurate tables and exact relationships. So the mainly functionality will be correct if the database and its relationships are accurate.

3.2.7 Performance:

For efficient performance, the programming languages capabilities will be investigated seriously. We will select the most efficient ways to implement our program. Therefore we aim at maximal throughput and minimal cost requirements. The program will be run fast, the results will be returned correctly and the user should be satisfied when he wants to entertain on our application.

3.2.8 Supportability:

Data tables and their diagrams will be well organized so in the future it will become easier to update, remove or add new data. For not being late, before the implementation, we consider about maintainability and well- organized configuration. Therefore the mistakes, incomplete parts will be corrected in design states. After the implementation, technical supporting and development can be formed as well.

3.2.9 Fault Tolerance

There will be minimum number of unpredictable application results. If an error conditions occurs, the system will continue operating properly in the event of the failure of some of its components.

3.3 Hardware Requirements

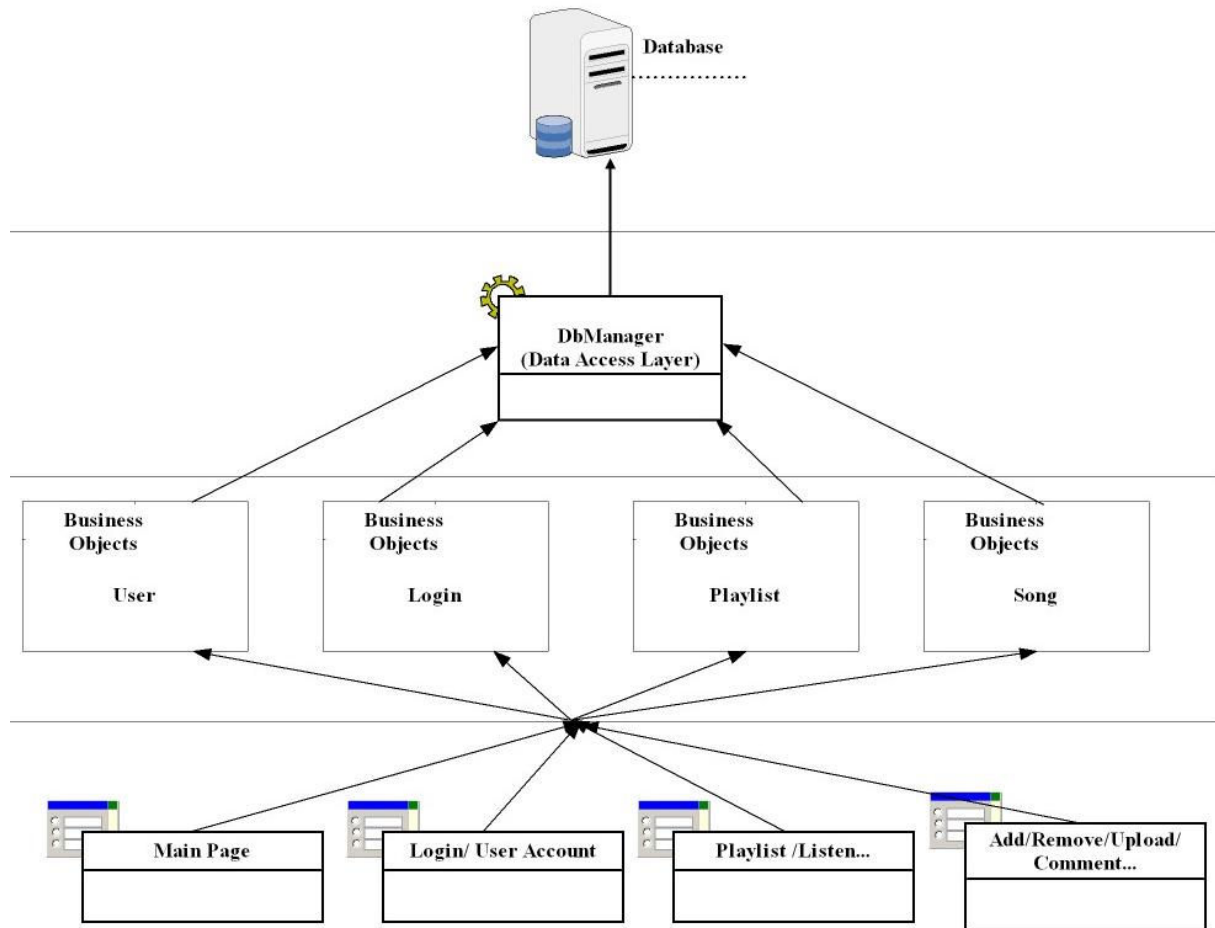
- At least Pentium processor or better
- Good internet connection probably better than 56 k modem connections.

3.4 Software Requirements

- Windows XP, Linux, Mac, Solaris
- Mozilla Firefox, Internet Explorer, Opera, Netscape, Chrome

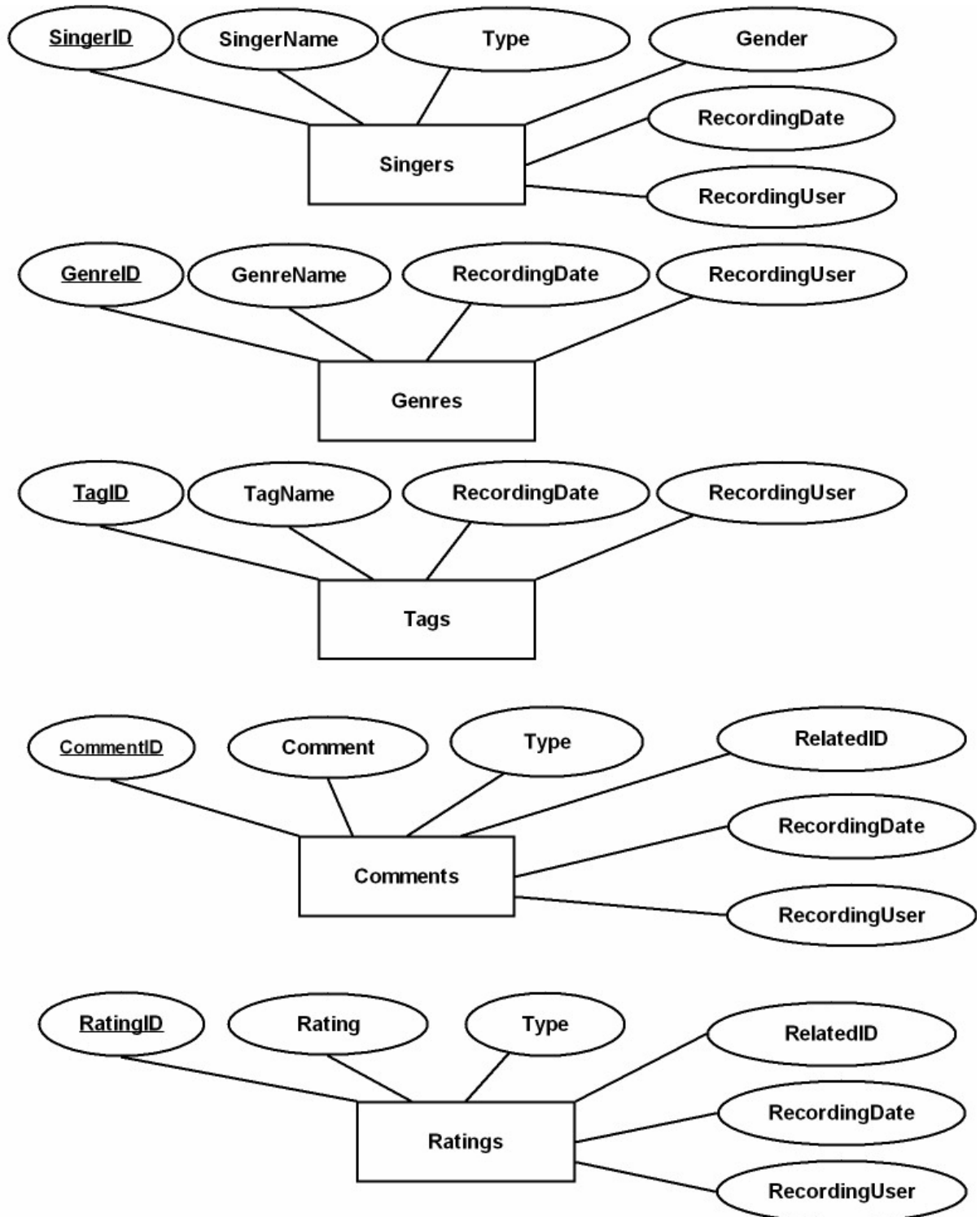
3 OVERALL ARCHITECTURE

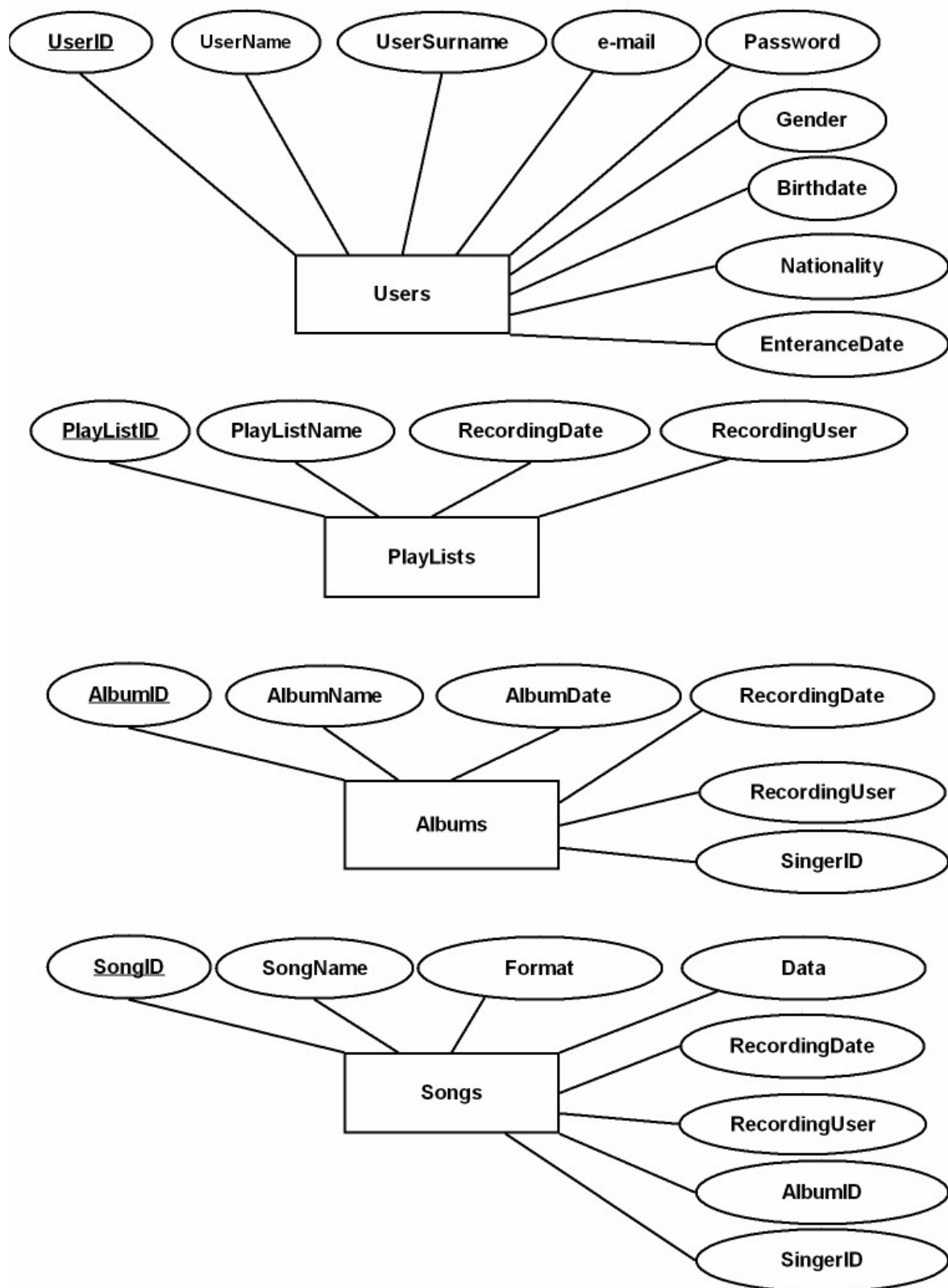
In the following the diagram, the overall architecture of the WEB-VR shown:

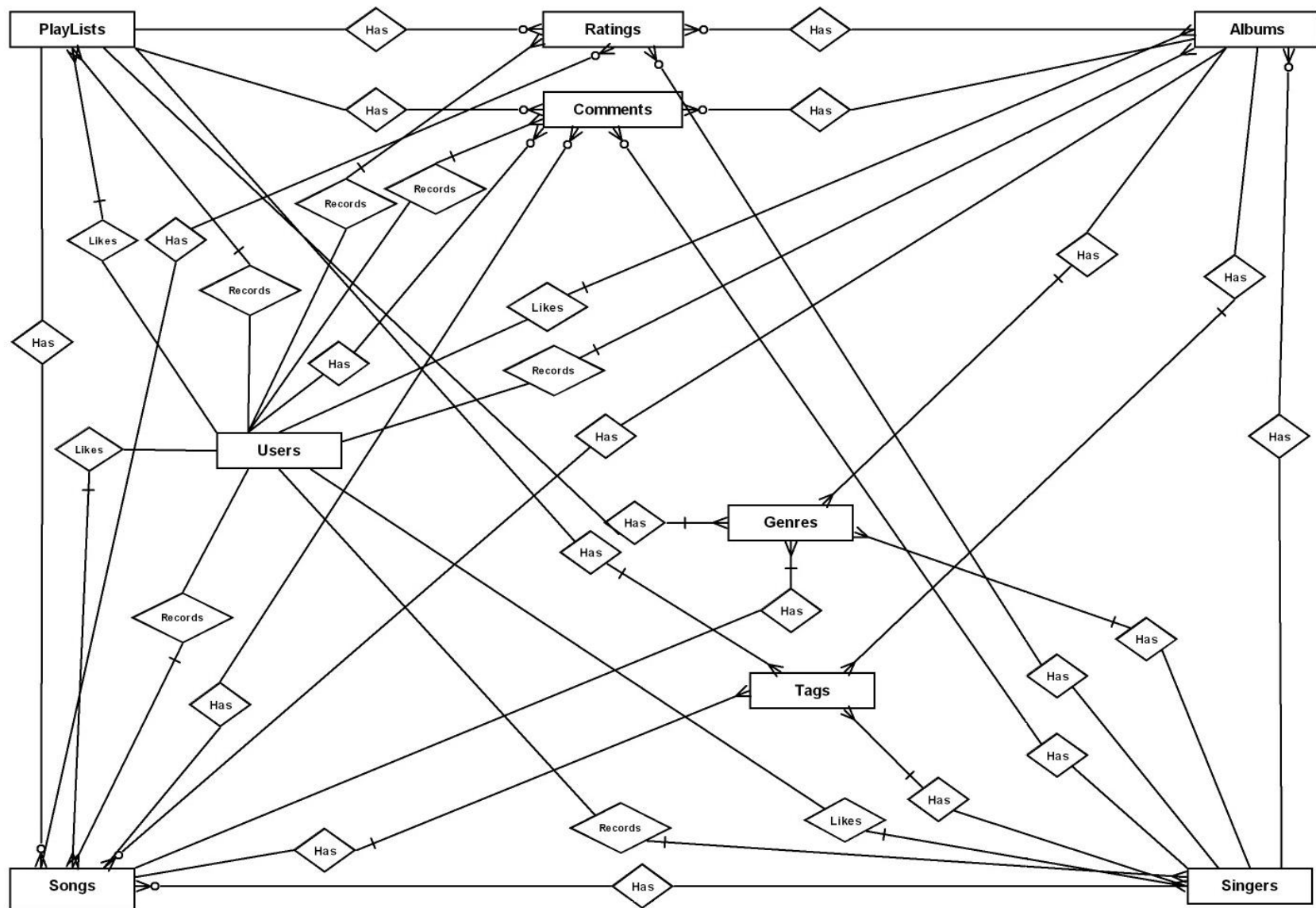


4 DATA DESIGN

4.1 ER Diagrams

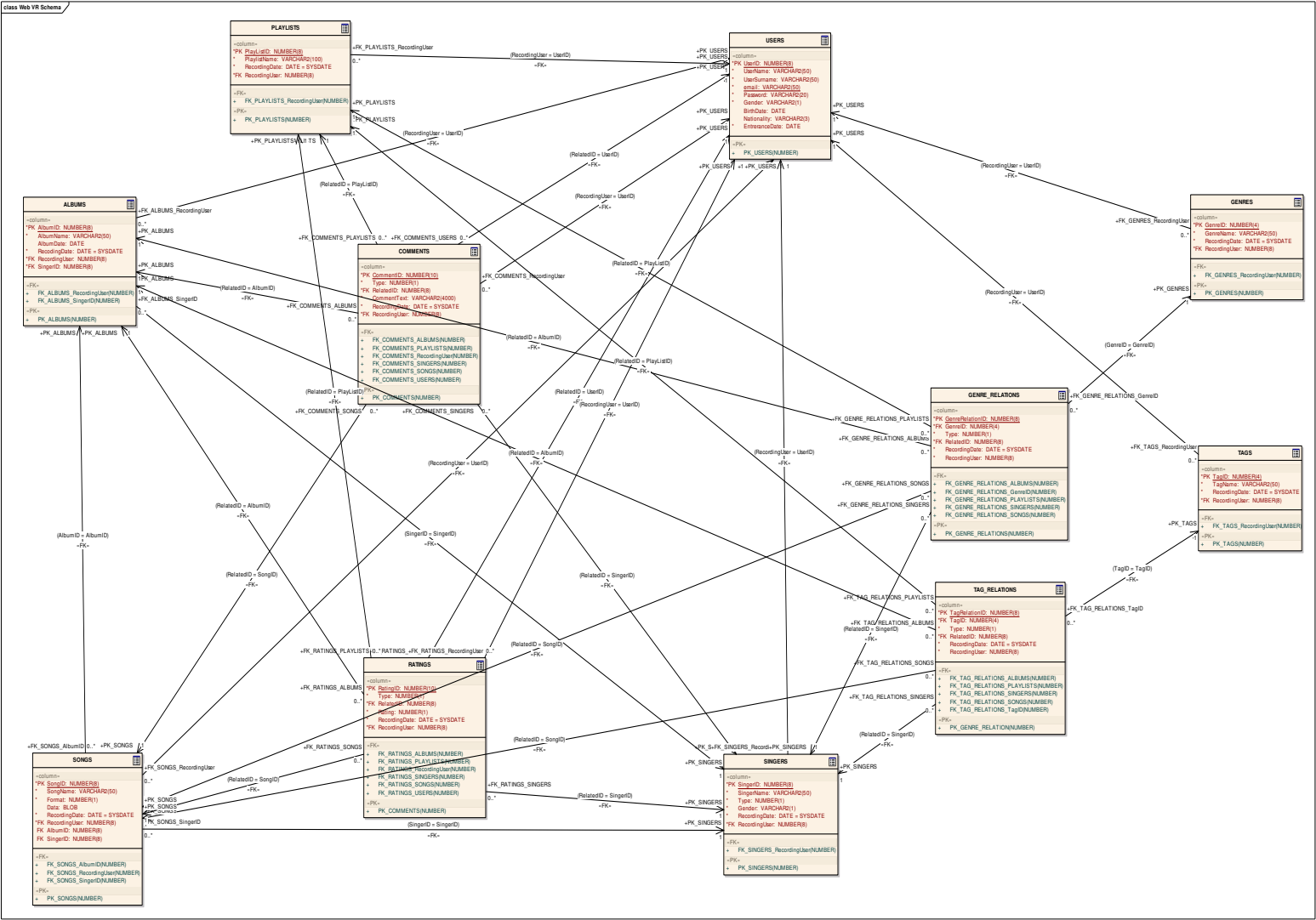






4.2 Data Dictionary & Diagram

4.2.1 Web VR Schema



4.2.2 ALBUMS

Notes: Which contains albums recorded by the users

Columns

Name	Type	Not Null	Unique	Len	Init	Description
AlbumID	NUMBER	True	True	8		Surrogate key of the table
AlbumName	VARCHAR2	True	False	50		Name of the album
AlbumDate	DATE	False	False			Release date of the album
RecordingDate	DATE	True	False	8	SYSDATE	Recording system date of the album
RecordingUser	NUMBER	True	False	8		Recording user of the album
SingerID	NUMBER	True	False	8		Singer (Person or Group) of the album.

Constraints

Name	Type	Columns
FK_ALBUMS_RecordingUser	«FK»	RecordingUser
FK_ALBUMS_SingerID	«FK»	SingerID
PK_ALBUMS	«PK»	AlbumID

Relationships

Columns	Association
(SingerID = SingerID)	0..* ALBUMS.FK_ALBUMS_SingerID 1 SINGERS.PK_SINGERS
(RelatedID = AlbumID)	0..* COMMENTS.FK_COMMENTS_ALBUMS 1 ALBUMS.PK_ALBUMS
(AlbumID = AlbumID)	0..* SONGS.FK_SONGS_AlbumID 1 ALBUMS.PK_ALBUMS
(RecordingUser = UserID)	0..* ALBUMS.FK_ALBUMS_RecordingUser 1 USERS.PK_USERS
(RelatedID = AlbumID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_ALBUMS 1 ALBUMS.PK_ALBUMS
(RelatedID = AlbumID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_ALBUMS 1 ALBUMS.PK_ALBUMS
(RelatedID = AlbumID)	0..* RATINGS.FK_RATINGS_ALBUMS 1 ALBUMS.PK_ALBUMS

4.2.3 COMMENTS

Notes: Which contains comments for users, playlists, songs, singers etc... Type attribute determines type of the comment (user comment, song comment...)

Columns

Name	Type	Not Null	Unique	Len	Init	Description
CommentID	NUMBER	True	True	10		Surrogate Key of the table
Type	NUMBER	True	False	1		Type of the comment 1 for user, 2 for playlist, 3 for album, 4 for songs, 5 for singers
RelatedID	NUMBER	True	False	8		Related id of the comment. For SONGS it is SongID, for users it is UserID so on...
CommentText	VARCHAR2	True	False	4000		Comment of user, song, playlist, singer entered by a user
RecordingDate	DATE	True	False		SYSDATE	Recording system time of comment
RecordingUser	NUMBER	True	False	8		Recording user of the comment

Constraints

Name	Type	Columns
FK_COMMENTS_ALBUMS	«FK»	RelatedID
FK_COMMENTS_PLAYLISTS	«FK»	RelatedID
FK_COMMENTS_RecordingUser	«FK»	RecordingUser
FK_COMMENTS_SINGERS	«FK»	RelatedID
FK_COMMENTS_SONGS	«FK»	RelatedID
FK_COMMENTS_USERS	«FK»	RelatedID
PK_COMMENTS	«PK»	CommentID

Relationships

Columns	Association
(RelatedID = UserID)	0..* 1 COMMENTS.FK_COMMENTS_USERS USERS. PK_USERS
(RecordingUser = UserID)	0..* 1 COMMENTS.FK_COMMENTS_RecordingUser USERS. PK_USERS
(RelatedID = AlbumID)	0..* 1 COMMENTS.FK_COMMENTS_ALBUMS ALBUMS. PK_ALBUMS
(RelatedID = SongID)	0..* 1 COMMENTS.FK_COMMENTS_SONGS SONGS. PK_SONGS
(RelatedID = PlaylistID)	0..* 1 COMMENTS.FK_COMMENTS_PLAYLISTS PLAYLISTS. PK_PLAYLISTS
(RelatedID = SingerID)	0..* COMMENTS.FK_COMMENTS_SINGERS

Columns	Association
	RS 1 SINGERS. PK_SINGERS

4.2.4 GENRES

Notes: Which contains genres (Rock, Pop, Turkish Folk Music, Arabic, Greek etc...)

Columns

Name	Type	Not Null	Unique	Len	Init	Description
GenreID	NUMBER	True	True	4		Surrogate Key of the table
GenreName	VARCHAR2	True	False	50		Genre Name (Rock, Arabic, Greek etc...)
RecordingDate	DATE	True	False		SYSDATE	Recording system date of the genre
RecordingUser	NUMBER	True	False	8		Recording user of the genre

Constraints

Name	Type	Columns
FK_GENRES_RecordingUser	«FK»	RecordingUser
PK_GENRES	«PK»	GenreID

Relationships

Columns	Association
(RecordingUser = UserID)	0..* GENRES.FK_GENRES_RecordingUser 1 USERS. PK_USERS
(GenreID = GenreID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_GenreID 1 GENRES. PK_GENRES

4.2.5 GENRE_RELATIONS

Notes: Holds singer genre relations, song genre relations, album genre relations.

Columns

Name	Type	Not Null	Unique	Len	Init	Description
GenreRelationID	NUMBER	True	True	8		Surrogate key of the table
GenreID	NUMBER	True	False	4		Genre ID of the relation table
Type	NUMBER	True	False	1		Type of the genre relation 2

	R					for playlist, 3 for album, 4 for songs, 5 for singers
RelatedID	NUMBER	True	False	8		Related id of the genre. For SONGS it is SongID, for users it is UserID so on...
RecordingDate	DATE	True	False		SYSDATE	Recording system date of the genre relation
RecordingUser	NUMBER	True	False	8		Recording user of the genre relation

Constraints

Name	Type	Columns
FK_GENRE_RELATIONS_ALBUMS	«FK»	RelatedID
FK_GENRE_RELATIONS_GenreID	«FK»	GenreID
FK_GENRE_RELATIONS_PLAYLISTS	«FK»	RelatedID
FK_GENRE_RELATIONS_SINGERS	«FK»	RelatedID
FK_GENRE_RELATIONS_SONGS	«FK»	RelatedID
PK_GENRE_RELATIONS	«PK»	GenreRelationID

Relationships

Columns	Association
(RelatedID = AlbumID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_ALBUMS 1 ALBUMS. PK_ALBUMS
(RelatedID = SongID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_SONGS 1 SONGS. PK_SONGS
(RelatedID = PlayListID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(GenreID = GenreID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_GenreID 1 GENRES. PK_GENRES
(RelatedID = SingerID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_SINGERS 1 SINGERS. PK_SINGERS

4.2.6 PLAYLISTS

Notes: Which contain playlists created by the users

Columns

Name	Type	Not Null	Unique	Len	Init	Description
PlayListID	NUMBER	True	True	8		Surrogate key of the table
PlaylistName	VARCHAR2	True	False	100		Name of the playlist
RecordingDate	DATE	True	False		SYSDATE	Recording system date of the playlist.
RecordingUser	NUMBER	True	False	8		Recording user of the playlist

Constraints

Name	Type	Columns
FK_PLAYLISTS_RecordingUser	«FK»	RecordingUser
PK_PLAYLISTS	«PK»	PlayListID

Relationships

Columns	Association
(RecordingUser = UserID)	0..* PLAYLISTS.FK_PLAYLISTS_RecordingUser 1 USERS. PK_USERS
(RelatedID = PlayListID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(RelatedID = PlayListID)	0..* COMMENTS.FK_COMMENTS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(RelatedID = PlayListID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(RelatedID = PlayListID)	0..* RATINGS.FK_RATINGS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS

4.2.7 RATINGS

Notes: Which contains ratings for playlists, songs, singers etc... Type attribute determines type of the rating (playlist rating, song rating ...)

Columns

Name	Type	Not Null	Unique	Len	Init	Description
RatingID	NUMBER	True	True	10		Surrogate Key of the table

	R					
Type	NUMBER	True	False	1		Type of the rating 2 for playlist, 3 for album, 4 for songs, 5 for singers
RelatedID	NUMBER	True	False	8		Related id of the rating. For SONGS it is SongID, for users it is UserID so on...
Rating	NUMBER	True	False	1		rating of related entity it can be from 0 to 5
RecordingDate	DATE	True	False		SYSDATE	Recording system time of rating
RecordingUser	NUMBER	True	False	8		Recording user of the rating

Constraints

Name	Type	Columns
FK_RATINGS_ALBUMS	«FK»	RelatedID
FK_RATINGS_PLAYLISTS	«FK»	RelatedID
FK_RATINGS_RecordingUser	«FK»	RecordingUser
FK_RATINGS_SINGERS	«FK»	RelatedID
FK_RATINGS_SONGS	«FK»	RelatedID
FK_RATINGS_USERS	«FK»	RelatedID
PK_COMMENTS	«PK»	RatingID

Relationships

Columns	Association
(RecordingUser = UserID)	0..* RATINGS.FK_RATINGS_RecordingUser 1 USERS. PK_USERS
(RelatedID = UserID)	0..* RATINGS.FK_RATINGS_USERS 1 USERS. PK_USERS
(RelatedID = PlaylistID)	0..* RATINGS.FK_RATINGS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(RelatedID = AlbumID)	0..* RATINGS.FK_RATINGS_ALBUMS 1 ALBUMS. PK_ALBUMS
(RelatedID = SingerID)	0..* RATINGS.FK_RATINGS_SINGERS 1 SINGERS. PK_SINGERS
(RelatedID = SongID)	0..* RATINGS.FK_RATINGS_SONGS 1 SONGS. PK_SONGS

4.2.8 SINGERS

Notes: Singers which recorded by the users

Columns

Name	Type	Not Null	Unique	Len	Init	Description
SingerID	NUMBER	True	True	8		Surrogate key of the table
SingerName	VARCHAR2	True	False	50		Singer (or Group) name
Type	NUMBER	True	False	1		1 for Single, 2 for groups, 3 for Choirs
Gender	VARCHAR2	True	False	1		Gender of the singer (type 1 single singers) M for Male, F for Female and O for Others
RecordingDate	DATE	True	False		SYSDATE	Recording system date of the singer
RecordingUser	NUMBER	True	False	8		Recording user of the singer

Constraints

Name	Type	Columns
FK_SINGERS_RecordingUser	«FK»	RecordingUser
PK_SINGERS	«PK»	SingerID

Relationships

Columns	Association
(SingerID = SingerID)	0..* ALBUMS.FK_ALBUMS_SingerID 1 SINGERS. PK_SINGERS
(SingerID = SingerID)	0..* SONGS.FK_SONGS_SingerID 1 SINGERS. PK_SINGERS
(RecordingUser = UserID)	0..* SINGERS.FK_SINGERS_RecordingUser 1 USERS. PK_USERS
(RelatedID = SingerID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_SINGERS 1 SINGERS. PK_SINGERS
(RelatedID = SingerID)	0..* RATINGS.FK_RATINGS_SINGERS 1 SINGERS. PK_SINGERS
(RelatedID = SingerID)	0..* COMMENTS.FK_COMMENTS_SINGERS 1 SINGERS. PK_SINGERS
(RelatedID = SingerID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_SINGERS 1 SINGERS. PK_SINGERS

4.2.9 SONGS

Notes:

Which contains songs recorded by the users

Columns

Name	Type	Not Null	Unique	Len	Init	Description
SongID	NUMBER	True	True	8		Surrogate key of the table
SongName	VARCHAR2	True	False	50		Name of the song
Format	NUMBER	True	False	1		Format of the song currently we support only 1 for mp3
Data	BLOB	False	False			Binary data of the song in supported format
RecordingDate	DATE	True	False		SYSDATE	Recording date of the song
RecordingUser	NUMBER	True	False	8		Recording user of the song
AlbumID	NUMBER	False	False	8		Album of the song (Null if user don't know album name or album doesn't exists in system)
SingerID	NUMBER	False	False	8		Singer of the song

Constraints

Name	Type	Columns
FK_SONGS_AlbumID	«FK»	AlbumID
FK_SONGS_RecordingUser	«FK»	RecordingUser
FK_SONGS_SingerID	«FK»	SingerID
PK_SONGS	«PK»	SongID

Relationships

Columns	Association
(RelatedID = SongID)	0..* COMMENTS.FK_COMMENTS_SONGS 1 SONGS. PK_SONGS
(SingerID = SingerID)	0..* SONGS.FK_SONGS_SingerID 1 SINGERS. PK_SINGERS
(AlbumID = AlbumID)	0..* SONGS.FK_SONGS_AlbumID 1 ALBUMS. PK_ALBUMS
(RecordingUser = UserID)	0..* SONGS.FK_SONGS_RecordingUser 1 USERS. PK_USERS
(RelatedID = SongID)	0..* GENRE_RELATIONS.FK_GENRE_RELATIONS_SONGS 1 SONGS. PK_SONGS
(RelatedID = SongID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_SONGS TAG_RELATIONS.FK_TAG_RELATIONS_SONGS

Columns	Association
	1 SONGS. PK_SONGS
(RelatedID = SongID)	0..* RATINGS.FK_RATINGS_SONGS 1 SONGS. PK_SONGS

4.2.10 TAGS

Notes: Tag table that contains tags recorded by the user

Columns

Name	Type	Not Null	Unique	Len	Init	Description
TagID	NUMBER	True	True	4		Surrogate key of the table
TagName	VARCHAR2	True	False	50		Name of the Tag
RecordingDate	DATE	True	False		SYSDATE	Recording system time of the table
RecordingUser	NUMBER	True	False	8		Recording user of the tag

Constraints

Name	Type	Columns
FK_TAGS_RecordingUser	«FK»	RecordingUser
PK_TAGS	«PK»	TagID

Relationships

Columns	Association
(RecordingUser = UserID)	0..* TAGS.FK_TAGS_RecordingUser 1 USERS. PK_USERS
(TagID = TagID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_TagID 1 TAGS. PK_TAGS

4.2.11 TAG_RELATIONS

Notes: Holds singer genre relations, song genre relations, album genre relations.

Columns

Name	Type	Not Null	Unique	Len	Init	Description
TagRelationID	NUMBER	True	True	8		Surrogate key of the table
TagID	NUMBER	True	False	4		Tag ID of the relation table
Type	NUMBER	True	False	1		Type of the tag relation 2 for

	R					playlist, 3 for album, 4 for songs, 5 for singers
RelatedID	NUMBER	True	False	8		Related id of the tag. For SONGS it is SongID, for users it is UserID so on...
RecordingDate	DATE	True	False		SYSDATE	Recording system date of the tag relation
RecordingUser	NUMBER	True	False	8		Recording user of the tag relation

Constraints

Name	Type	Columns
FK_TAG_RELATIONS_ALBUMS	«FK»	RelatedID
FK_TAG_RELATIONS_PLAYLISTS	«FK»	RelatedID
FK_TAG_RELATIONS_SINGERS	«FK»	RelatedID
FK_TAG_RELATIONS_SONGS	«FK»	RelatedID
FK_TAG_RELATIONS_TagID	«FK»	TagID
PK_GENRE_RELATION	«PK»	TagRelationID

Relationships

Columns	Association
(TagID = TagID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_TagID 1 TAGS. PK_TAGS
(RelatedID = PlayListID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_PLAYLISTS 1 PLAYLISTS. PK_PLAYLISTS
(RelatedID = AlbumID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_ALBUMS 1 ALBUMS. PK_ALBUMS
(RelatedID = SongID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_SONGS 1 SONGS. PK_SONGS
(RelatedID = SingerID)	0..* TAG_RELATIONS.FK_TAG_RELATIONS_SINGERS 1 SINGERS. PK_SINGERS

4.2.12 USERS

Notes: Which contains users of the Web VR system

Columns

Name	Type	Not Null	Unique	Len	Init	Description
UserID	NUMBER	True	True	8		Surrogate key of the table
UserName	VARCHAR2	True	False	50		Name of the user
UserSurname	VARCHAR2	True	False	50		Surname of the user
email	VARCHAR2	True	True	50		e-mail of the user
Password	VARCHAR2	True	False	20		Password of the user
Gender	VARCHAR2	True	False	1		Gender of the user. M for male, F for Female
BirthDate	DATE	False	False			Birth date of the user
Nationality	VARCHAR2	False	False	3		Nationality of the user. (TR) - Turkiye, (USA) USA
EntreranceDate	DATE	True	False			EntreranceDate of the user

Constraints

Name	Type	Columns
PK_USERS	«PK»	UserID

Relationships

Columns	Association
(RecordingUser = UserID)	0..* RATINGS.FK_RATINGS_RecordingUser 1 USERS. PK_USERS
(RelatedID = UserID)	0..* RATINGS.FK_RATINGS_USERS 1 USERS. PK_USERS
(RelatedID = UserID)	0..* COMMENTS.FK_COMMENTS_USERS 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* COMMENTS.FK_COMMENTS_RecordingUser 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* GENRES.FK_GENRES_RecordingUser 1 USERS. PK_USERS

Columns	Association
(RecordingUser = UserID)	0..* SINGERS.FK_SINGERS_RecordingUser 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* SONGS.FK_SONGS_RecordingUser 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* ALBUMS.FK_ALBUMS_RecordingUser 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* PLAYLISTS.FK_PLAYLISTS_RecordingUser 1 USERS. PK_USERS
(RecordingUser = UserID)	0..* TAGS.FK_TAGS_RecordingUser 1 USERS. PK_USERS

5 DESIGN DIAGRAMS

5.1 Class Diagram

Although WEB-VR has detailed database tables and element, it has simple classes that mainly link the interface with the WEB-VR database. Our two most important classes are user and DbManager.

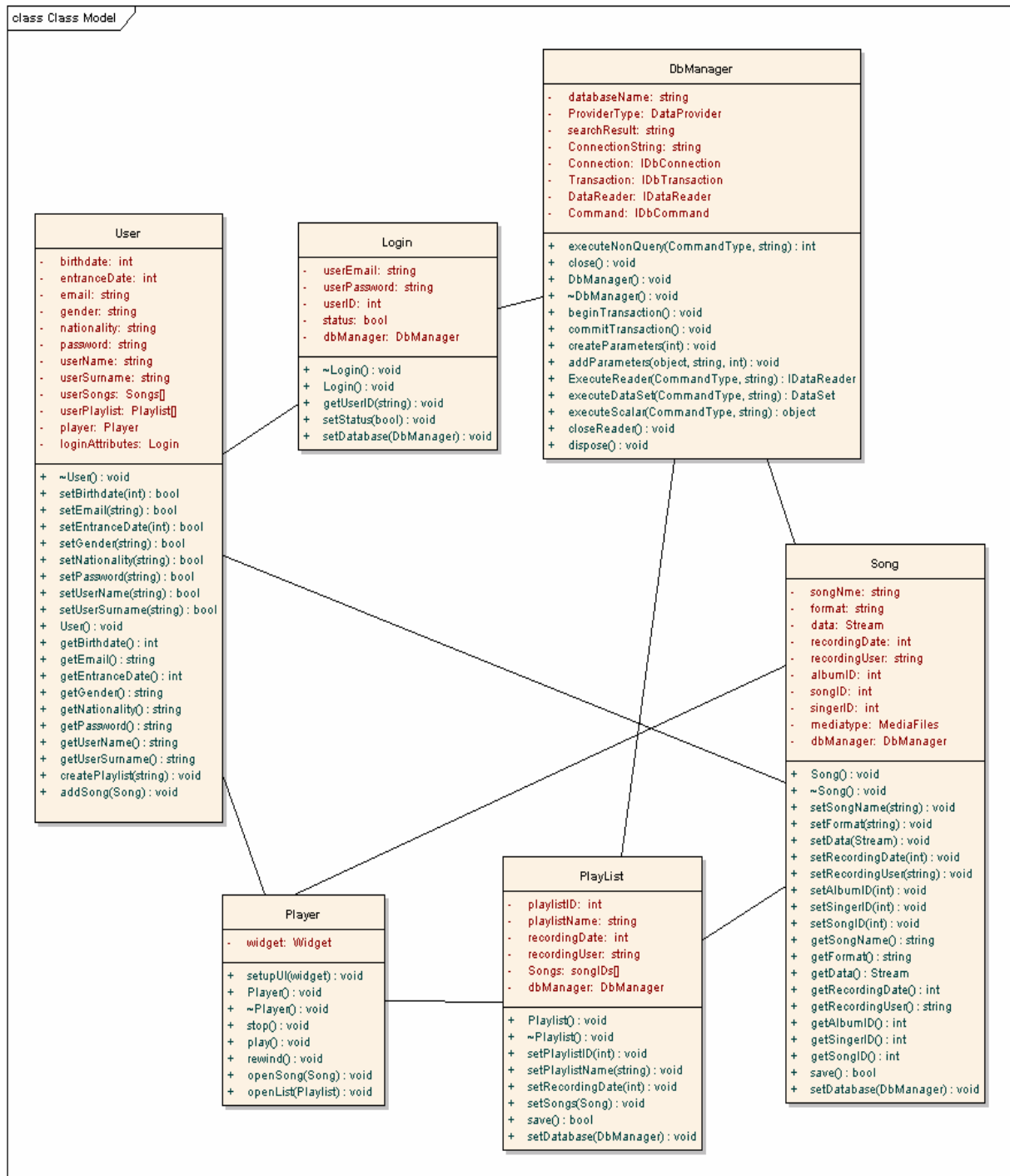
User classes are important since user is directly in relationship with user class. All the songs, videos, playlist elements and other options such as search, detailed search are controlled from this class. This class is directly under control of user. Also this class plays a role as container class since it is a main class where other classes are created in this class.

DbManager acts as a library since it is necessary to the database connection. All classes that have functionality with the database are using this class in order to connect to the database.

DbManager includes specific operations for the programming language that will be used in the implementation phase. On the other hand the attributes such as databaseName, searchResults and their corresponding operations are not specific to the programming language. So the some of the functionalities of DbManager is taken from out source. That outsource will be given in the references of DEMO.

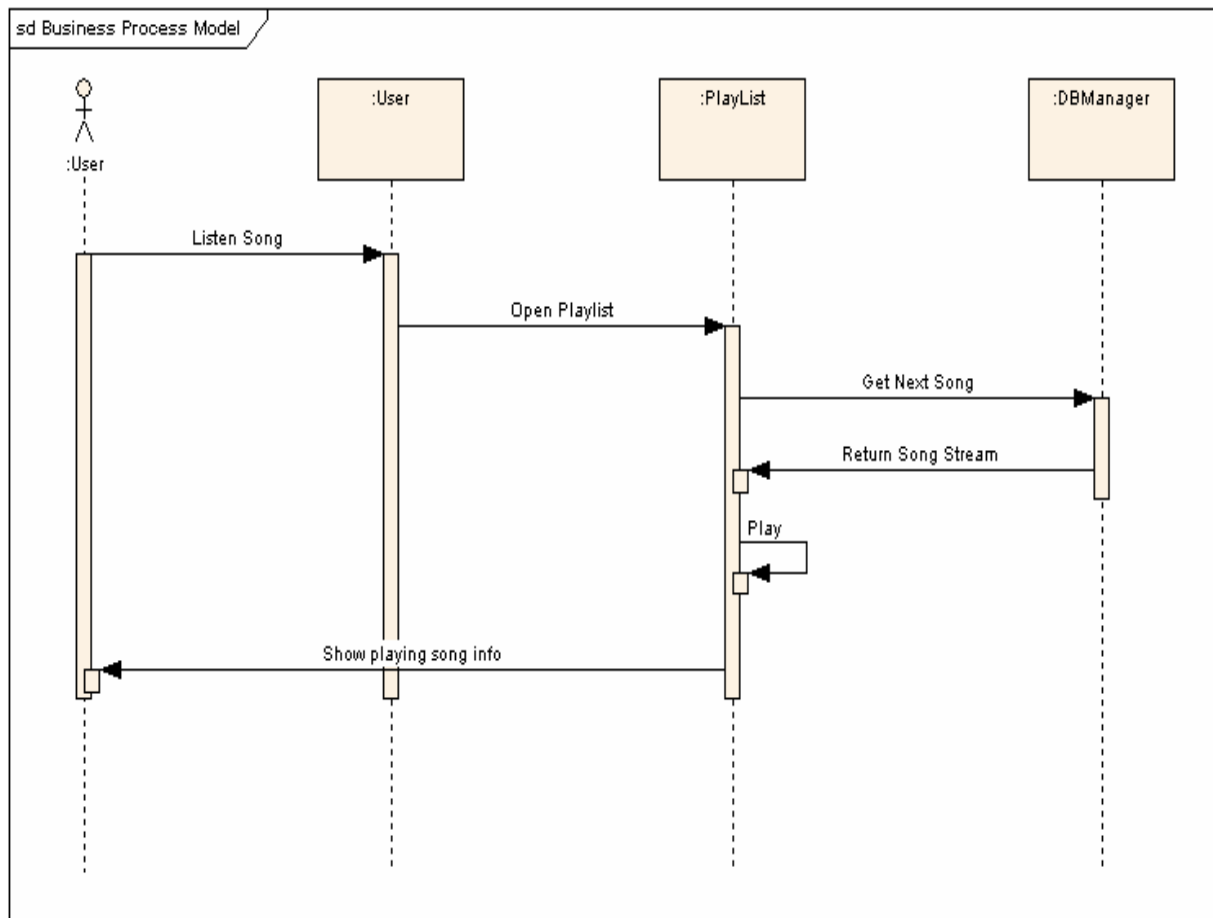
In our design we tried to eliminate all the possible complexities in order to show our design in an understandable way as much as possible. In order to do so, We did not include the minor classes that are only include small GUI functionalities or passes their functionalities to the our major classes. So in the below class diagram only shows the major classes. Minor classes such as help menus, search and detailed search screen are eliminated.

WEB-VR Class diagram is shown below:

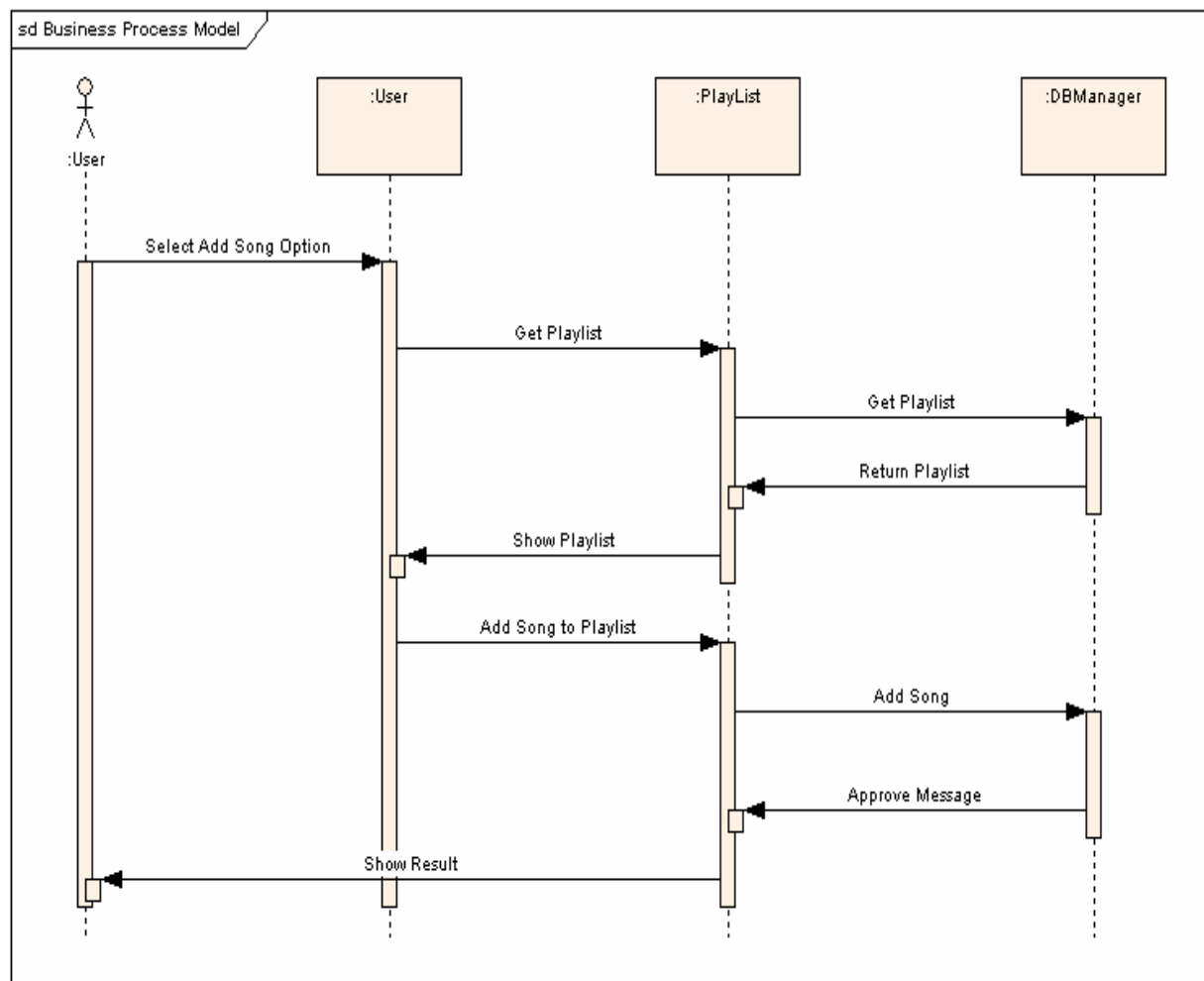


5.2 Sequence Diagrams

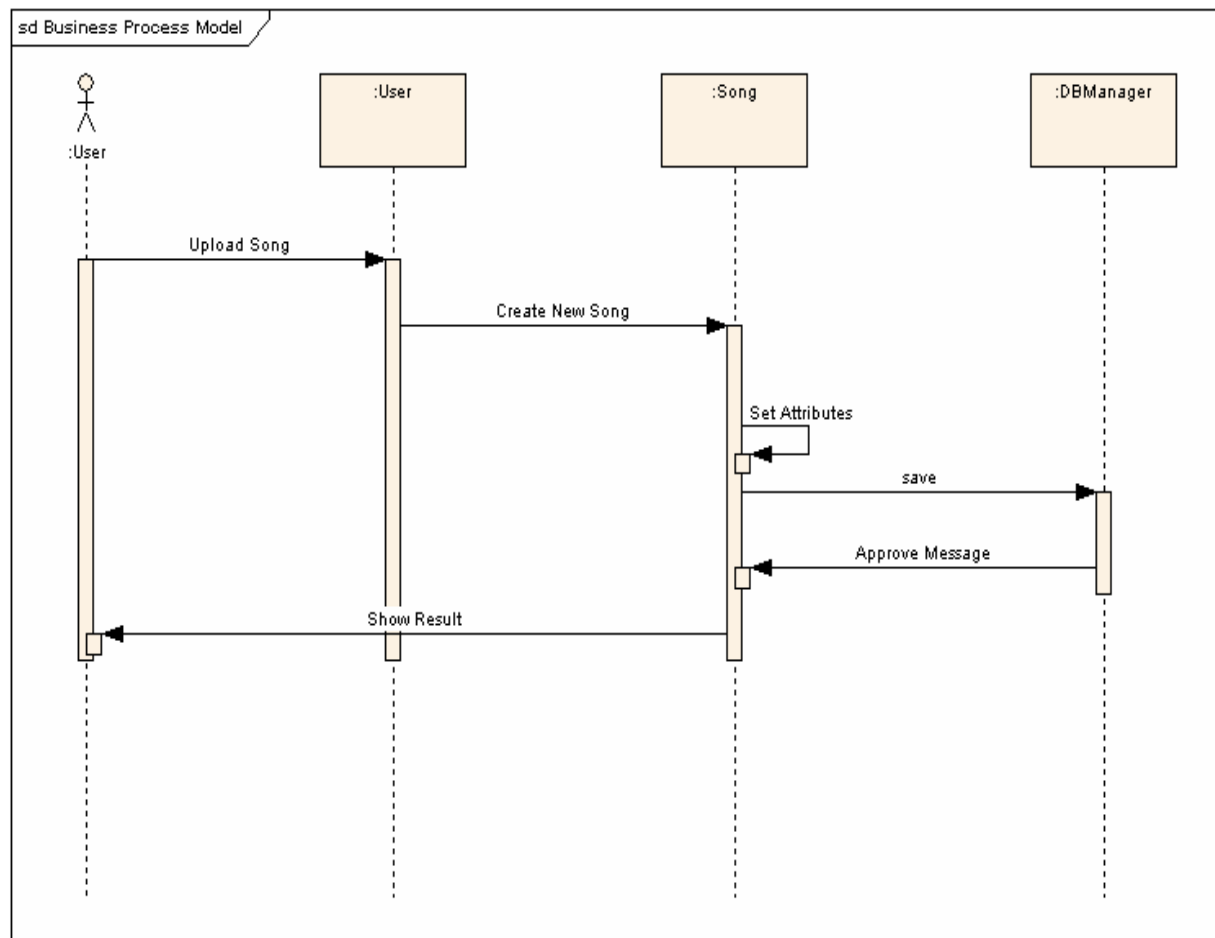
5.2.1 Listen Songs



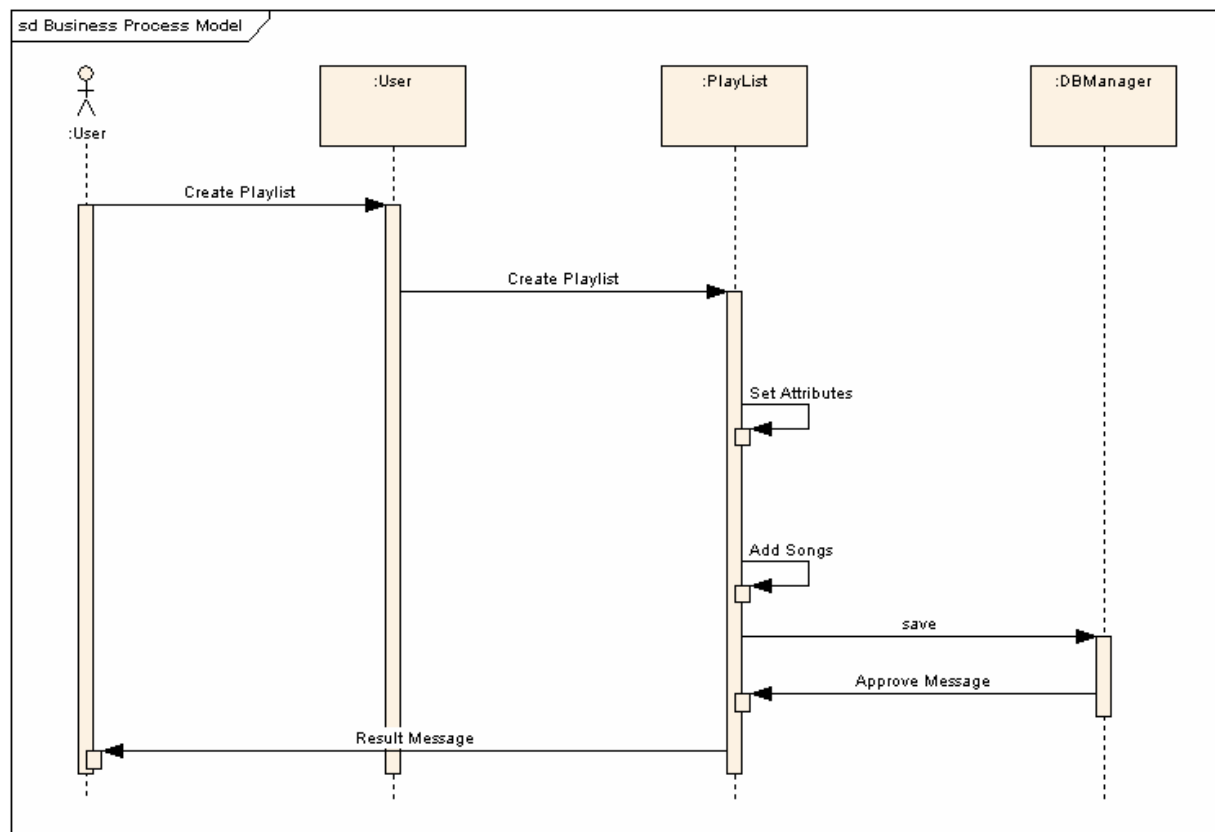
5.2.2 Add Song to Playlist



5.2.3 Upload Song to Database

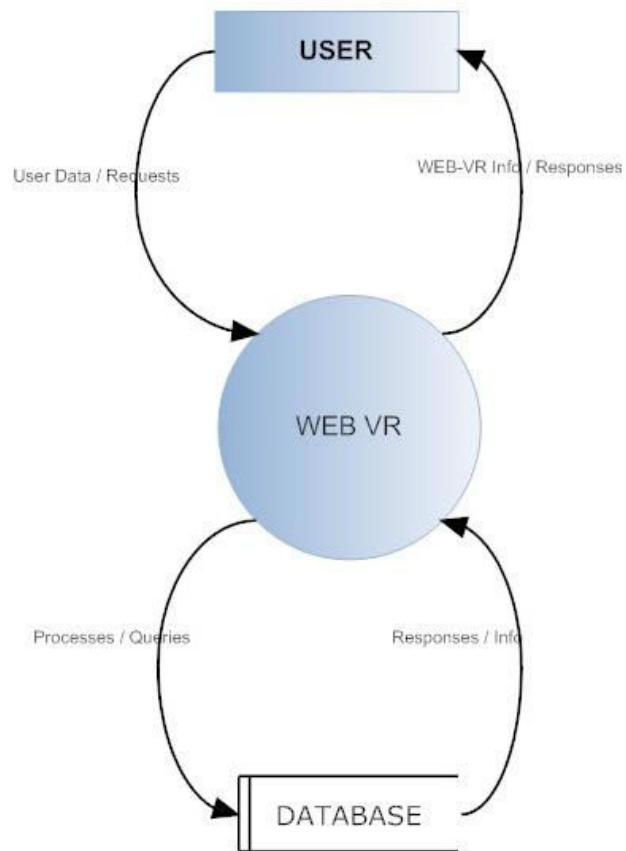


5.2.4 Create Playlist

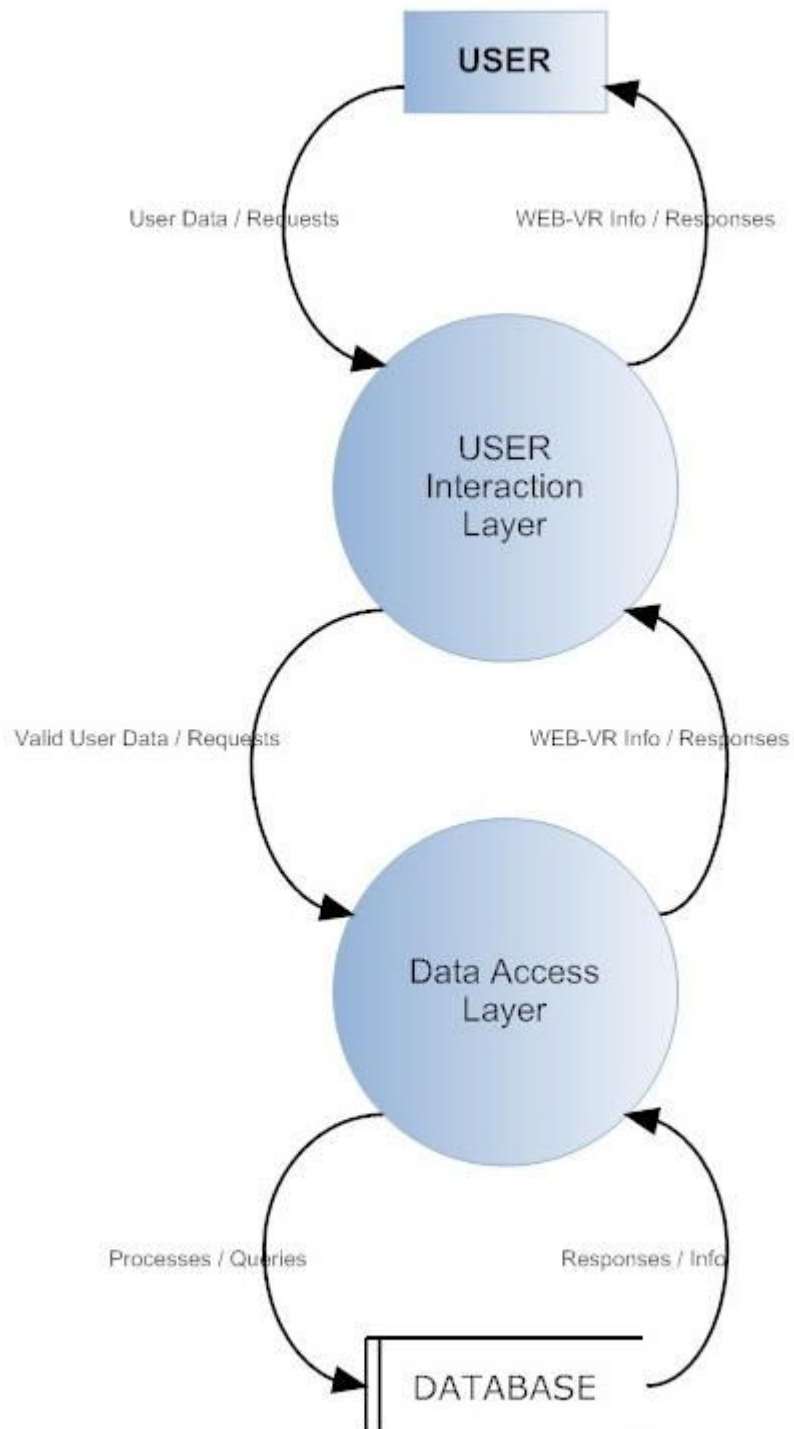


5.3 Data Flow Diagrams

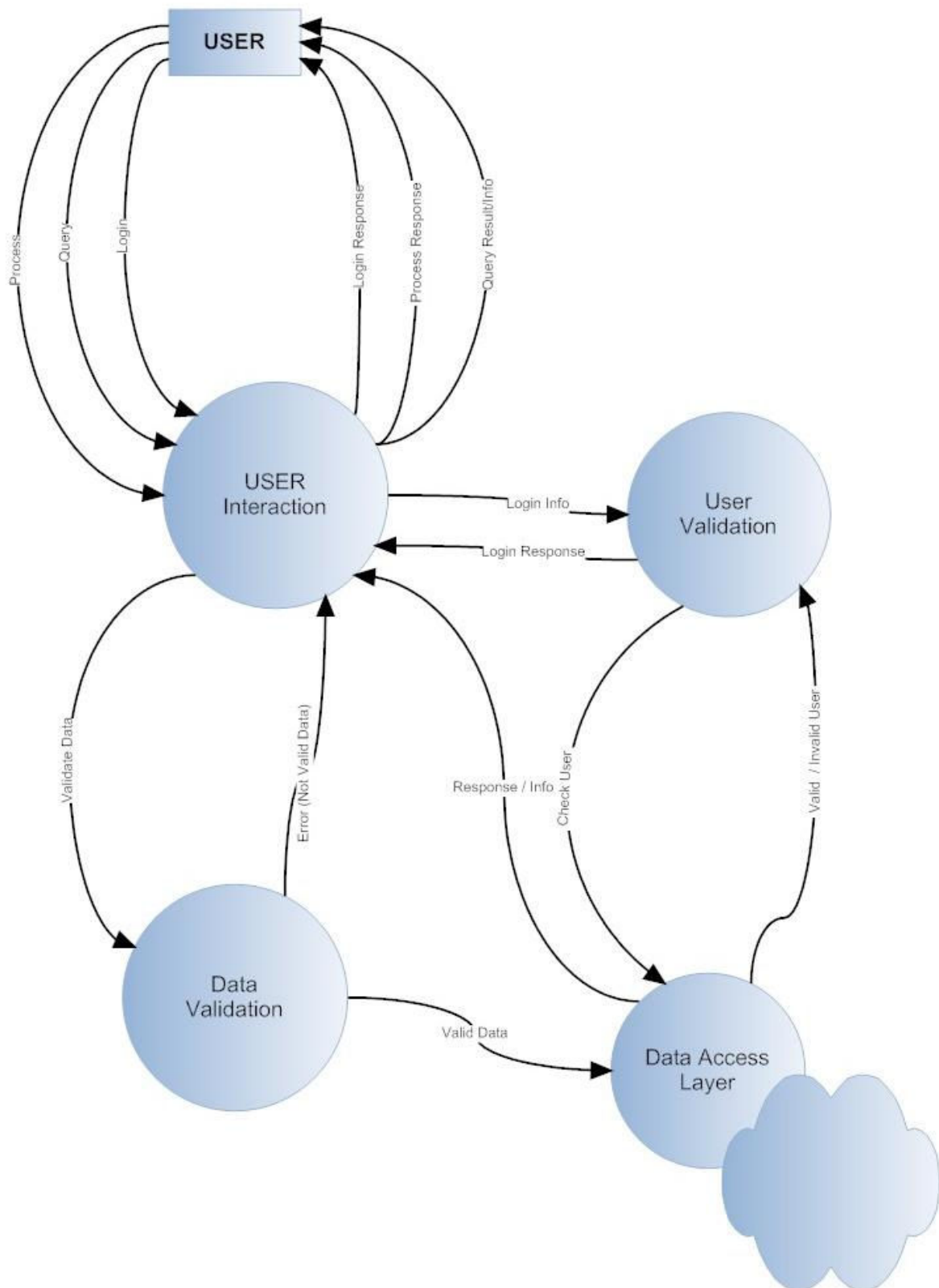
WEB-VR LEVEL-0 DFD



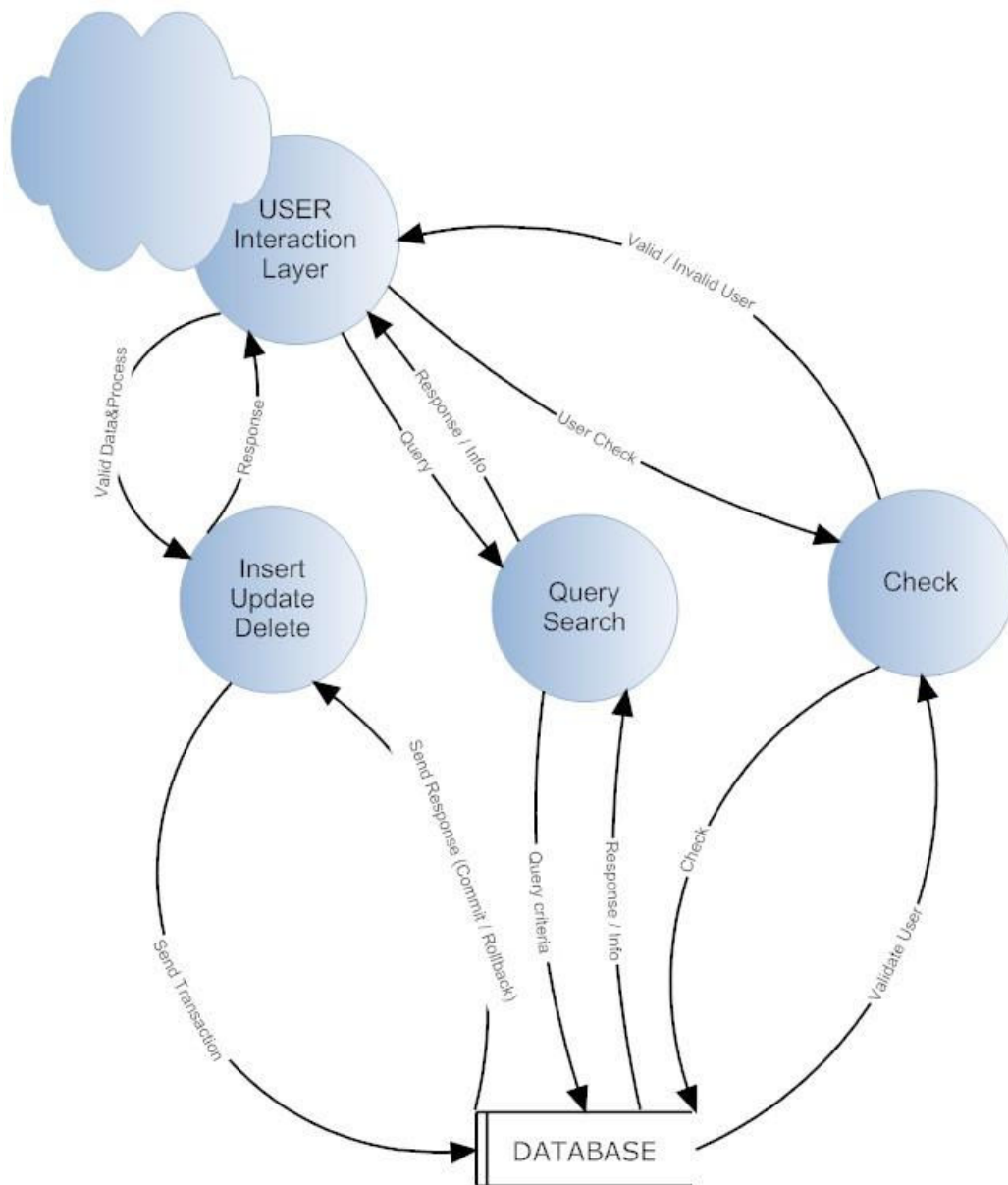
WEB-VR LEVEL-1 DFD



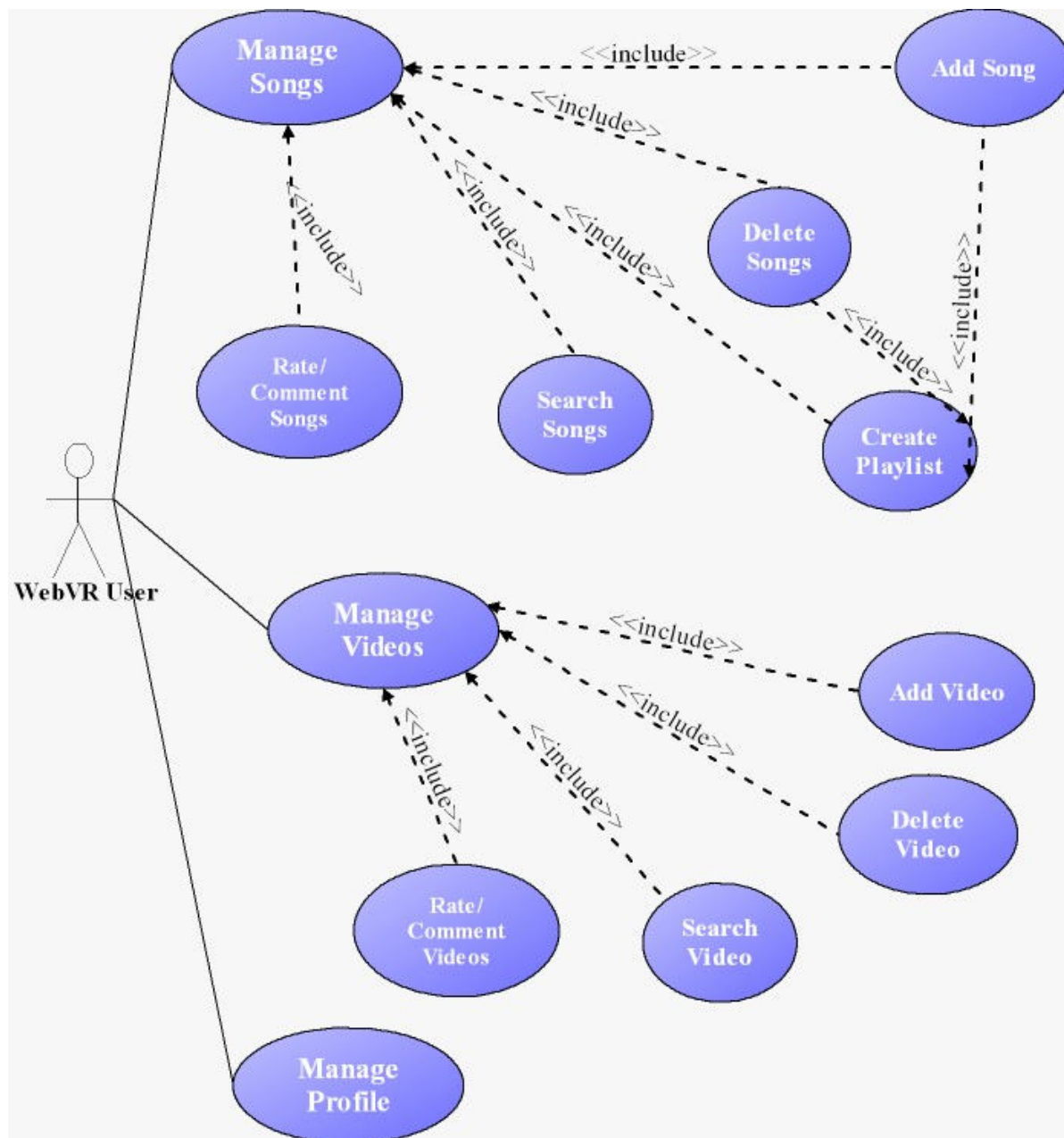
USER Interaction Layer LEVEL-2 DFD



Data Access Layer LEVEL-2 DFD

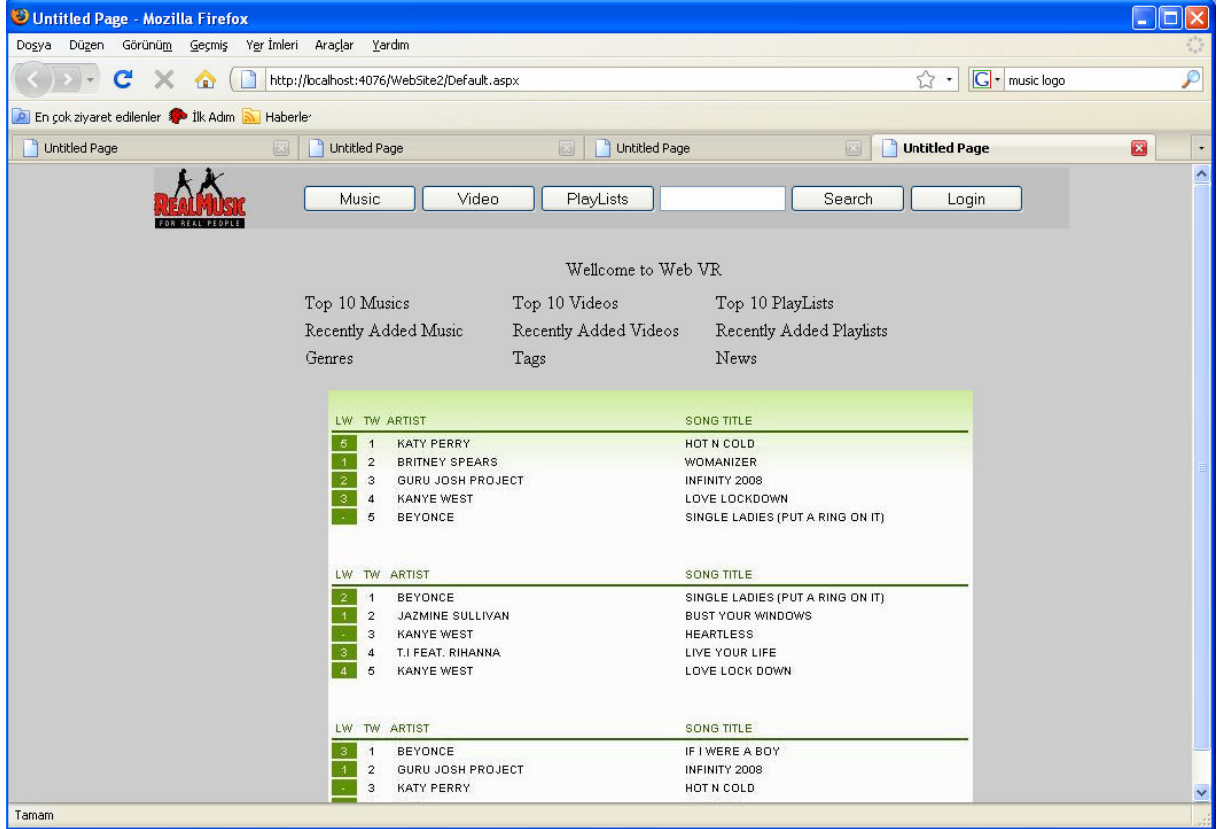


5.4 Use Case

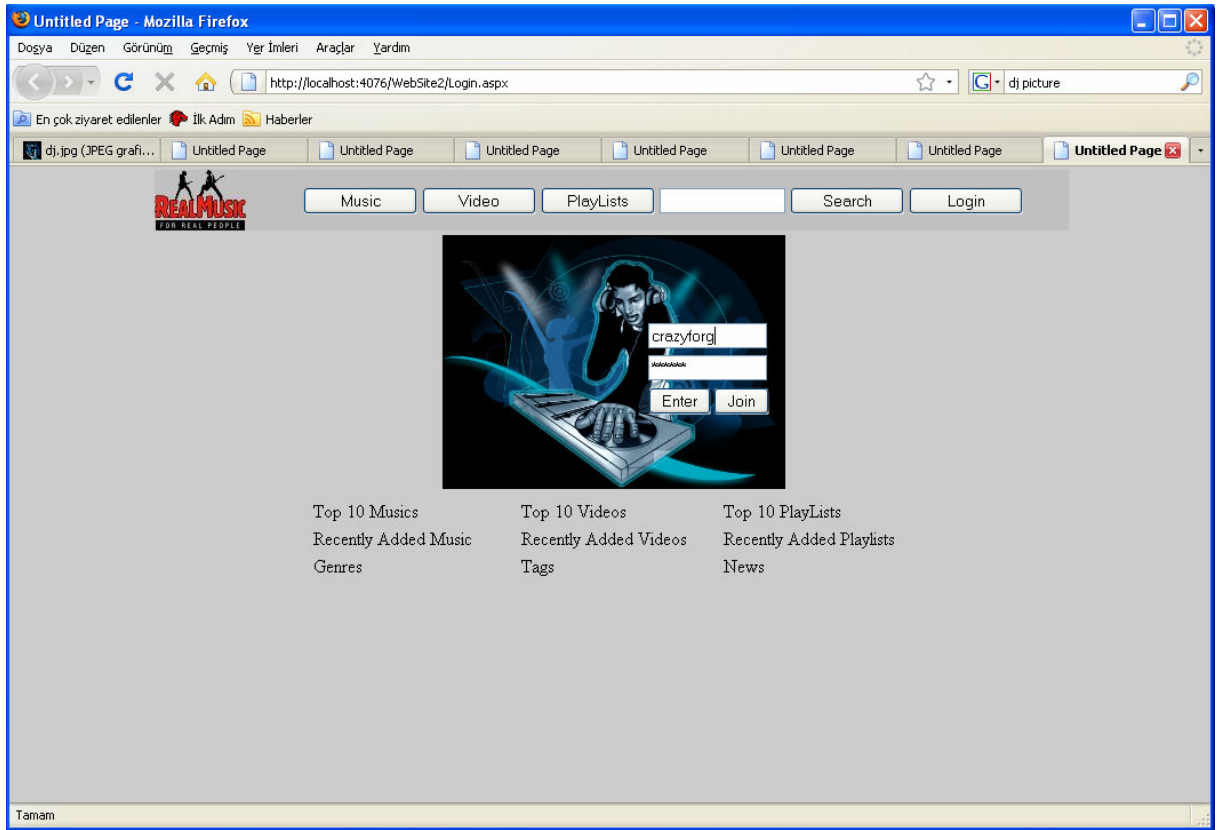


6 USER INTERFACE DESIGN

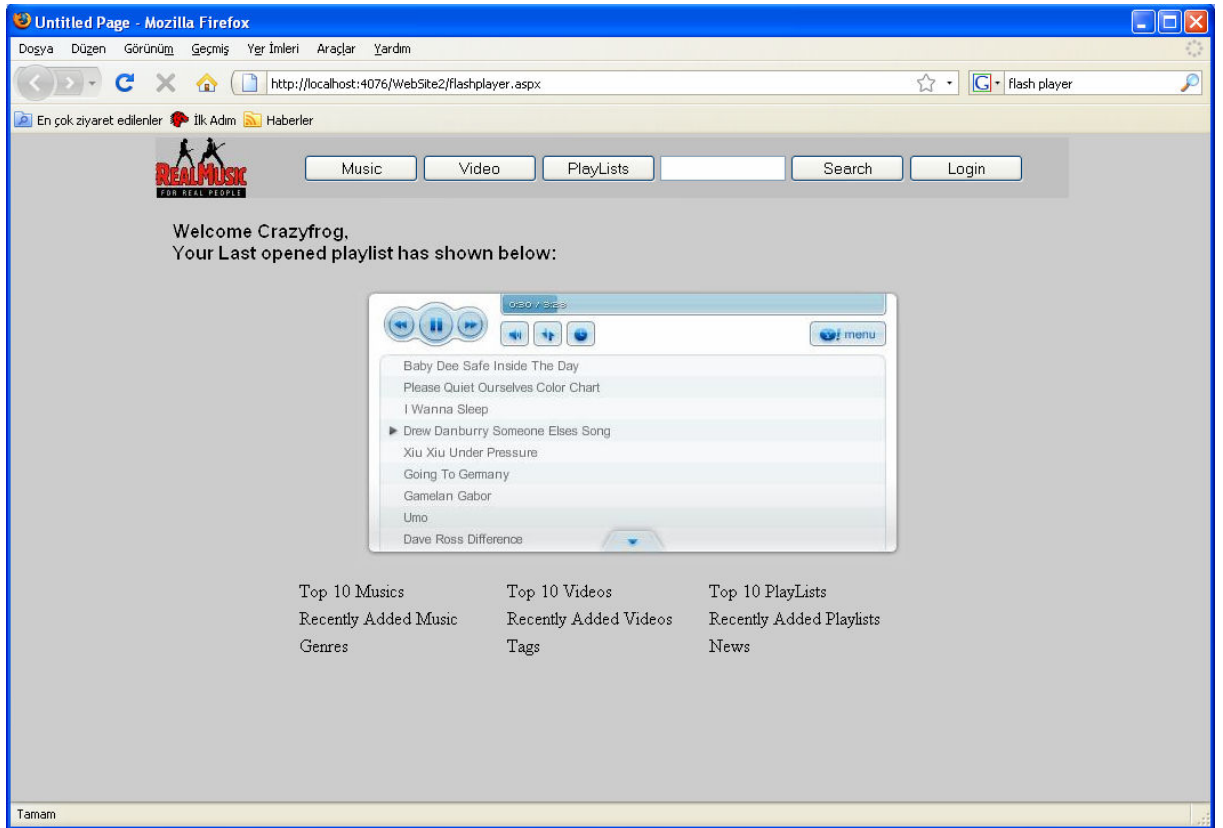
Our user Interfaces are simple and they are just demonstrate the usage of WEB-VR.
These interfaces will be changed in the implementation phase:



This is our welcome screen. User can easily top rated songs and videos from this screen. Also the news part will be available.



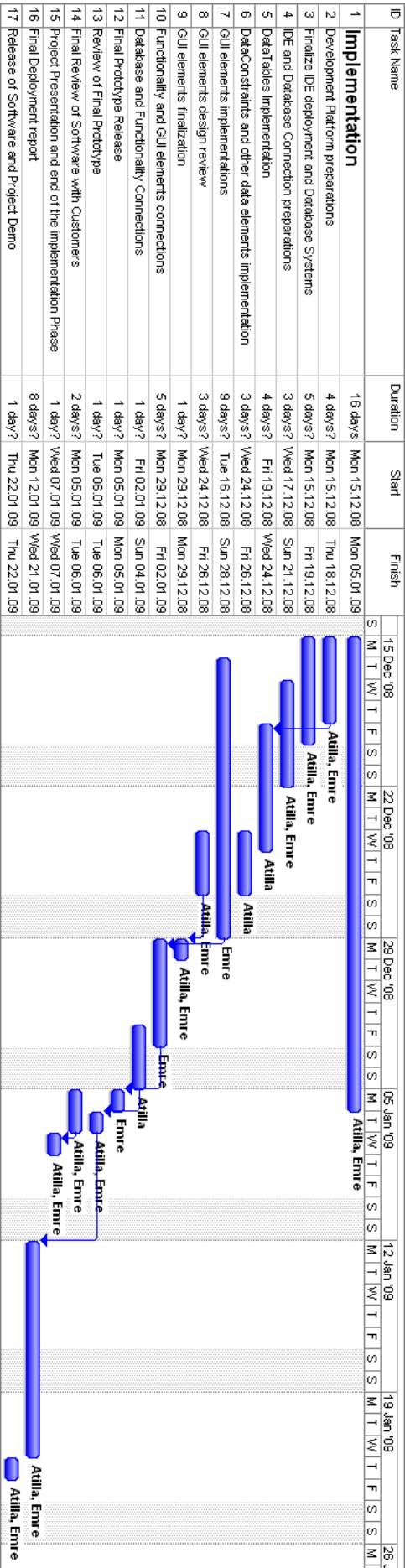
This is our login screen when user can enter his name and password in order to access the WEB-VR music and radio portal.



This is a sample player which is used our prototype. We implement and use this player in order to observe the possible advantages and disadvantages of the flash players. This player is capable of showing songs and playlists. However it is lack of showing videos. Our final project includes different flash players for showing audio and video files.

7 UPDATED PROJECT PLAN

We updated our project plan in a more efficient and reliable way. We have too much work to do in a short period of time. We keep our deadlines short and strict as possible. The updated project plan will be shown below:



8 CONCLUSION

In this design report, we as team described what we have done so far. Although the final design may show small changes during implementation, the overall architecture of the system will be the one as described in this report.

In this report also we mentioned about changes from requirement analysis and the future works.

The preparation phase of this report provides us the opportunity to review our previous design and requirements. With the help this opportunity we revised our requirements and we changed our design to more reliable and efficient one. Also we expanded some of our requirements, we revised our ER diagrams, we changed our database design and we changed our development environment.

We briefly described our design and its goal in this report. After the submission of this report we will immediately start the implementation phase of the project. Since our time is short and we have too much work to do, we keep our deadlines short and strict. As we show in the project plan part, we will work more efficiently to produce a qualified web based application.

In the end of this semester we will be finished WEB-VR and it will be a reliable and efficient web based application.