Erasmus Mundus
master course in
Data Mining and
Knowledge Management
a european master

<u>Course:</u>

Advanced Databases

<u>Project:</u>

Vinyl Record Collection Manager

<u>Authors:</u>

Brindusa Smaranda
Diana Nicoleta Popa
Jean Paul Stocca
Mohammad Ghufran
Rajani Chulyadyo

Table of Contents

Abstract

The main purpose of this project is to develop a Web Application that supports a Relational Database on the back-end. In this document, the design of the Database, the Web application and its functionality will be presented. This presentation will engulf three topics: Use Case Diagrams, Database Modeling (through an Entity Relationship Diagram) and a view of how the Web Application will look like.

The Web application follows two goals:

(i) Sharing a genre-oriented reference database for sound experts
(ii) Managing personal record libraries.

Audiophiles, serious collectors and DJs are willing to maintain a shared reference database of rare and cool vinyl records from their preferred genre. They also require tools to manage their own library of records and gather community of sound experts in an open platform. This document will describe how such users can manage their records and soundtracks, and also how they can share them with other users.

1. Use Cases

In this section, a set of use cases will be presented along with their corresponding work-flow. This analysis will aid the understanding of functional requirements of the project. Initially, there will be a general view of the system, and in the subsequent sections a detailed view will be explained. It is important to be noted that for each use case diagram (except the general view) there will be at least one description table

1.1. General system use case

Figure 1.1 presents a general view of what the user can do with Record Manager application

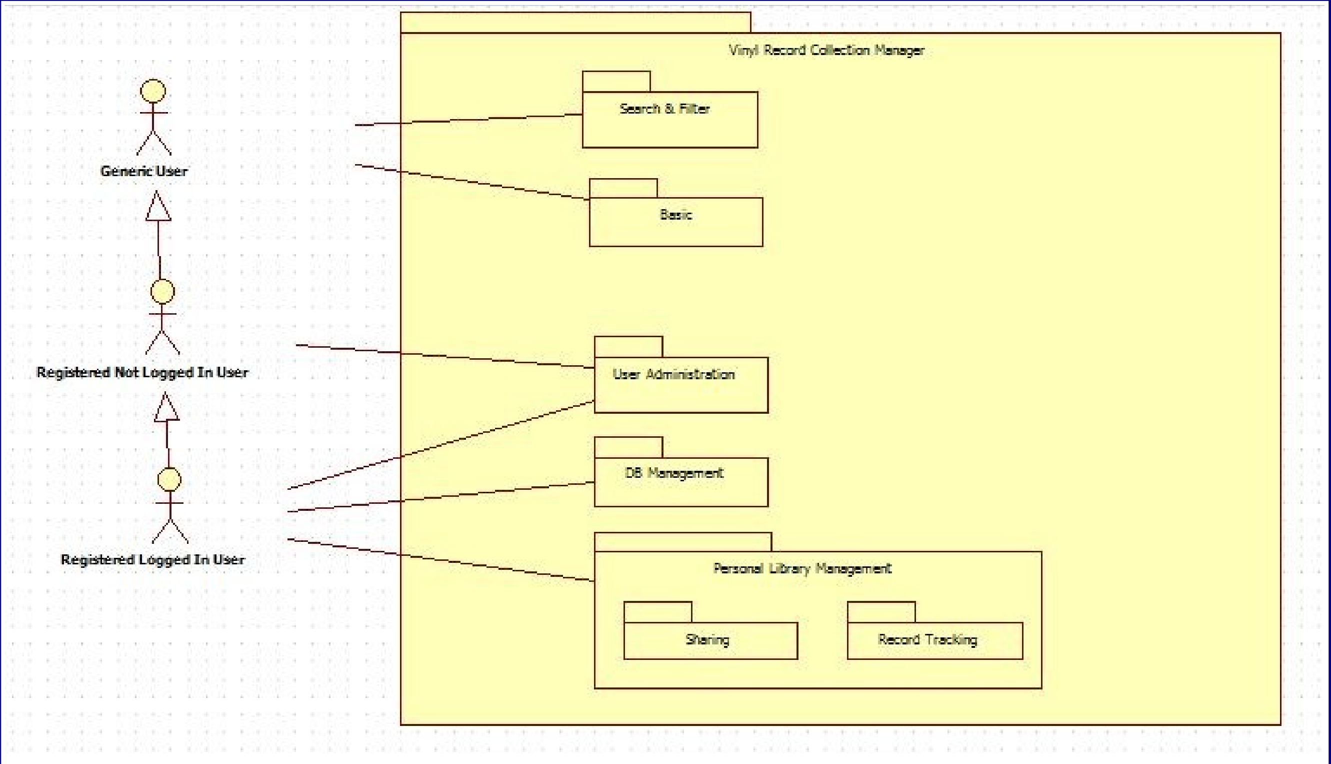

Vinyl Record Collection Manager
Search & Filter
Basic
User Administration
DB Management
Personal Library Management
Sharing
Record Tracking
Generic User
Registered Not Logged In User
Registered Logged In User

Figure 1.1- Use Case Diagram: General View

What is important to notice in Figure 1.1 is:

1 The different sub-systems where the user will interact:

a) Search & Filter
b) Basic
c) User Administration
d) DB management
e) Personal Library management, sharing and Record tracking.

2 There are 3 actors involved:

a) Generic user, not registered.
b) Registered but not logged in User
c) Registered and logged in User

In the following sections, the interactions between actors and sub-systems will be expanded and thoroughly analyzed.

1.2. Use cases for generic users

Figure 1.2 shows what a generic user can do within the Record Manager application. In order to search, the user is not required to be logged in nor registered. On the other hand The generic user can change the language of the application interface and he can also register.

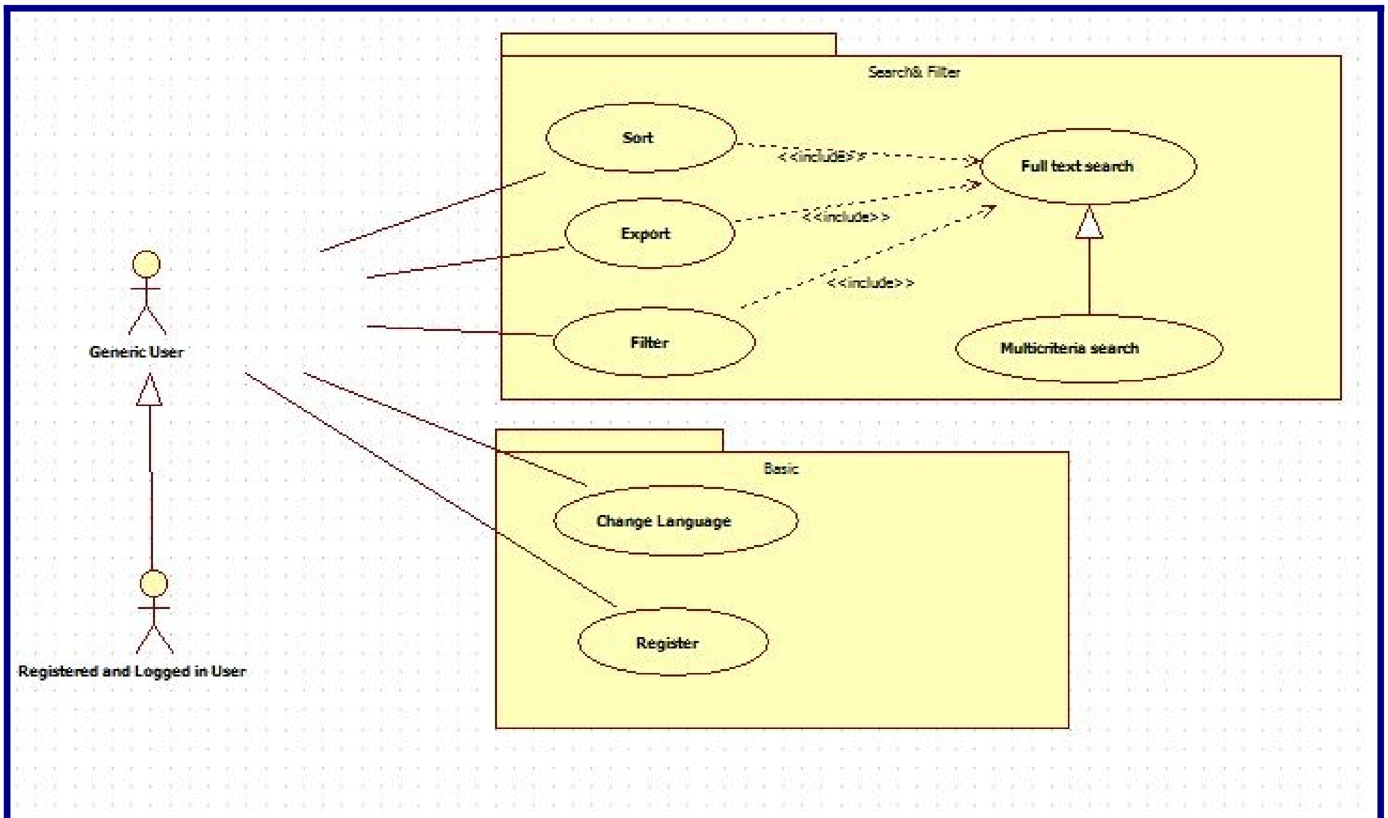


Figure 1.2- Use Case Diagram for generic user: Search & Filter and Basic

As a hypothesis, it has been agreed that the user can only sort, export and filter, only after performing a search. This latter use case is a generalization of the multi-criteria search. In the below tables the description of these use cases will be presented, rendering the corresponding work-flow.

Use Case	Change language
Description	User changes the language of the web application interface.
Actors	Generic User
Assumptions	User is within the web application
Steps	1 User selects a new language by clicking a flag icon on the screen. 2 System changes the language of the web application.

Table 1.2.1- Use Case Description: Change Language

Use Case	Register
Description	The user creates an account for the Web Application Manager
Actors	Generic user
Steps	1 The user presses register 2 System redirects user to registration and prompts the user to fill in a form 3 User enters his first name 4 User enters his last name 5 User enters his email 6 REPEAT 6.1 User enters his username 6.2 System checks uniqueness and displays an agreement sign. UNTIL username is unique. 7 REPEAT 7.1 User enters password 7.2 System checks password format and displays agreement sign UNTIL password meets requirements 8 REPEAT 8.1 User re-enters password 8.2 System checks if re-entered password matches initial password and displays agreement sign. UNTIL re-entered password matches initial password 9 User presses submit 10 System adds the information to DB
Variations	#1. The user may click login or he may attempt other tasks that requires login in (DB management, Personal library management, Record tracking, Sharing, etc)

Table 1.2.2- Use Case Description: Register

Use Case	Export **uses** full text search
Description	User exports the search results
Actors	Generic user
Assumptions	User is within the web application search screen and there are search results
Steps	1 User clicks export drop down menu 2 System displays the export format options 3 User selects a format (ex: csv) 4 System pops out a window prompting for a location in the user's hard drive to store the results 5 User provides location. 6 User clicks export. 7 System appends the information under the desired format and saves the file in user's hard drive

Table 1.2.3- Use Case Description: Export

Use Case	Search
Description	User enters a random text to query the common DB.
Actors	Generic user
Assumptions	User is within the web application
Steps	1 User points to the search form 2 User enters a text. 3 User clicks search button. 4 System attempts to match the text against the common DB using a predefined criteria 5 IF there is at least one successful match THEN 5.1 System displays the matched results ELSE 5.2 System displays a 'no results found' sign.

Table 1.2.4- Use Case Description: Full Text Search

Use Case	Multi-criteria search
Description	The user can provide additional details to his search in order to refine it.
Actors	Generic user
Assumption	User is within the web application search screen
Steps	1 User clicks on advanced filtering button 2 System redirects user to Advanced filtering screen. 3 User fills additional search parameters 4 User clicks search 5 System attempts to match the text against the common DB using a predefined criteria 6 IF there is at least one successful match THEN 6.1 System displays the matched results ELSE 6.2 System displays a 'no results found' sign.

Table 1.2.5- Use Case Description: Multi-criteria Search

Use Case	Filter **uses** full text search
Description	User filters the search results by predefined filters
Actors	Generic User
Assumptions	User is within the web application search screen and there are search results
Steps	1 Use Table 1.2.5 steps but at #3, the user only fills one parameter.

Table 1.2.6- Use Case Description: Filter

Use Case	Sort **uses** full text search
Description	User sorts the search results according to predefined options
Actors	Generic User
Assumptions	User is within the web application search screen and there are search results
Steps	1 User click on a column name 2 System re orders the information according to user's click

Table 1.2.7- Use Case Description: Sort

1.3. Use cases for registered users: database management

In Figure 1.3 the use cases regarding database management are illustrated. Each of these are described with the word entry.

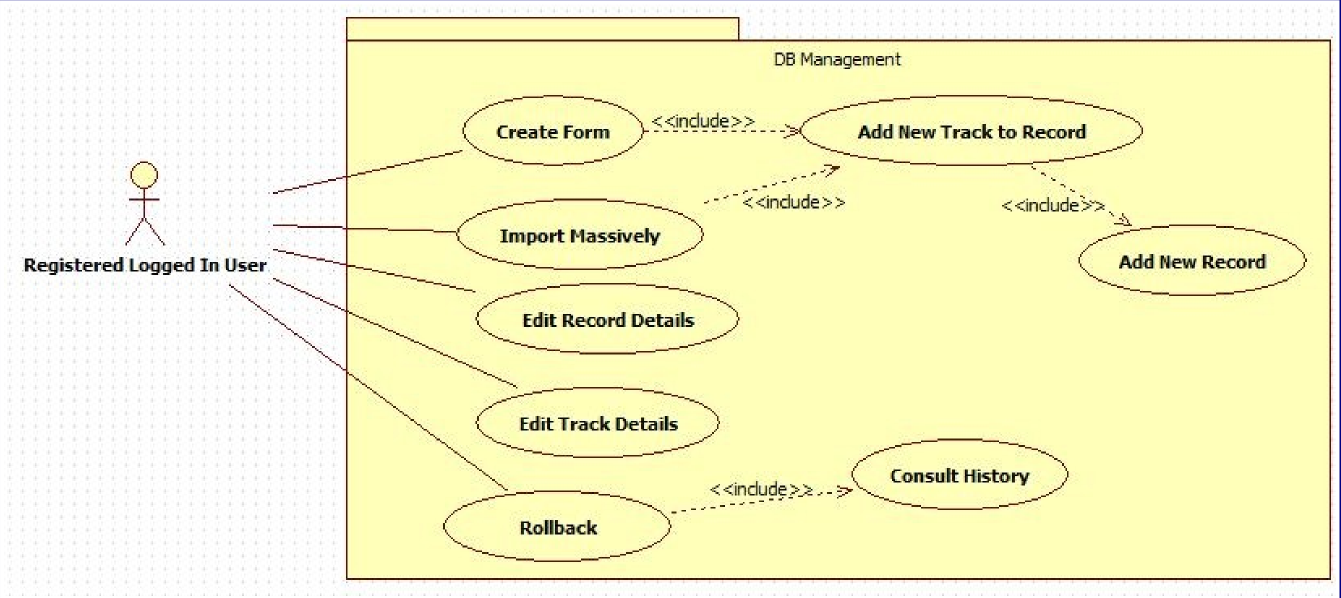


Figure 1.3- Use Case Diagram: DB Management

In the below tables the description of these use cases will be presented, rendering the corresponding work-flow

Use Case	Add new track **uses** add new record
Description	User can add a track into the reference database.
Actors	Registered Logged In User
Assumptions	The user is already logged in.
Steps	1 User wants to add a new track to the reference database and clicks 'Add New Track' button. 2 System opens a dialog box and provides the user with two opportunities: 'Import from CSV' button and 'Add manually' button. 3 IF the user presses 'Import from CSV' button THEN 3.1 User imports massively (see Table 1.3.2) ELSE 3.2 User creates a form (see Table 1.3.3)

Table 1.3.1: Use Case Description: Add new track (same case for Add New Record, but clicking on "Add new record")

Use Case	Import massively **uses** add new track
Description	User can perform massive import when creating an entry.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to import massively in order to create an entry.
Steps	REPEAT 1 System opens an import window. 2 User chooses his CSV file to import from his computer and clicks 'Import'. 3 IF the file is in CSV format THEN 3.1 System imports the file to the database along with the timestamp and the username, performing also duplicate elimination and gives the user a message such as: 'Your file has been imported successfully'. ELSE 3.2 System displays a message such as: 'Your file is in invalid format. Please try again'. UNTIL the user successfully uploads a CSV file or the user presses 'Cancel' button and the system takes him back to the previous stage of choosing between 'Import from CSV' and 'Add Manually' buttons

Table 1.3.2- Use Case Description: Import Massively

Use Case	Create form **uses** add new track
Description	User can create a form in order to further create an entry.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to fill a form in order to create an entry.
Steps	REPEAT 1 The system opens a form to create a new entry. 2 The user fills in the required fields in the form and presses 'Done' 3 The system verifies the information in the form. 4 IF the information consistency in the form is validated by the system THEN 4.1 The system displays an agreement message and checks if the entry already exists. 4.2 IF the entry already exists THEN 4.2.1 The system notifies the user and cancels the creation of the entry ELSE 4.2.2 The system adds the entry to the database along with the timestamp and username. ELSE 4.3 The system displays the errors in the creation of the entry and lets the user try again. UNTIL the entry is created successfully or the user presses “Cancel” button and the system takes him back to the previous stage of choosing between “Import from CSV” and “Add Manually” buttons

Table 1.3.3- Use Case Description: Create form

Use Case	Edit Record details
Description	User can edit a record from the reference database.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to edit an entry from his owned list or from a shared playlist.
Steps	1 User wants to edit an existing record of the reference database and clicks it. 2 System opens a box containing the record details and allows the user to edit them. 3 REPEAT 3.1 The user edits the record details. 3.2 System checks for information consistency. 3.3 IF the information is valid THEN 3.1.1 Systems displays an agreement message. 3.1.2 System saves the old value of the record for roll back purposes. 3.1.3 System updates the record details in the database. 3.1.4 System adds the timestamp and username to the updated record. ELSE 3.1.5 System displays an error message letting the user try again. UNTIL the record details are inserted correctly or the user chooses to press 'Cancel' button and no changes are made to the record.

Table 1.3.4- Use Case Description: Edit Record (same case for Edit track, replacing 'record' for 'track')

Use Case	Consult history
Description	User can consult the history of updates of the entries from the reference database.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to consult the list of updates to the entries from his owned list or from a shared list.
Steps	1 User clicks the “Consult History” button next to the entry of the reference database for which he wants to see the history. 2 System displays a list of all updates that have been performed to that entry

Table 1.3.5- Use Case Description: Consult history

Use Case	Rollback **uses** Consult history
Description	User can consult the history of updates of the entries from the reference database.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to consult the list of updates to the entries from his owned list or from a shared list.
Steps	1 User clicks an entry in the page. 2 System enables the user to click 'Rollback' button if he wants to restore to the previous state or 'Cancel' button if he doesn't want to do anything. 3 User presses one of the two buttons. 4 IF the user presses 'Cancel' THEN 4.1 No changes are made ELSE 4.2 System restores to the previous state and saves the changes to the DB

Table 1.3.6- Use Case Description: Rollback

1.4. Use cases for registered users: Personal library management

In this section the most important cases of the record manager will be presented, those intended to fulfill the very purpose of the application: personal list managing. This includes record tracking and sharing.

It has been taken as a hypothesis that there is only one owned list, one tracked list, but several playlists. A playlist can be composed by either elements from the tracked list or elements from the owned list. Another thing to take into consideration is the 'Import massively' use case. This is the same use case that appears in Figure 1.3, but although it has been duplicated in Figure 1.4, it must be interpreted as a single use case. This duplication has been done as a result of having two separate groups of uses cases as opposed to a single diagram combining 'Personal Library Management' and 'Database Management'.

In the below tables the description of these use cases will be presented, rendering the corresponding work-flow

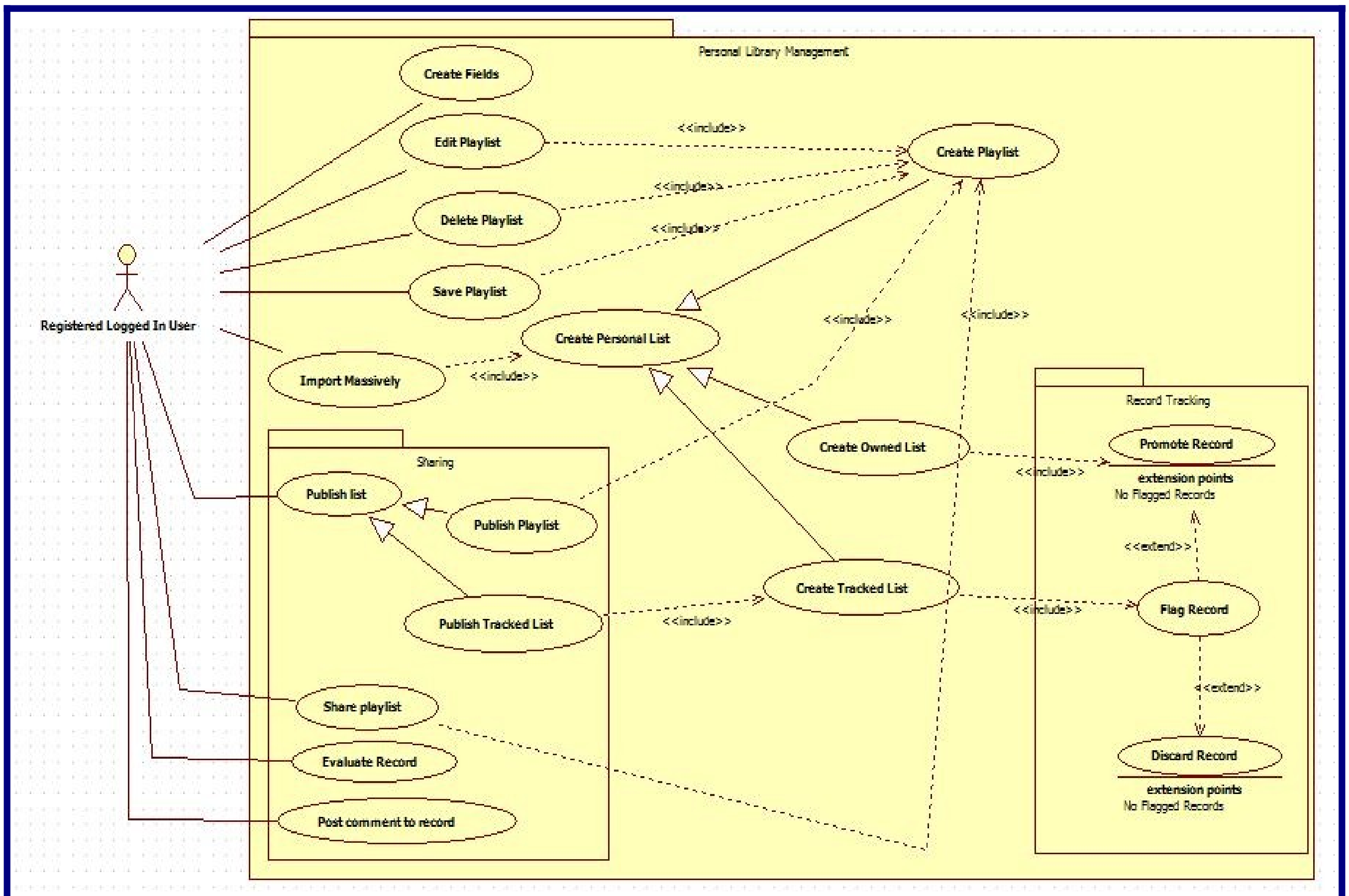


Figure 1.4- Use Case Diagram: Personal library management

Use Case	Edit playlist **uses** create playlist
Description	User can edit a playlist.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to edit his playlist. A playlist must be created first.
Steps	1 User wants to edit an existing playlist of the reference database and clicks on it. 2 System opens a box containing the playlist details and allows the user to edit them. 3 REPEAT 1.1 The user edits the playlist details 1.2 System checks for information consistency 1.3 IF the information is valid THEN 1.3.1 Systems display an agreement message ELSE 1.3.2 System displays an error message letting the user try again UNTIL the entry details are inserted correctly or the user chooses to press 'Cancel' button and no changes are made to the entry or the user has finished his edit.

Table 1.4.1- Use Case Description: Edit playlist

Use Case	Create fields
Description	User must be able to create fields to describe owned records into the reference database.
Actors	Registered Logged In User
Assumptions	User is within the web application
Steps	1 User clicks the 'Consult History' button next to the entry of the reference database for which he wants to see the history. 2 System displays a list of all updates that have been performed to that entry. 3 User clicks 'Add Description' button. 4 System opens a text field and allows the user to write in. 5 After the user finished his description, he has two possibilities: to save what he wrote or to delete it. 6 IF the user presses the 'Save' button THEN 6.1 System saves the user description and gives the user a message such as: 'Your description has been saved' ELSE (if the user presses 'Delete' button) 6.2 The system is returning to user a message window alert: 'Do you want to delete this description?' and it is providing to user two buttons: 'Yes' or 'No' 7 IF the user presses the 'Yes' button THEN 7.1 The system is going to empty the text field. ELSE (if the user presses 'No' button) 7.2 The system is going to close the message window alert and allow to user to continue his description.

Table 1.4.2- Use Case Description: Create fields

Use Case	Delete playlist **uses** create playlist
Description	User can delete a playlist from the library.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to edit his playlist. A playlist must be created first.
Steps	1 User clicks the 'Delete' button next to the playlist. 2 System displays a message: 'Are you sure?' having 2 options: 'Yes' and 'No'. 3 The user presses the button corresponding to his choice. 4 IF the user presses 'yes' THEN 4.1 The system deletes the play list from the library. ELSE 4.2 The system does not delete it.

Table 1.4.3- Use Case Description: Delete playlist

Use Case	Create playlist
Description	User can create a playlist or several playlists for an upcoming event.
Actors	Registered Logged In User
Assumptions	User is within the web application
Steps	1 The user chooses to create a play list from the left said menu of the page interface. 2 The user clicks on 'Create New Playlist' option 3 The system is going to open a new window called 'Create New Playlist', where the user can put the name of the playlist in the field 'Name' and he can add a description in the field called "Description". After this the user can press the button: 'Close' or 'Create'. 4 IF the user presses the 'Close' button THEN 4.1 The system closes the 'Create New Playlist' window ELSE 4.2 The user fills in the fields and clicks 'create' 5 System closes the 'Create New Playlist' window and the new playlist name appears in Playlist from the 'My Music Library' but as being empty (no songs in it). He can add songs by manual search or by massive import. 6 IF the user chooses to create a playlist by using 'Search' (see Table 1.2.4) THEN 6.1 REPEAT 6.1.1 The system returns all the songs, artists and albums of the specific search. 6.1.2 The user adds songs to his playlist, by selecting it and clicking on the '+' button next to the it. 6.1.3 The system will display a pop down menu, named 'Add To', and the user has the possibility to add the song to one of his created playlists. UNTIL the user doesn't want to add any more songs. ELSE 6.2 The user selects massive import by selecting a playlist and clicking 'massive import' (see Table 1.3.2)

Table 1.4.4- Use Case Diagram: Create playlist

Use Case	Save playlist **uses** create playlist
Description	User can save a playlist.
Actors	Registered Logged In User
Assumptions	The user is already logged in and wants to edit his playlist. A playlist must be created alaredy.
Steps	1 User clicks the 'save' button next to the playlist. 2 System displays a message: 'Are you sure?' having 2 options: 'Yes' and 'No'. 3 The user presses the button corresponding to his choice. 4 IF the user presses 'yes' THEN 4.1 The system saves the playlist from the library. ELSE 4.2 The system does not save it.

Table 1.4.5- Use Case Description: Save playlist

Use Case	Create owned list **uses** promote record
Description	User can create a single owned list
Actors	Registered Logged In User
Assumptions	The user is already logged in. There must be a tracked list if the user wants to create an owned list by promoting tracked records of the tracked list.
Steps	1 User clicks on Owned List option from the 'My Music Library' 2 The system opens a new 'Owned List' section. 3 The user may promote his tracked records from the tracked list, by clicking on the button 'Promote' or the user can use 'Import massively' button to import his favorite records; 4 IF the user presses the 'Promote' button THEN 4.1 REPEAT 4.1.1 User Promotes a record (see Table 1.4.7) UNTIL the user has promoted enough records. ELSE User imports massively (see Table 1.3.2)

Table 1.4.6: Use Case Description: Create owned list

Use Case	Promote Record
Description	User can add a record to his owned list by promoting from his tracked list
Actors	Registered Logged In User
Assumptions	There are records in the tracked list
Steps	1 The user may promote flagged record from the tracked list, by clicking on the button 'Promote' 2 The system opens a list of tracked records which has a '+' button close to it 3 The user presses the '+' button to add the selected records to his owned list.

Table 1.4.7: Use Case Description: Promote record

Use Case	Create tracked list **uses** flag record
Description	User can create a single tracked list
Actors	Registered Logged In User
Assumptions	User is within the web application
Steps	1 User clicks on Traked List option from the 'My Music Library' 2 The system opens a new 'Tracked List' section. 3 The user may create his tracked list, by clicking on the button “Flag” or the user can use “Import from CSV” button to import his favorite records. 4 IF the user presses the 'Flag' button THEN 4.1 The system process the flagged records as being tracked ones ELSE User imports massively (see Table 1.3.2)

Table 1.4.8- Use Case Description: Create tracked list

Use Case	Flag Record **extends** promotion
Description	User can add a record to his tracked list by flagging it Deals with the assumption that there are records in the tracked list
Actors	Registered Logged In User
Assumptions	User is within the web application
Steps	1 The user clicks the "Flag' button next to the record 2 The system is considering that record as a tracked record.

Table 1.4.9- Use Case Description: Flag record

Use Case	Discard
Description	The user can discard a record from his tracked list
Actors	Registered Logged In User
Assumptions	User is within the web application
Steps	1 The user can press the 'X' button in order to delete a record from the tracked list 2 The system deletes the tracked record from the tracked list.

Table 1.4.10: Use Case Description: Discard

Use Case	Share list
Description	User can share his playlist with other registered users for co-working purposes.
Actors	Registered Logged in user
Assumptions	The user already has a playlist that he wants to share. The user is already logged in.
Steps	1 User presses the 'Add people to share with' button to share his playlist with other registered users. 2 System provides the user with the opportunity of choosing who to share the playlist with by displaying an empty box in which the user types the username of the person he wants to share the playlist with. 3 REPEAT 3.1 The user types the username of the person he wants to share the playlist with and clicks 'Add' button. 3.2 The system checks the existence of the desired username and displays a verification sign. 3.3 IF the username exists THEN 3.3.1 the system displays the check sign and allows the user to continue adding people by displaying another empty box. ELSE 3.3.2 the system notifies the user that the desired username does not exist and lets him try again in the same box. UNTIL the user doesn’t have any other persons he wants to add to the sharing group. 4 The user presses 'Done' button. 5 The system makes the user's playlist available to the indicated users and sends a notification to each of them regarding these aspects.

Table 1.4.11- Use Case Description: Share list

Use Case	Publish
Description	User can publish an owned list to the community of registered users.
Actors	Registered Logged In User
Assumptions	The user already has a playlist/tracked list list in order to further publish it. User is already logged in and is currently viewing the page containing his owned list.
Steps	1 User presses 'Publish' button to publish his list. 2 System adds the list of the user to the community of registered users.
Non-functional	Performance Mean: the time to update in the database the list of owned lists should be less than 5 seconds.

Table 1.4.12- Use case Description: Publish

Use Case	Evaluate record
Description	User can evaluate records in a list.
Actors	Registered Logged in user
Assumptions	The user is already logged in.
Steps	1 User clicks one of the 5 stars available next to each record in order to evaluate it. 2 System records the evaluation given by the user and, if necessary, updates the percentage displayed on the page.
Non-functional	The user can see only his choice of rating the record and the percentage computed by the system as a result of analyzing all users' choices.

Table 1.4.13- Use Case Description: Evaluate record

Use Case	Post comment to record
Description	User can post comments on records.
Actors	Registered Logged in user
Assumptions	The user is already logged in and is currently in the page displaying the shared playlist.
Steps	1 User clicks a track belonging to the record he wants to post a comment to. 2 System opens a box right next to the track containing the name of the record the track belongs to and 2 buttons: "Add Comment to Record" and "See other comments". 3 IF user presses "Add Comment to Record button" THEN 3.1 System opens an empty box allowing the user to add a comment in it. 3.2 User types the comment and presses "Done" button. 3.3 System adds the comment to the database and to the list of comments to that record. ELSE (if the user presses "See other comments" button) 3.4 System displays a list of all comments posted to that record.

Table 1.4.14- Use Case Description: Post comment

1.5. Use cases for registered users: User administration

In this section the uses cases for user administration will be illustrated and explained. It is relevant to point out that a special situation is considered in Figure 1.5, where 2 users can are interacting with the system: a registered and logged in user, and a registered but not logged in user. The first can execute certain administrative tasks, while the latter can only log in or reset his password.

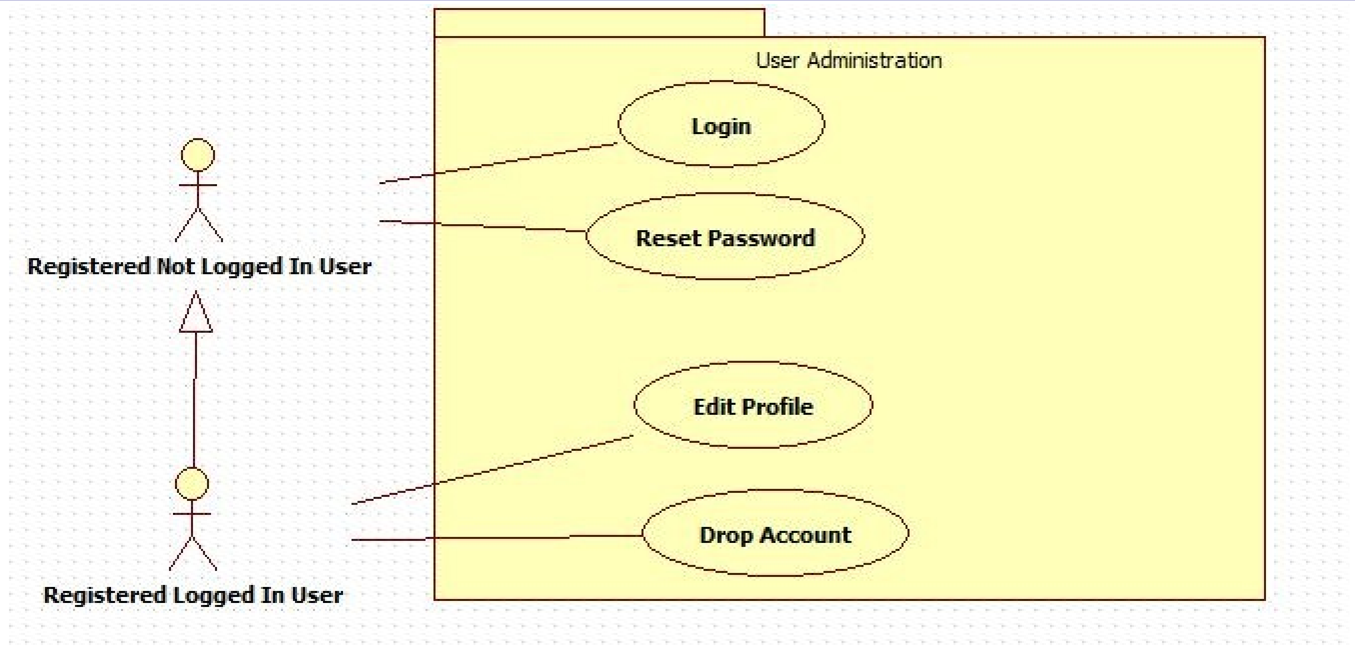


Figure 1.5- Use Case Diagram: User administration

The use cases from Figure 1.5 will be presented in the below tables, rendering the corresponding work-flow for each of them.

Use Case	Log in
Description	User goes through authentication to log in to the Vynil record manager
Actors	Registered but not logged in user
Assumptions	-The user is already registered -The user knows his password
Steps	1 System prompts user to log in 2 REPEAT 2.1 User enters his user-name 2.2 User enters his password 2.3 User presses log in 2.4 System verifies the log in information UNTIL username and password is correct 3 System logs in the user in the application
Variations	#1. The user may click log in or he may attempt other tasks that require log in (DB management, Personal library management, Record tracking, Sharing, etc)

Table 1.5.1- Use Case Description: Log in

Use Case	Drop Account
Description	User drops his account
Actors	Registered and logged in user
Assumptions	User is within the web application
Steps	1 User clicks profile button 2 System displays user's profile information. 3 User clicks 'Drop Account' button 4 IF user presses 'Yes, drop my account' button. THEN 4.1 System deletes the profile of the user along with all the data it contains. ELSE 4.2 No changes are made to the user’s account.

Table 1.5.2- Use Case Description: Drop account

Use Case	Reset password
Description	How to reset password in case user forgets it or if he wants to change it
Actors	Registered but not logged in user
Assumptions	-The user is already registered
Steps	1 The clicks on reset password within the log in screen 2 System informs that the will receive a follow up URL in his personal email (the one used for registration) and asks for user's username. 3 User types his username and presses send email button 4 System sends a follow-up link to the user's email inbox. 5 User clicks the follow up link 6 System prompts user for new password and new password re typing 7 REPEAT 7.1 User enters his password 7.2 System checks password format and displays agreement sign. UNTIL password meets requirements 8 REPEAT 8.1 User re enters password 8.2 System checks if re-entered password matches initial password and displays agreement sign . UNTIL re-entered password matches initial password 9 User presses submit 10 System adds the information to DB
Non-functional	Follow-up link expiration: Time for user to complete registration should be less than 24 hours.

Table 1.5.3- Use Case Description: Reset Password

Use Case	Edit Profile
Description	User changes information within his profile
Actors	Registered and logged in user
Assumptions	User is within the web application
Steps	1 User clicks profile button 2 System displays user's profile information 3 User clicks edit. 4 REPEAT 4.1 User clicks on a specific profile field 4.2 System enables editing of that field 4.3 User changes the information within the field UNTIL username and password is correct 5 User clicks save changes 6 System updates the profile information and the DB

Table 1.5.4- Use Case Description: Edit profile

2. Database Design

In this section, the database model for the Vinyl Record Manager application will be presented, including the necessary Entity Relationship Diagrams and a detail description for each of the tables.

The online Record Database will need to have a persistent data storage that can be used to store, and update the Record Database as well as store information about users, their collections and their activity in general. In order to achieve that, an adequate RDBMS is needed to store the information as well as the state of the system. The proposed structure of the database that will be used to store and maintain information is shown on Figure 2.1, in page 19 . And the tables contained in this figure are described in the subsequent sections.

2.1. Main Tables

Soundtrack
This table is the basic building block of the database and refers to a single track or song in the system. Soundtracks table will be used to store all the information about a single track along with foreign keys for different tables like genre and a self reference in case it has an original version.

Record
This table is going to store the basic information about a collection of tracks including the title, the matrix number which uniquely identifies the record, its size and so on.

User
This table is used to store the information of the users of the system. It will contain the username, authentication details like the email, hashed password, and language preferences etc. This is an integral table which will be used by the authentication system as well as used across the site to represent users.

Playlist
This table is used to store information about different playlists a user makes. It only contains the common information about a playlist like its name, creator, whether it is publicly shared or not and also whether its shared with others for editing.

Artist
This table is used to store different artists. A reference for this table is kept in the TrackArtists table for a list of 'invited' artists for a given track as well as the Records table. This table will be maintained to avoid duplicate artists as well as suggestions for users who are adding/editing information about Records or Sountracks.

MusicPlayer
The MusicPlayers table stores a list of music players. This is a list of all the music players a soundtrack can be played on. A reference to records in this table is used in the SoundtrackPlayers table.

Genre
This table is used to store a list of possible Genres a particular Soundtrack or Record can have. Unique keys in this table are referenced in the respective tables to obtain the genres.

2.2. Secondary Tables

This section describes the tables which are used to store information like lists of information - primarily concerning with 1-to-many or many-to-many relationships between some of the entities mentioned above.

CustomAttribute
This table is used to store the custom attributes users want to associate with the records that are in their library. This table is referenced by the record id and the user id to obtain a list of all the custom attributes a user has added for a particular record.

RecordTrack
RecordTracks table contains, for each record the tracks that are part of it. This table will have a record id as a reference to the records table and a track id as a reference to the tracks table and has been created to normalize the many-to-many relationship between Records and Tracks table. RecordTracks will also contain some relationship variables like the order and the disk number.

TrackArtist
Since there can be any number of artists in a track, this information will be stored in a separate table. TrackArtists will contain a foreign key to the artists table as well as reference to the soundtracks table. This table is needed since there can be multiple artists performing a particular track.

PlaylistItem
This table keeps the details about different songs that have been added to a playlist. This table also tracks the user who added the item in case the playlist is shared. This table is used in order to normalize the 1-to-many relationship between a playlist and the Tracks present in it. Each row will uniquely identify a playlist and a Soundtrack - thus providing information about the tracks present in a playlist.

RecordLibrary
This entity keeps track of users' personal library. It is used to maintain a list of owned and tracked lists of all the users as well as information about whether it is published or not. Details about the records in these lists are stored separately as foreign keys for the records table and the record library entry in RecordLibraryItems entity.

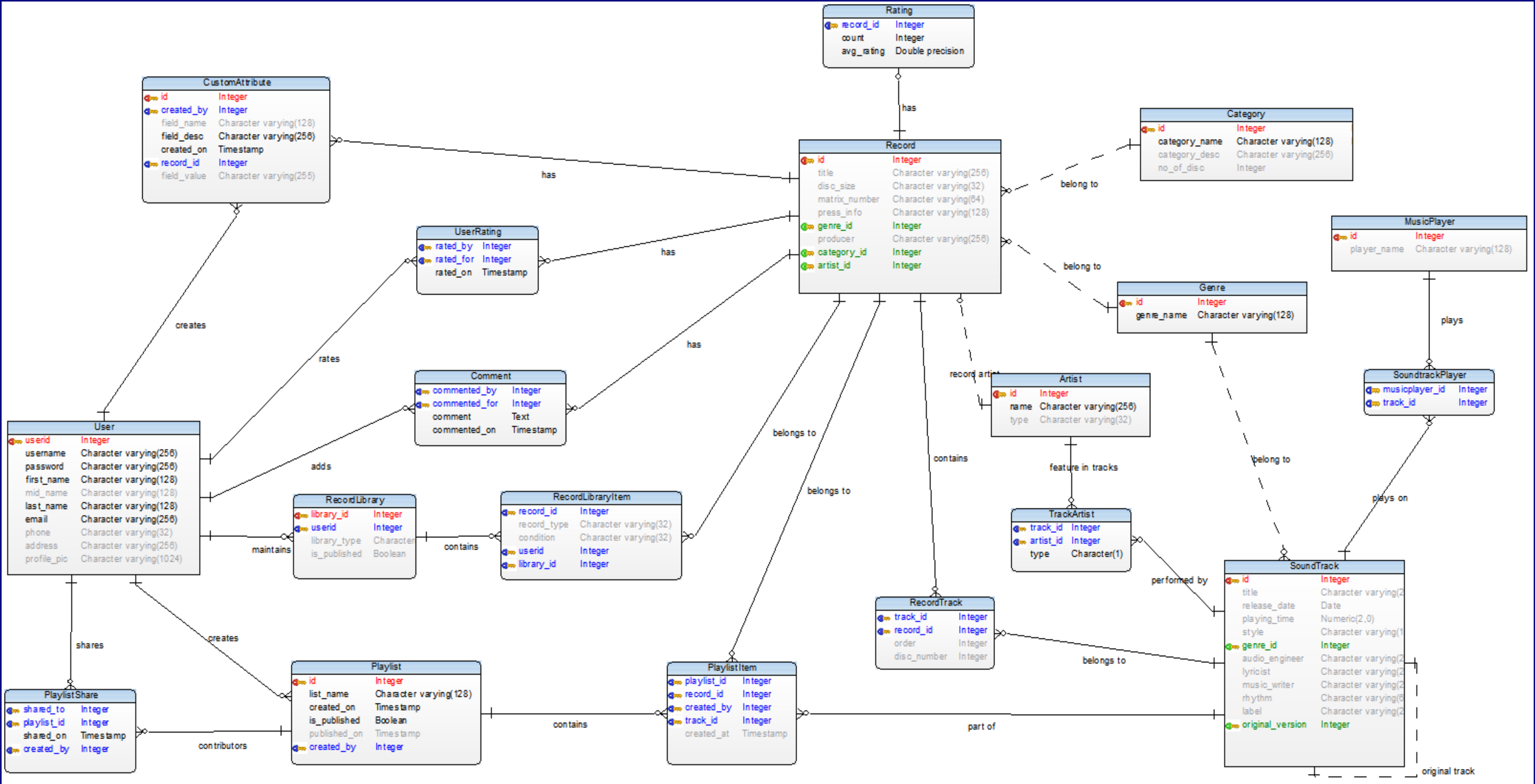


Figure 2.1- Entity Relationship Diagram for Vinyl Record Manager

PlaylistShare
This table has 1-to-many relationship with the Playlists table. It is used to store the users a particular playlist is shared with. This will be essential to check permissions for viewing/editing playlists. Though not required at this stage, it will be possible to extend this table to incorporate permissions for individual users through this table. By individual permissions, we mean that we will be able to specify the access level of any given user.

SoundtrackPlayer
Since a Track can be played on multiple MusicPlayers, the information about what music players a particular track can be played on needs to be stored. In order to efficiently resolve the 1-to-many relationship between Soundtracks and MusicPlayers, the SoundtrackPlayers table will be maintained which is used to maintain a list of all the music players that a particular song can be played on.

2.3. *Managing User Ratings and Comments:*

In order to maintain user ratings and to make it work in an efficient manner, we have sub-divided the ratings process into two tables.

Rating
This table stores a reference to a particular Record as well as a count an average rating of a particular record. This is done in order to avoid the need to calculate the average and count of the votes each time this information is required. Each row in the ratings table will give information about the overall rating of a Record at any given time, along with the total number of votes and the average rating of the record.

UserRating
This table is used to store an 'individuals' rating for a particular record. This table stores a reference to the record id, the user id as well as the rating the user chose to give for a particular record. This is necessary in order to store what rating a particular user has given to the record. If a user wishes to change their opinion, their existing entry in the database will be changed to reflect the change.

Comment
Since the users are able to post comments on records, and there is no restriction on the number of comments posted on any record, it is best to store comments in a separate table. This table will store a reference to id of the record on which the comment was posted, a reference to the user who posted the comment, a timestamp and the text of the comment itself.

2.4. *Revision Management*

Keeping track of changes made to Records is one of the requirements. In order to achieve this, three new tables were created to store the information efficiently. The revision management was subdivided into two sub-parts for ease of implementation. This will in no way have an impact on the user experience. Any changes made to information about the record will be stored separately while the list of soundtracks that are in a record at any given time are stored separately. For implementation purposes, a change to the record related information will be treated separately from changes to the list of soundtracks associated with a record. This means that a 'revision' can either contain changes to the record information like its title, artist, etc. or a change to the list of soundtracks it contains.

This scheme has been chosen to maintain revisions because it allows to find the difference between two successive or any selected revisions easily. An alternate approach was considered where only the changes that are made during any particular revision could be stored. While this approach is more concise in terms of storing minimal information - it is, however, potentially problematic in terms of performance and scalability. The problem arises when one needs to find out the exact state of a Record at any given revision. The corresponding entity relationship diagram is displayed on Figure 2.2.

Revision

This table will contain information about each revision in order to identify further information about it. It will contain the id of the user, the record, a revision number for that particular record and the timestamp.

RecordRevision

This table will contain a reference to a particular revision from the revisions table along with all the fields of Records table that are editable.

RecordTracksHistory

This table will contain multiple all the revisions of list of tracks in a record. It will have a reference of a record from the revisions table and will have a list of soundtracks that were present in a record at a given time.

There are also revision tables for soundtracks, soundtrack artists and music players for soundtracks designed in a similar way.

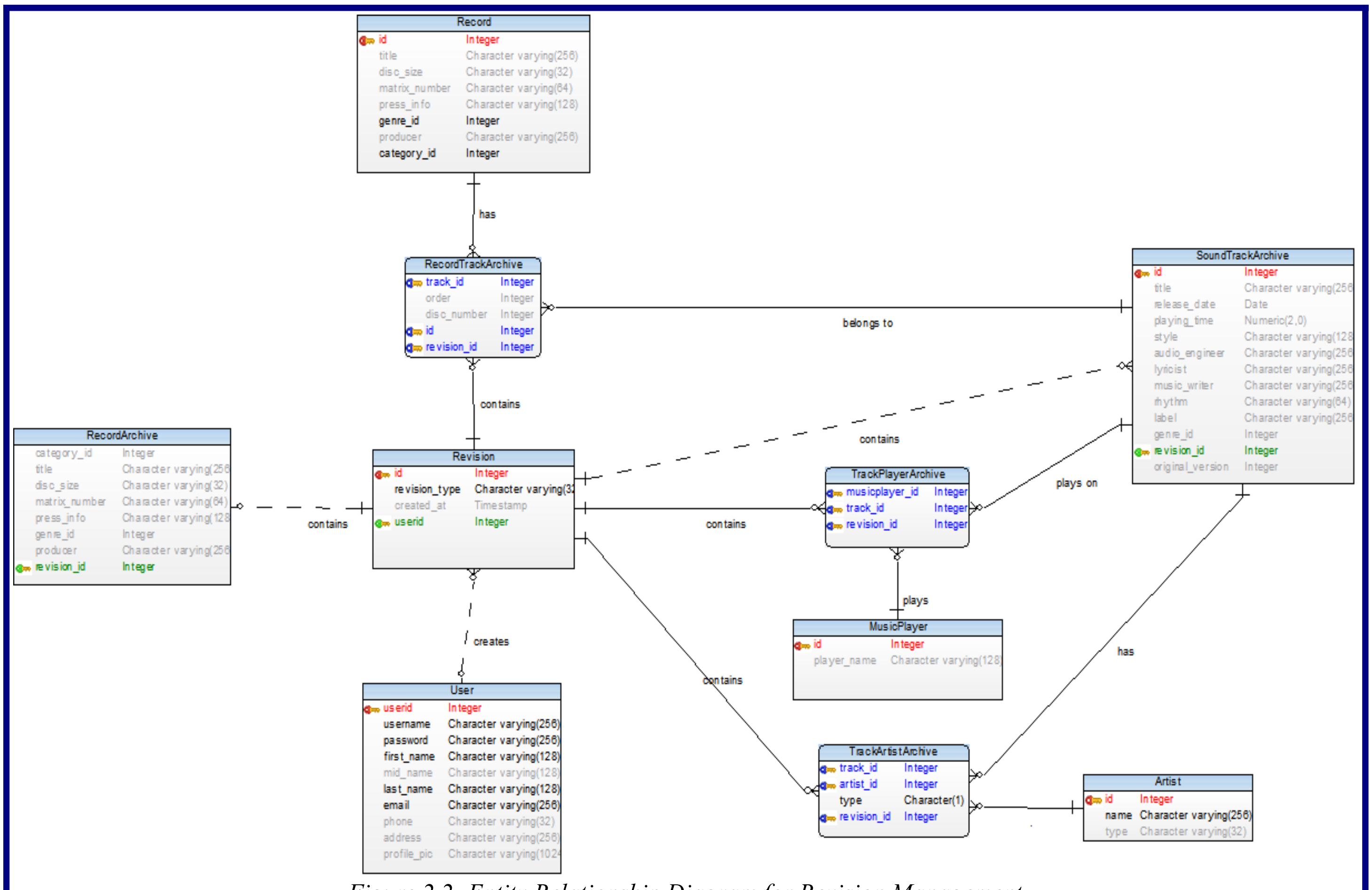

Record
id Integer
title Character varying(256)
disc_size Character varying(32)
matrix_number Character varying(64)
press_info Character varying(128)
genre_id Integer
producer Character varying(256)
category_id Integer
has
RecordTrackArchive
track_id Integer
order Integer
disc_number Integer
id Integer
revision_id Integer
belongs to
SoundTrackArchive
id Integer
title Character varying(256)
release_date Date
playing_time Numeric(2,0)
style Character varying(128)
audio_engineer Character varying(256)
lyricist Character varying(256)
music_writer Character varying(256)
rhythm Character varying(64)
label Character varying(256)
genre_id Integer
revision_id Integer
original_version Integer
contains
RecordArchive
category_id Integer
title Character varying(256)
disc_size Character varying(32)
matrix_number Character varying(64)
press_info Character varying(128)
genre_id Integer
producer Character varying(256)
revision_id Integer
contains
Revision
id Integer
revision_type Character varying(32)
created_at Timestamp
userid Integer
contains
TrackPlayerArchive
musicplayer_id Integer
track_id Integer
revision_id Integer
plays on
plays
MusicPlayer
id Integer
player_name Character varying(128)
creates
contains
has
User
userid Integer
username Character varying(256)
password Character varying(256)
first_name Character varying(128)
mid_name Character varying(128)
last_name Character varying(128)
email Character varying(256)
phone Character varying(32)
address Character varying(256)
profile_pic Character varying(1024)
TrackArtistArchive
track_id Integer
artist_id Integer
type Character(1)
revision_id Integer
Artist
id Integer
name Character varying(256)
type Character varying(32)

Figure 2.2- Entity Relationship Diagram for Revision Management

3. Web Design

In this section, the wireframing of the Web application for the Vinyl Record manager will be presented. This will be done by means if illustrations that will intend to portray the basic look of each window/ situation. The green frame in Figure 3.1 is not part of the design. Its inside is a variable area that will show different statuses, while its outside will remain fixed for any status. Refer to this frame when looking at other statuses.

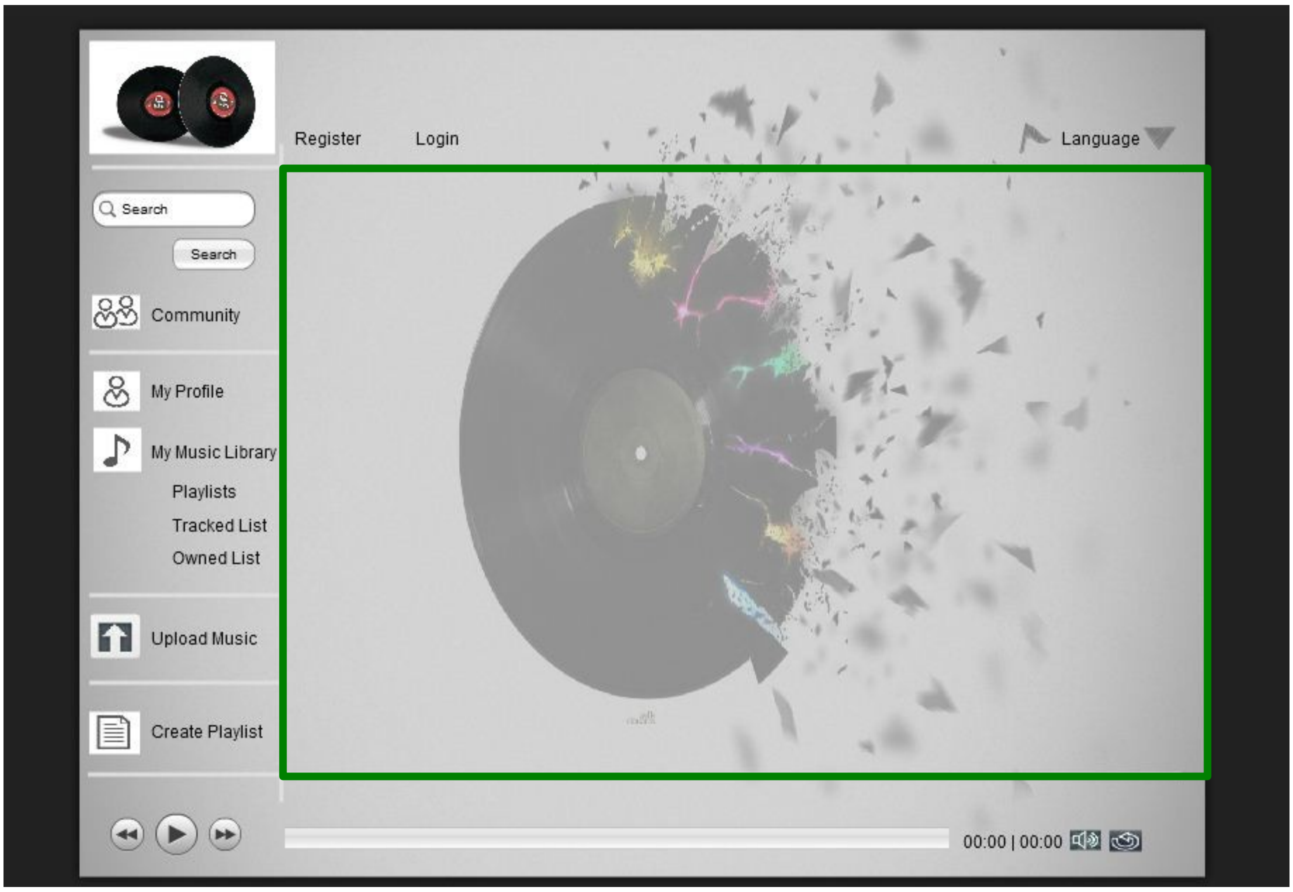


Figure 3.1- Web site look: Main Layout

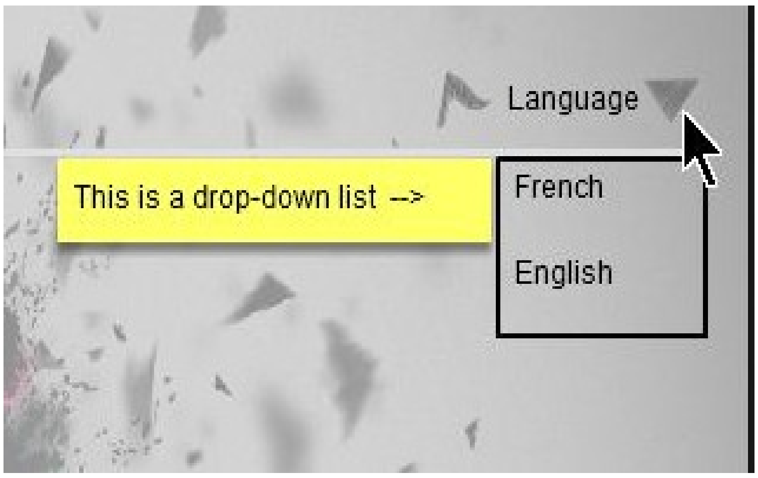


Figure 3.2- Web site look: Change Language (see Figure 3.1 on the upper right)

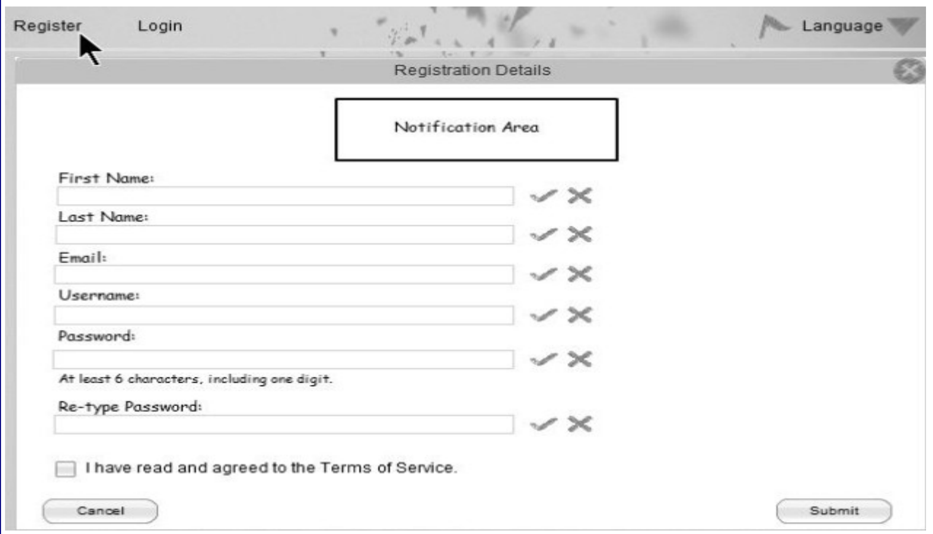


Figure 3.3- Web site view: Registration

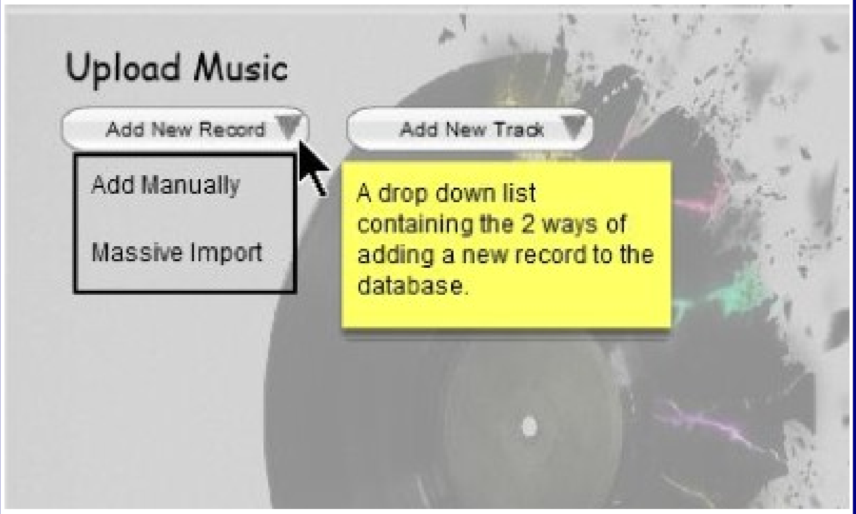

Upload Music
Add New Record
Add New Track
Add Manually
Massive Import
A drop down list containing the 2 ways of adding a new record to the database.

Figure 3.4- Web site view: Add new record (see 'Upload Music' on Figure 3.1)

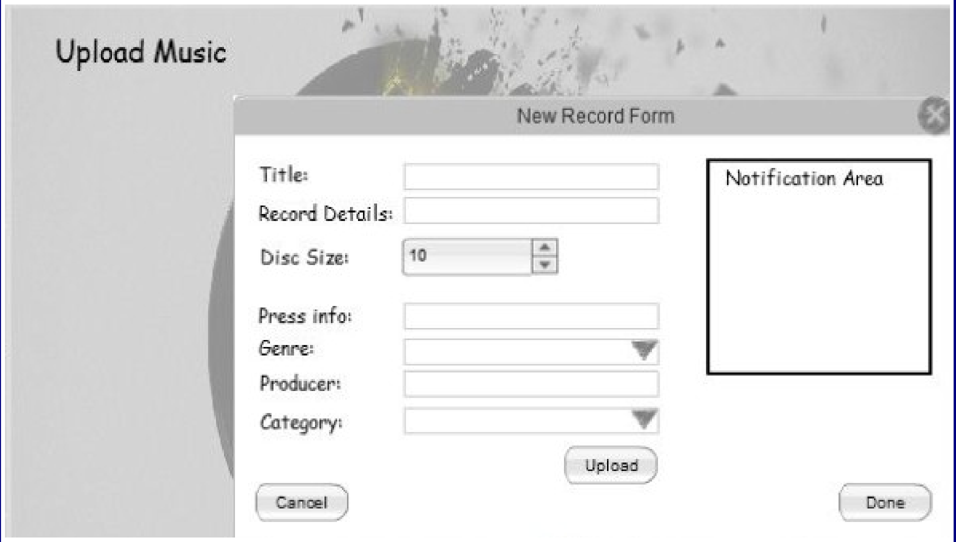


Figure 3.5- Web site view: Add record manually

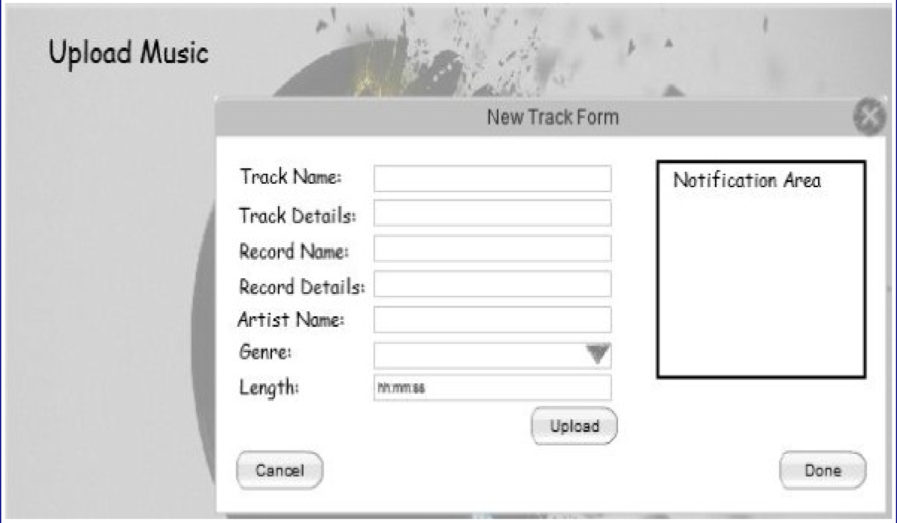


Figure 3.6- Web site view: Add track manually (First, the user would click on Add new track as showed in Figure 3.4, having the same two options)

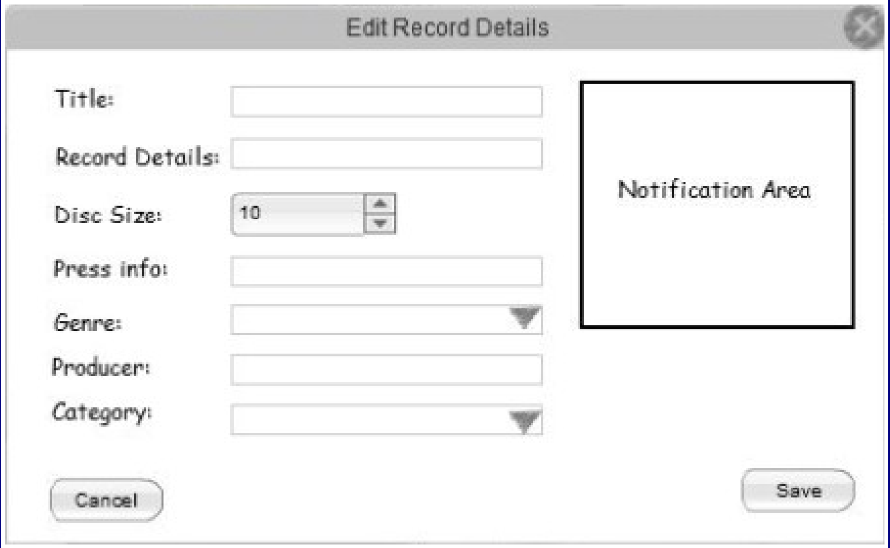


Figure 3.7- Web site view: Edit Record

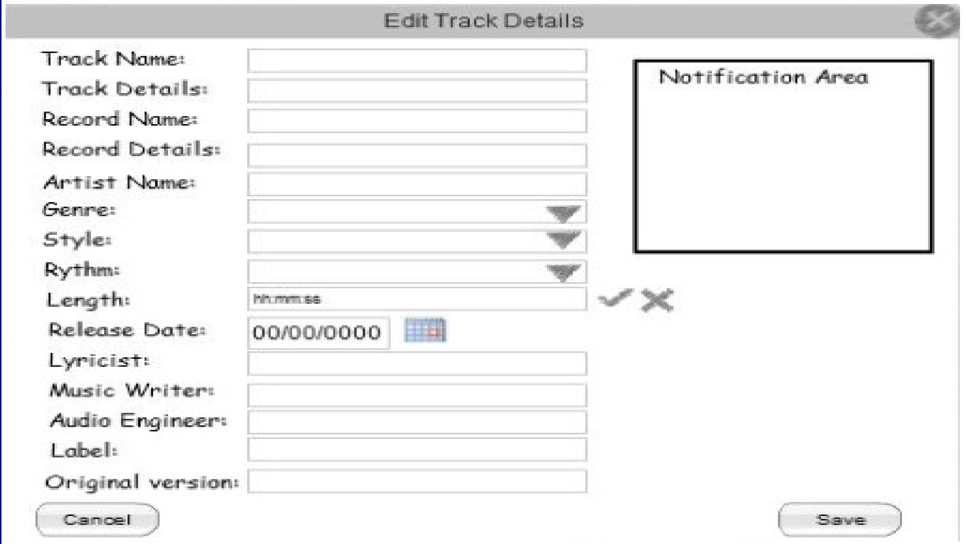


Figure 3.8- Web site view: Edit track

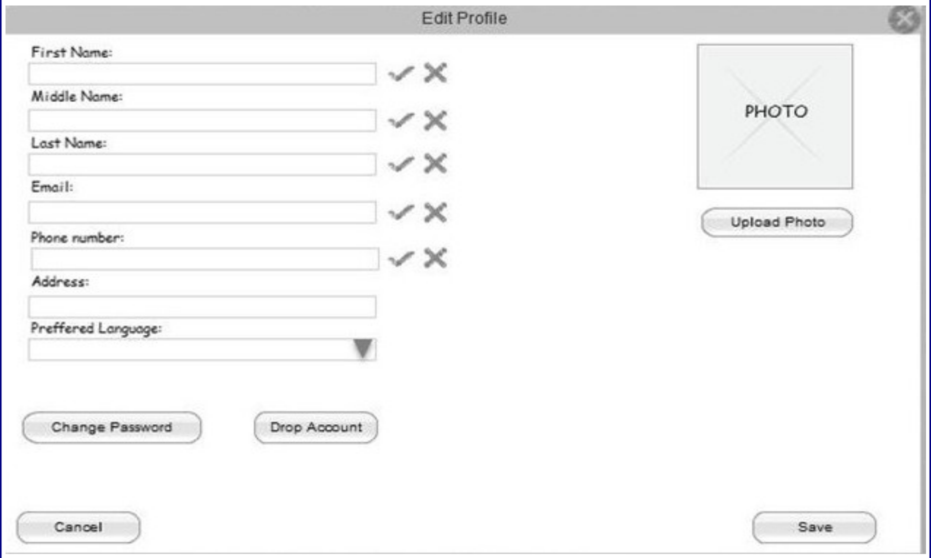


Figure 3.9- Web site view: Edit profile

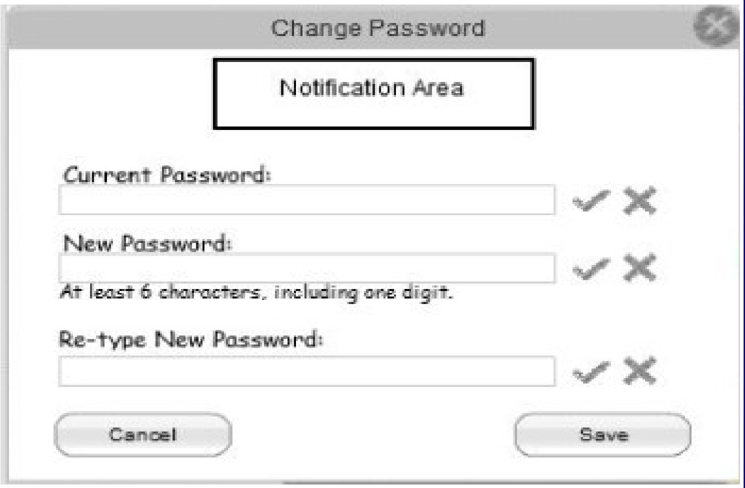
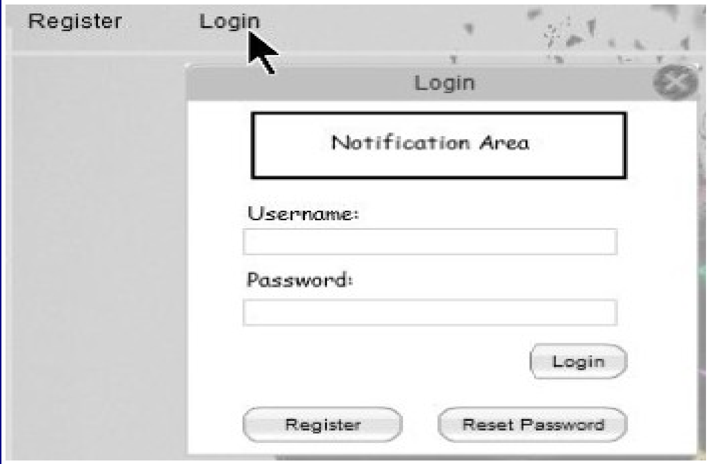


Figure 3.10- Web site view: Change password (see Figure 3.9)


Register Login

Login

Notification Area

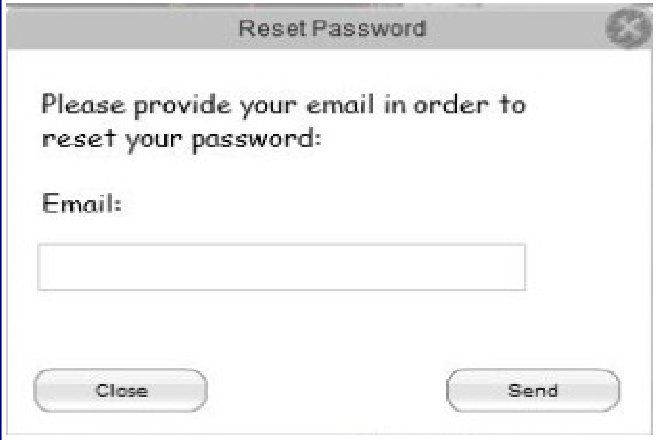
Username:

Password:

Login

Register Reset Password

Figure 3.11- Web site view: Log in (see Figure 3.1)

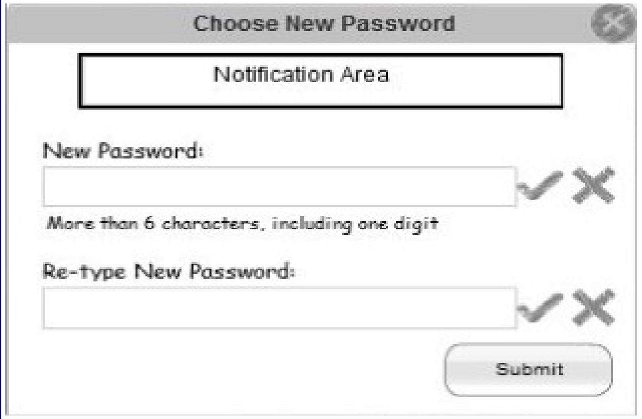

Reset Password

Please provide your email in order to reset your password:

Email:

Close Send

Figure 3.12- Web site view: Reset Password (see Figure 3.11)


Choose New Password

Notification Area

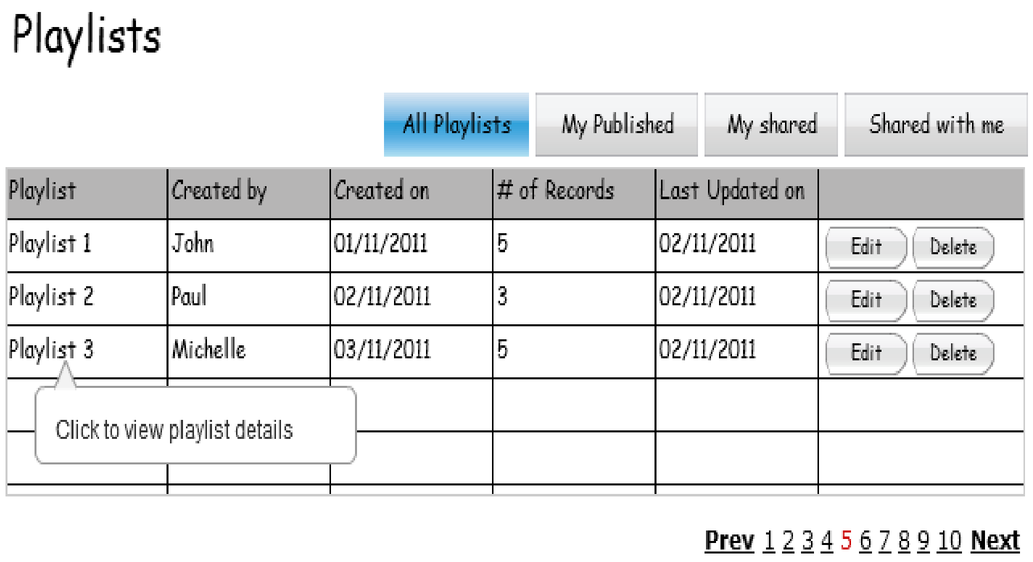
New Password:

More than 6 characters, including one digit

Re-type New Password:

Submit

Figure 3.13- Web site view: Reset password #2 (see Figure 3.12 and Table 1.5.3)


Playlists

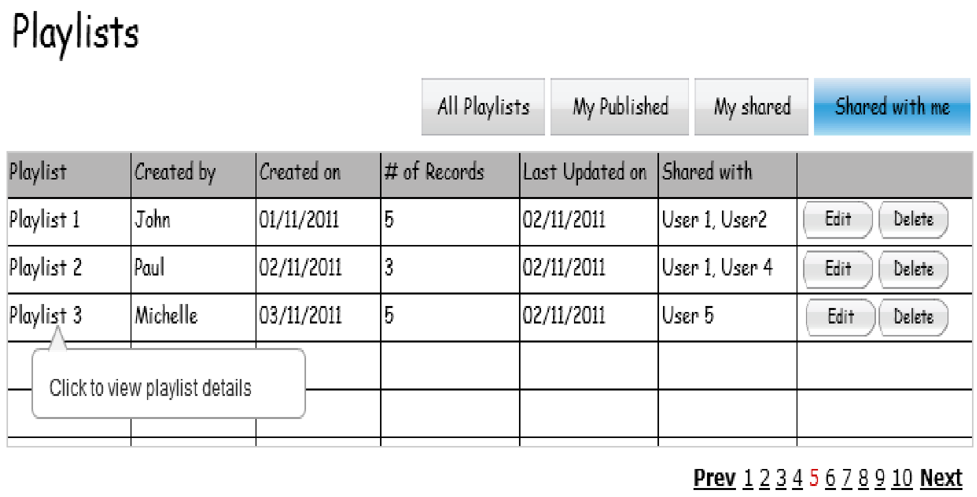
All Playlists | My Published | My shared | Shared with me

Playlist	Created by	Created on	# of Records	Last Updated on	
Playlist 1	John	01/11/2011	5	02/11/2011	Edit Delete
Playlist 2	Paul	02/11/2011	3	02/11/2011	Edit Delete
Playlist 3	Michelle	03/11/2011	5	02/11/2011	Edit Delete

Click to view playlist details

Prev 1 2 3 4 5 6 7 8 9 10 Next

Figure 3.14- Web site view: Playlists (see left column on Figure 3.1)


Playlists

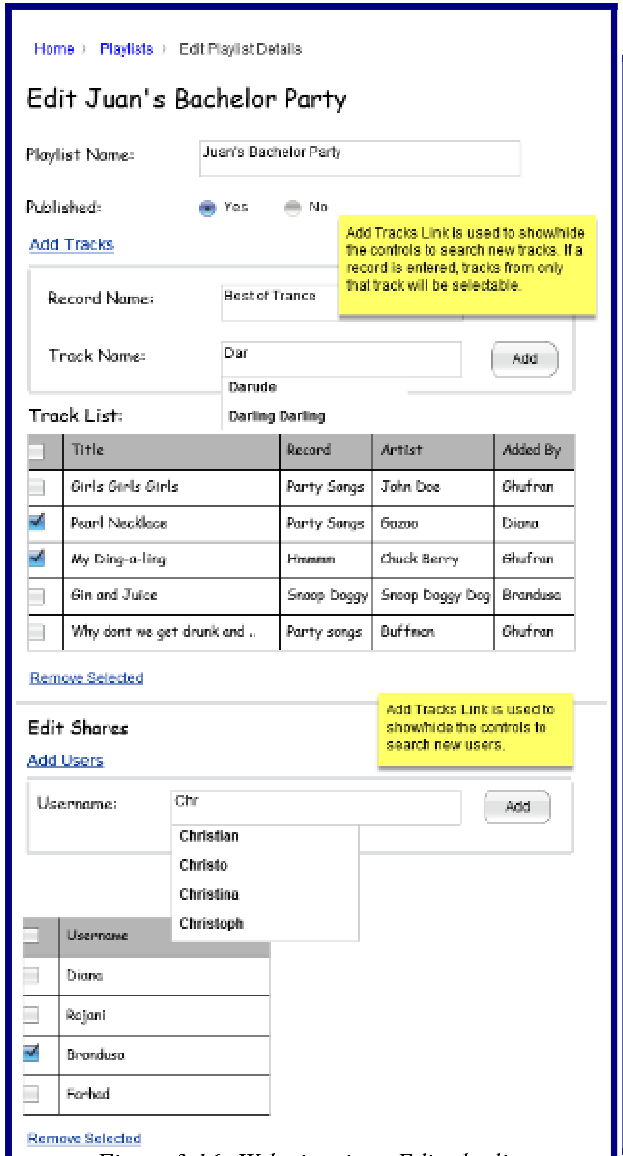
All Playlists | My Published | My shared | Shared with me

Playlist	Created by	Created on	# of Records	Last Updated on	Shared with	
Playlist 1	John	01/11/2011	5	02/11/2011	User 1, User2	Edit Delete
Playlist 2	Paul	02/11/2011	3	02/11/2011	User 1, User 4	Edit Delete
Playlist 3	Michelle	03/11/2011	5	02/11/2011	User 5	Edit Delete

Click to view playlist details

Prev 1 2 3 4 5 6 7 8 9 10 Next

Figure 3.15- Web site view: Playlists shared with me (should the user click published/shared, the view would be the same



Edit Juan's Bachelor Party

Playlist Name:	Juan's Bachelor Party
Published:	[✓ Yes] No

Record Name:	Best of Trance	
Track Name:	Dar	Add

Add Tracks Link is used to show/hide the controls to search new tracks. If a record is entered, tracks from only that track will be selectable.

Track List:

☐	Title	Record	Artist	Added By
☐	Girls Girls Girls	Party Songs	John Doe	Ghufran
✓	Pearl Necklace	Party Songs	Gazoo	Diana
✓	My Ding-a-ling	Hmmmm	Chuck Berry	Ghufran
☐	Gin and Juice	Snoop Doggy	Snoop Doggy Dog	Brandusa
☐	Why dont we get drunk and ..	Party songs	Buffman	Ghufran

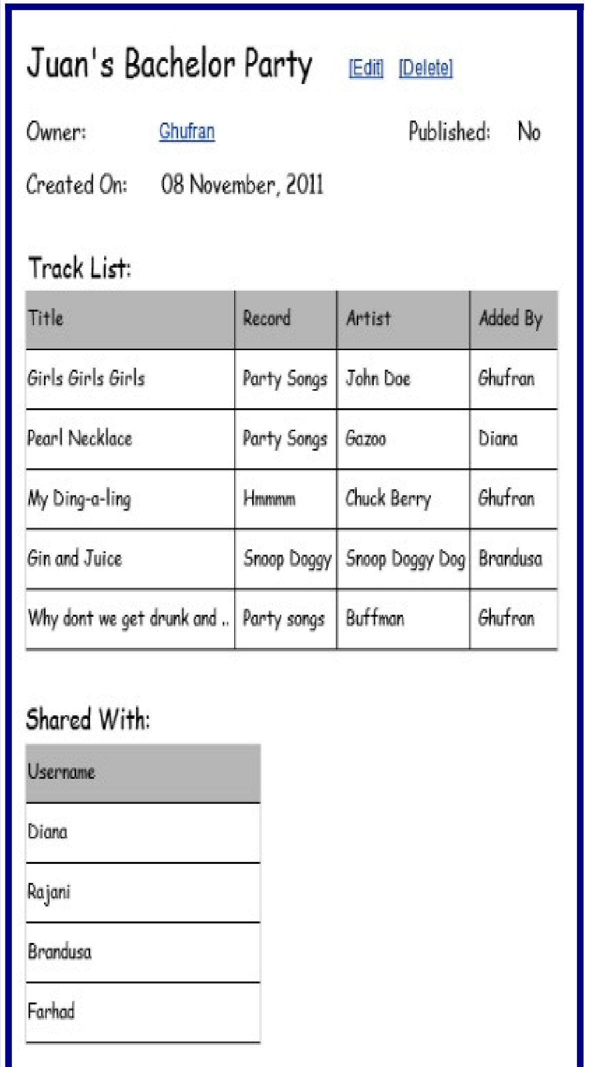
Edit Shares

Add Tracks Link is used to show/hide the controls to search new users.

Username:	Chr	Add

☐	Username
☐	Diana
☐	Rajani
✓	Brandusa
☐	Farhad

Figure 3.16- Web site view: Edit playlist


Juan's Bachelor Party

Owner:	Ghufran	Published:	No
Created On:	08 November, 2011		

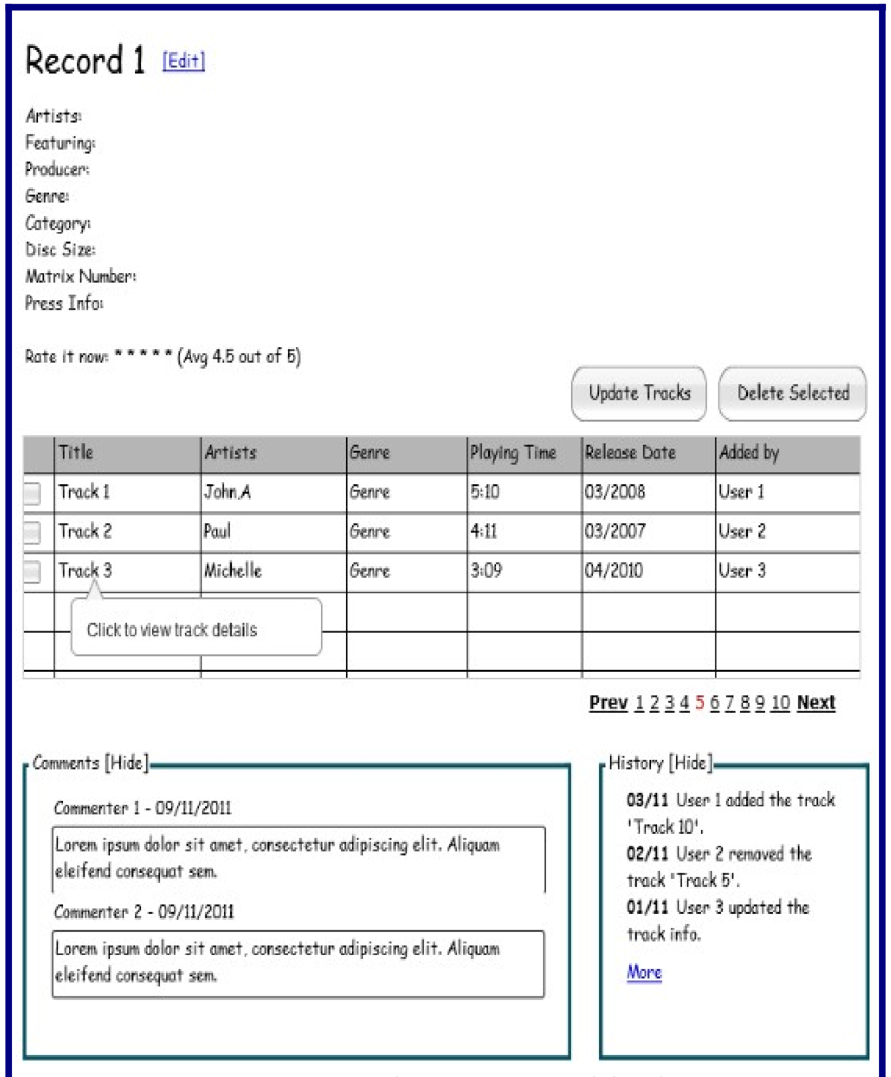
Track List:

Title	Record	Artist	Added By
Girls Girls Girls	Party Songs	John Doe	Ghufran
Pearl Necklace	Party Songs	Gazoo	Diana
My Ding-a-ling	Hmmmm	Chuck Berry	Ghufran
Gin and Juice	Snoop Doggy	Snoop Doggy Dog	Brandusa
Why dont we get drunk and ..	Party songs	Buffman	Ghufran

Shared With:

Username
Diana
Rajani
Brandusa
Farhad

Figure 3.18: Web site view: Playlist details


Record 1

Artists:
Featuring:
Producer:
Genre:
Category:
Disc Size:
Matrix Number:
Press Info:

Rate it now: * * * * * (Avg 4.5 out of 5)

	Title	Artists	Genre	Playing Time	Release Date	Added by
☐	Track 1	John.A	Genre	5:10	03/2008	User 1
☐	Track 2	Paul	Genre	4:11	03/2007	User 2
☐	Track 3	Michelle	Genre	3:09	04/2010	User 3

Comments [Hide]

Commenter 1 - 09/11/2011

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eleifend consequat sem.

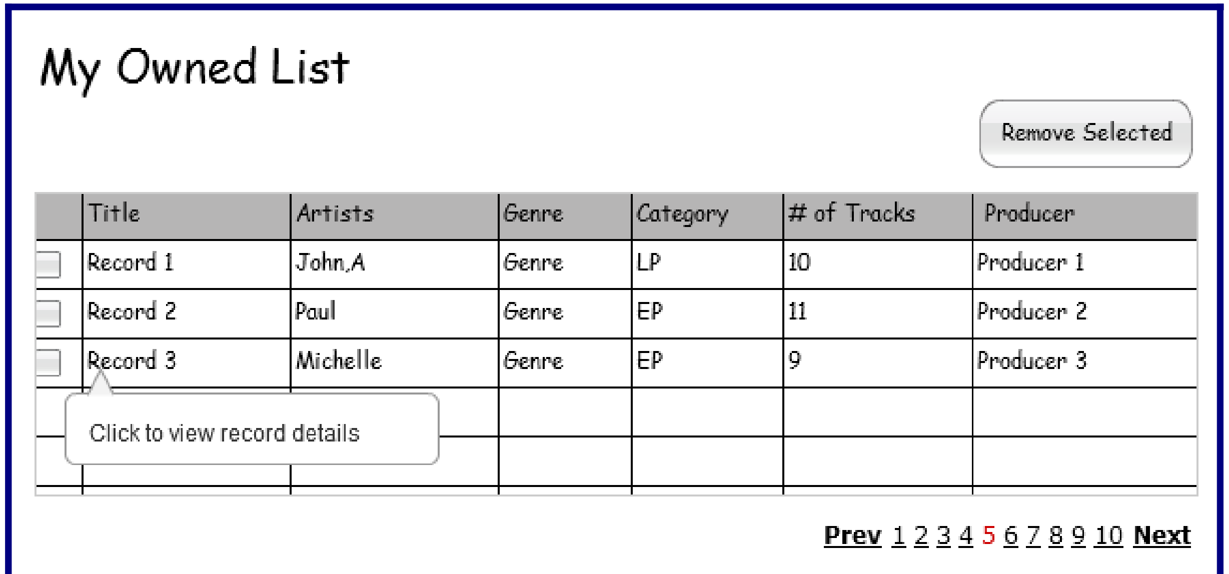
Commenter 2 - 09/11/2011

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eleifend consequat sem.

History [Hide]

03/11 User 1 added the track 'Track 10'.
02/11 User 2 removed the track 'Track 5'.
01/11 User 3 updated the track info.

Figure 3.17: Web site view: Record details



My Owned List

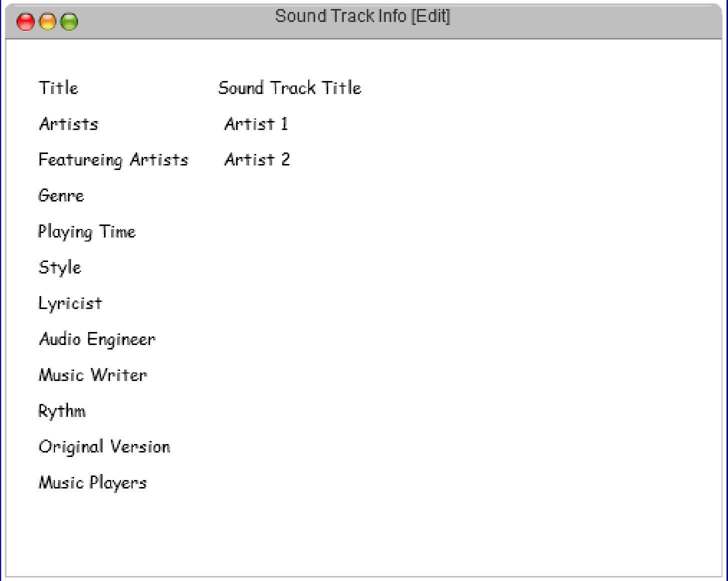
Remove Selected

	Title	Artists	Genre	Category	# of Tracks	Producer
☐	Record 1	John,A	Genre	LP	10	Producer 1
☐	Record 2	Paul	Genre	EP	11	Producer 2
☐	Record 3	Michelle	Genre	EP	9	Producer 3

Click to view record details

Prev 1 2 3 4 5 6 7 8 9 10 Next

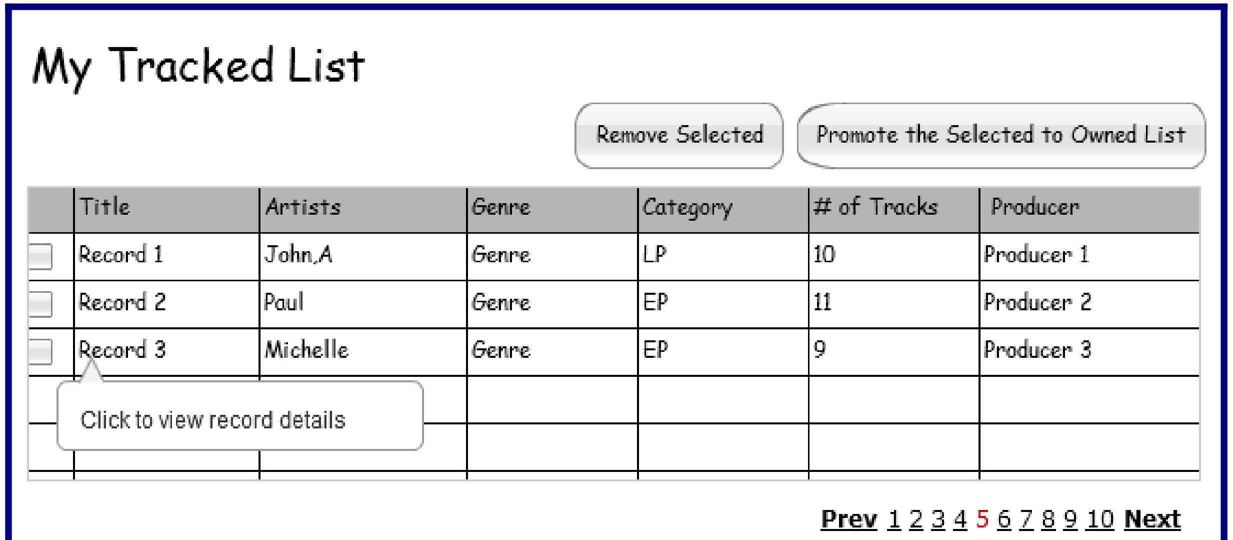
Figure 3.19- Web site view: My owned list (see left column on Figure 3.1)



Sound Track Info [Edit]

Title	Sound Track Title
Artists	Artist 1
Featureing Artists	Artist 2
Genre	
Playing Time	
Style	
Lyricist	
Audio Engineer	
Music Writer	
Rythm	
Original Version	
Music Players	

Figure 3.20- Web site view: Track details (see Figure 3.19 and Figure 3.21)



My Tracked List

Remove Selected

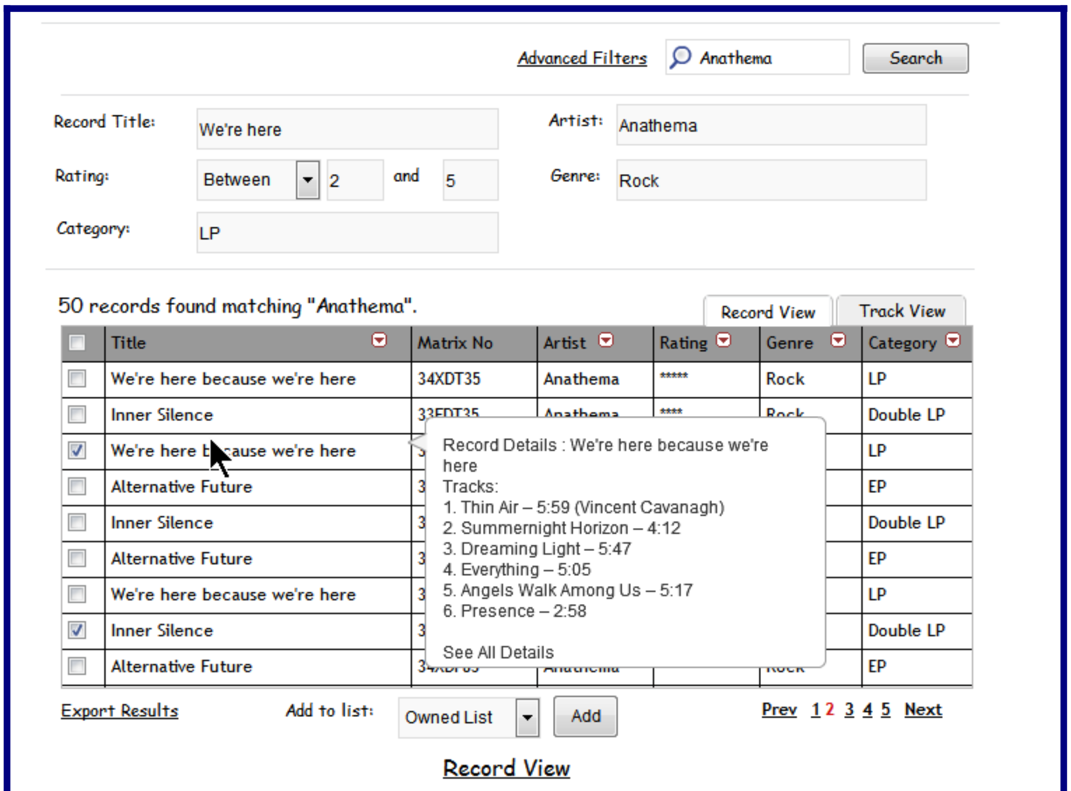
Promote the Selected to Owned List

	Title	Artists	Genre	Category	# of Tracks	Producer
☐	Record 1	John,A	Genre	LP	10	Producer 1
☐	Record 2	Paul	Genre	EP	11	Producer 2
☐	Record 3	Michelle	Genre	EP	9	Producer 3

Click to view record details

Prev 1 2 3 4 5 6 7 8 9 10 Next

Figure 3.21- Web site view: My tracked list



Advanced Filters | Anathema | Search

Record Title:	We're here	Artist:	Anathema
Rating:	Between 2 and 5	Genre:	Rock
Category:	LP		

50 records found matching "Anathema".

Record View | Track View

☐	Title	Matrix No	Artist	Rating	Genre	Category
☐	We're here because we're here	34XDT35	Anathema	*****	Rock	LP
☐	Inner Silence	33FDT35	Anathema	****	Rock	Double LP
✓	We're here because we're here	3				LP
☐	Alternative Future	3				EP
☐	Inner Silence	3				Double LP
☐	Alternative Future	3				EP
☐	We're here because we're here	3				LP
✓	Inner Silence	3				Double LP
☐	Alternative Future	34XDT35	Anathema		Rock	EP

Record Details : We're here because we're here
Tracks:
1. Thin Air – 5:59 (Vincent Cavanagh)
2. Summernight Horizon – 4:12
3. Dreaming Light – 5:47
4. Everything – 5:05
5. Angels Walk Among Us – 5:17
6. Presence – 2:58

See All Details

Export Results | Add to list: Owned List | Add | Prev 1 2 3 4 5 Next

Record View

Figure 3.22- Web site view: Search (For Advanced filtering, see Table 1.2.5 and Table 1.2.6)

4. Project management

4.1. Management of tasks, communication and documents

Since the team for the project is not in the same location, collaborating efficiently is imperative to success. We realized the importance of quick and efficient communication as well as sharing of ideas and opinions. We also realized that establishing excellent communication mechanisms in not only important for the first phase but also important for the development phase of the project.

Understanding the project and its requirements is perhaps the most important and critical aspect that governs the success of any given project. We realized that and we decided to dedicate the first week as well as the first few meetings to discuss, share ideas, as well as clarify questions any of the team members had. We also posed some unresolved questions to the clients (in this case the professor and the tutors of the course) in order to clarify the requirements and the overall workflow of the project. During this time, we also focused on team building and establishing means and methods of communication, schedules for future meetings, as well as processes for escalating any issues there may be so that anyone in the group is aware and is able to help, if possible.

Since the members of the team were in two different locations, we decided to have weekly sync-up meetings. In the meetings, progress from previous week was discussed along with open issues. Content of the discussions in local teams or feedback from the clients was shared with the other team members. This was done to ensure that everyone in the team is up-to-date with the project and updates.

For the first part of the project, we subdivided the tasks into smaller ones which could be taken up by two smaller teams. This was done because we found it to be more efficient and constructive to have resources in same geographic locations to discuss and work on a task together. This not only helped with quick brainstorming but also with verifying and identifying any errors quickly.

In order to keep change management easy to manage, we initially used Zoho Projects and later shifted to Github. This helped us in easily managing files and updates as well as track changes. We transitioned to Github mostly because we realized that we are going to be using the same tool in the second phase of the project and it would be a good idea to start using it so the members of the team are accustomed to it.

The first phase was sub-divided into four phases. Database design, identification of use cases, wireframing and finally creating the report. While the first two tasks were primarily done on one of the two locations; the rest were done collaboratively, further dividing the tasks into smaller ones. Once the first two phases were complete, the work done by both teams was shared and discussed with the other teams and several meetings were dedicated to answer questions and incorporate improvements.

We plan to continue with similar strategy from the management perspective for the next phase of the project. We also plan to use the tickets system that is available in github to track change management for code and assign tasks for each of the members of the group

4.2. Management of time (Gantt Diagram)

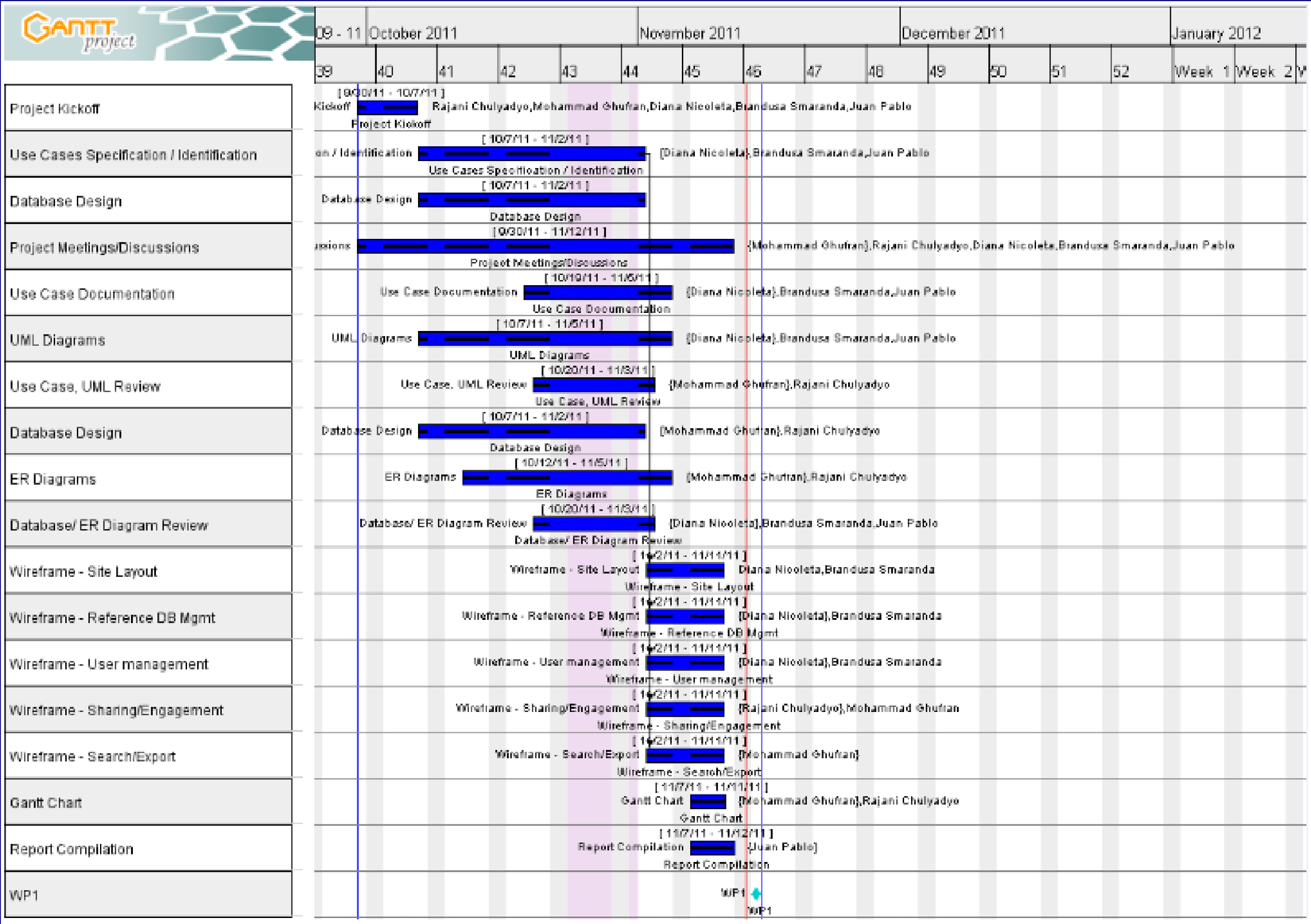


Figure 4.1: Gantt diagram for Vinyl Record Manager project (WP1)