# Umusic : Phase 2

## Information & Knowledge Engineering Project

**Berend Ottervanger / Bojana Dumeljic / Hylke Visser**
**Group 7**

**9/12/2011**

Begeleider: **Pascal Wiggers**
Studentassistent: **Joris Albeda**

# Table of contents

# Goal

The goal of our application is to provide users with music recommendations, based on their listening behavior. It should help users by finding music they like, but would have not discovered themselves. It should let users connect to other users with the same taste in music and let them recommend music to each other by sharing their playlists.

It should also let users share songs, artists or their playlist throughout the web on social networks like Facebook and Twitter.

The application should be able to:

- ✔ Provide authentication methods for users

- ☐ Remember users favorite genres

- ☐ Let users rate recommendations

- ☐ Remember personal ratings

- ☐ Recommend music based on the genres and ratings provided by the user

- ☐ Let user have a playlist

- ☐ Stream users playlist

- ☐ Share playlist with other users

- ☐ Share playlist on the web (Mail, Facebook, Twitter, Google+)

- ☐ Provide information about the artist or song like a biography or news

- ☐ Provide information about the artist gathered from Twitter or YouTube

# Scope

The application is focused on the end user. It uses data, gathered from various music websites such as Last.fm to find music for the user. This data has to be analyzed and linked to the main dataset, the Million Song Dataset. We will use the Playme.com API to stream music and the Echonest API for more data about music.

With frequent use the system learns the user's music taste and is able to make more accurate predictions about what the user might like.

The application will show the user recent posts about a certain artist or song. This information will be collected from Twitter or YouTube.

# Functional Description

In the application, users can register for an account. At the first login we will ask the user to select the music genres he or she likes. This selection can be revised any time. Based on this information the application is able to narrow down the set of songs it recommends music from.

The system displays a list of recommendations based on the selected genres. If a user likes a song from the list, they have the option of adding it to their playlist or removing it from the suggestion list. Based on this feedback the probability that this song or a similar one will be recommended again, is adjusted. The user can also give feedback about songs while they are in their playlist. The chance that the certain song will be recommended again is thus respectively highered or lowered. If a user really doesn't like a certain artist or song they can block them hereby preventing the blocked item from being recommended again.

The application will use PlayMe.com to stream the music in the playlist to the user.

The application will also display recent information about artists or songs, gathered from social websites like Twitter or YouTube.

Users will be able to save their playlist and share them with other users of the application. It will also be possible to share a playlist or a song on Facebook or Twitter.

## Diagrams

The first figure, Figure 1, shows the whole system which is made out of the various databases, the recommendation algorithm and the part that handles the user interaction.

Figure 2 displays a use-case diagram of the system. It shows the actions the user will be able to make. The base system will allow the user to register. It will also handle signing in and out.

The music part of the system will keep the users playlist up to date, stream the music the user has in his playlist, manage the genres the user wants to hear, allow the user to rate the recommendations made and make it possible for the user to share their playlist.

Figure 3 shows the database structure. This structure is made up of two databases. The first one, which is shown on the left of the figure, consists of all the datasets that possess information about music. The second on, which is on the right, contains the user information.

These two databases are combined to form one database which is then used for music recommendation.
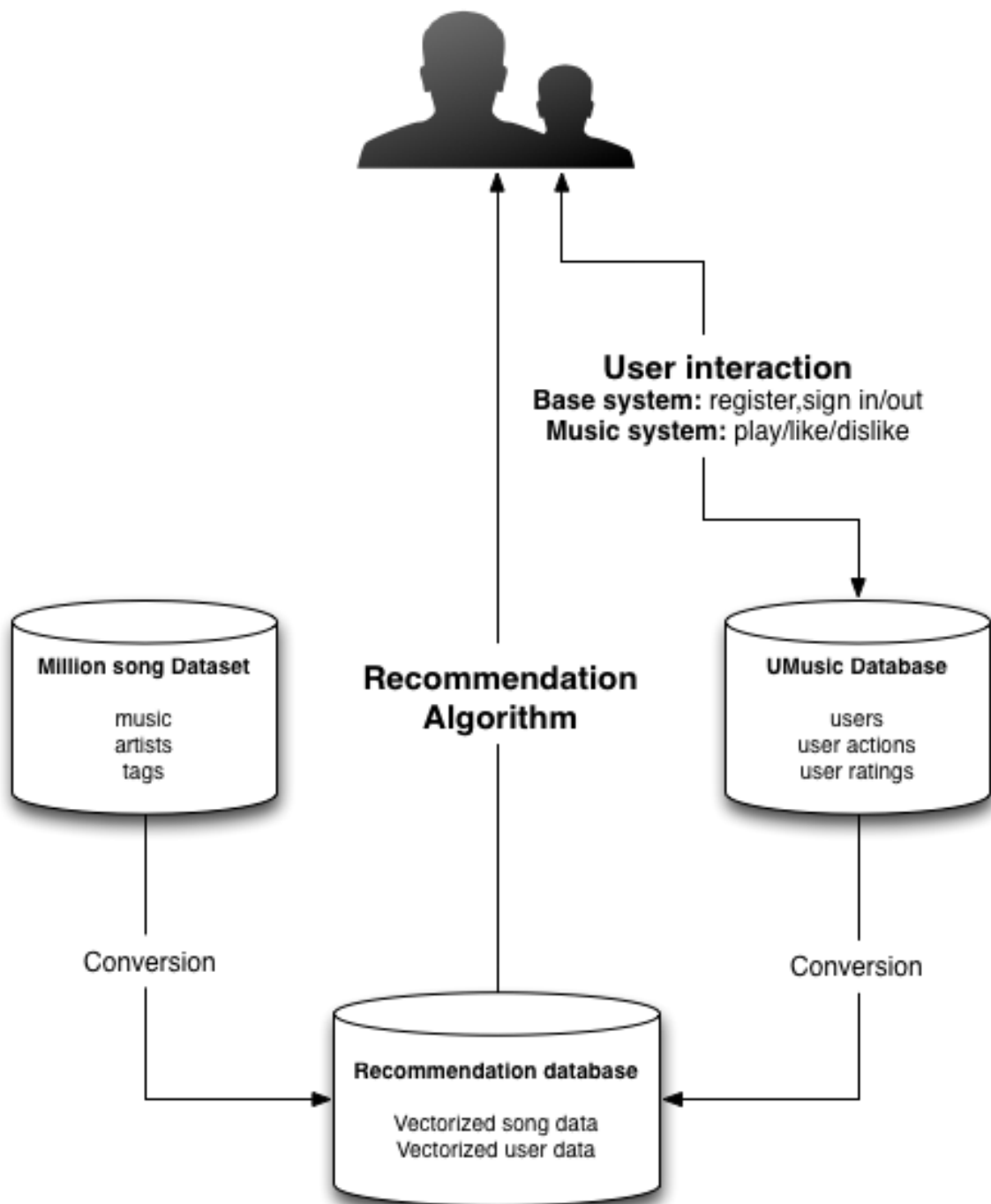
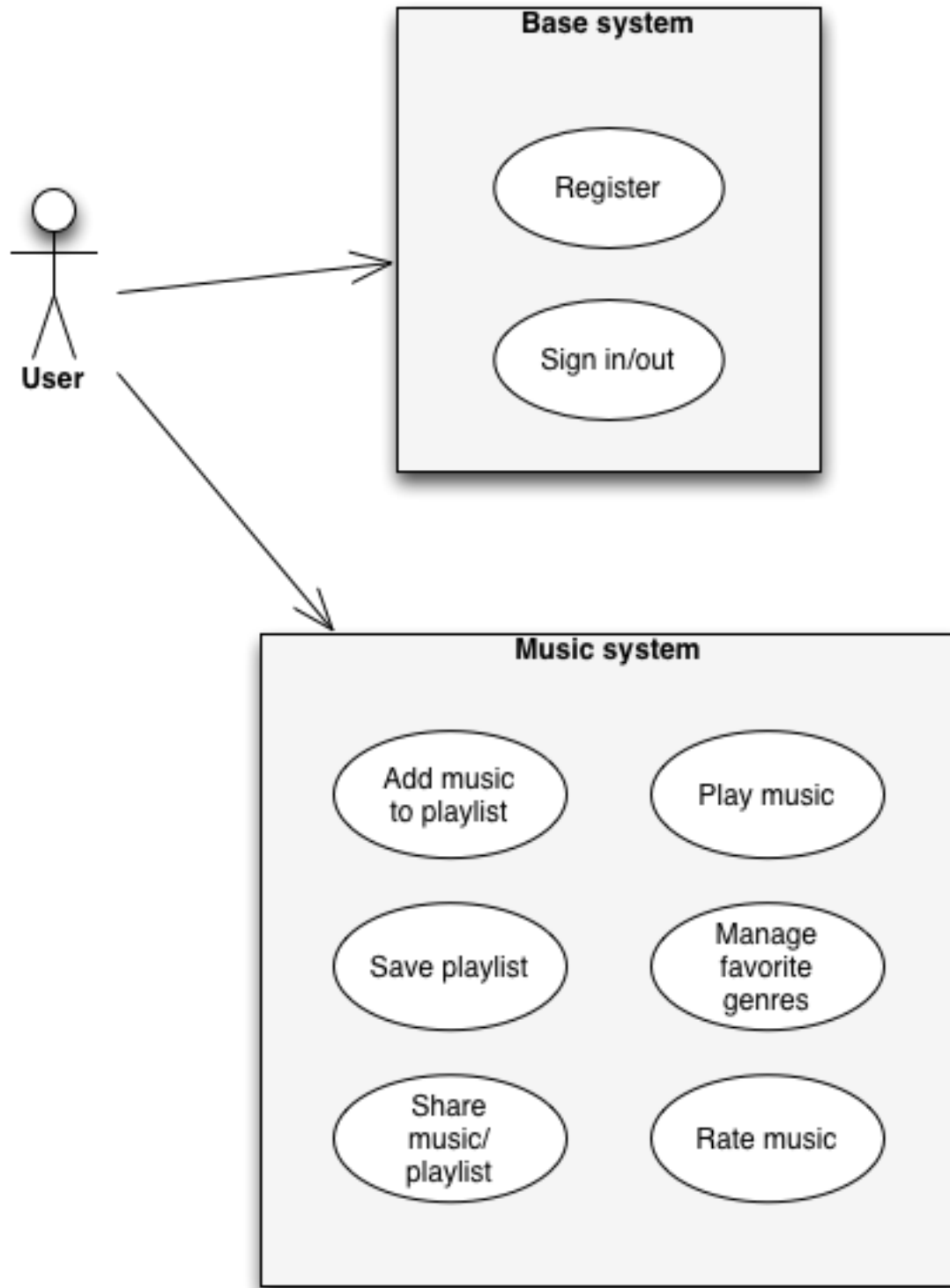FIGURE 1: A DIAGRAM OF THE SYSTEM
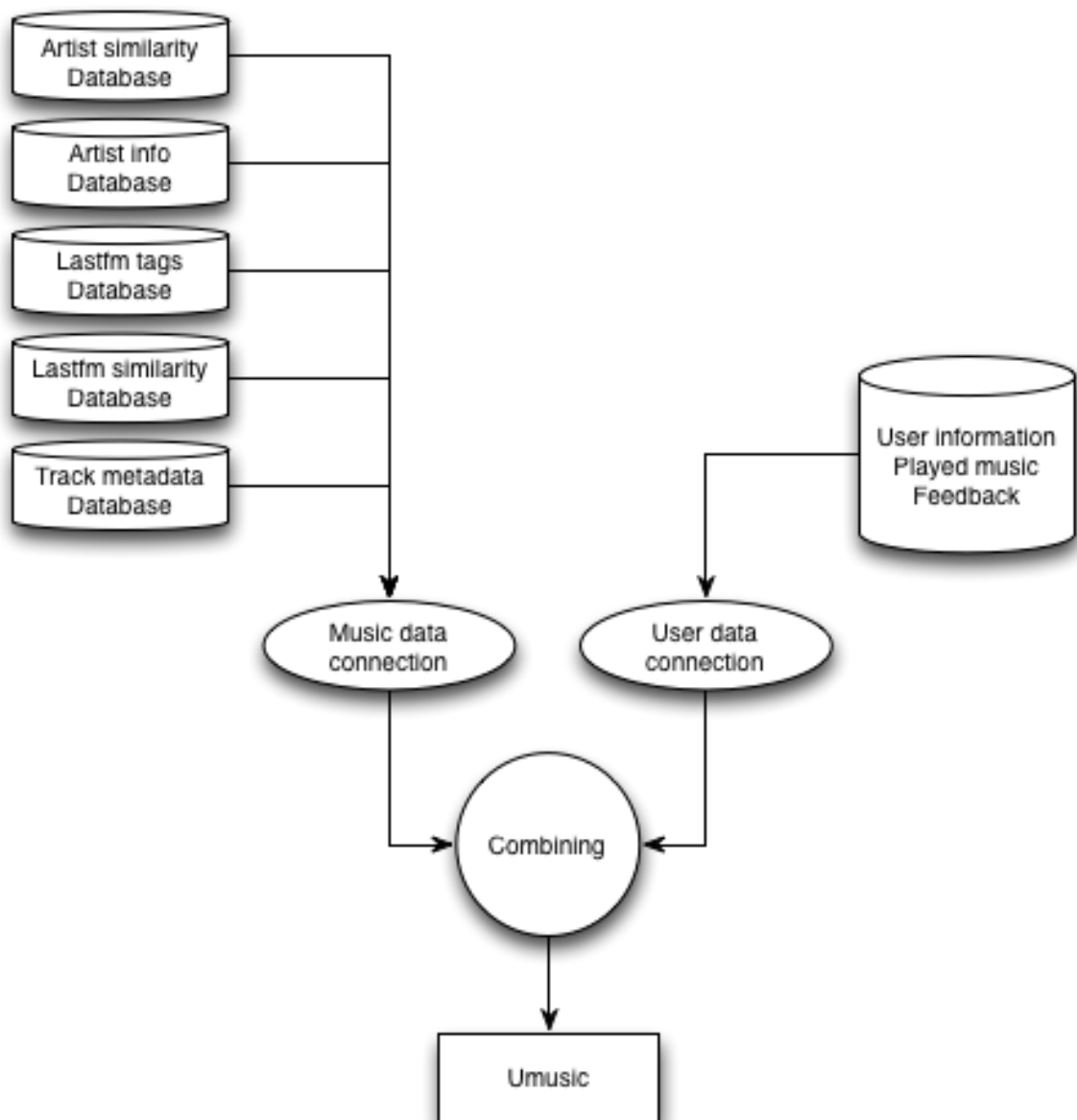
FIGURE 2: THE USE-CASE DIAGRAM

FIGURE 3: THE DATABASE STRUCTURE

# Planning

## Week 1-2

- ✔ Analyse available data
- ✔ Link data
- ✔ Global Design of the application
- ✔ Set up Kohana Framework
  - ✔ Controllers
  - ✔ Connect to Million Song dataset
- ✔ Look for ways to stream the playlist
- ⟳ Design Recommendation Algorithm + data structure

## Week 3-4

- ✔ Update Application design
- ⟳ Implementation of:
  - ✔ Kohana Models
  - ✔ User Account system
  - ✔ User feedback storage in database
  - o Recommandation Algorithm
    - ⟳ Updating database according to user feedback
    - ▪ Recommendation of new songs
- ✔ Specify evaluation plan

## Week 5-6

- Implementation
  - o Unit tests
  - o Music Streaming
- ⟳ Application Lay-out design (website)
  - o Kohana Views
  - o HTML / CSS
- Integration with social networks (Twitter, Facebook, Google+)
- Design Test cases

- Execute tests

- Fix problems in design and implementation

## Week 7

- Testing

- Fix problems in application

- Write project documentation

## Week 8

- Final tests

- Fix problems

- Update documentation

- Create presentation

- Evaluation

# Implementation

The implementation of the product will consist of two parts, a web application for presentation and an underlying service for data gathering and analysis.

## The Web application

The web application will be implemented with the PHP language. We will use the Kohana framework to provide a MVC basis. Kohana also contains various tools for communication with databases and other web services. The website itself will be built on HTML5, CSS and Javascript. It will use the jQuery library and the Playme.com streaming API. We will use the Echonest API to fetch news items and artist biographies and display these as information to the user.

## The Data service

The data service will convert the data from the Million Song and Last.fm dataset to a vectorized format which will make searching more efficient by our application. This service will also add data for new songs that are fetched from the Echonest API.

## The Recommendation Algorithm

Our algorithm is based on finding similarities in songs by comparing the tags they have in common. We will use the million song data set and represent the songs as vectors, so that we can calculate the similarity. For this we will use the cosine similarity function.

We would also like to incorporate the knowledge we gather from users liking or disliking songs. This could be done by constructing a vector of the user. To do this we will use liking information to determine what genres a user likes. Every time a user clicks the 'like' or 'dislike' button, the rating of the tags of the corresponding song get updated. This information will ultimately lead to a good representation of what kind of songs the user likes. This data will be represented as vectors, so that we are able to calculate the similarity (between two users) to recommend songs to the user. Because when two similar users have been found, we can use the songs one user likes to recommend to the other user. We will also need to track the songs the user has already come across, to be sure to recommend 'new' songs. New songs being songs the user does not know (according to the system). The same holds for disliked songs, we wouldn't want to bother people with songs disliked in the past.

Data needed: songs with their corresponding tags and user liking information both represented as vectors. Per song: similarities with other songs. Per user: known songs, disliked songs and similarities with other users.

# Evaluation

To be able to estimate if the system is doing what it is supposed to we will need to test and evaluate it thoroughly. In this chapter we will set up a plan how we will handle this for every goal that we have set up earlier in the relative chapter.

□ *Provide authentication methods for users*
> This part will be evaluated with unit tests.

□ *Remember users favorite genres*
> This part will be evaluated with unit tests.

□ *Let users rate recommendations*
> Check if the functionality is present. No further tests needed.

□ *Remember personal ratings*
> Check if the data is saved correctly into the database.

□ *Recommend music based on the genres and ratings provided by the user*
> To evaluate if the system is actually recommending music that suits the users taste we will ask a group of 10 to 20 people to help us test it. We will ask them to make an account and use the system for an hour or two. We will ask them to write down the number of songs that they actually added to their playlist from the list with recommended songs. Then we will compute the percentage of songs that were selected from the total number of songs in the recommendation list. We will compare these numbers to see if they increase with the time the user has used the system.
> If the percentages do increase then the system is in fact learning from the users' preferences. Otherwise the system is not doing what it was supposed to and we will have to make adjustments.

□ *Let user have a playlist*
> This part will be evaluated with unit tests.

□ *Stream users playlist*
> Check if the music played is in the user's playlist and if every song in the playlist is being played.

□ *Share playlist with other users*
> This part will be evaluated with unit tests.

□ *Share playlist on the web (Mail, Facebook, Twitter, Google+)*
> Check if the playlist has been shared for all the possible cases.

□ *Provide information about the artist or song like a biography or news*
> Check if the output is correct.

□ *Provide information about the artist gathered from Twitter or YouTube*
> Check if the output is correct.

# Retrospective

## Phase 1

We have analyzed a lot of data and finally found a few promising datasets that we can use for out system.

The web framework has been set up successfully using Kohana. The databases have been linked and connected with the system.

The first thing that went wrong was collecting data. In the beginning every single data set we tried turned out to be useless or not working. We wasted a lot of time and effort on this problem. We finally decided to use the One Million Song Dataset with additional information from Last.fm.

Furthermore we haven't yet been able to find a way to stream the music. More importantly the recommendation algorithm hasn't been completed yet.

Next time we should keep in mind to not waste a lot of time and effort on one place.

## Phase 2

In this phase we have updated our design and a few Kohana models. We have also implemented the user account systems and user feedback storage.

We have also written an evaluation plan.

We were a little bit too enthusiastic with the planning of this phase and have had to move a few things to the next phase due to time issues. Also the algorithm isn't finished yet.

Next time we should keep in mind to not delay the biggest and most important task, the recommendation algorithm.