



Umusic Final Report

Information & Knowledge Engineering Project

Berend Ottervanger / Bojana Dumeljic / Hylke Visser
Group 7

18 January 2012

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

Table of contents

Table of contents.....	1
Introduction.....	2
Design	3
Goal	3
Scope	4
Functional Description	5
Diagrams.....	5
Implementation.....	11
The Web application	11
The Data service	11
The Recommendation Algorithm	12
Planning.....	13
Week 1-2	13
Week 3-4	13
Week 5-6	13
Week 7.....	14
Week 8.....	14
Evaluation.....	15
Measures	15
Results	16
Retrospective.....	17
Phase 1	17
Phase 2	17
Phase 3	17
Phase 4 - Final.....	17
Discussion.....	18
Problems.....	18
Tags.....	18
Further directions.....	18
People involved	19
Groep 7	19

Introduction

The web is large and vast. It has become time consuming to go through all of that information. To be able to steer ourselves efficiently through this large pool of information we could use some tools. To be more precise, we could use more web applications.

For this project the information has been limited to music. Throughout the years countless of artist have made an enormous amount of songs. In this time and age of technology we have a fairly good access to most of them online. Besides this even more people have written about music in various ways. Thus we have a very large amount of information available on this subject.

Finding new music to suit your taste or information about music that you love has become an endlessly time consuming task. To make our lives a little bit simpler we could use an application that can provide us with music recommendations based on our music preference, information about music and also play this music all in one place.

For the past eight weeks we have been working on this problem. This report describes our progress throughout this time period. It contains our goal and our idea for the problem solution, how we have implemented this, an evaluation of the project and a discussion about the whole.












Design

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 YouTube to stream music and the Echo Nest 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 Echo Nest, 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 songs from this genre 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 a YouTube player to stream the music in the playlist to the user. This will be done within the application. The user will not have to leave the website to actually hear the recommended music. Everything is kept in one place.

The application will also display recent information about artists or songs, gathered from social websites like Twitter or YouTube and Echo Nest's array of music data.

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. This way users will have a richer music experience.

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 UML 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 4 displays the recommendation algorithm. A detailed description about the algorithm can be found in the chapter about the implementation of the system.

Finally in figure 5 we have a simple sequence diagram which displays the recommendations action.

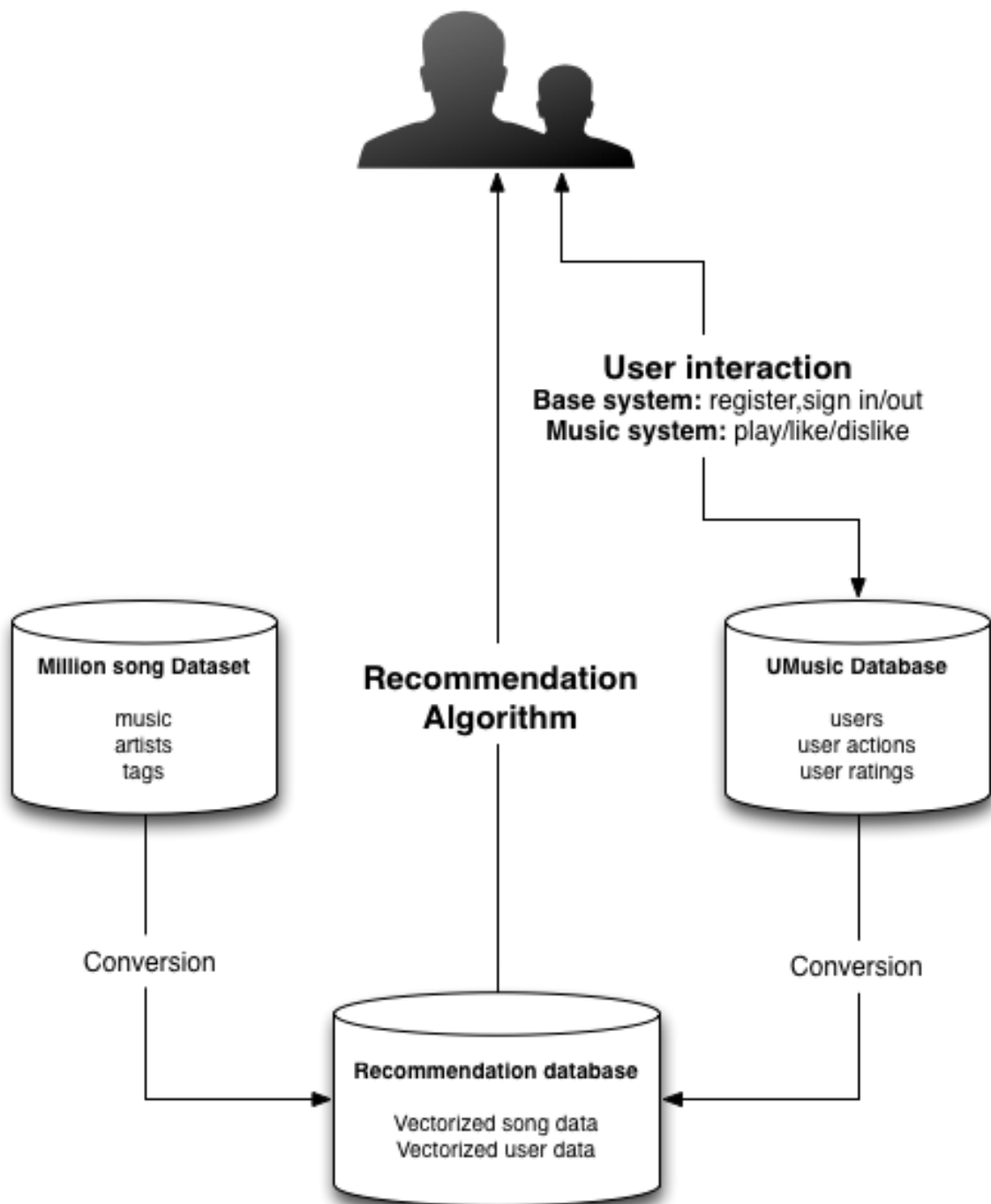


FIGURE 1: A DIAGRAM OF THE SYSTEM

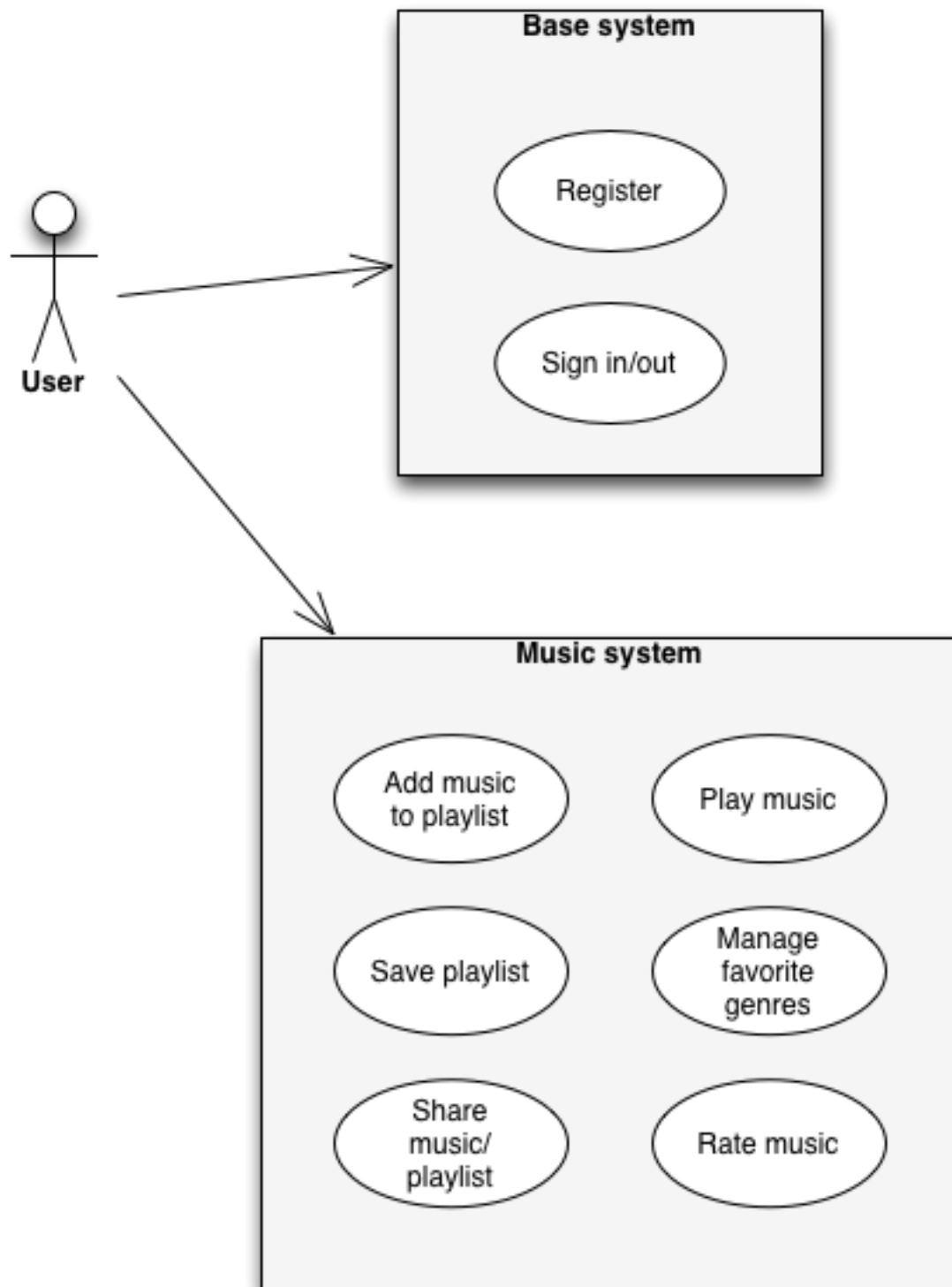


FIGURE 2: THE USE-CASE UML DIAGRAM

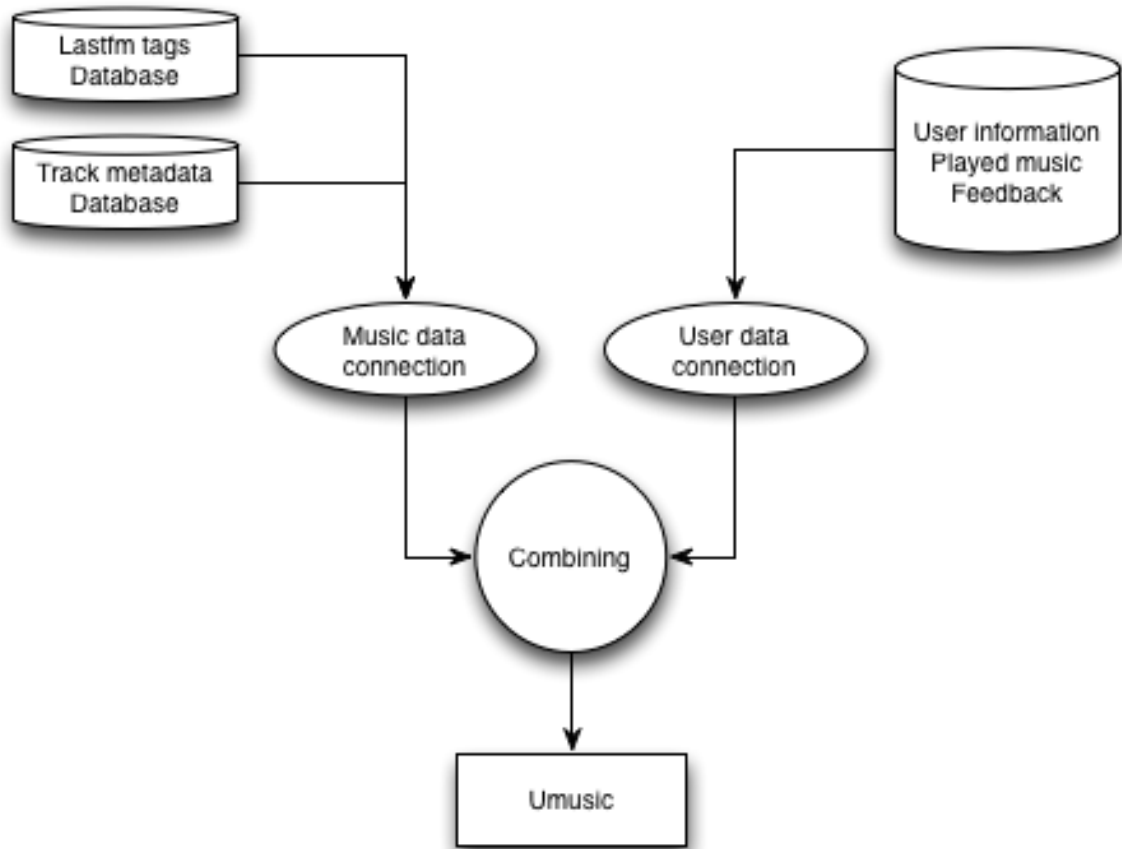


FIGURE 3: THE DATABASE STRUCTURE

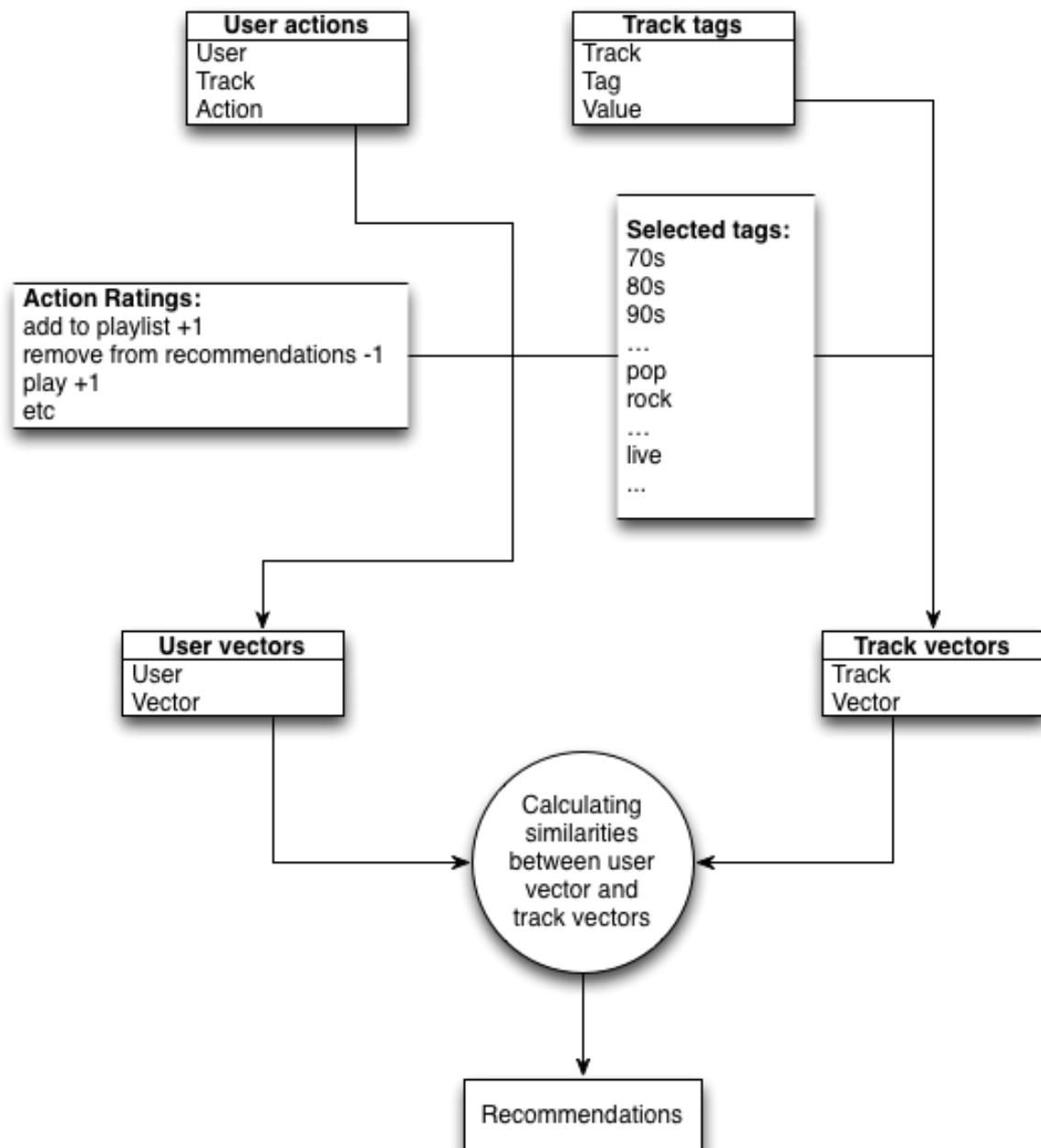


FIGURE 4: THE RECOMMENDATION ALGORITHM

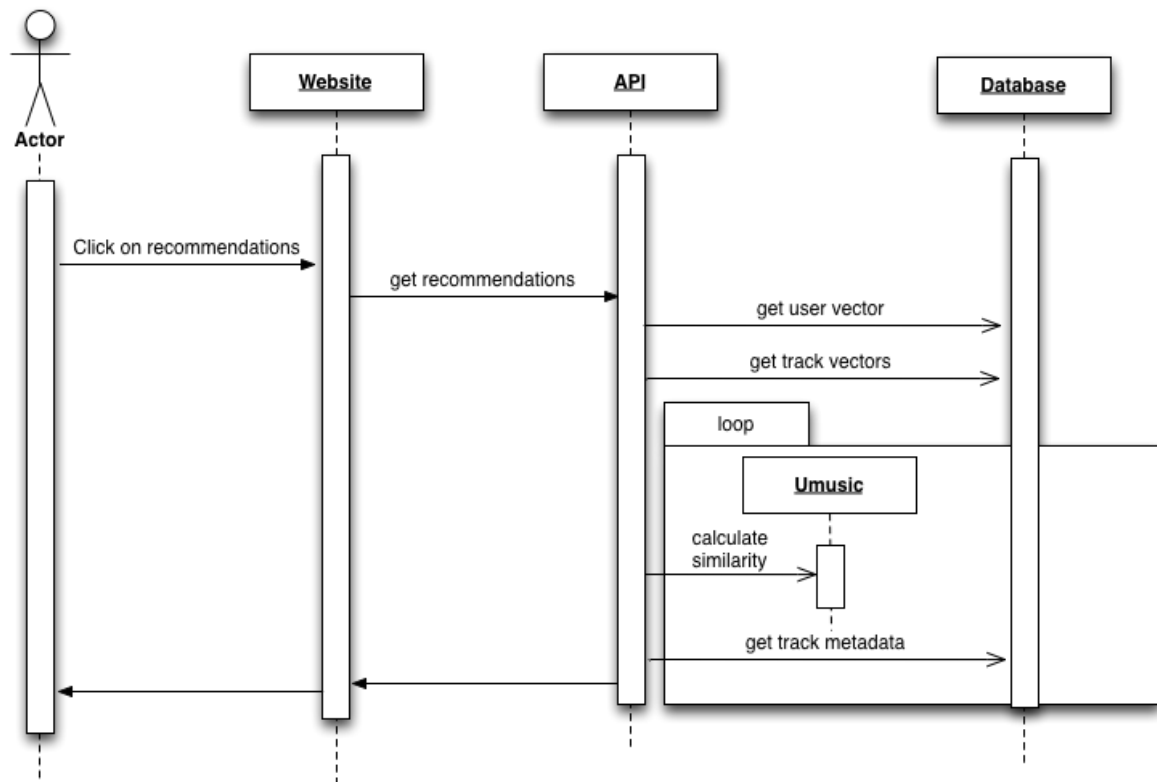


FIGURE 5: THE RECOMMENDATIONS SEQUENCE DIAGRAM

Implementation

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

The Web application

The web application is implemented with the PHP language. We used the Kohana framework to provide a MVC basis for the system. Kohana also contains various tools that help communicate with databases and external web services. The website itself will be built on HTML5, CSS and JavaScript. It will use the jQuery library to handle interaction with users.

We used the Echo Nest API to fetch information, news and reviews about artists and display these as extra information to the user. We will use the wrapper¹ for the Echo Nest API written in PHP5 by Brent Shaffer for information gathering from Echo Nest.

The album art is requested from Last.fm. Their API provides the system with the location of the album art images. These are downloaded and stored on the server.

UMusic provides a streaming option through YouTube. We used the YouTube Data API to obtain the video ids of the songs in the playlist. These are passed to the YouTube player JavaScript API, which starts playing them.

The Data service

The Data service has two main functions, the conversion of track information to a vectorized format and the conversion of user information to a vectorized format.

The conversion of the track information is done only once. We have selected a number of tags (or genre) to use. For each track in our database the data service checks for each of the selected tags which tags apply to the current track. These results are vectorized and stored in the database.

The conversion of user information can be done periodically. For each action a user has performed the tags of the affected track get a value corresponding to the action type. For example, adding a track to the playlist has a value of +1, disliking a track is -1 etc. These vectors are summed and stored in the database.

This conversion process can be started by a user, by clicking the update button on the "My Tags" page.

¹ A simple, Object Oriented API wrapper for the Echo Nest API written with PHP5 (<https://github.com/bshaffer/php-echonest-api>)

The Recommendation Algorithm

Our algorithm is based on finding the similarities between users and songs by comparing the tags they have in common. We use the million song data set and represent the songs as vectors (as described above), so that we can calculate the similarity. This approach is called item-based recommendation.

Our first approach was to calculate the similarities between the user and all tracks in the database, sort them and return the songs that have the highest similarity. This process had an enormous impact on the performance of our server, so we had to come up with a different approach.

The final method still calculates the similarities between users and songs. The difference is that now we only return recommendations with a similarity higher than 80%. We also stop calculating when we have 100 recommendations.

The similarity between two vectors is calculated using the cosine similarity function.

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
 - ✓ Recommendation Algorithm
 - ✓ Updating database according to user feedback
 - ✓ Recommendation of new songs
- ✓ Specify evaluation plan
- ✓ Get the server running

Week 5-6

- ✓ Implementation
 - ✓ Playlist
 - ✓ Display user's tag profile to user
 - ✓ Music Streaming
- ✓ Application Lay-out design (website)
 - ✓ Kohana View
 - ✓ HTML / CSS / Javascript
- ✓ Design Test cases

- ✓ Fix problems in design and implementation

Week 7

- ✓ Implementation
 - ✓ Music information
 - ✗ Sharing with other users
 - ✗ Sharing on social networks
- ✓ Testing
- ✗ Integration with social networks (Twitter, Facebook, Google+)
- ✓ Fix problems in application
- ✓ Write project documentation

Week 8

- ✓ Final tests
- ✓ Fix problems
- ✓ Update documentation
- ✓ Create presentation
- ✓ Evaluation

Evaluation

Measures

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*

Check if the user is being registered and can successfully sign in and out.

✘ *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*

Check if the playlist is saved correctly in the database and if it is displayed and updated correctly.

✔ *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.

Results

We have decided to not implement a few functionalities because we did not have time for them. The user will not be able to manually change their user vector. They will only be able to see their vector in a graphic form. It would be too complicated to add this functionality.

We have also not been able to implement any kind of sharing between users or on the web and we have not implemented a Twitter feed.

Therefore an evaluation of the above functionalities was not necessary. We have checked that the other functionalities that we have implemented work correctly.

We have performed a user test. Due to a lack of time we have only been able to test the application with five different people excluding ourselves. We have asked them to make an account and add 10 to 15 songs to their playlist and then ask for recommendations. We have asked them to give us an indication on how many of the recommendations they actually liked. Then we asked them to repeat this process a few times. We have come to the conclusion that there is room for improvement in the recommendation algorithm. The recommendations are far from accurate and it can take quite a long time to compute them in the beginning. They do get better in time. We will discuss ways to improve this in the discussion.

The general opinion about the application was that it was user friendly and had pleasing graphics. Suggested improvements were more artist information and more player options (shuffle, loop, mute, etc.). A way to be able to control your own user profile would indeed be handy.

Retrospective

Phase 1

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

Phase 3

During this phase we have finally been able to complete and implement our recommendation algorithm. We have also implemented the user feedback and playlist. We are still working on the streaming of the said playlist and the storage of user's favorite genres. The HTML and CSS design has been updated and polished.

Thorough tests have yet to be done.

So far we have finished half of the goals that we have set. We have not been able to write the test cases yet. Thus we are falling behind on our schedule.

Phase 4 - Final

This phase concludes the project. We have had to drop a few functionalities: manual adjustment of user's favorite genres, sharing in any kind of way and information from Twitter or YouTube. There was no time to implement these. We have also had very little time to test the system.

More about this in the chapter Discussion.

Discussion

Problems

During the design and development of the UMusic system, we kept encountering problems. In the first phase we ran into the problem that every data source we wanted to use was offline or just not working. When we finally found a data source that was actually working, we realized that our plans were a little bit too ambitious. We had to drop several items.

The following functionalities were dropped because we were low on time:

- Sharing in any way
- Twitter feed
- Selection of genres at registration and user adjustment of these genres
- Artist block list
- Recommend same song twice

Tags

Because we were using a dataset with a million songs we encountered the problem that there were a lot of tags. Because of the huge amount of tags the system was very slow. We managed to narrow down the thousands different tags to 80. We have had to sacrifice accuracy for more speed.

Unfortunately, the current system still has some problems. The process of calculating track vectors is very time consuming. Calculating the track vectors in the current system took our server about six hours of work.

The recommendation algorithm is also quite slow. Sometimes a user has to wait about 10 seconds before recommendations are shown. This is not acceptable for a production website.

Further directions

Some of the problems can be solved by upgrading the resources of the server. For others we would need a better solution. By implementing the hybrid system that we actually wanted to make we could solve some of the problems with the accuracy. So a user based algorithm should be added to this system. But then a new problem would arise that we would need a high amount of actual users for the user based algorithm to actually work.

The actual functionality that we had to drop could be added later.

People involved

Groep 7



Berend Ottervanger
netID: bottervanger
studentnummer: 4082427



Bojana Dumeljic
netID: bdumeljic
studentnummer: 4103246



Hylke Visser
netID: htdvisser
studentnummer: 4082222

Supervisor: Pascal Wiggers
Studentassitent: Joris Albeda