CONTEXTPROJECT

Product Planning, TI2806

Authors:
S.A. BOODT, sboodt, 4322258
T. HEINSOHN HUALA, theinsohnhuala, 4326318
A. HOVANESYAN, ahovaneyan, 4322711
D. SCHIPPER, dschipper, 4155270
J.R. WEIJGERTSE, jweijgertse, 4247124

Delft University of Technology

Friday 15th May, 2015

Abstract

This document describes the plans to make the product Discover. Discover is an new way to listen to music quick and easy. The planning of exactly how, what and when we are doing during the period we are working on the product.
We will discuss several things in this document. First of all we will focus on the product in chapter 2. Which contains a high-level product backlog and a roadmap of what we will create each sprint. Second we will look to a more detailed product backlog in chapter 3. After this will look at the definition of done in chapter 4 where we define when a part of our product is really finished. Finally we will explain some terms we used in this document in chapter 5.

Contents

1 Introduction

"One cannot deny the power of music." This is what Laurence O'Donnell(1999) wrote in his article about the effect of music on people. He saw that music and humans are almost always connected with each other throughout history. Music has a great influence on people. It can help people achieve great things, but it can also have a negative effect on people.

The fact that music can be so valuable to humans, is a good reason to create our product, called Discover. We want to have a way to improve the current listening experience and which can improve the way people explore new music. This new music can be a whole other genre or artist they normally listen to. Music is the most important thing of our product. Our product consists of music that is fetched from another product: SoundCloud.

Another important part of our product is the user. Mainly the part where the user likes or doesn't like the song or where the user has something to say about the song. In our product we use this data to improve the exploring of the song.

The main feature of our product (besides the part where you can listen to music) is that users should be able to listen to a small preview of a song to check whether they like it or not. We call this preview a snippet. In our product we will implement a way to get the best snippet of a song, based on various metadata. The use of a snippet will increase the user experience of exploring music. It will save time when a user doesn't like the snippet and chooses to listen to the next snippet. The user will also be able to hear a snippet of an album or mix of songs. This snippet will consist of other snippets which will be the most important moments of an album or a mix.

2 Product

In this chapter we will specify the High-level product backlog(2.1) using the MoSCoW method and the Roadmap(2.2), which specifies which features will be implemented in which sprint.
We divide the features in four different groups:

Must haves features that are essential for the application to function.

Should haves features that have a high priority and should be done if possible, but could be satisfied in a different way if needed.

Could haves low priority features that could be implemented if time permits.

Won't haves features that will not be implemented but could be features for a follow-up project.

2.1 High-level product backlog

Must Haves:

- The application working with the SoundCloud widget.
- Being able to play a part of the track with a press of a button. This preview shouldn't be too long and contain the drop or an other important part of the track.
- Have a database containing the metadata of a track. Data such as track ID, user ID and information about the feature of the track. For example: the track's bpm.
- An input field to reload the widget with a new track or list. The fields should be able to accept the SoundCloud links for lists and other sets.
- An algorithm that searches the most interesting parts in a track based on comments or features.

Should Haves:

- A function to play a random track from the database by pressing a button.
- Extracting metadata from an unknown track through the SoundCloud api. This data includes the id's of tracks, comments from that track and features like bpm.
- Being able to add a song to a users collection from the application, when the user finds a song that he or she likes.
- A way to preview a whole mix. With the preview containing a snippet of the most important parts of the mix, for example a part of each song in the mix.
- A preview of a list or an album. With the preview containing a snippet of every song in the list.

Could Haves:

- A recommendation system based on the previously liked tracks.
- Browsing SoundCloud through the web application. Showing currently popular tracks and other groups from the SoundCloud website.
- The use of user feedback and behavior to test the snippets validity.

Won't Haves:

- Integration with other services like Spotify or previewing snippets of songs that are not on SoundCloud.
- Added visualization of the playing track apart from the SoundCloud widget.
- Way of uploading own tracks through the web application.

2.2 Roadmap

Design phase

In this phase we will decide our approach of Discover. We have two goals:

- Determine the most interesting part of a song or a compilation of a mix/playlist.
- Determine how we are going to achieve this goal(in terms of programming language(s) to use, frameworks and api's).

Sprint 1

The first week of the implementation of Discover.

- Working web player.
- A database with all the comments.
- The backbone of a snippet selector.

Sprint 2

The main goal of this sprint is to implement some must-haves.

- Link all the different components together.
- Improve the snippet feature by using the comment intensity.

Sprint 3

- Improve the selection of the snippet by using the feature essentia of a song.
- Filtering out the comments that are valuable to use for the selection.

Sprint 4

- Refine and improve the selection of the snippet.
- Improve the snippet feature by using the comment intensity.
- The user interface will be improved.

Sprint 5

This sprint we should focus on making our product ready for the upcoming presentation.

- Refactoring of the code and be sure the code is well tested.
- Finish the user interface.

Sprint 6

- Determine how mixes will be split up.
- Begin with implementation of creating a compilation of a mix or playlist.

Sprint 7

- Improve splitting up mixes.
- Making snippets of these pieces to create a compilation.
- Also make compilation for playlists.

Sprint 8

Final sprint

- Refactoring code.
- Be sure the product is done according to the Definition of Done in chapter 4.
- Completion of all previous sprints.

Release

- Make the final report.
- Make documentation about Discover.

3

Product Backlog

In this chapter we will discuss the product backlog more in detail using User Stories(3.1) and we will specify the due dates of the releases in the Initial release plan(3.2).

3.1 User Stories

In this section we will describe the user stories for the features that were discussed in chapters 2 and 3. The user stories all have and identifier: 'M', 'S', 'C', 'D' or "K". The identifiers 'M', 'S' and 'C' tell the priority of the user story feature, being from the Must haves, the Should haves or the Could haves section respectively. The 'D' identifies the user stories for defects and the identifier "K" is used for the know-how acquisition.

3.1.1 User Stories of features:

ID	Story
M1	As a user I should be able to listen to a preview of a song instead of the whole song. The preview shouldn't be too long and contain the most important part of the track. For example: the drop.
M2	As a user I should be able to use the application through my browser.
M3	As a user I should be able to listen to the music with the Soundcloud widget. The player should have the same features as the one on SoundCloud's official websites.
M4	I want to be able to choose a track or a list of tracks with an URL.
S1	As a user, when I like a song, I want to be able to add it to my Soundcloud likes.
S2	As a user I want to be able to play and preview any song that Soundcloud has.
S3	As a user I want to be able to preview a mix of songs, with the preview showing me multiple snippets from the mix.
S4	As a user I want to be able to get a preview of an album or a list of tracks of my choice.
C1	As a user I want the system to recommend me a different track I might like based on a song I have liked using the system.
C2	As a user I want the preview snippet get better at predicting the best part of the song by looking at my behavior and feedback.

3.1.2 User Stories of know-how acquisition:

ID	Story
K1	As a user I want tool tips to show me what steps I have to take to generate a preview for a track or a mix.
K2	As a user I want the application to tell me how it came up with the recommended songs.

3.1.3 User Stories of defects:

ID	Story
D1	Sometimes if I switch during a preview I cannot preview the next song, but somehow can preview the song after that.
D2	As a user sometimes when I already have listened to a snippet I need to press the button twice to preview the track again.
D3	Sometimes when the preview button is pressed and the page is reset the widget loses its current list and only show the song I wanted to preview.

3.2 Initial release plan

We have the following milestones for our product. At the end of the third sprint we should have a functional web player linked with a database. On this web player the user should be able to enter an url (provided this song is under the correct license) and receive a snippet based on the comment intensity. The minimum requirements features for this release are a web player based on the SoundCloud widget, a linked database and the snippet based on the comment intensity. The next milestone will be at the end of the fifth sprint, the product should be able to provide a correct snippet of a song based on the comments and the feature essentia of the song. The MRF of this release is the same features as the previous milestone, but the snippet is now based on the comment intensity and the feature essentia. The final milestone will be after the eight sprint. Our product can now process a mix or a playlist and generate a compilation of the songs. The compilation will be based on the snippets from the previous milestones. MRF are again the same, plus the web player should be able to process a mix/playlist and generate a compilation.

releases	added features	due dates
0.1	Capability to play music.	01-05-2015
1.0	A web player based on the SoundCloud widget A linked database Snippet is based on the comment intensity	08-05-2015
1.1	Snippet is now based on feature essentia.	15-05-2015
1.2	GUI is looking better and more practical	22-05-2015
2.0 (prototype)	Snippet is now based on the comment intensity and the feature essentia.	29-05-2015
2.1	Determine how to split up mixes.	05-06-2015
2.2	Combine the snippets for mixes into a compilation.	12-06-2015
3.0 (final)	The web player should be able to process a mix/playlist and generate a compilation.	19-06-2015

Table 3.1: The releases in a nutshell

The whole schema can be seen in table 3.1. This table provides a quick glance at the release plan. The release plan is very influenced by the Roadmap in chapter 2.2.

4

Definition of Done

This chapter describes what needs to be done on the different subjects in order to consider said subject done. When we talk about test coverage in this chapter we refer to the default coverage provided by the plugin Cobertura, or in case this is not compatible with the projects programming language (or version) the line coverage of the project.

4.1 Features

A feature is considered to be done when the following conditions are fully met. If for some reason a feature does not satisfy all conditions the reason should be explained.

- The code should at least have 75% coverage. In case the code doesn't need this coverage, it should be explained in the testreport why this feature doesn't need this amount of coverage.
- All the code has javadoc statements to allow for easier understanding of the given variable, method or class.
- All the implemented tests should pass on the local machine, Maven and Travis.
- PMD and FindBugs should report no errors when being run over the code.
- CPD, the add-on for PMD, which stands for copy paste detector should also issue no errors that are not explained by comments or by some document that backs up why CPD throws this error.
- CheckStyle should report few to no errors other than the magic number errors in the test classes. A checkstyle list will be made specifically for this project.
- The code is provided with comments to explain certain parts where needed.

This ensures that the feature works as intended and is correctly documented. After all these requirements are met a feature can, after a review of at least two other members of the team, merged with the main project.

4.2 Sprints

We consider a sprint done when all the tasks of the sprint have been fulfilled. A task can be delayed if it is not considered important enough for this weeks sprint and is moved to next week. Every sprint should have a release of the product that is an improvement of last week, either in looks, speed, features or quality. The release should have a version number and tag before it can be considered done. A testing coverage of 75% overall has to be achieved as well. Last if the sprint has a milestone it should have been reached in order for it to be considered done.

4.3 Releases

A release is considered done when all of the sprints have been done and thus appropriate milestones have been reached. The release is only considered done as the stakeholders are satisfied.

A release is considered done when all the sprints and features are fully completed. It is of importance that the stakeholders are satisfied with the release. All the must haves, described in chapter 3 should be implemented. Besides the mandatory functionality a part of the should haves should be implemented as well, we aim to implement 30% of these features to deliver a more complete product. It is also of importance that all the code is efficiently implemented and up to the standards. The SIG test of our product is an importance part in the completeness of our product.

5

Glossary

API Application program interface. An interface that is provided so other programs can call the methods provided by the Application.

BPM Beats Per Minute, a unit to indicate the tempo of a song.

Bug Flaw in software.

CPD Copy paste detector.

CheckStyle Static analysis tool that checks the style and format of the code.

FindBugs Static analysis tool that finds bugs in the code.

Framework According to the page 'Software Framework' on Wikipedia: "An abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software."

Line coverage The percentage of lines that has a test that covers them.

Magic number error A style error that is caused by hard coded numbers in a class.

Maven A tool that makes it easy to ship the product in a platform independent way.

Metadata Data about data, like the bpm and the length of a song.

MRF Minimal releasable features, minimal set of requirements to release the software.

PMD A static analysis tool.

SIG Software improvement group.

Snippet Small part of a song or mix.

SoundCloud A popular music platform.

Test report A document that contains information about why certain parts are not entirely tested.

Travis A service that allows the project to be continuously tested on a independent machine.

Tool tip A small message that pops up when you move over a button.

URL Uniform Resource Locator, way to find a document.

References

- O’Donnell, L. (1999). *Music and the Brain* Retrieved from http://www.cerebromente.org.br/n15/mente/musica.html
- Software framework. (n.d.) Wikipedia. Retrieved May 15, 2015 from http://en.wikipedia.org/wiki/Software_framework