Music Manager for UTM CSCI 352

Andrew Marshall, Kyle Rolland

Abstract

The goal of this project is to create an interface that allows a user to store and manage music of their choice. Users will be able to create play lists, favorite certain songs, and sort songs by the artist, or the song name. Our goal with this project is to make a simple version of music applications like Spotify or iTunes, for people who don't like using their current music player application.

1. Introduction

Music is an everyday part of life, and many people go everyday without even thinking about the luxury that we have in the modern era, being able to listen to what we want, wherever we want to. Following this, there are many many platforms that one can use to access music. Some of them work well, some of them aren't so lucky. We aim to reproduce one such platform, creating an interface that allows someone to add music of their choosing, organize it, and play it.

Obviously there's nothing groundbreaking about an undertaking like this, but providing people with more options is never a bad thing. Even if there are only a few things that differentiate one software from another, there are always going to be users looking for new experiences. We would like to be able to appeal to as wide an audience as possible, people who have problems with the existing flaws in other software, which are ignored or put off of being fixed by developers. Helping those people find a new place where they feel they can store their favorite songs can make a large difference, especially because music can be a very valuable, memorable resource for some people.

1.1. Background

There aren't too many things that an average user wouldn't be able to understand, when it comes to a project like this. There is a term called ”Metadata”, and in this case, it's really just a technical term for basic music information. It pertains to things like album art, release date, track/song length, genre, artist, and the songs title.

We decided on a project like this because music is something that is easy to make connections between. Thankfully, The two of us had common ground in a genre that we used to listen to, but even if we didn't, it would be easy to come together to work on a project like this. It originally sprouted as an idea for an audio waveform visualizer, the flashing bars/lights that you'll see with music videos, that correspond to the beat of the song, or the lyrics. Neither of us have looked into what it takes to make those, and some further consideration led us to decide that it would be better to take a step back and just make a music management interface.

Andrew has run into problems with CDs on iTunes, so if we could work out a way to transfer music from the manager to the CD, and various other abilities for handling music that way, it may provide a new place for him to use.

I (Kyle) listen to music in the mornings and when I go to sleep, and I frequently find myself wanting to mix up what I'm listening to, but not at the immediate moment. My main music application is Apple Music (which is really just iTunes but with more steps), and I always find myself pining for the simple option to shuffle a play list, and then add it to the end of my listening queue, and unfortunately Apple Music does not offer something that seems as easy to implement as that. If we could work the capability into this project, I think that would be great.

1.2. Impacts

Creating another music management system isn't really something that will make waves and affect the core of many people, but it would be wonderful to see if we can even just reach some people, and give them another place to handle the songs that they hold dear. It doesn't take much to brighten someone's day, and hopefully with this project we can manage to do that for a handful of people.

1.3. Challenges

It may be difficult to organize and handle the previously mentioned metadata, since there is a lot of information that goes along with it. We also may run into trouble trying to create a system to accept different file types, like .mp3 versus .mp4 versus .wav, but working a system like this out would prove very useful for attracting more users. Making a display

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

for things like album art, and giving the user the option to upload their own art to go with songs or created play lists may also prove to be difficult, partially because of accepting images, and partially reaching back to accepting different image file types, like .jpeg and .png. Lastly, we are still sort of hoping to implement an audio waveform visualizer somewhere within the interface, and I think this will prove to be the biggest hurdle, especially if we can't find a framework to start with.

2. Scope

Our project will be completed when we have an interface that allows the user to add songs of their choosing, modify their metadata, add those songs to play lists and edit those play lists freely and easily, search up songs within their database, and sort them by given data types, like song duration, artist, genre, and song title

We have some ideas for stretch goals. One of them would be implementing the previously mentioned audio waveform visualizer. Along with this, being able to easily export your play lists or songs to various different platforms would be very handy. Lastly, we would love to be able to implement a system that allows the user to upload images for artwork that will display when a song is playing, be it album art, or an image that will act as the ”face” of a given play list, allowing for more user customization.

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.1.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number:	1
Use Case Name:	Add item to cart
Description:	A shopper on our site has identified an item they wish to buy. They will click on a “Add to Cart” button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item

2) User left-clicks on “Add to Cart” button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on “Add to Cart” button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2

Use Case Name: Checkout

Description: A shopper on our site has finished shopping. They will click on a “Checkout” button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.