DIGITAL SPLITTER

Exploring ways to synchronize audio across devices

Naveen Ram[1], Gianna Giancarlo
Gunn High School[1], Hero Digital[2]

INTRODUCTION

Purpose

In order to explore the connection between engineering, design, and business, I decided to pick a problem I experience in my community and to design, code, and market a software solution over the course of the year. I planned to use Java in order to create a product, which I would refine and market with the help of my mentor, Gianna Giancarlo, at Future Perfect Labs.

Problem

When I work, I find it helpful to listen to music, and I'm not alone. If you walk through any public workspace, you can observe people with headphones on, immersed in their own soundscapes. It is clear that music can be a great catalyst for productivity, but there is a problem. Today's music streaming services do not allow people to listen to music together. I often find myself working with peers in a library, coffee shop, or classroom, faced with the dilemma of whether to work together, and go through the awkward process of sharing headphones, or listening to different music and collaborating less. I solved this problem by creating an application that allows two people to listen to music from one computer on two different devices. This allows multiple people to be engaged in the same auditory environment, while not disturbing others in a public workspace. I believe this can increase collaboration and productivity, as well as bring the communal and social aspect back into music.

MATERIALS & PROCESS

Libraries
•Swing (Javax)
•JavaFX

Language
•Java

Computer
•MacBook Pro

This project was one of my first programming projects, and the only project of this scale which I have attempted, so I decided to use the language I was learning in school, Java. Java is the most popular programming language in the world, but its syntax is more complicated than languages like Python, so I had to spend a large portion of the year learning how to implement various graphics and auditory elements.

In Java, many libraries, or compilations of pre-written classes or templates, are available to make programming quicker and easier. Selecting the graphical library (Swing or JavaFX) to use was tricky. I initially wanted to use Swing because I already knew it a little, so I started writing some Swing methods to play music. I looked for open source music application templates online to get ideas, and I found that almost everywhere I looked, people were using JavaFX, which seemed to be intended for media applications. I explored a few JavaFX designs, putting Swing elements (which I knew how to code) into a JavaFX application template (which I found online). This turned out to be complicated, as explained below. I then decided to write the entire app from scratch with Swing.

Figure A (JavaFX)

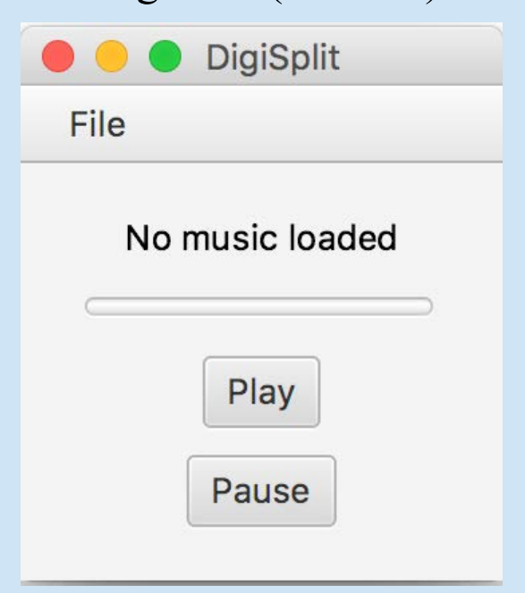
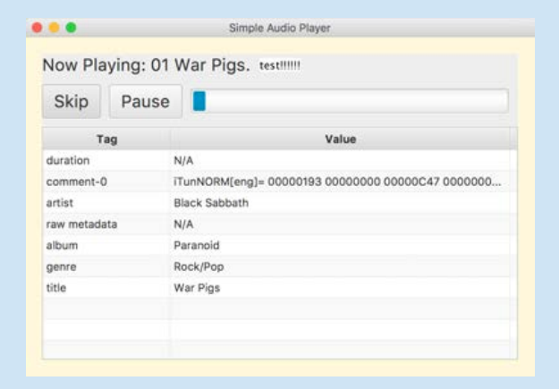
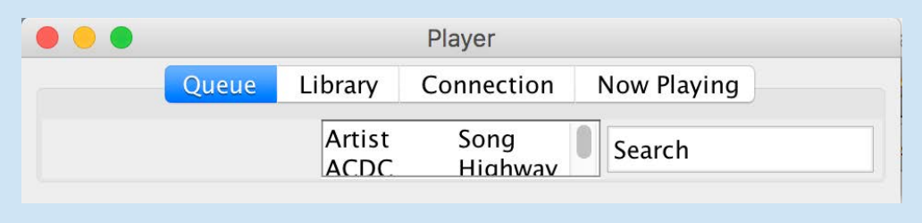


Figure B (JavaFX)



Tag	Value
duration	N/A
comment-0	iTunNORM[eng]= 00000193 00000000 00000C47 0000000...
artist	Black Sabbath
raw metadata	N/A
album	Paranoid
genre	Rock/Pop
title	War Pigs

Figure C (Swing)



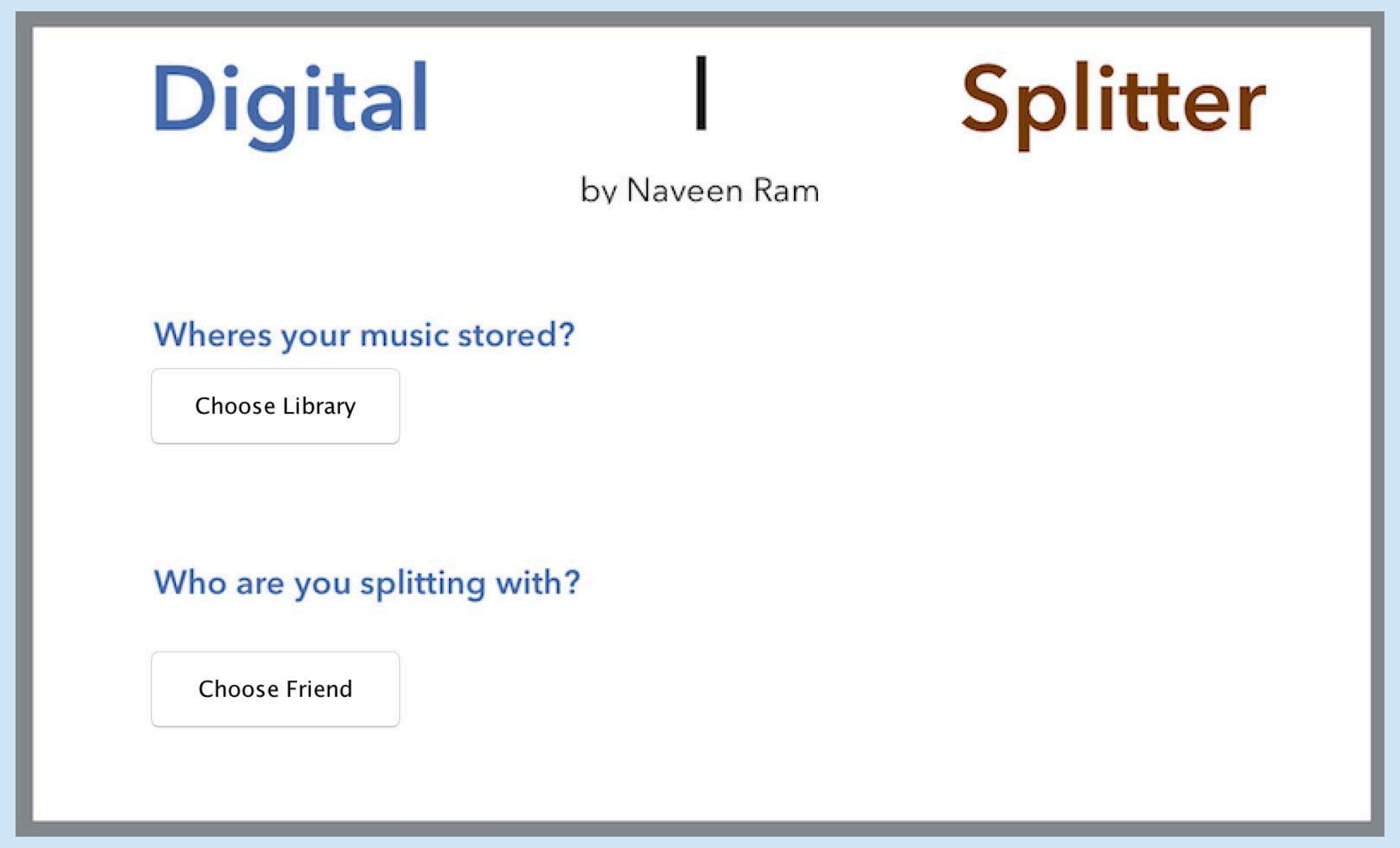
I started by loading and playing music from a file. I experimented with different Java functions, and found a good JavaFX starter on the internet, from which I built the app shown in **Figure A.** Once I finished building the interface to choose a song, pause, play, and show a progress bar, I started exploring how to make a queue. I wanted users on both devices to be able to scroll through the library, and pick which songs would play next. I found a JavaFX snippet which performed a similar function, and tried to build my app around that (Figure B). This is when I started running into problems with JavaFX. I didn't have time to learn all the intricacies of the library, so I tried to place a Swing node directly into the JavaFX app. That is the string ("test!!!") in the middle of Figure B. Eventually, I decided to build the app from scratch with Swing so that I could understand all the parts. By not using templates, I would also learn how to construct apps from the ground up. The result is still a work in progress, but Figure C shows part of what I have so far.

RESULTS

Interface

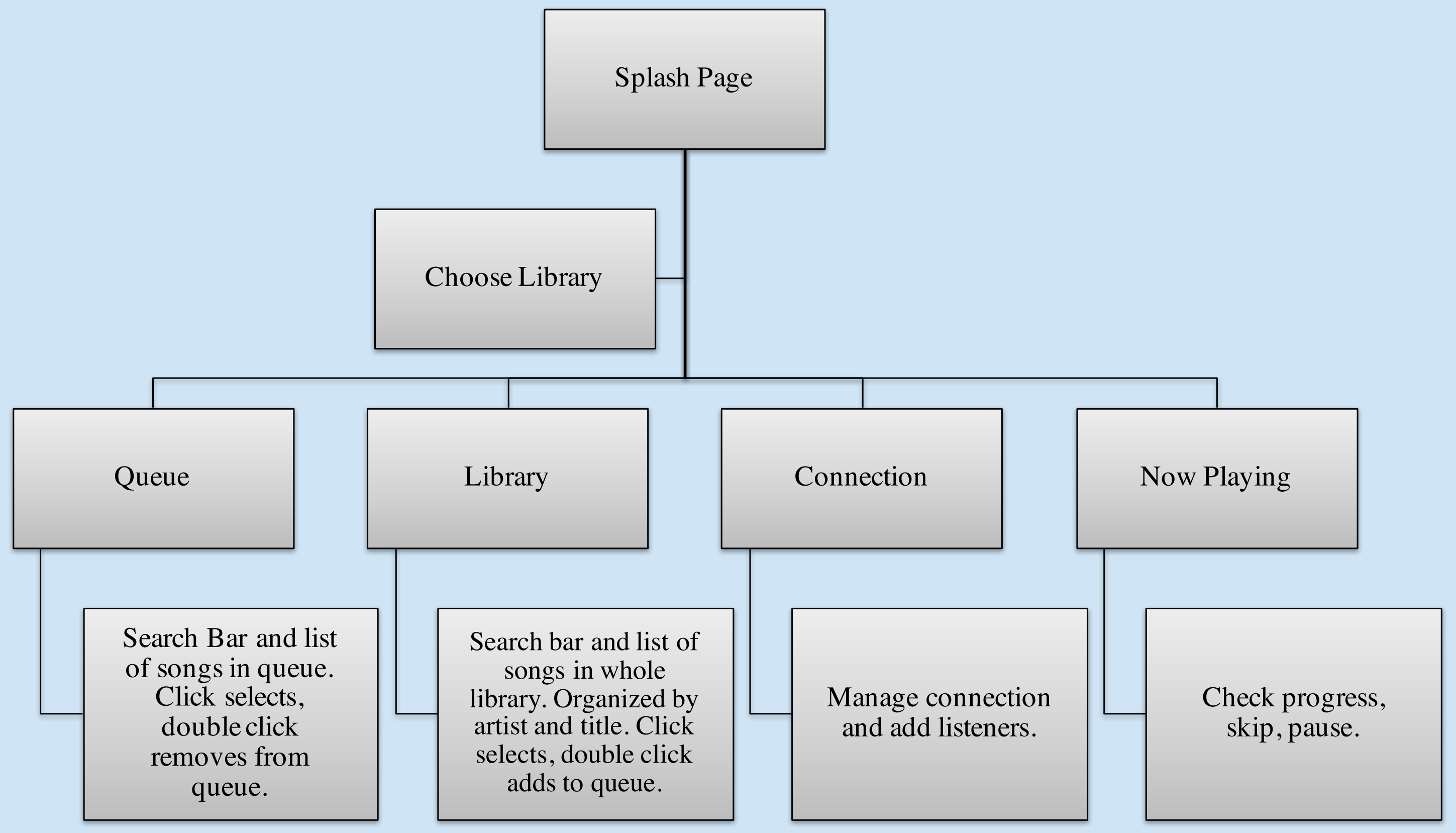
Initialization:

The first thing I wanted to enable the user to do when they opened the app was to select their library: the folder which contains the MP3s they wished to play. I wanted this to happen in a popup window so that once they were done, the popup would disappear, and they could operate the app like iTunes, Spotify, or any other music app. To accomplish this, I designed this splash page:



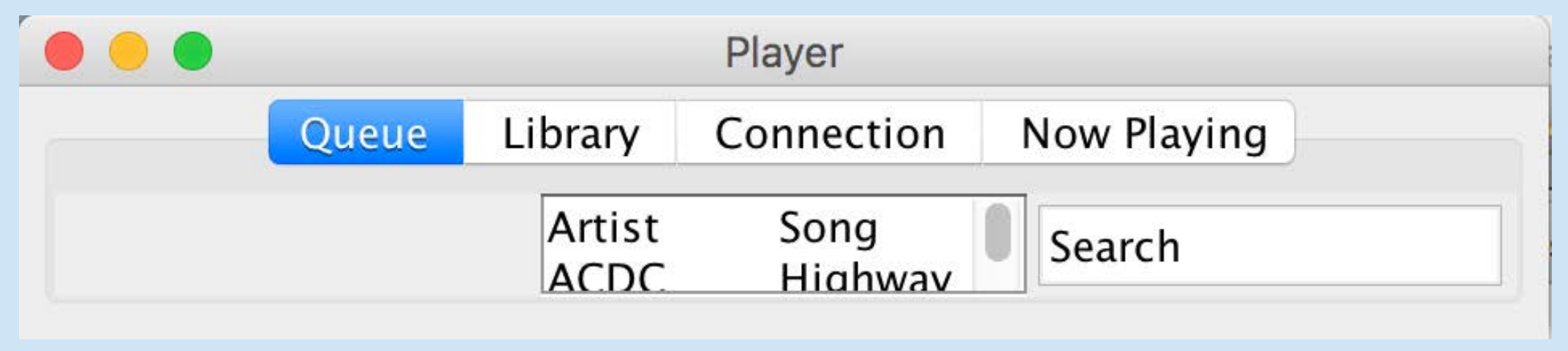
Main Application:

I wanted to design the main application to be intuitive and easy to use. I started by diagramming what users could do from each tab.


Splash Page
Choose Library
Queue
Library
Connection
Now Playing
Search Bar and list of songs in queue. Click selects, double click removes from queue.
Search bar and list of songs in whole library. Organized by artist and title. Click selects, double click adds to queue.
Manage connection and add listeners.
Check progress, skip, pause.

The app works as follows. One person connects to another's computer during the "init" phase, and they both see their libraries in the "library" tab. They can then scroll through and add songs from the library tab into the queue, which will play synchronously through both computers.

Figure C (enlarged)



Currently the app looks like Figure C. The Connection button doesn't yet work, but I have run a successful test of my approach, sending a song through a `server` class and playing it synchronously through a `client` class. All I have left to do is implement those classes in the current code,.

NEXT STEPS

Its which I

Finishing Up

I'm proud of the progress I've made as a novice coder, but my app still some work to complete. All my code is finished, but some of it is in older versions of the app. I need to copy the chunks into the final version of my app and reconfigure them to match the new system. Some classes just need to be connected; for example, I have successfully written `server` and `client` classes to play songs across devices, but they need to be integrated into the main class where the user selects what to play. After connecting these last few snippets of code, I should have a complete project.

Expansion & Evaluation

After completing the application, I plan to do some user testing, and make the app compatible with more operating systems. I know that I would use a digital splitter application, and I suspect it would be useful to other people with the same problems as me, but to verify it is a useful product I need to test it with actual users, starting with students at my high school.

If testing is successful, I want to expand the reach of the app, making it available on Android and iOS. I also want to allow users to split music from Spotify, SoundHound, or YouTube. I did a lot of research on getting audio data from streaming services towards the beginning of my project, but I didn't find a feasible way to do this. With enough time, I could write a script that determines which source the audio is coming from, and performs the appropriate methods to get and split the audio.

CONCLUSION

I set out to build an application which would allow small groups of people to listen to music together but privately, while not disturbing the rest of the people in a workspace. I wanted to see whether I could solve this problem which I faced everyday as I worked in computer science class, and design a simple solution to it. Although I was striving for simplicity, I encountered many complex problems along the way. I went through three different app designs, two libraries, twenty classes, and countless compilation errors.

Overall, *Digital Splitter* offered a unique and original solution to a common problem. It has already successfully connected two devices, and with some minor adjustments, will soon be a complete app read for testing. More importantly, it allowed me to got through the whole process of building an application from start to finish, from idea to design to implementation to product. I'm proud of the progress I made, from someone who barely knew Java to a expert on JavaX and Swing. I'm excited to continue working on *Digital Splitter* in the future, and eventually get feedback from my peers in order to refine my design. Hopefully, I will be able to sit in AP Computer Science next year, working on a group project while listening to music with an application which I have created myself.

ACKNOWLEDGEMENTS

I'd like to thank my mentor Gianna Giancarlo for helping me figure out what I wanted to do. I'd also like to thank Skyyler Seijko for helping me out with difficult coding issues.

I used a lot of online resources throughout the project, including StackOverflow answers, Oracle documentation, and GitHub projects.