My (Musical) Life

Design Document

Team Number: sddec20-13
Adviser & Client: Dr. Henry Duwe

Team Members:
Christian Hernandez - Project Manager
Chaz Clark - iOS Developer
Daksh Goel - Backend Developer
Vignesh Krishnan - Frontend Developer
Vatsal Bhatt - Backend Developer

Team Email: sddec20-13@iastate.edu
Team Website: http://sddec20-13.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- Development Standards
  - Commented Code
  - Quality
  - Efficiency
  - Apple Developer Standards
  - Waterfall Design
- Practices
  - Test code regularly
  - Agile Development
- Engineering Standards
  - Quality
  - Performance
  - Safety

Summary of Requirements

- **Functional requirements**
  - User Data (from their mobile device)
    - Location
    - Weather
    - Schedule
  - A music Streaming service account
  - Music Recommendations
  - Mapping Sensor Inputs to Songs/Playlists
  - Volume Control

- **Non-functional requirements**
  - Security (SSL, TLS, WPA2)
    - Account logins
    - Location information
    - Calendar data
    - Music preferences
  - AWS Security
    - Database
    - Lambda data
  - Response time/performance
    - Crash Rate: 1-2%
    - API Latency: 1 sec
    - End-to-end app latency: <3 sec
- **Economical requirements**
  - Spotify Premium Subscription

- **Environmental requirements**
  - Network reception in user's mobile device
  - iOS device (iPhone)

- **Apple Design Guideline Requirements**
  - Consistency
  - Feedback
  - Direct manipulation
  - User control

Applicable Courses from Iowa State University Curriculum

- S E 185 - Problem Solving in Software Engineering
- CPR E 185 - Introduction to Computer Engineering and Problem Solving I
- COM S 227 - Object-Oriented Programming
- COM S 228 - Introduction to Data Structures
- COM S 309 - Software Development Practices
- CPR E 310 - Theoretical Foundations of Computer Engineering
- COM S 311 - Introduction to the Design and Analysis of Algorithms
- S E 319 - Construction of User Interfaces
- S E 329 - Software Project Management
- S E 339 - Software Architecture and Design
- COM S 363 - Introduction to Database Management Systems
- ENGL 314 - Technical Communication

New Skills/Knowledge acquired that was not taught in courses

- Swift
- iOS Development
- Machine Learning
- Using Spotify's API
- Amazon Web Services

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

We would like to first thank Dr. Henry Duwe for meeting with us weekly and providing us with guidance and advice as we develop our senior design project. Dr. Duwe has done an amazing job in regards to helping us set up this project by giving us small assignments to complete each week for him. We would also like to thank the Electronics Technology Group for providing us with a website and a Git project. Lastly, we would like to thank the TAs and the professors (Dr. Lotfi Ben-Othmane and Dr. Daji Qiao) for their guidance and help thus far.

1.2 Problem and Project Statement

Have you ever been in a sad mood and listened to sad music despite it not helping and making you even sadder? Do you typically play high-tempo music when heading to the gym and while working out at the gym? Do you listen to softer, calmer music as you study for your next exam at the library? If you answered yes to some of the questions, *My (Musical) Life* app will be perfect for you! For people who love music, this app will be great to use on the daily.

Our app will use data from multiple different, possible sources (location, calendar, weather, schedule, time of day, etc.) to determine which song is the best to pipe directly into your ears. The app will require little user input, and the music suggestions will improve as the user continues to use the app. Overall, as long as the app is open on your phone, the app will continue to play music based on the different sources listed above.

1.3 Operational Environment

The end product of our project is an iOS mobile application, as demanded by the client. *My (Musical) Life* will be able to be installed and used by anyone owning Apple's mobile device, namely, iPhone. Upon installation and registration, our app will require some permissions from the user including access to their mobile device's location and some user data. Our app is only supported by iPhone's Operating System and will be not available for use in mobile devices running Android.

1.4 Requirements

The requirements of our project are as follows:

- **Functional requirements**
  - User Data (from their mobile device)
    - Location
    - Weather
    - Schedule
  - A Spotify account

    - Music Recommendations
    - Mapping Sensor Inputs to Songs/Playlists
    - Volume Control

- **Non-functional requirements**
    - Security (SSL, TLS, WPA2)
        - Account logins
        - Location information
        - Calendar data
        - Music preferences
    - AWS Security
        - Database
        - Lambda data
    - Response time/performance
        - Crash Rate: 1-2%
        - API Latency: 1 sec
        - End-to-end app latency: <3 sec

- **Economical requirements**
    - Spotify Premium Subscription
    - Apple Developer Account

- **Environmental requirements**
    - Network reception in user's mobile device
    - iOS device (iPhone)

- **Apple Design Guideline Requirements**
    - Consistency
    - Feedback
    - Direct manipulation
    - User control

1.5 Intended Users and Uses

An intended user for our iOS mobile application, *My (Musical) Life*, will be someone who loves to listen to music and wants to listen to their favorite music with minimal to no user input at specific times and during specific events of their day. Our app will create personalized playlists for each user depending on their mood, location, and schedule. Other factors including time of the day, and weather will also play a role in creating these playlists. The main purpose of the app is to start playing music in a user's mobile device without their input, during specific times of the day when the user would possibly be wanting to listen to music. An example use case will be a student wanting to listen to soft and calm music while studying in the library. Our app will determine that the user i.e., the student is in the library through the location of their mobile

device and will start playing soft and calm music in their mobile device, while having five other playlist recommendations in case the user did not like the music that is being played currently.

Pictured below is a venn diagram consisting of all the use cases for our project.

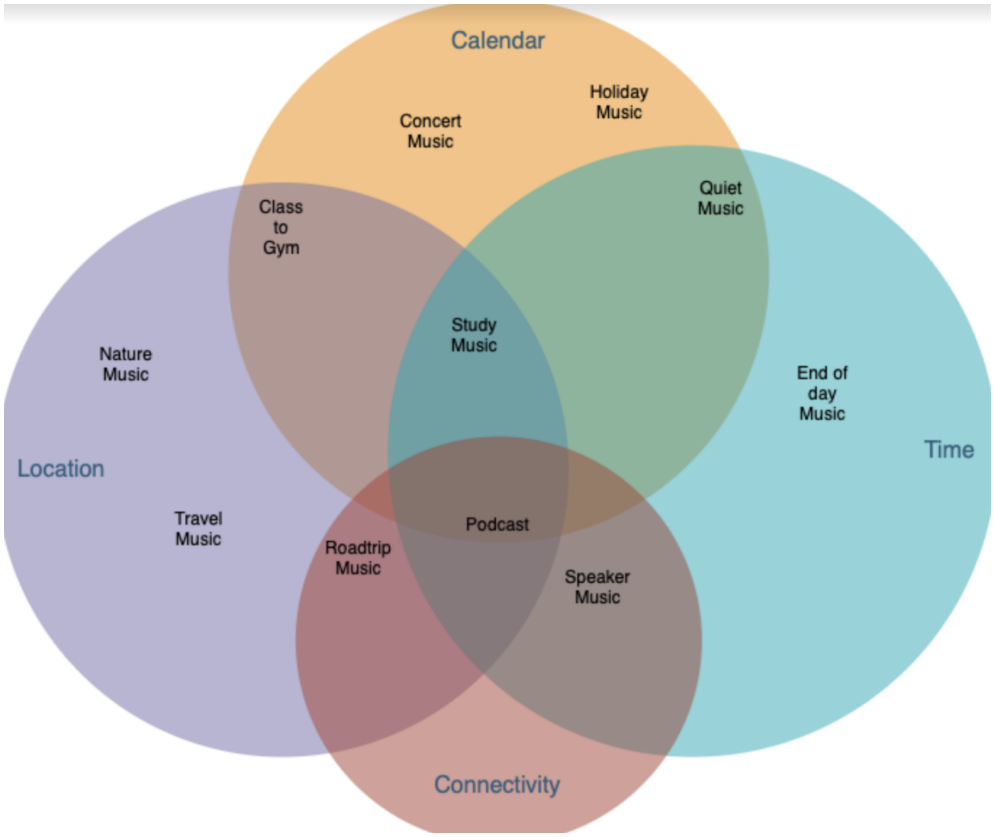

Figure 1: Use Case Venn Diagram

1.6 Assumptions and Limitations

Assumptions:

1. The user will have an existing account with Spotify music streaming service or will be willing to create one for the use of the application.
2. The user will have an existing Spotify library to choose songs from, otherwise songs may be chosen from a random Spotify playlist for specific mood.
3. The user will have an existing Google account or will be willing to create one for the use of the application.
4. The user is comfortable with the private data and permissions that the application requires to provide the most intelligent song selections. Ex. location, calendar, bluetooth.

Limitations:

1. The application will only be available on iOS devices as required by the client.
2. Users with streaming services other than Spotify or local music storage will not be able to integrate their music library.
3. Users without a Google account will not be able to use features that use personal data for selecting songs.
4. There will be minimal user input and songs will only be able to be played automatically with a skip feature, not chosen specifically.
5. The user will need to allow device location permissions at all times to receive location based song selections in real time.
6. The user will need to allow bluetooth device connection permissions at all times to receive device based song selections in real time.

1.7 Expected End Product and Deliverables

The *My (Musical) Life* iOS application will be the final deliverable to our client and it will be commercialized on the Apple App Store.

Description:

My (Musical) Life is a music streaming application that changes your music based on what you do every day. This is a great app for listeners who love having music playing throughout the day, almost like a soundtrack for their life! Finding the right song, playlist, or genre for a particular activity can be difficult and time consuming. *My (Musical) Life* can take the pain out of choosing music by predicting and playing the songs you like. Powered by Spotify, *My (Musical) Life* will select your favorite songs that fit your daily activities. Whether it be working out at the gym, working in your office, studying in the library, or taking a road trip, never worry about changing the song again. *My (Musical) Life* will recommend and play music that best fits your daily routine. Using your location, calendar, and Bluetooth connectivity data, it will select the genre of music that best matches your mood for a specific activity. As you provide feedback on selections made by *My (Musical) Life*, it will build a personalized music profile that will only play the songs that you love.

2. Specifications and Analysis

2.1 Proposed Approach

The proposed approach to our app is to develop an iOS application for our users to interact with. The iOS application will then communicate to various third-party APIs. In order to play music, our app will use Spotify's API and stream music from there.

The application will be mapping users' sensor data to songs and playlist. The mapping of this data will be organized using a bin model approach where users will have bins that are built upon the user's sensor data and will contain songs associated to that data. To see if a bin already exists for the user's current situation we will be designing AWS lambda functions to search for appropriate bins, if none exist a new one will be created.

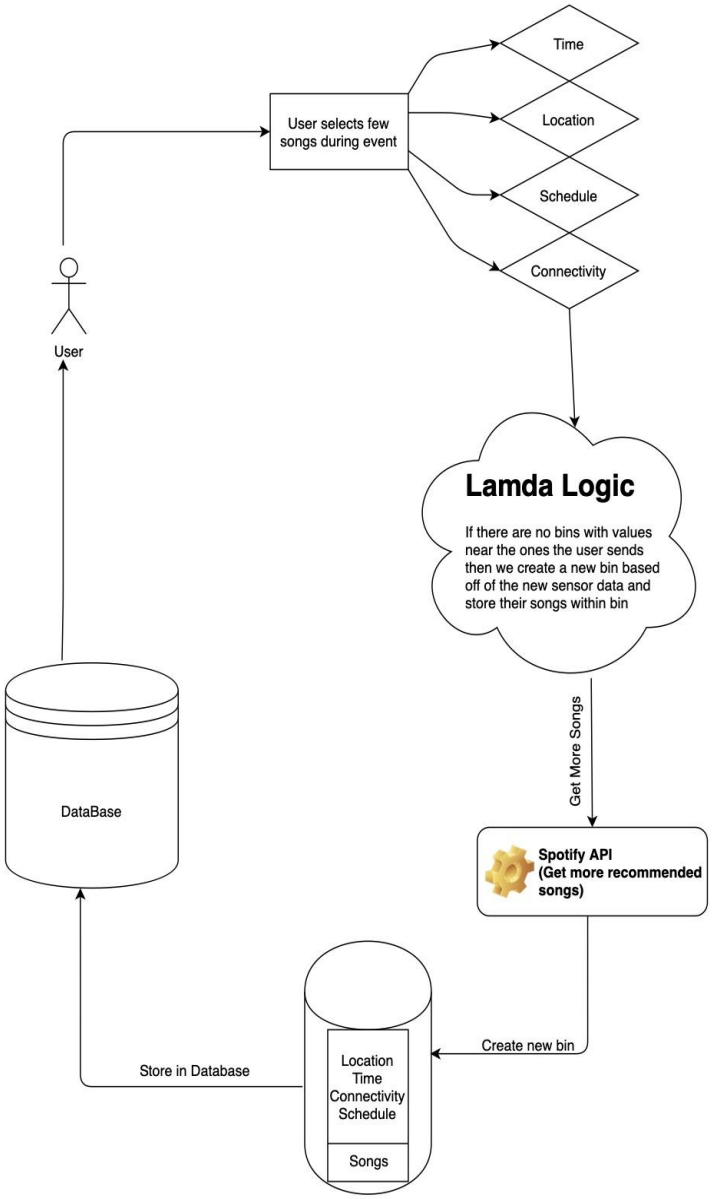

Figure 2: Bin Creation Diagram

For our data storage we plan to use a hybrid of on-device storage and AWS. The on-device storage will store user-login, and settings. Our AWS backend will build a profile of the type of music the users listen to, which will help us recommend music. We will also use the backend to recommend the next song(s) to the user. This will either be ML or a predictive algorithm (as stated above).

So far, we have implemented a POC application that uses the spotify API to play music and search for music. Additionally, we have all been learning Swift as most of us are brand-new to iOS development. We have been spending most of our time doing research on several APIs, data storage, and ways that we can predict mood.

2.1.1 Sensor Inputs

Pictured below is a list of sensor inputs and the results for each of these. More specifically, these are APIs or frameworks the team will be using to gather the inputs. Using these APIs or frameworks we will be able to gather the result needed. For example, the app will be able to use the Core Location API in order to obtain the location, movement, etc. of the user.

Inputs (API/Framework)	**Result**
Core Location	Location, Movement
Healthkit/Core Motion	Steps, Heart Rate, Movement, Date of Birth
IO Kit	Bluetooth Connectivity
Core Motion	Start Date, End Date, Number of steps, Distance, etc.
IOBluetooth UI	Bluetooth Connectivity
CarPlay	Bluetooth Connectivity to car
Asset Playback	Control Volume
Google API	Schedule/Calendar, Sign-In
Spotify	Return Songs, connection status, etc.
Open Weather API	Temperature, Wind Speed, Cloudiness Percentage, etc.
Dates and Times	Date and Time

Figure 3: Sensor Input Table

2.1.2 Database Design

Below is a diagram of our database design. As you can see, we will use the User table to store user information. Since we are using Google authentication we won't need to store passwords. We1 will store hashed email addresses as well as the Age of the user and the region they reside in. Next, we have the UserSettings table which is where we store the values for the personalized UserSettings such as if they would like to enable or disable

volume control, location services, etc. Each User will have many bins. To model this one-to many relationship we have the Bin table, each Bin will have a user associated with it. The bin will be defined by the various sensor inputs that we have discussed so we have a field for each of these inputs. Similar to the user-bin relationship, each bin will have many songs, however different bins may share the same song, therefore this is a many-to-many relationship. To represent this, we have created a Songs table, where each SongName will be associated with a bin. The primary key here ensures no duplicate values.

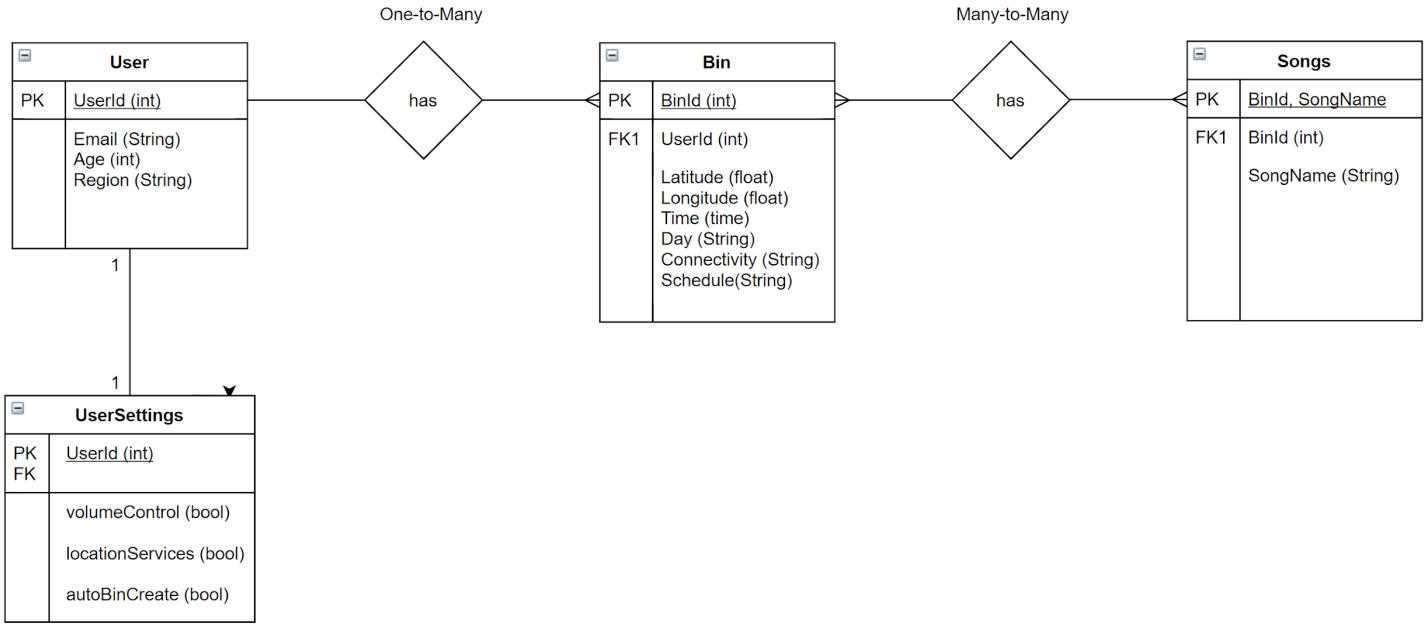

Figure 4: Database Design

2.1.3 Onboarding Diagram

When a user first logins in to use or account, we will take them through an initial process to get their recommendation system setup. This diagram shows our onboarding process for each user. First, they will login to Spotify or Google and simply choose a song from their own library or any song on Spotify they'd like to listen to at that moment. From that particular song, their feature vector will be derived from their current sensor inputs. The combination of these will create their first bin with this song in it and feature vector to describe the bin. We will then give similar song recommendations from their library or Spotify based on the song metadata. The user will give feedback on these

recommendations which will help us decide whether to keep adding these songs to the same bin or create new bins if the user wants to differentiate any particular song.

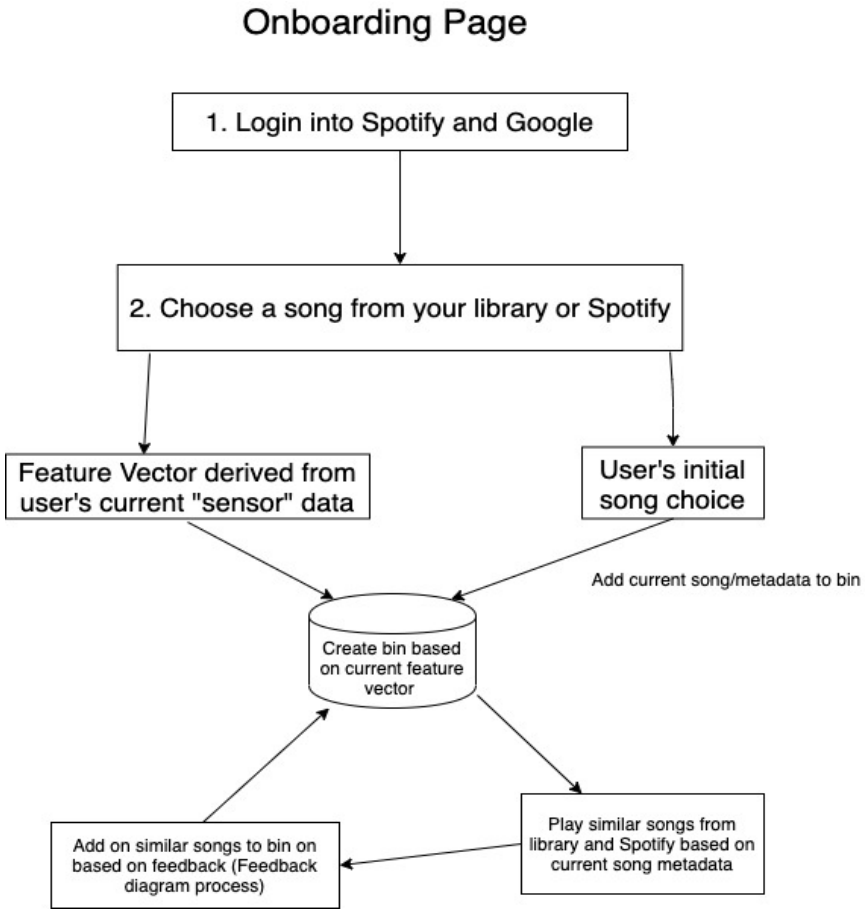

Figure 5: Onboarding Diagram

2.1.4 Feature Vector Workflow

Once our user is onboarded, in order to choose songs accurately the next time they use the application, we will use this feature vector workflow method. First, we will receive all of the sensor inputs in their respective format and send it to our backend lambda function. This lambda function will uniquely check whether this feature in the vector corresponds to any specific existing feature in another bin. Depending on the correlation, we will give a priority weight to each feature that was checked and these weights will be passed to another lambda which will decide which feature is most relevant based on the weighting. The highest correlating feature will decide which bin we choose songs from at that moment.

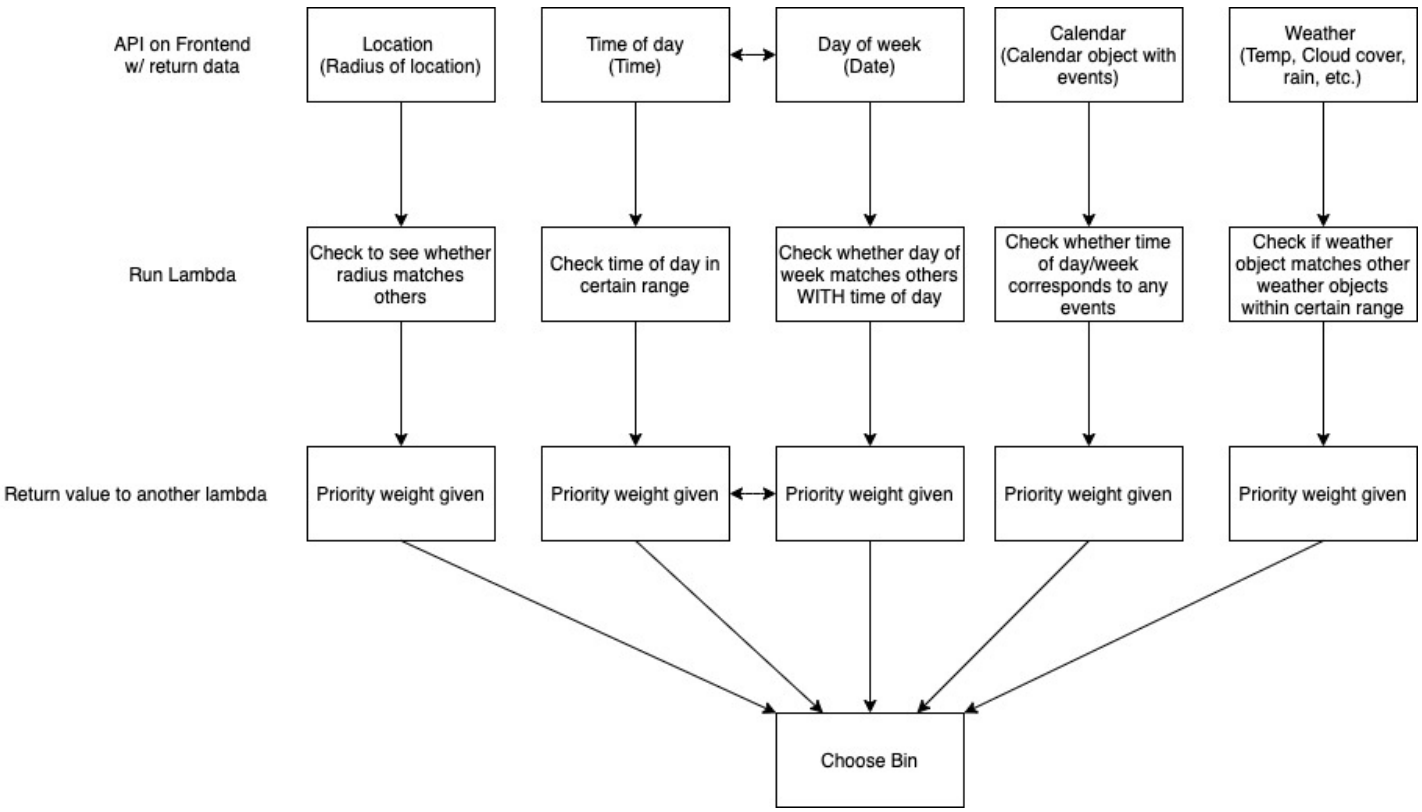

Figure 6: Feature Vector Workflow

2.1.5 Feedback Diagram

Another important part of our recommendation system is the feedback process. The diagram below highlights our feedback process flow. First, our feedback page will ask the user if they like the current song first. If they like the song then it will keep playing and be added to the bin if it doesn't already exist there. If they respond no, we will change the song to another one in that bin and increase a counter. If the user responds no to 3 songs in a row, we know that type of music doesn't belong in that bin and they need a change. We will then run our backend function to find the next best bin and play a song from that bin. The entire process continues cyclically.

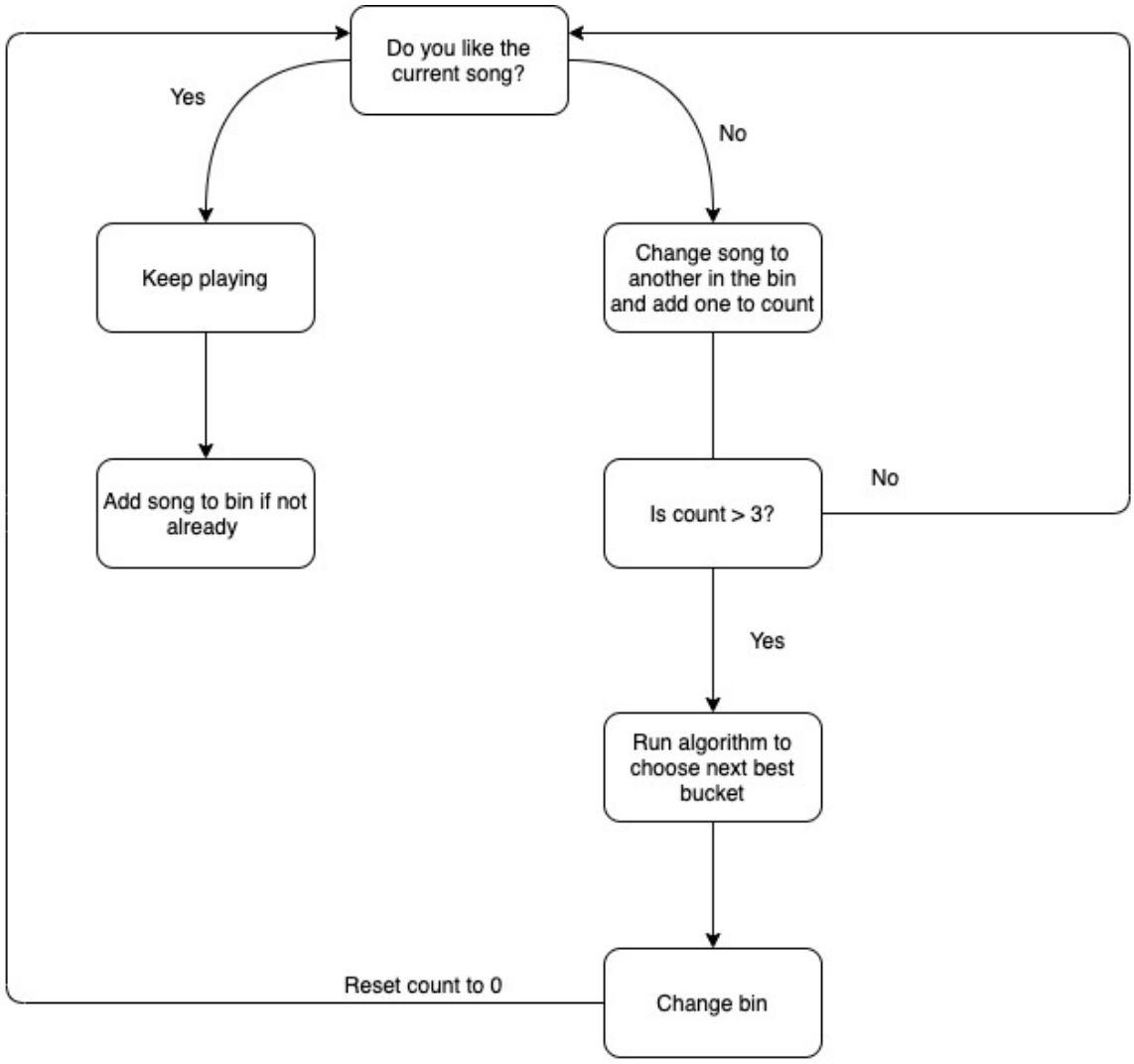

Figure 7: Feedback Diagram

2.2 Design Analysis

There are multiple modules that our application will need to function. We start with the User. The User will be responsible for providing feedback on our music predictions as well as inputting their music preferences. The application will then take in the user feedback to send to our processing co1mponents as well as storing local api logins. The application will then send the user's sensor data to our AWS Cloud functions to predict the next song that should be played for the user. Once that prediction is complete it will then call Spotify API to request a song and load that into the queue to be played. We will store the users liked and disliked prediction in our AWS database

2.3 Development Process

To satisfy the requirements of our client and the senior design course we are currently following a Waterfall method. We have completed the Planning & Requirements stage and currently in the System & Software development design stage. This helps the team gain a high-level perspective on the project. Once software development comes around, we will begin to adopt an agile approach towards the project. This will allow the team to implement continuous client feedback into our application.

2.4 Conceptual Sketch

Our System-level diagram contains multiple modules that will deliver the necessary requirements. *My (Musical) Life* will consist of 5 modules. Local Storage, AWS Database, AWS Cloud Processing, Third-party API, and User Feedback. The application is designed to take minimal user input and provide a nice automated experience. To do so there will be a minimal user interface. The user will input password and login credentials that will be stored locally on the device. This data will be used to access their third-party accounts such as their google calendars and schedule. The local storage data will also be used to deliver audio streaming service. Our largest requirement is to predict what type of sounds the user wants to listen to and deliver it. Part of this processing will be done on Amazon web services. We are storing detailed user models in the database to build a personalized profile for each user. This will serve as a key factor when we try to predict the user's sounds. To minimize the amount of on device processing we will utilize AWS cloud functions to perform more of the heavy-duty computation in prediction.

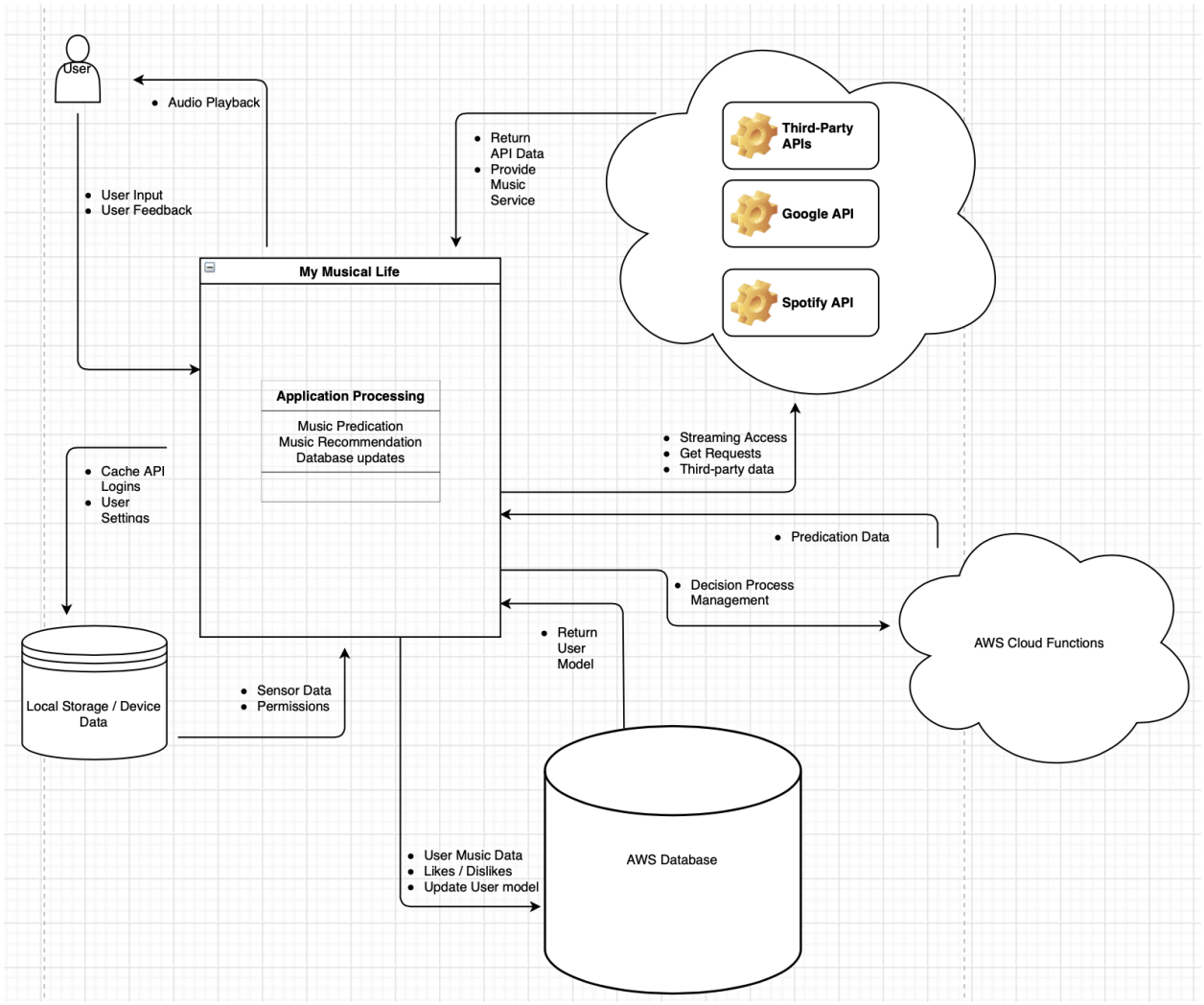

Figure 8: Conceptual Sketch

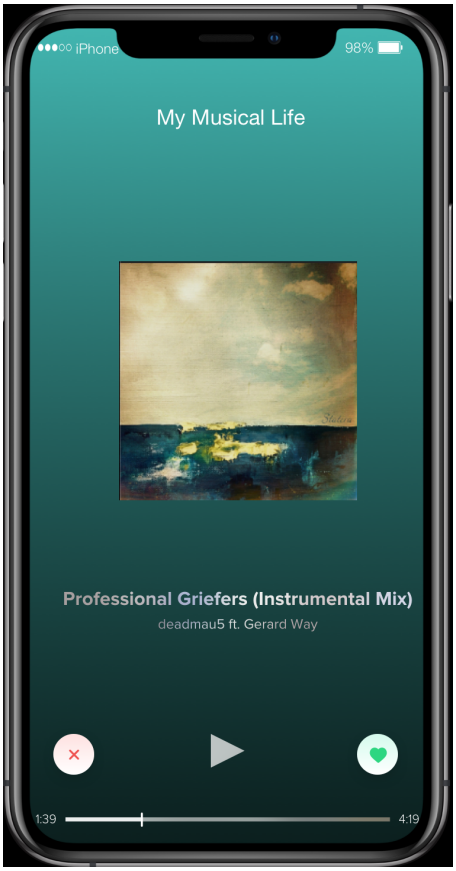

Figure 9: iOS Mockup

Our application will consist of a simple UI to encourage an autonomous feel to the experience. We will gather feedback from our users about their listening experience through the like and dislike buttons.

3. Statement of Work

3.1 Previous Work And Literature

As for previous work that has been done, there are apps that have been created that do something similar to the app we plan on making. However, our app will be different than the three that we will mention.

First, there is an app by the name of *MusicFit.* The goal of this app is to generate music based on the user's body movements through the iPhone's sensors [3]. Ultimately, this app will generate music based on the user's change of pace while working out. The app will only generate music based on four genres (techno, electro, idm, chill) [3]. *My (Musical) Life* will be different because our app will generate music based on a couple of factors (calendar, time of day, weather, connectivity, location, etc.). Our app will predict what type of music the user would like to listen to based on those factors. Additionally, our app will not be limited to only four genres.

Next, another similar application that exists is *Musicovery.* For this app, "recommendations of tracks, artists, genres, and playlists are personalized in real time to each listener, according to his music preferences, listening behavior, and listening history" [2]. Again, this is different from *My (Musical) Life* since our app will generate music based on different factors. Furthermore, *Musicovery* seemed to have been an app at some point, but now it is only a website. We could not find it in the app store.

Lastly, the last application that seemed to be similar to our project is *Songza*. Songza would use date, time, and past listening history to generate "playlists based on predictions about the user's mood and/or activity at the time" [1]. However, this app would allow the user to also search for playlists based on genres, mood, and artists. This is a feature that *My (Musical) Life* will not have. Adding on, the app was shut down, but it was integrated into Google Play Music. The difference between *My (Musical) Life* and *Songza* (now integrated in Google Play Music) is the fact that *My (Musical) Life* is planned to be an app with minimal to no user input. The app is planned to be extremely simple to use for the user. Furthermore, Songza had the ability to ask the user what he/she is doing [4]. *My (Musical) Life* will be different because, with the help of Google APIs, *My (Musical) Life* will have access to the user's calendar. The app will know what the user is doing at a specific time of the day. Overall, *Songza* seems to have the most similar idea to *My (Musical) Life,* but with the plan of making our app play music based on location, connectivity, weather, movement, number of steps, etc., *My (Musical) Life* will be different than Songza since Songza does not use more factors other than date, time, and past listening history.

3.2 Technology Considerations

- **Spotify API**
  - **Strengths**: Spotify has tutorials and many webpages with how to integrate their API into an app.
  - **Weaknesses:** Each member of the team has not had much experience using Spotify's API. Therefore, there will be a learning curve.
- **AWS**

    - **Strengths:** AWS seems to be simple to use and integrated into an app. There are many tutorials that can be followed to use AWS.
    - **Weaknesses:** Some members of the team do not have experience with using AWS
- **Google API**
    - **Strengths:** There are many tutorials one can follow with Google API. Additionally, this API seems to be relatively simple to integrate into an app.
    - **Weaknesses:** Experience with using Google API.
- **iOS**
    - **Strengths:** There are many nice tools to use while developing iOS applications. Apple provides some tutorials for iOS development as well.
    - **Weaknesses:** Apple has many restrictions on iOS apps. There are many guidelines we must follow. Also, one out of the five team members has experience with iOS development.

3.3 Task Decomposition

Below are the current tasks:

- Research/Learning APIs, Swift User Interfaces, Spotify API, AlamoFire, AWS, etc.
- Setting up a database to store data
- Setting up front-end
- Connect app to Spotify and Google APIs

3.4 Possible Risks And Risk Management

As we develop the app, there will be multiple issues that we will run into. The first issue is the user not having a Spotify Premium account. This may make it difficult to access all types of music for the user. Although we may not be able to provide the same experience for non-premium users, we will be able to work with the Spotify API in order to still access songs and playlists to recommend to our users.

Next, trying to figure out an algorithm to select which bin needs to be selected or if a bin needs to be created will be a challenge. This algorithm can be somewhat subjective which adds to the challenge. However, even if we are not able to come up with a perfect solution, the rest of the application will still have its intended functionality as we will still be able to recommend songs for the users.

Developing an iOS app will create some roadblocks for the team. Apple has many restrictions and guidelines one must follow. Creating an app in iOS rather than Android is a bit more difficult. Only one out of the five team members is familiar with iOS development. Thus, there will be a learning curve for the four other team members.

Obtaining user data, such as accessing the user's calendar, location, etc. may be a difficult task for the team. There may be some restrictions on the amount of data we can receive from each user.

3.5 Project Proposed Milestones and Evaluation Criteria

Overall, the most important milestone for the team is to have a fully functional app by December 2020. Another key milestone is, by the end of May 2020, the team plans on having parts of the app completed. That includes the setup of AWS, the setup of the front-end and back-end, setup of the database, and our app should be able to connect to Spotify and Google's APIs.

As for next semester, the team plans on beginning to collect data during the month of August. Additionally, the team will develop front-end calls to Spotify's API in order to play songs. Constructing the front-end UI will also be completed in the beginning of the semester. Furthermore, the Fall semester will include creating lambda functions and setting up lambda function triggers. We will also develop front-end calls to the lambda functions. Throughout the Fall semester, the team plans on going through an Agile development cycle. We will be in the process of developing, testing, and receiving feedback from Dr. Duwe.

Some of the milestones we have completed include research into tools, technologies, APIs, frameworks, etc needed for the development of our app. Additionally, we completed the planning and designing phase of our app. This included the development of multiple diagrams and sketches to help us understand how to go about the design of our app, and also, how everything connects.

Tests will include many types of tests. Currently, we plan on using Unit Testing, System Testing, Regression Testing, Stress Testing, etc. Additionally, the evaluation criteria will involve receiving input from our advisor/client, Dr. Duwe.

3.6 Project Tracking Procedures

The group will use GitLab to track progress throughout the course of this and next semester. GitLab has a section called "Issues" where one can add issues to an "Issues Board." Thus, this is the project management tool the team will use throughout the duration of the project.

3.7 Expected Results and Validation

The desired outcome is to have a fully functional app by December 2020. *My (Musical) Life* app will be in the app store free for users to use. The application will predict the user's mood based on many factors and generate music based on that mood. Additionally, the application will be bug-free.

We will confirm the application will work at a high level by running an extensive list of tests (System Tests, Unit Tests, Stress Tests, Regression Tests, etc). Additionally, it is important the app satisfies our client (Dr. Duwe) and other potential users.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

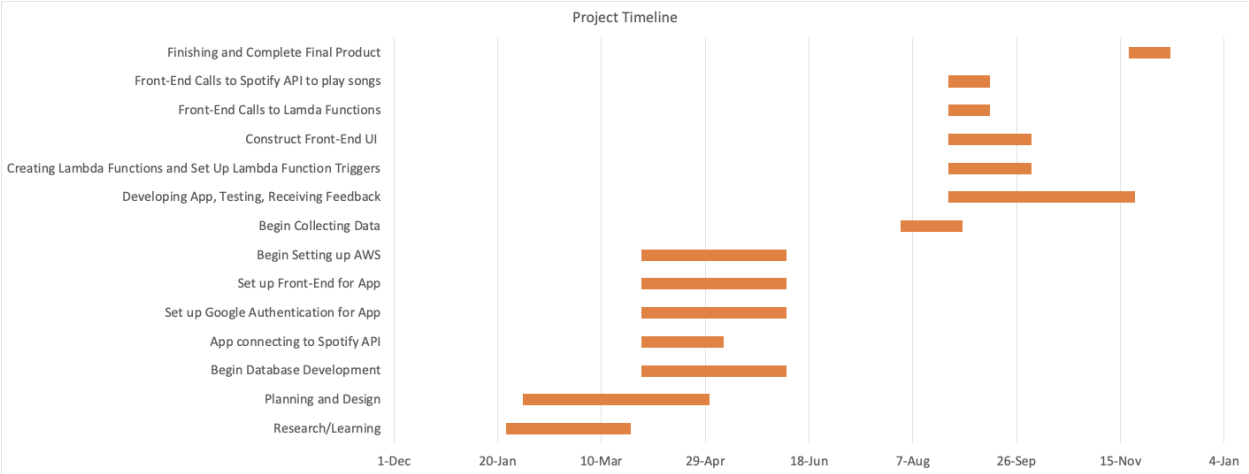

Figure 10: Gantt Chart of Project Timeline

The Gantt chart above shows some of the short term and long term goals that we have for this project. We are now slowly moving out of the phase that primarily consists of design and development. However, these are some core pieces we must cover in the next couple of weeks. First, we would like to finish setting up AWS for our app. The team is projecting that this will be completed by the end of May. Next, setting up a basic front-end will be completed by the end of May. The app will not consist of many pages, but the initial set up of the front-end may take awhile. Going forward, we need to set up Google authentication for the app. Thus, this will involve having the users logging into his/her Google account via the app. This will also take up until the end of May to complete. Additionally, the app will need to connect to Spotify and use Spotify. Thus, connecting to Spotify's API is something else the team must accomplish. As mentioned, users will need a Spotify account. Setting up the database is another task that the team will continue working on. Setting up the database can be difficult and take a while to accomplish. Thus, this may take 70 days or more to complete. Furthermore, research and learning may take longer as we discover new technologies and software to potentially use during the development process.

We would also like to mention that, by next semester, the team will begin collecting data for the development of the app. The team will use the data for development. The team will go through a cycle of developing, testing, and receiving feedback from our client/adviser throughout the semester. To be more specific, by the beginning of the semester, the team will begin working on front-end calls to Spotify's API in order to play songs. Also, the team will begin making the front-end calls to lambda functions. Both of these tasks will take about 20 days to complete. Adding on, next semester will involve constructing and finishing the front-end UI of the app as well. This will take 40 or more days to complete. Two other tasks the team plans on completing are creating lambda functions and setting up lambda function triggers. This will also take about 40 or more days to complete. By November, the hope is to make finishing touches and continue testing the app. In the end, the app is projected to be completed by December.

4.2 Feasibility Assessment

Overall, the expectation of this project is to be an iOS app that our client (Dr. Duwe) will be able to download and use. By the end of the Fall semester (December 2020), the goal is for the app to be fully functional with regards to requirements. The application should also meet the expectations of our client, Dr. Duwe.

As for challenges the team may run into, the team may run into challenges using new technologies we do not have experience. Thus, we will have to allocate time for learning new technologies. Another challenge is time. We are given a year to create this app; therefore, given our class schedules and other obligations, this will be a challenge.

4.3 Personnel Effort Requirements

Task	Description	Projected Effort
Research/Learning of APIs, Swift User Interfaces, Spotify API, AlamoFire, AWS, Machine Learning, etc.	This task simply includes research and learning any new technologies needed for the project.	This takes quite a bit of effort. Most likely a month is a good timeframe for this task.
Create a database to store data locally and in the cloud	We need to set up a database to store information locally and in the cloud. Currently, we plan on storing data in both places.	Setting up databases can be difficult and confusing. Ultimately, a month will be enough time to set up a fully database.
App(Frontend) connects to APIs (Google API, Spotify API, Third-Party APIs)	After the research/learning phase, we would like to begin to connect to these APIs. That is, we need to integrate some of these APIs into our app and learn how to use them.	This should only take about two weeks to accomplish. With the right training and tutorials, this can be done in a short time frame.
Use Lambda functions in AWS to update DB	Write push, pull function to update and read our database	This should take about 2-3 weeks depending on how many tables we use.
Connect backend to Spotify API	Backend needs to query /recommendations endpoint in order to get the next song(s)	This should take about one week. However, building the logic for recommendation may take about a month
Implement Google login	Frontend will use google to authenticate users	This should take about 2 weeks.

4.4 Other Resource Requirements

Our Project has a few resource requirements in order to function.

Device: Since we have chosen to utilize the newest Swift frameworks available our app needs to be running on an Apple device that is running iOS 13 or higher.

Accounts: A Spotify account will be required to use the app in order to enable streaming data to the user.

Cellular Data or Wifi: A constant connection will need to be enabled in order for the program to function.

4.5 Financial Requirements

Currently, the only requirement is a Spotify Premium subscription.

5. Testing and Implementation

5.1 Interface Specifications

The interface should respond to user requests and obtain user data. We will be using Apple's built-in UI testing to ensure UIkit components are responsive

5.2 Hardware and software

- Testflight
    - This allows us to distribute our ios builds among our testers without submitting a build to the app store
- PostMan
    - This will allow us to send and retrieve data to our custom-built APIs providing us a detailed view of our JSON packets

5.3 Functional Testing

To ensure the application is fully functional we will be performing various forms of integration testing.

XCTest & XCUITest - Apples built in testing software suite. This is what we will be using to test the iOS portion of the application.

- API Integration
    - Third party APIs will be a big component in our application to meet our deliverables. Testing of these APIs will be done utilizing PostMan and Black box testing
- Music Integration
    - Volume and audio testing will take place to ensure that music is delivered consistently and on demand
- Sign up / Sign In
    - Authentication testing will ensure users will have profiles with saved personalized models.
- Decision Making Algorithm
    - To test the decision making algorithm we will be conducting user testing on the developers on the team. We will collect data on recommendation satisfaction and record the progress of this satisfaction as time continues

5.4 Non-Functional Testing

Once the app is developed, we will perform some performance, usability, security, and compatibility tests.

- Authentication Testing
- Memory Leak Testing
- UI Testing

5.5 Process

Each team member will be responsible to write their own test cases towards code they push to the project. This will help lower the exposure to bugs and complications that the app will have. We will be using various tools in Xcode and Amazon Web Services.

5.6 Results

We will have a well tested application that will be reliable and secure. It will deliver the functional deliverables as specified. Testing will help ensure that our application will be a great experience.

6. Closing Material

6.1 Conclusion

In conclusion, we are in the beginning stages of developing our app. The planning phase included an abundance of research and learning. Our research consisted of tools and technologies we can potentially use for this app. Currently, the plan is to finish connecting the app to Spotify's API and Google's API. Additionally, we are working on setting up the database and AWS. These tasks will be completed towards the end of May. As we slowly begin developing more for our app, we are working with Dr. Duwe to make sure we are on the right track. Overall, the goal is to create an app that is fully-functional by December 2020. We want our adviser/client (Dr. Duwe) to be satisfied with the final product. Therefore, we are making sure to follow the tasks and advice given to us by Dr. Duwe in order to stay on track.

6.2 References

[1] J. Crook, "Google Will Shut Down Songza App, Songza.com To Fold Into Google Play Music," *TechCrunch*, 02-Dec-2015. [Online]. Available: https://techcrunch.com/2015/12/02/google-will-shut-down-songza-app-songza-com-to-fold-into-google-play-music/. [Accessed: 24-Feb-2020].

[2] "Musicovery B2B," *Musicovery B2B*. [Online]. Available: http://b2b.musicovery.com/. [Accessed: 24-Feb-2020].

[3] "MusicFit," *App Store*, 02-Jan-2018. [Online]. Available: https://apps.apple.com/us/app/musicfit/id1186085097. [Accessed: 24-Feb-2020].

[4] E. Van Buskirk, "Songza's Concierge Picks Free Music for Specific Situations (Now for iPad)," *evolver.fm*, 05-Mar-2012. [Online]. Available: http://evolver.fm/2012/03/05/songzas-new-concierge-picks-free-music-for-your-specific-situation/. [Accessed: 24-Apr-2020].

[5]"iOS App Testing Tutorial: Manual & Automation," Guru99, 24-Mar-2020. [Online]. Available: https://www.guru99.com/getting-started-with-ios-testing.html. [Accessed: 25-Apr-2020].

[6] Guo, Bingkun, "iOS Security" WUSTL, 1-Dec-2014. [Online]. Available: https://www.cse.wustl.edu/~jain/cse571-14/ftp/ios_security/index.html. [Accessed: 25-Apr-2020].

[7] "16 Metrics to ensure mobile app success" App Dynamics, 2015. [Online] Available: https://www.appdynamics.com/media/uploaded-files/1432066155/white-paper-16-metrics-every-mobile-team-should-monitor.pdf. [Accessed 26-April-2020].