Team #1

Tony Nguyen, Ian Farris, Herbert Aruya, Pumposh Bhat, Teddy Kahwaji

Project Name: (Segue™)

Project Synopsis:

- Predictive music transitioning mobile app that predicts and mixes songs according to the listener's best interests.

Project Description:

- Despite the many features that music streaming services provide, only a few features exist that target passionate and devoted music lovers. Segue aims to reach beyond basic curated playlist; by utilizing music theory into machine learning algorithms, Segue will provide extraordinary transitions between songs. For instance, by taking advantage of the BPM (beats per minute) of a song and additional metrics, Segue curated playlists will contain songs that flow harmoniously together. Additionally, for those users who may not have complete music libraries, Segue allows users to select from an abundance of artists effortlessly and it will provide the curated transition playlist for them. Furthermore, Segue will expand on music theory concepts and aim to reach a diversity of genres; this will be done not only by machine learning algorithms but also by selective research. Utmost Segue will act as an extension to music service providers in the form of a mobile app; it will administer an enriching experience for music connoisseurs and casual listeners regardless of whichever streaming service they subscribe to.

Project Milestones:

1st semester:

- Project requirements/plan (October 18th)
- Project proposal video (October 28th)
- Finish researching programming tools (machine learning, react native, etc) (December 6th)
- Documentation (No specific due date; document as we design/implement)

2nd semester:

- Front end implementation (January 31st)
- Back end implementation (February 28th)
- Database installed (March 10th)
- Mobile app front-end styled (March 17th)
- Documentation (No specific due date; document as we design/implement)

Project Budget:

- Estimated cost: $15 for music subscription services (Spotify)
- Special training: machine learning and full-stack
- Subscription services will be required for the whole lifetime of the project.

Work Plan:

Tony:

- Creating and storing user login info
- Research machine learning and full stack tools
- Implement ML algorithms and design backend
- Help style the front end
- Organize documentation

Pumposh:

- Configure web hosting & APIs on backend framework
- Help front-end design/development
- Design algorithms for song transitions

Ian:

- Design intuitive front-end
- Implement and style front-end
- Document use cases for front end
- Help design algorithmic logic for song transitions

Teddy:

- Research of music theory and ML libraries.
- Help create pages within the mobile application.
- Research proper backend that supports ML.

Herbert:

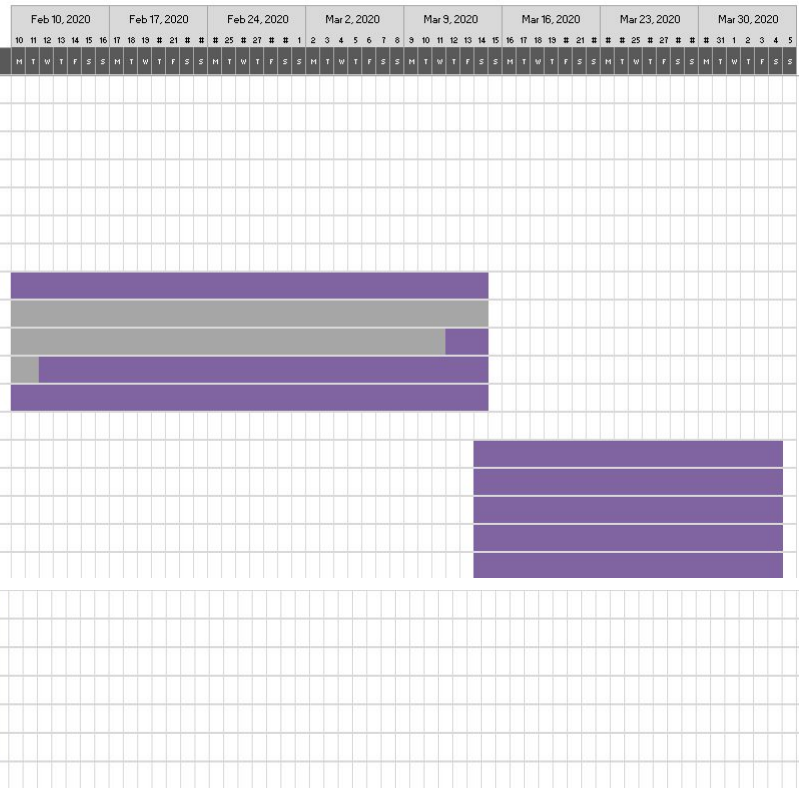
- Work with and implement machine learning framework
- Backend configuration
- Research on the most optimal full-stack to use
- Documentation help

Gantt Chart:

Segue

Project Start: Wed, 8/28/2019
Display Week: 25

| TASK | ASSIGNED TO | PROGRESS | START | END |
|---|---|---|---|---|
| **Research & Design** | | | | |
| Research ML | Tony & Herbert | 90% | 8/28/19 | 12/12/19 |
| Web Hosting Setup | Tony | 100% | 8/28/19 | 11/10/19 |
| Music Theory R&D | Pomposh & Teddy | 80% | 8/28/19 | 11/10/19 |
| Front-End Initial De | Ian | 100% | 8/28/19 | 11/10/19 |
| Initial Documentatic | Entire Team | 100% | 8/29/19 | 12/10/19 |
| **Initial Development** | | | | |
| ML algorithm Imple | Pomposh & Herbert | 20% | 1/23/20 | 3/14/20 |
| Functional Web Ser | Tony | 100% | 1/23/20 | 3/14/20 |
| Backend Configure | Herbert | 95% | 1/23/20 | 3/14/20 |
| Mobile Pages Setu | Teddy & Ian | 40% | 1/23/20 | 3/14/20 |
| APIs configured | Entire Team | 12% | 1/23/20 | 3/14/20 |
| **Development & Integration** | | | | |
| Backend Fully Cont | Tony & Herbert | 0% | 3/14/20 | 4/12/20 |
| Fully functional Alg | Pomposh & Herbert | 0% | 3/14/20 | 4/12/20 |
| Mobile Pages Fully | Teddy & Ian | 0% | 3/14/20 | 4/12/20 |
| APIs communicate ( | Entire Team | 0% | 3/14/20 | 4/12/20 |
| Functional Song Tr | Entire Team | 0% | 3/14/20 | 4/12/20 |
| **QA and Tweaks** | | | | |
| Mobile Performanc | Teddy | 0% | 4/12/20 | 5/12/20 |
| ML Algorithm Twe | Pomposh, Herbert, Tony | 0% | 4/12/20 | 5/12/20 |
| UX/UI | Ian | 0% | 4/12/20 | 5/12/20 |
| Minor Feature Add | Entire Team | 0% | 4/12/20 | 5/12/20 |
| Regression Testing | Entire Team | 0% | 4/12/20 | 5/12/20 |

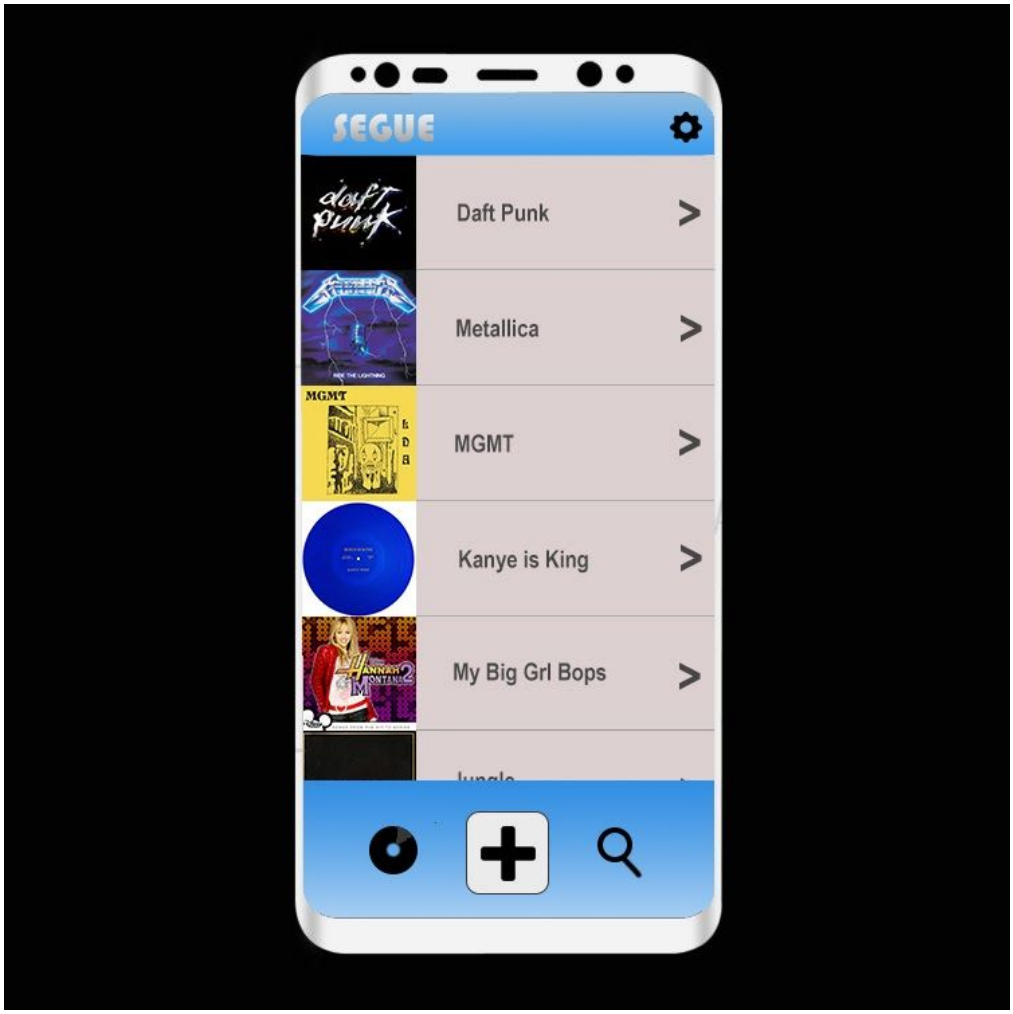



Preliminary Project Design:

We have set technical specifications according to what technologies we believed would allow us to accomplish the goals of our project with the most ease. We decided to go ahead using React native for constructing the front-end of the application. Django along with Rest API are what we have decided to go with for our backend. Machine learning is an important piece of the application we are aiming to create, and Tensorflow is our choice for a machine learning platform. For choosing these specific technologies are as follows:

- React Native: React native makes cross-platform development a breeze because most of the codebase can be reused between android and apple versions of an application. This is especially useful because we would waste valuable time for this project learning different languages for different platforms we might not be used to coding for. React also allows developers to create applications like building blocks, creating reusable components to save both time in initial development, as well as updates. Not only will this save time in development, but also in catching bugs because this style of coding allows less variety in the way bugs can manifest.

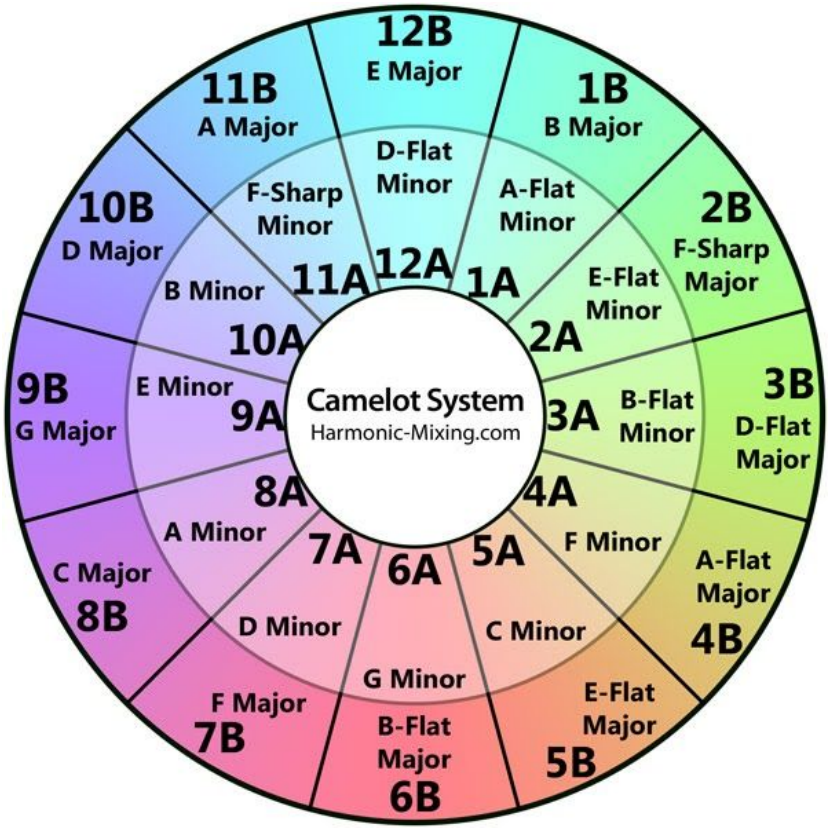
- Django: Django is designed with ease of use in mind, with the specific goal of allowing developers to write their application as quickly as possible. It contains tons of additional built-in utility that can help developers with things such as user authentication and content administration.
- Rest API: Rest allows for ease of scalability due to the separation between client and server. Rest has additional advantages such as using a relatively small amount of resources and being simple to build and update.
- TensorFlow: Tensorflow is an open-source machine learning platform that was developed by Google. One of the biggest advantages is the backing of Google, which means its widely used and has excellent performance as well as a host of features through community support. This level of community support could result in our group identifying and leveraging machine learning applications that could be applicable to our use case.




(Figure 1 Mockup of front-end UI)

Firstly for Segue to work we will require our users to connect a premium account for a music streaming service, such as Apple Music, Spotify, or Pandora, as these applications allow

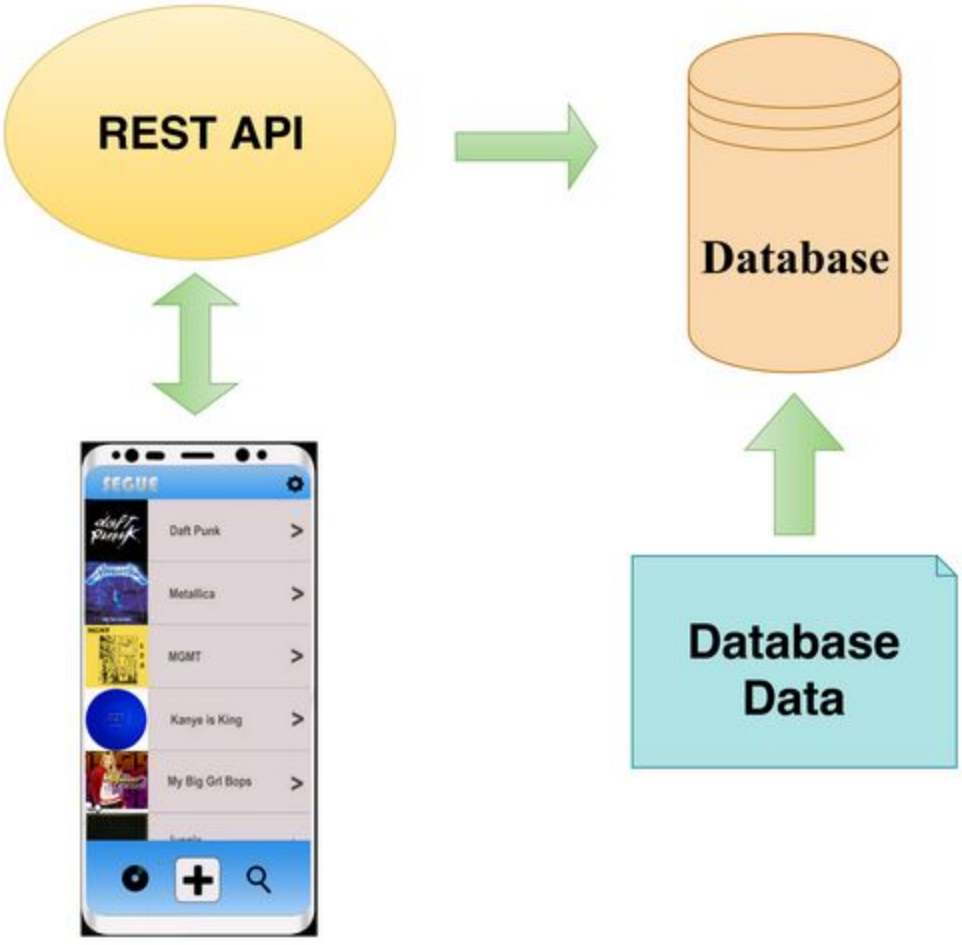
for users to export their music with the use of a premium account. At a high level our application will work by allowing users to import playlists that they have already created in their choice of music streaming service. Next we can discuss how the users will interact with Segue by referencing Figure 1. As mentioned above, users will be required to connect a premium music streaming service account with our application. This can be done through the options/setting icon in the top right corner of Figure 1. Clicking on this icon will also allow the user to update email address and password and musical preferences. Once the user has connected their account with their account on their preferred music service they will be able to use the next functionality which is importing a playlist to our application. This is done by pressing the plus icon in the center of the bottom banner in figure 1. Another piece of functionality that Segue will include is the browse tab, accessible by clicking on the magnifying glass icon. This page will show users other Segue user's playlist with the ability to rate them, leading to our application highlighting well-rated playlists. The last icon to be discussed in the disc on the bottom left of figure 1, clicking this will take the user to a media player type page. This page will show the previous song, the current song being played, and the next song to be played according to our algorithm. This will also have functionality such as play, pause, and skip.




(Figure 2. Camelot wheel for matching key transitions)

The next thing to discuss is our algorithm will determine which transitions are optimal within a playlist as well as how machine learning will be utilized in improving that algorithm. The most important aspects of a song to determine how well a transition will go between two songs is the key to both songs, beats per minute, and the genre of both songs. Initially, our

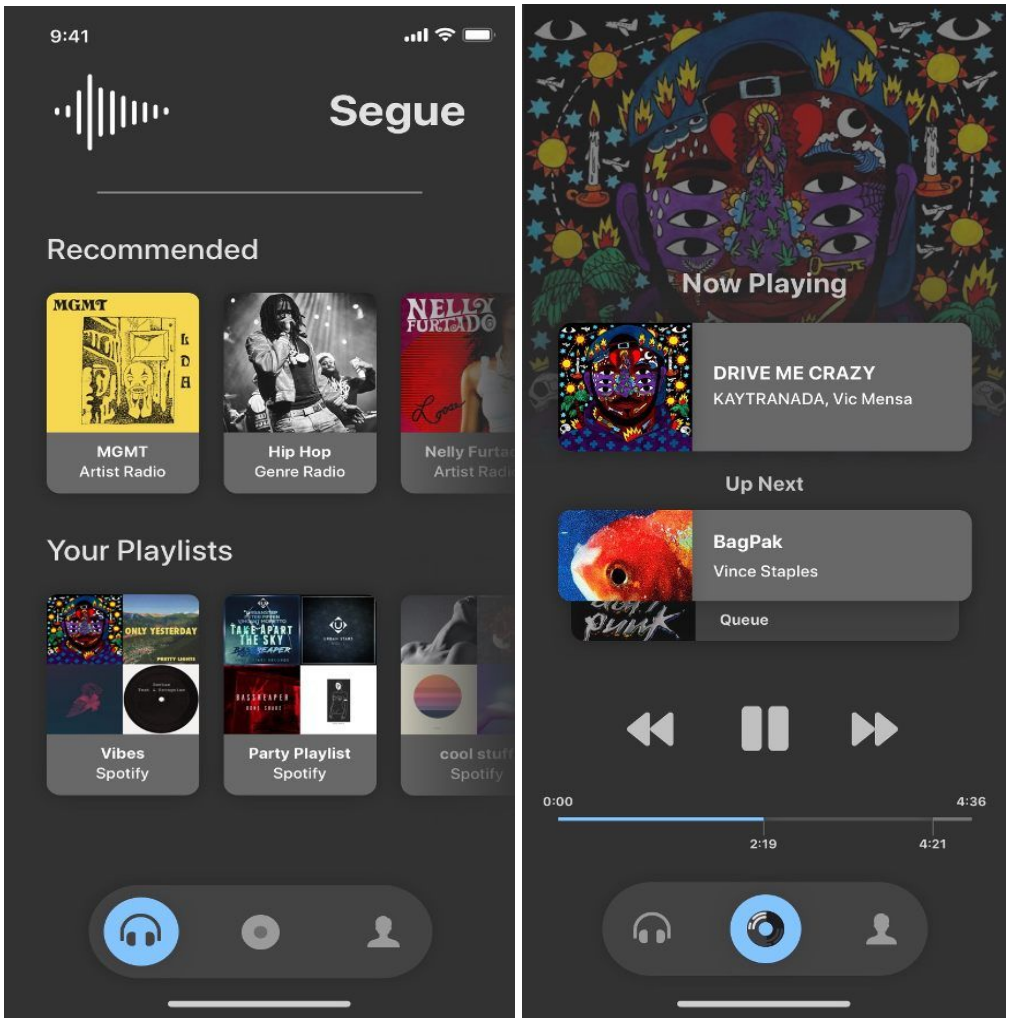
algorithm will determine the soundness of a transition by keeping the bpm difference within a certain threshold and referencing the Camelot wheel as shown in figure 2. According to the wheel, E Minor would transition well with B Minor, A Minor, or G Major. An example of a good transition would be track 1: 140 bpm  key: D Minor with track 2: 135 bpm key: G Minor. This simple algorithm will be supplemented by machine learning. Our algorithm will use a transition matchability index as a target variable. Taking metadata provided by music streaming service api's to feed into our algorithm. We have found an initial data set of good musical transitions to leverage in the form of a subreddit dedicated to this subject. Skimming the data from this subreddit may be a little difficult as most posts do not contain the text of song names. We have discussed many possible workarounds for this such as analyzing samples of the audio for both tracks and finding a match for them that contains the song name in text somewhere easy to grab. In addition to the training of our algorithm using data from this subreddit, we will also prompt Segue users to rate transitions from song to song in order to get more data to work with as our target variable does have an element of subjectivity.




(Figure 3. application to database connections)

The last aspect of our application to be discussed is how the front-end and machine learning algorithms will interact with our databases holding transition data. Referencing Figure 3 the application will both send data to our databases in the form of transition ratings done by users, and the application will receive data such as matchability ratings for potential transitions from our database. This movement will be monitored and assisted using Rest API.

The biggest obstacle we face is being able to leverage good data to improve our algorithm. This has already been touched on a bit but although we have found some good data to utilize it may not be as much data as would be needed to really refine our algorithm and as discussed before some difficulty will exist in scraping that data so that we can utilize it.



(Figure 4. Playback UI)

Ethical Issues:

- One of the ethical issues we'll be running into is ACM's code of ethics 2.6. Code 2.6 requires us to work only in areas of competence. For all of our group members, this is our first project where we will be implementing a machine learning algorithm. As of now, we do not have the required knowledge to call ourselves competent in machine learning. This

is something that we will have to do thorough research on before we can decide roles for our machine learning design. Since we all want to have some role on the machine learning aspect of our project, all of us will have to continue our research to avoid this issue.

- Another ethical issue we'll be running into is ACM's code of ethics 2.9. Code 2.9 requires us to design and implement systems that are robustly and usably secure. This is especially important since we'll be retrieving and storing user information. Security is going to be one of our top priorities to protect important user information. Our mobile application should be robust as well. We'll need to make sure that each component in our application is free of dangerous bugs. We'll avoid robustness issues with a lot of unit testing.

Intellectual Property Issues:

- One of the biggest intellectual property issues we'll run into is patenting. Our machine learning algorithm has never been done before so we'll want to protect our algorithm with a patent. This is especially important because it prevents huge music streaming services like Spotify and Apple Music from taking our work without our acknowledgment. We eventually want to push our mobile app in the global market which we'll need a patent for. This also keeps our algorithm competitive. No one will be able to steal our algorithm or build on top of it without our permission.
- Another intellectual property issue we'll run into copyrights. Similar to parenting, we want to protect our software/mobile app as a whole. That way, we can prevent any occurrences of someone copying our program and potentially selling it under a different name. As of now, we don't have any plans on making our project open-source. This is something we'll worry about once we have a robust and working application. We're also focused on the scalability of our project, so setting up copyrights to our program will be needed as soon as it's functioning.

Change Log:

- Project budget reduced since the application will now be created specifically for Spotify users only; therefore, Apple & Soundcloud subscriptions will not be needed.