Team 13 Project Proposal

Team Number: 13

Team Members
Chance Penner, Haonan Hu, Markus Becerra, Thomas Gardner, Ziwen Wang

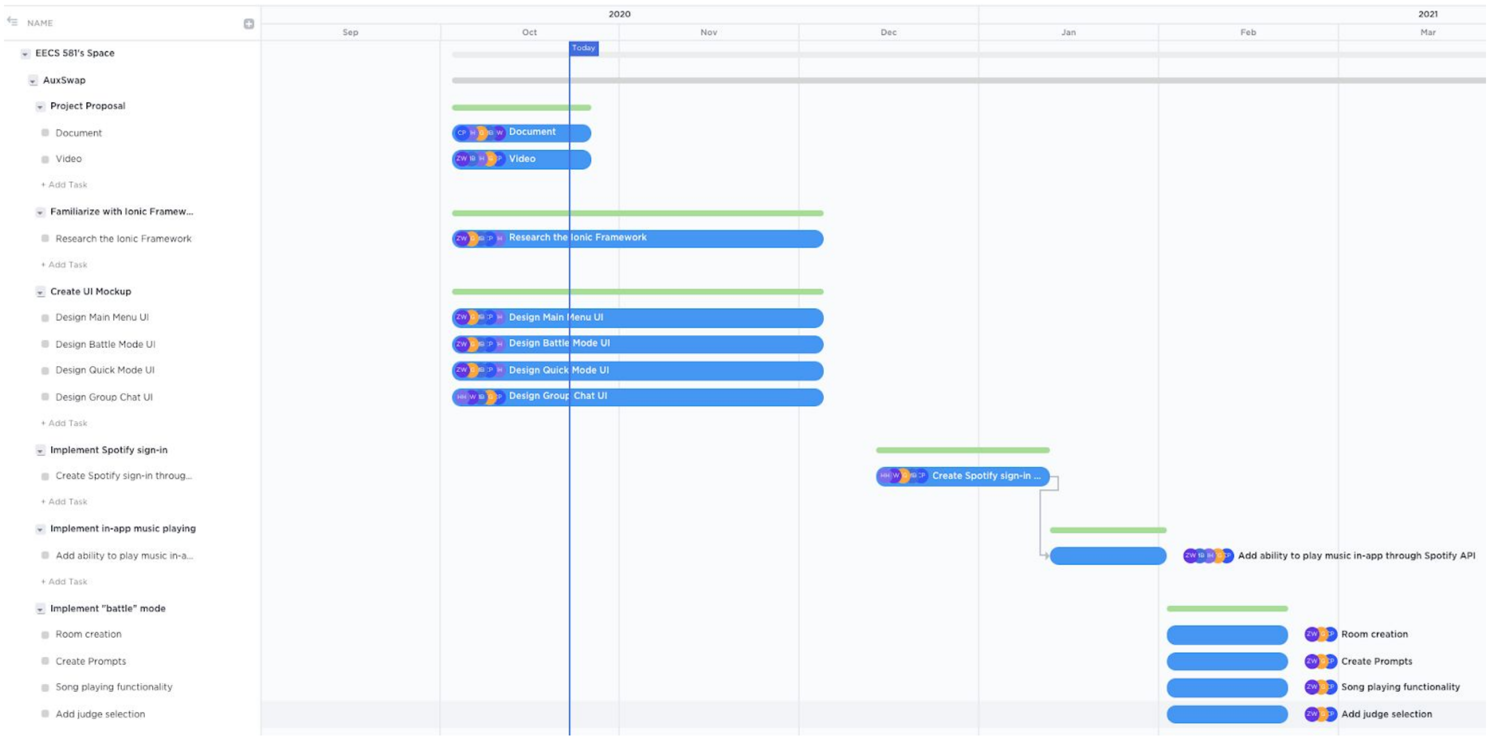
Project Name
AuxSwap

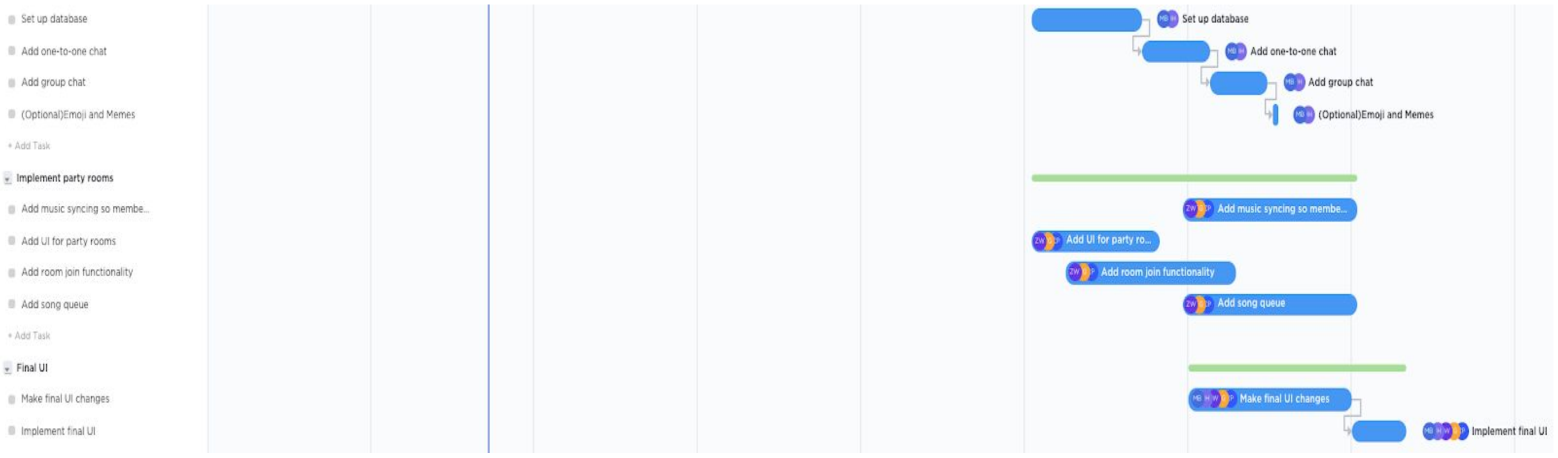
Project Synopsis
Web and Mobile application connecting Spotify users to allow for fun and interactive group song battles, joinable party lobbies, and social chat functionality.

Project Description
This project is being undertaken to provide an interactive and social music sharing experience. Users can battle in rounds to see who has the best song choice for a given prompt, with the capability of adding the winning songs to your Spotify playlists. Once the round is completed, a snippet of the song is played to reveal the winner. With chat room functionality, this allows for social sharing of songs with friend groups. Party lobbies will host different joinable rooms where AuxSwap users can add songs to the room's queue. Users will have the option to listen in to the current lobby. The end result of this project will be a fully implemented music sharing application, developed for web and smartphones. We hope to release it on all app stores, so long as it meets our quality standards and app store criteria, with the main focus being the website.

Project Milestones

Milestone	**Description**	**Date**
Semester 1		
Make Gantt Chart	Create Gantt chart and assign tasks to team members	Oct 5, 2020
Project Proposal	Create Project Proposal document and video	Oct 26, 2020
Familiarize with Ionic Framework	Research the Ionic Framework to understand how it works	Dec 4, 2020
Create UI Mockup	Create a rough draft UI mockup.	Dec 4, 2020
Semester 2		
Implement Spotify sign-in	Allow users to link their Spotify accounts	Jan 12, 2020
Implement in-app music playing	Utilize Spotify API to allow users to stream Spotify songs in-app	Feb 1, 2020
Implement “battle” mode	Given a prompt, players suggest a song and a judge chooses the best one for the round	Mar 15, 2020
Implement chat functionality	Create group chat functionality for sharing and playing music	Mar 20, 2020
Implement Party Lobbies	Users can join different rooms to listen to songs with others	Apr 1, 2020
Redesign and implement UI	Finalize UI decisions and implement UI to app	Apr 10, 2020





Project Budget

Item	Cost	Required Date
Website Domain	~$20	Feb 1 2021
Annual Apple App Store developer fee	$99	TBD
One time Google Play Store developer fee	$25	TBD

Preliminary Project Design

How the software works

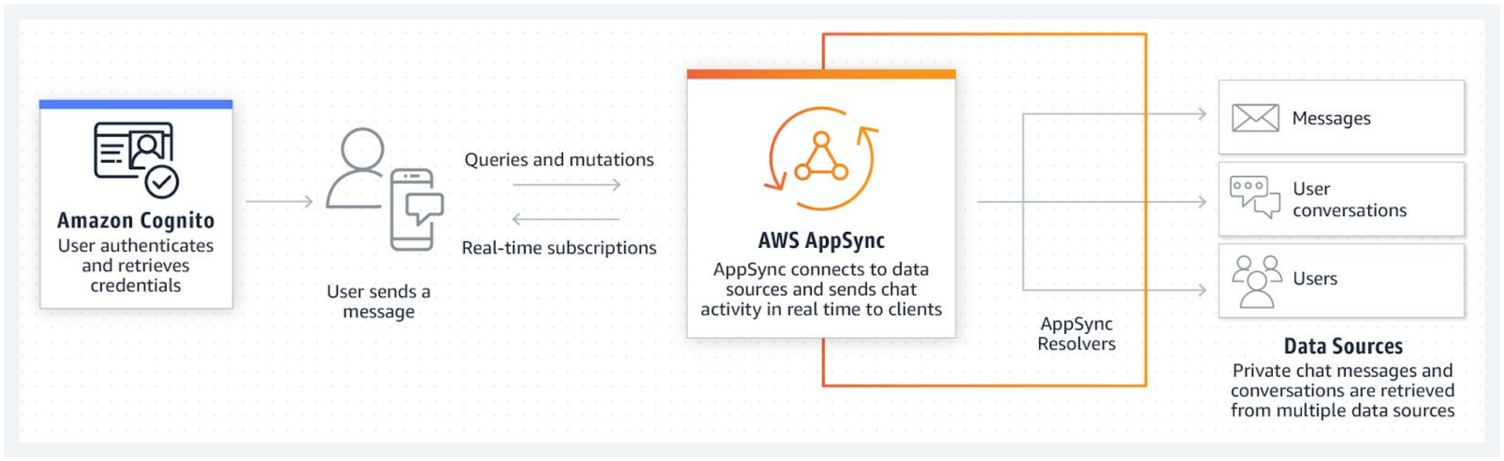
AuxSwap will be a multi-platform web and mobile application. We will be using the Ionic Framework. Thus, our application will only require separate builds for each platform, rather than rewriting the code for each platform. This is the power of hybrid app development. Operating on a GitHub repository, our team will be able to program and implement all of the features collaboratively. We will incorporate music streaming and sharing into our application using the Spotify API. We can access Spotify playlists, search tracks, and more. Users will be greeted to our app by a login screen for Spotify. Once a user is logged in, our app is able to have functionality of their Spotify application. We can stream songs, add to playlists, and more. The main features of our app will be the “Battle” mode, Party mode, and chat functionality.

The Party mode will allow a user to create a room, and other users will be able to connect to this said room. Users then can add to the room’s queue so each user has input into what will be played during the listening session. Each user will have the option to mute the room’s audio, so if, for example, you are at a party, and there is an AuxSwap room open, you can connect to this room and queue a song. The host of the party is also the host of the AuxSwap room, and has their phone connected to a bluetooth speaker. You do not need to have the music also playing out of your phone, so you will have the option to mute the audio, but the music will still be playing from the host’s phone. The host of the room will also be the admin. They will have the ability to set various settings and permissions for the room. it is likely these will be determined as we get closer to the implementation phase. Users will also have the ability to chat within the room.

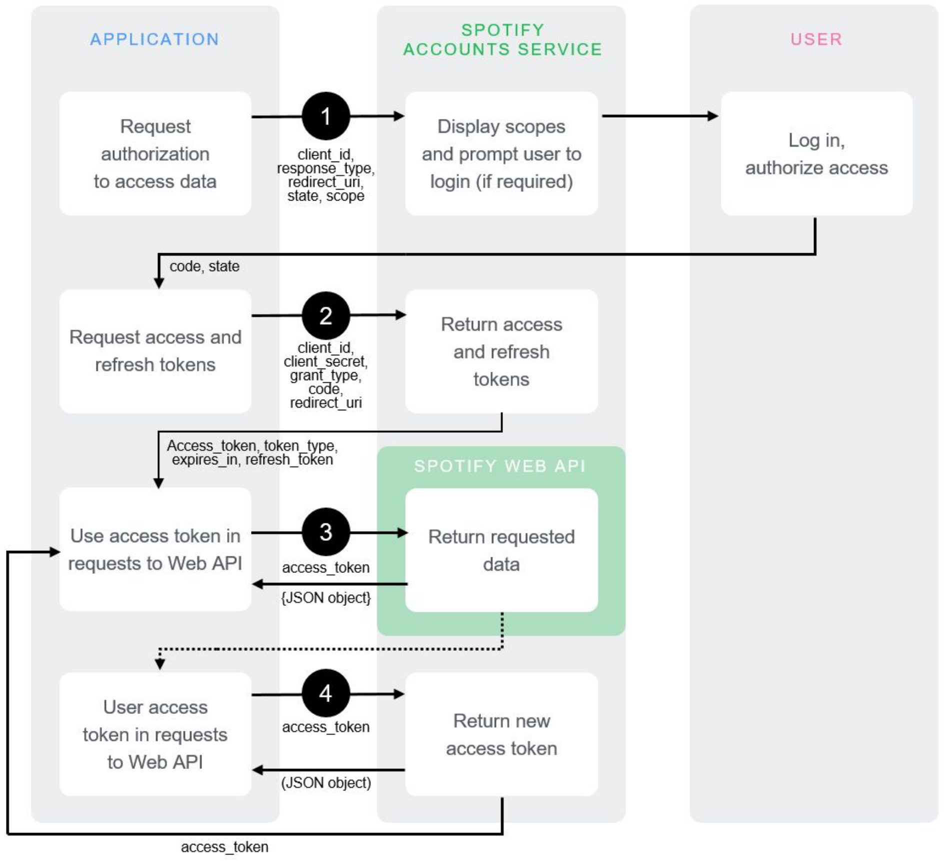
The next mode is a battle mode, which is modeled after Apples To Apples, where users will select a song to play based on a prompt, and there will be a “judge” who chooses the song that best fits this prompt, and the user who submitted this song will have their score increased by one. The winner will be the person with the most amount of points after a certain amount of rounds.

Chat functionality will allow multiple users to chat about anything and share music. Within a chatroom, users have the ability to play shared songs in the app, rather than taking them out of the app. This will make users more connected to AuxSwap as it can be used to chat, share, and play music, all in one location. Perfect for many use cases.

Utilizing these capabilities in the features of our app, the user experience will be seamless (without any disrupting login or confirmation screens). The Spotify API requires that we include certain metadata and artist information wherever we are streaming music. We will be careful in verifying that we are not violating any intellectual property of Spotify or Spotify artists. In order to safely store user data for the messaging portions of our app, we will use Amazon Web Services (AWS). By storing user data and private messages in the cloud, we keep this data away from potential threats to our application's security.


Amazon Cognito
User authenticates and retrieves credentials
User sends a message
Queries and mutations
Real-time subscriptions
AWS AppSync
AppSync connects to data sources and sends chat activity in real time to clients
AppSync Resolvers
Messages
User conversations
Users
Data Sources
Private chat messages and conversations are retrieved from multiple data sources

AWS AppSync can be used to build real-time messaging functionality within an application.


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

Technical Constraints

Framework and Programming Language

We have chosen to use the Ionic Framework for AuxSwap, which will likely mean TypeScript will be our language of choice. This decision was made due to how efficient and useful the Ionic Framework is for building multiple platforms in one code base. Although this means many of the team members will have to learn the Ionic Framework and TypeScript, which will use up some of our development time, it will ultimately pay off as we can write the code once and deploy to multiple platforms with ease.

For AuxSwap, we will also be using the Spotify API. This means that we have decided on the sole music streaming application that we will use, with no ability to add another. Also, this will limit AuxSwap to only be available to users that have Spotify Premium, since that is a requirement of the Spotify API. This will of course limit the amount of users that can use our application, but it is a necessary requirement in order to utilize the Spotify API.

This shows the flow of user login with the Spotify API.

Platforms

Our platforms of choice are web and mobile. We are primarily focusing on the web application as this will allow for ease of sharing the application without having to have a finished and polished mobile app on the app store. However, we are not constraining ourselves to a full release on either web or mobile. This is due to concerns regarding budget, privacy, and even legal concerns with publishing an application made with the Spotify API. We primarily want to develop the application to potentially be released, but we do not expect to have a fully released web and mobile application due to these limitations.

Another concern with using the Spotify API is that there are many guidelines within the terms of service that we must abide by. This means doing things such as using the Spotify logo correctly and placing the necessary metadata in each of our features, ensuring the UI and styling of AuxSwap fits the required guidelines required by Spotify, and other more trivial guidelines. This can be quite tedious, but of course is necessary.

Business Constraints

Schedule

Since the final delivery date is fixed at May, 2021, we need to make good use of our time to meet all of our requirements by this time. Once the deadline arrives, we will not be able to make any more changes prior to our project presentation. We will want to make sure we can not only meet our product’s design requirements, but also our presentation requirements, such as our final proposal video and all related documentation and charts.

Budget

We did not set out to create a revenue generating product, but rather something for fun and free use. Of course, this relies on the capabilities of the free version of Amazon AWS and the Spotify API, so if AuxSwap were to become incredibly popular, then this may not be as sustainable. Generating a revenue stream would be quite difficult for this project. First, we would need to apply to build a commercial app and get approval from Spotify to do so. Then, we would need to decide on a method of generating an income, such as advertisements. Again, we would need to get permission from Spotify to be allowed to use their API to generate an income, and especially for having ads in our product. Our main concern is not to make money, but to be able to sustain the application's costs, such as Amazon AWS. But, if spotify does not allow us to earn revenue with their API, we need to come up with a legal way to get funded in order to sustain AWS services. Thus, legal consideration could be extremely constraining.

Ethical issues

A harmful condition

An ethical issue AuxSwap will be facing is that it may create a harmful environment or situations to its users. Similar to other 'battle' games or applications configured with any sorts of competition among its users, even though the initial intention for those particular applications could be leading their users positively for some degree of entertainment. However, a user can take information differently among the aspects of their ages and cultural background as well as social level, the consequences could be vital with misleading a user. To prevent this ethical issue taking place in AuxSwap and emphasizing our intention for AuxSwap to be more entertaining among its users, by weakening the importance for 'battle' in the application such as adding alternative features like chatting and focus on the in-App word choices as well as structure design in AuxSwap's user interface to make it universally understandable for our users to use.

User’s privacy

Another ethical issue AuxSwap will be facing is that it may disclose sensitive information. It is quite understandable to assume that the user information on AuxSwap could be disclosed for some unintended purpose, since AuxSwap is both web and mobile based application, all the data transformations rely heavily on the internet. Even though AuxSwap is based on the Spotify API, which is quite reliable sources, however, to keep our user safe on using AuxSwap at all time, basic hashing mechanism will be applied for data transfer such as in-App chatting feature. Moreover, by keeping an object-oriented programming style on most AuxSwap source code, which can achieve more reliability of handling with unintended behaviors.

Intellectual Property issues

Spotify API

Throughout this project, we will be utilizing the Spotify API to play music from each user’s device. The Spotify website clearly notes “by using Spotify developer tools, you accept the Spotify Developer Terms of Service”. These terms of service outline the limitations of the API and outline the features that API users are eligible to use. We must be responsible with our use of Spotify’s code base, and ensure that we give credit wherever is needed. We cannot “use or register any trademark or domain name that includes the word ‘Spotify,’ any other Spotify trademark, or any name that is confusingly similar to any of them” We must name our features something dissimilar to “Spotify” to avoid harming Spotify’s intellectual property. If we were to launch AuxSwap commercially, we would also have to be wary of the fact that Spotify discourages the use of the API for “games” and “quizzes” of any sort. We would have to ensure that the use of the Spotify API would not harm or put Spotify at a disadvantage in any way.

Changelog

- Adjusted the “Battle” mode to have one judge per round instead of each person voting. We felt that this would more often allow each person to have the ability to choose a song they like.
- Removed “quick play” mode. We felt that this mode was too similar to the battle mode, so we decided to focus on the chatting functionality instead of this feature.
- Added Party mode. These lobbies replaced quick mode, and provide a more relaxed feature than battle mode. This feature is more of a “hang out” kind of user experience.
- Switched from Google Firebase to Amazon Web Services for cloud messaging. We made this switch after researching the benefits of each service.