Team 13 Final Project Design

Team Number: 13

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

Project Name
AuxSwap

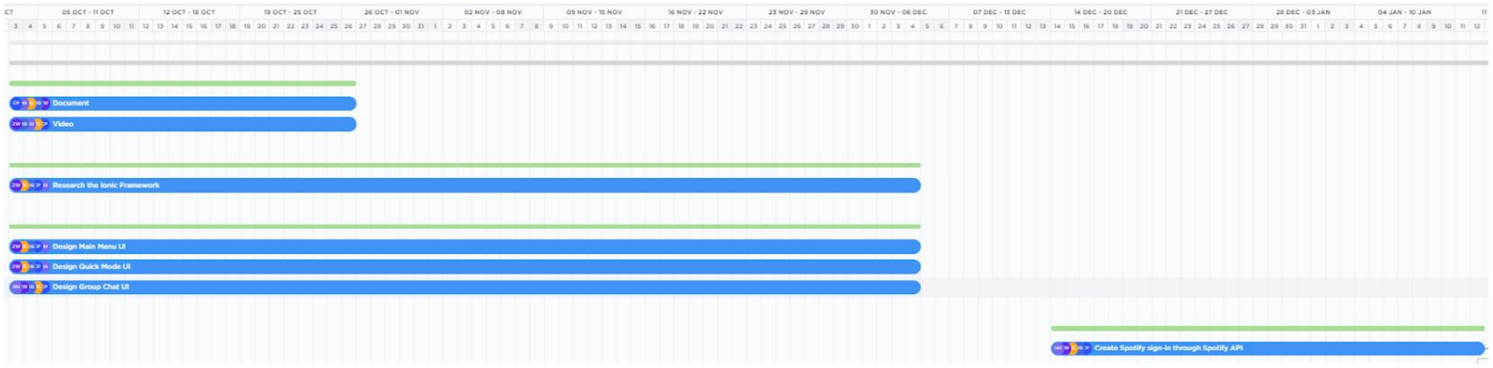
Project Synopsis
Web application connecting Spotify users, enabling in-sync music listening and social chat functionality for seamless music sharing.

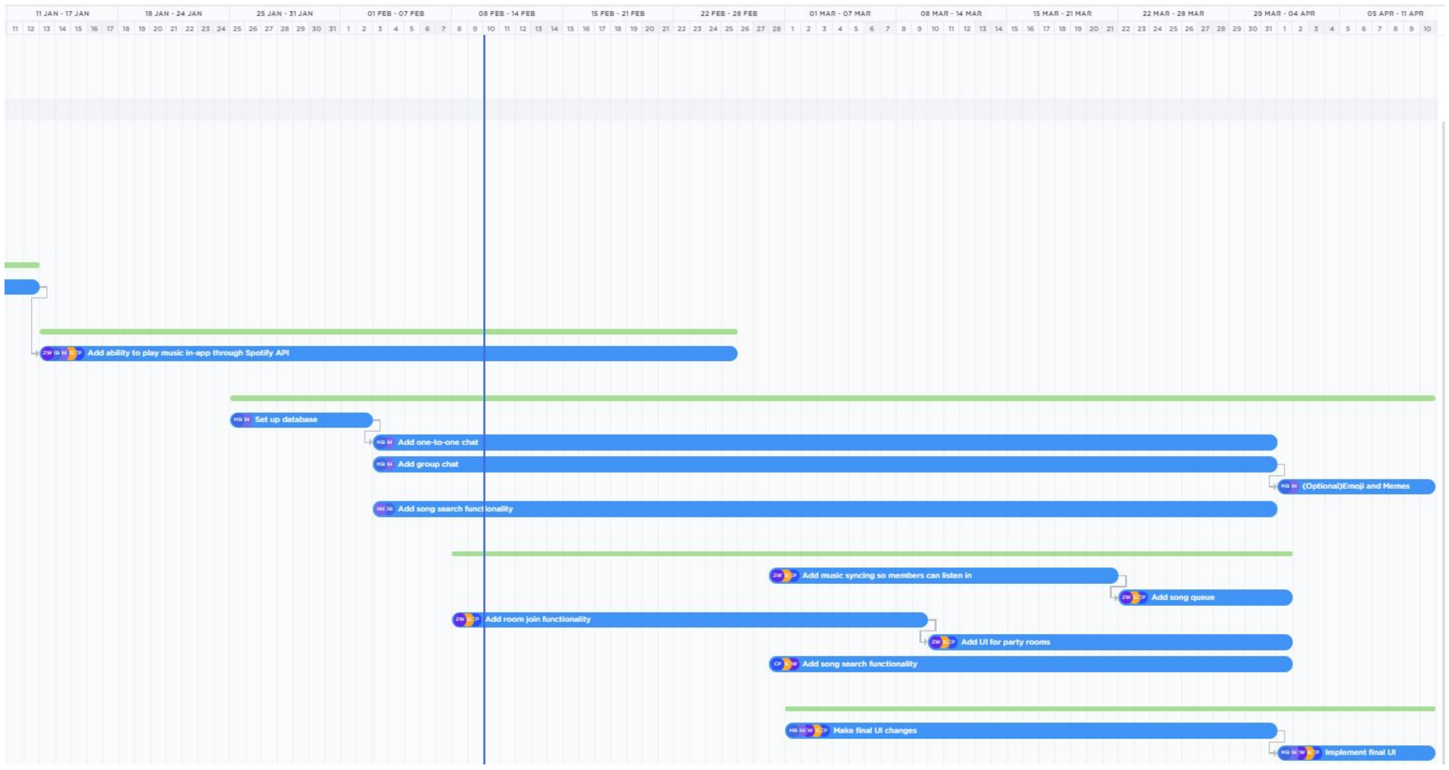
Project Description
This project is being undertaken to provide an interactive and social music sharing experience. We feel that AuxSwap will provide users with a unique social experience, with a strong focus on music sharing. The two main features of our project are chat rooms and party lobbies. With chat room functionality, this allows for social sharing of songs with friends. Create your own chat rooms with friends to begin discussing new music and share your favorite songs. Built in song sharing allows for easy song playing without even leaving the app. Party lobbies will host a room 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 the web, giving users the ability to chat with friends, share songs, and listen together with ease.

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 Tech Stack	Research the Tech Stack 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, 2021
Implement in-app music playing	Utilize Spotify API to allow users to stream Spotify songs in-app	Feb 28, 2021
Implement chat functionality	Create group chat functionality for sharing and playing music	Mar 31, 2021
Implement Party Lobbies	Users can join different rooms to listen to songs with others	Mar 31, 2021
Redesign and implement UI	Finalize UI decisions and implement UI to app	Apr 10, 2021

Gantt Chart


05 OCT - 11 OCT
12 OCT - 18 OCT
19 OCT - 25 OCT
26 OCT - 01 NOV
02 NOV - 08 NOV
09 NOV - 15 NOV
16 NOV - 22 NOV
23 NOV - 29 NOV
30 NOV - 06 DEC
07 DEC - 13 DEC
14 DEC - 20 DEC
21 DEC - 27 DEC
28 DEC - 03 JAN
04 JAN - 10 JAN
Document
Video
Research the Ionic Framework
Design Main Menu UI
Design Quick Mode UI
Design Group Chat UI
Create Spotify sign-in through Spotify API


11 JAN - 17 JAN
18 JAN - 24 JAN
25 JAN - 31 JAN
01 FEB - 07 FEB
08 FEB - 14 FEB
15 FEB - 21 FEB
22 FEB - 28 FEB
01 MAR - 07 MAR
08 MAR - 14 MAR
15 MAR - 21 MAR
22 MAR - 28 MAR
29 MAR - 04 APR
05 APR - 11 APR
Add ability to play music in-app through Spotify API
Set up database
Add one-to-one chat
Add group chat
(Optional)Emoji and Memes
Add song search functionality
Add music syncing so members can listen in
Add song queue
Add room join functionality
Add UI for party rooms
Add song search functionality
Make final UI changes
Implement final UI

Project Budget

Item	Cost	Required Date
Website Domain	~$20	April 1, 2021

Final Project Design

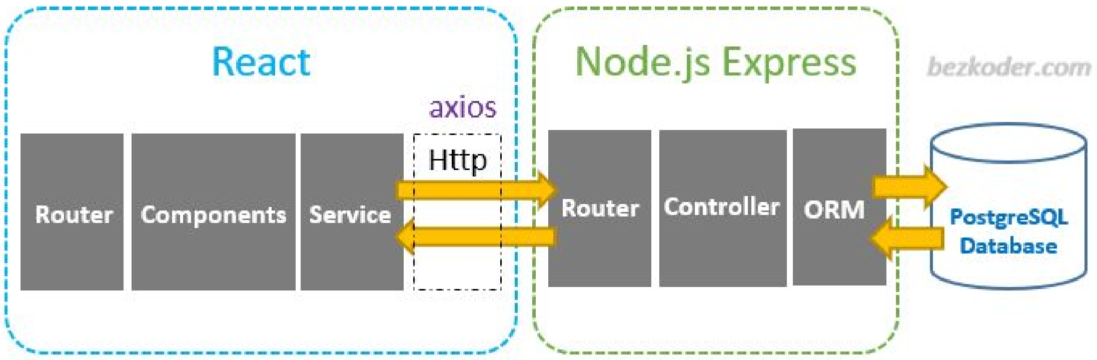
How the software works

AuxSwap will be a web application. We will be using ReactJS. 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 will have access to certain user information and allow the app to control the user's Spotify playback. The main features of our app will be the Party Lobbies and chat functionality.

The Party mode will allow a user to join 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 control playback of the room from their account, so if you are at a party, lets say, you will be able to mute the audio but still have the ability to queue songs. Users can also choose what device to play music from using the Spotify Connect API. If time allots, we will implement the ability to run/host several rooms and also integrate host features. Host features will include the maximum amount of songs that can be added by a user, allowed genres, if explicit songs can be added, etc.

Chat functionality will allow multiple users to chat about anything and share music. Within a chat room, users have the ability to play shared songs in the app, rather than taking them out of the app. Songs can easily be found and shared utilizing the search functionality. 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.



Our tech stack consists of a React JS front-end, which will call functions from the middleware (Node JS backend) to store data in the secure database.

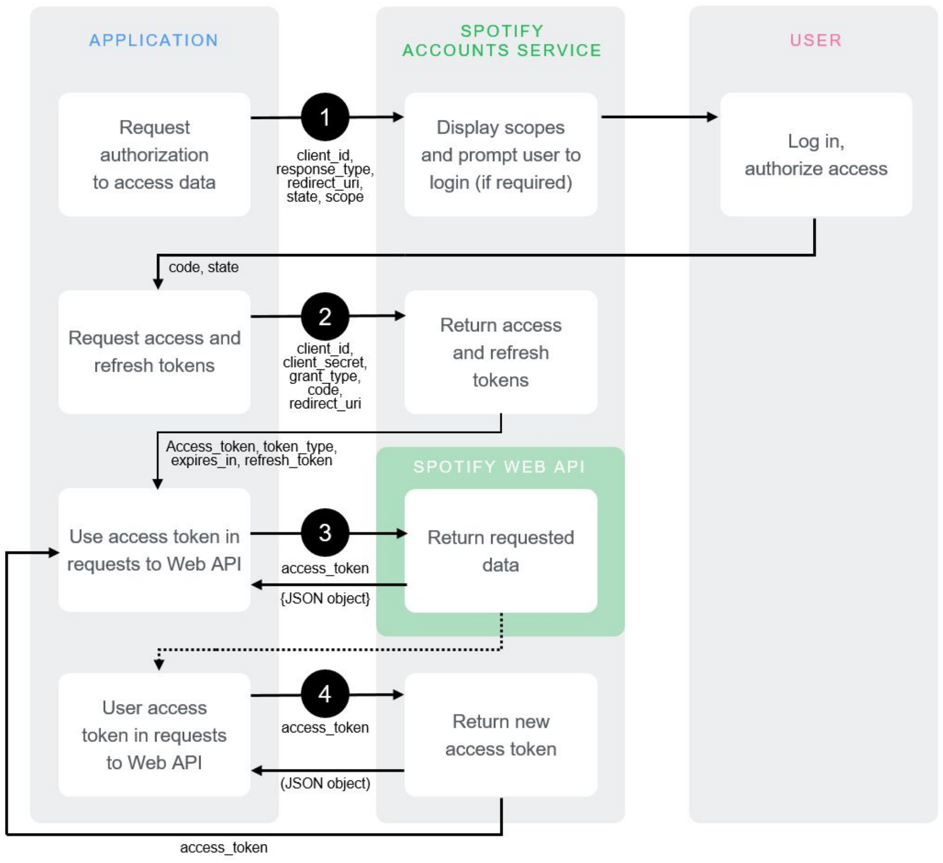
<u>Technical Constraints</u>

Framework and Programming Language

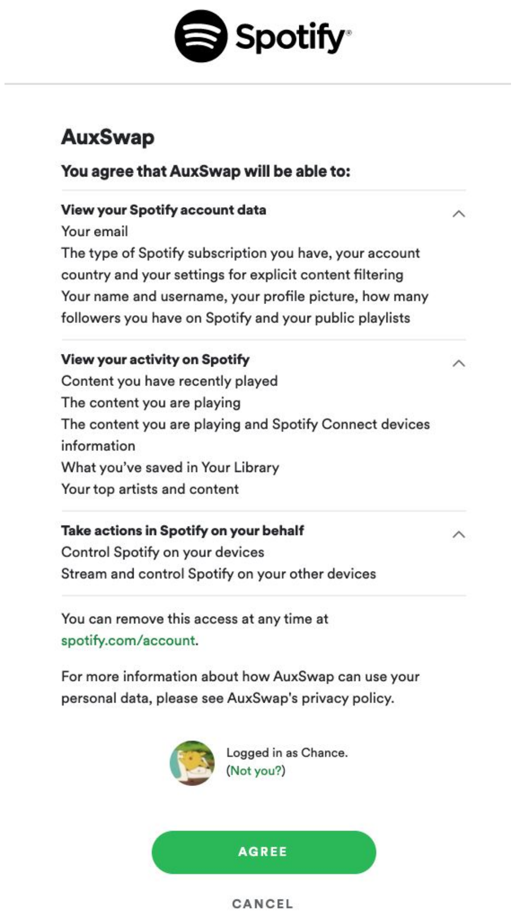
We have chosen to use ReactJS for AuxSwap. This decision was made due to how efficient and useful ReactJS is, as well as the plethora of learning resources available. This means many of the team members will have to learn React, which will use up some of our development time. However, it should prove to be much simpler than other choices and ultimately will provide us with all the necessary tools to create a great web application.

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.


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

The Spotify API user authorization flow chart shows how to refresh an access token for a given user.



Spotify

AuxSwap

You agree that AuxSwap will be able to:

View your Spotify account data
Your email
The type of Spotify subscription you have, your account country and your settings for explicit content filtering
Your name and username, your profile picture, how many followers you have on Spotify and your public playlists

View your activity on Spotify
Content you have recently played
The content you are playing
The content you are playing and Spotify Connect devices information
What you've saved in Your Library
Your top artists and content

Take actions in Spotify on your behalf
Control Spotify on your devices
Stream and control Spotify on your other devices

You can remove this access at any time at spotify.com/account.

For more information about how AuxSwap can use your personal data, please see AuxSwap's privacy policy.

Logged in as Chance.
(Not you?)

Our AuxSwap Spotify authorization page, verifying that the user grants our team access to their Spotify data.

Platforms

Our platform of choice is purely web. We are primarily focusing on a 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. We decided not to also create a mobile application due to concerns regarding budget, privacy, and even legal concerns with publishing an application made with the Spotify API. By focusing solely on a web-based application, we can ensure that we can meet our deadline.

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 April 21, 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

Illicit usage

An ethical issue AuxSwap will be facing is that it may be used as a tool for inciting or spreading illegal or harmful content since chatting is the primary function of AuxSwap. To defend and respect users' freedom of speech and prevent illicit usages that could occur on AuxSwap, we could consider a step that could be taken to create a positive and enjoyable environment. According to the Entertainment Software Rating Board guideline, some strong violent words could be automatically filtered out as the code build-in, and the user will only send out the string after appropriate replacement by AuxSwap.

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 a web-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 precautions will be implemented to prevent any privacy issues from arising. 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. We have to ensure that the use of the Spotify API would not harm or put Spotify at a disadvantage in any way.

Changelog

- Removed the “Battle” mode as this feature is the weakest of the three and we wanted to ensure that we could get everything done as best as we could. By removing this feature, we now are confident in our ability to create a polished product by the deadline.
- Switched from Ionic Framework to ReactJS. This change was made because React has more learning resources, a smaller learning curve, and proved to be just as efficient for this project.
- Changed due dates of milestones to better reflect the updated Gantt Chart.
- Switched to web only for our project, since React does not support the multi-platform model that the Ionic Framework does.
- Removed app-related budget costs, since our app is now web only.