AnarQ

Team 18 — Product Backlog

Amul Chaulagain, Logan Kulinski, Nick Pappas, Patrick Crowne, Siddharth Madapati

Problem Statement

As is often the case, deciding what music to play next while amongst a group can be a long, arduous process, prone to arguments and leading to wasted time and unhappy people. AnarQ aims to simplify this process by allowing users to upvote or downvote songs in a queue, ensuring fairness and leading to a general consensus on what to play next. Other programs with the idea of voting-based music queues exist, but ours will provide better features and greater customizability to both the host (DJ) and clients (Jammers), allowing further user-friendly interactions and customization.

Background Information

Target Audience
In today's society, it is commonplace for the host of a gathering or party to be in charge of the music. That's not inherently a problem, but it can be discouraging when the music isn't connecting with the majority of the guests. Our application is designed for people who want to contribute to what songs are played at a social gathering.

Similar Platforms
There are a few applications that have the same idea of a vote-based music queue for parties, such as Jukestar and Festify. With Jukestar, they have you download an app if you're a host, and an app if you're a guest for the vote-based music queue system. This can be annoying as a guest if you don't want to download an app to take part in it, or as a host if you don't want to download two different apps. Festify allows for web-based community queue creation and interaction, but doesn't have many features past that. The limitations for both of these pre-existing applications is that they are very contained in scope. Our application would provide a wide variety and range of customization options and host settings.

Limitations
The biggest limitation of pre-existing services with similar functionality is that they require you to download an app in order to use the service. Given that these services are meant to be used in large social settings, downloading apps, especially if they are large apps, can be a big discouraging factor as it is a waste of time and data. We solve this problem by giving clients the option of using a web-based service. The limited services that use web-based community queue

creation and interaction (such as Festify) do not offer the vast number of features that AnarQ can. Moreover, in order to connect to the host, users have to go to the service's website, enter a unique code provided by the host and sign in with their account (if they have one). We simplify this process by generating unique QR codes for the hosts that users can simply scan with any smartphone and instantly connect with the host.

Functional Requirements

1. As a user, I would like to be able to create an account with a username and password.
2. As a user, I would like to be able to create an account with an existing account that I already have, like Sign In with Google (if time allows).
3. As a user, I would like to be able to add a profile picture to my account during its creation.
4. As a user, I would like to be able to add biographical information to my account during its creation.
5. As a user, I would like to be able to update my account's username.
6. As a user, I would like to be able to update my account's password.
7. As a user, I would like to be able to update my account's profile picture
8. As a user, I would like to be able to update my account's biographical information.
9. As a user, I would like to be able to use two-factor authentication to sign into my account (if time allows).
10. As a user, I would like to be able to log out of my account.
11. As a user, I would like to temporarily disable my account.
12. As a user, I would like to be able to delete my account.
13. As a user, I would like to be able to enable a dark UI (if time allows).
14. As a user, I would like to be able to have a blue light filter (if time allows).
15. As a client/jammer, I would like to be able to use the application without an account.
16. As a client/jammer, I would like to be able to create a temporary anonymous username.
17. As a client/jammer, I would like to be able to keep my information private from others.
18. As a client/jammer, I would like to be able to connect to the host/DJ using a web-based client.
19. As a client/jammer, I would like to be able to connect to the host/DJ using a downloadable application (if time allows).
20. As a client/jammer, I would like to be able to access a community playlist through a short link or QR code.
21. As a client/jammer, I would like to be able to add songs to the community playlist by searching for their names.
22. As a client/jammer, I would like to be able to like any song, excluding my own, on the community playlist.

23. As a client/jammer, I would like to be able to dislike any song, excluding my own, on the community playlist.
24. As a client/jammer, I would like to be able to remove any songs I have requested.
25. As a client/jammer, I would like to be able to have the option to link my Spotify account.
26. As a client/jammer, I would like to be able to access any song on the playlist using the API of my preferred music service (Spotify API, Apple Music API, etc.).
27. As a client/jammer, I would like to be able to save the playlist to my device.
28. As a client/jammer, I would like to be able to report other users for abusing the system.
29. As a host/DJ, I would like to be able to create a music session using a web-based client.
30. As a host/DJ, I would like to be able to create a music session using a downloadable application (if time allows).
31. As a host/DJ, I would like to be able to contribute songs to the music session in the same way as a user/jammer.
32. As a host/DJ, I would like to be able to pause the song currently playing in the music session.
33. As a host/DJ, I would like to be able to advertise my music session using a short link or QR code.
34. As a host/DJ, I would like to be able to delete any song on the playlist.
35. As a host/DJ, I would like to assign certain clients/jammers as moderators.
36. As a host/DJ, I would like to view and act on reports made my other users.
37. As a host/DJ, I would like to be able to restrict the actions of certain clients/jammers that are causing trouble for a certain amount of time (like a penalty box).
38. As a host/DJ, I would like to be able to permanently ban certain clients/jammers that repeatedly cause trouble.
39. As a host/DJ, I would like to be able to restrict the genres of music that can be played.
40. As a host/DJ, I would like to be able to restrict the BPM of music that can be played.
41. As a host/DJ, I would like to be able to restrict the explicitness of music that can be played.
42. As a host/DJ, I would like to be able to change the visibility of a playlist.
43. As a host/DJ, I would like to be able to set a cooldown period between adding songs for individual users.
44. As a host/DJ, I would like to be able to set a cooldown period for the same song to be played again.
45. As a host/DJ, I would like to have a random song play if the queue is currently empty.
46. As a host/DJ, I would like to have an AI that goes through the songs in the list and recommend similar songs when the playlist is getting empty (if time allows).

Non-Functional

Architecture and Scalability
In our implementation we intend to separate the back-end and front-end components. This will make it easier to divide the work among our group members, but it will also ensure that the system follows the principles of modularity.

Our back-end will consist of a Java application that uses the Spring framework or Jakarta EE (formerly known as Java EE). While our familiarity with Java is one reason to use it, the language and platform are known for their stability and scalability. We want to ensure that the languages and frameworks that we use will continue to be reliable and scale further when needed.

Our front-end will consist of a web application that uses a front-end framework like AngularJS. The application will communicate with the back-end in order to perform the desired operations. There will be a distinction between the interface of the DJ/host and that of the jammer/client. Given the time, dedicated desktop, iOS, and Android apps may also be implemented.

Performance
It will be a priority in our design and implementation to complete requested actions in a fast, efficient way. We aim to be able to service single requests in two seconds or less. Additionally, the scalability of our application will be factored in. Given the various places that our application can be used, we aim to support thousands of concurrent sessions and users. Both of these priorities will involve carefully analysis on our chosen algorithms and corresponding implementations.

Security
The security of our system and our user's data will be the top priority. This involves actions such as obtaining an SSL certificate from a valid issuer, using secure sockets, and encrypting requests and user data. Two encryption algorithms that we are looking to use are Triple DES and RSA. We want our users to trust our platform, and ensuring its security is just one way to aid that.

Usability
The design and implementation of our front-end will include a focus on the usability of our system. While we are the developers of this system, we are also users of other software systems. We have used our fair share of software with poor usability and great usability. We would like to apply our knowledge and experience to our own system. Some design characteristics include a sleek, tidy layout and well-formed controls.

Deployment

Due to the modular nature of our back-end and front-end, we aim to have the ability to update either independently. We also intend to have the ability to switch server providers without significant issues like downtime.