Pineapple Music

Team 24

Sam Tarr
Tiger Lee
Jason Paik
Kichul Kang
Thomas Wiegand
Alan Zeng

Index

Purpose

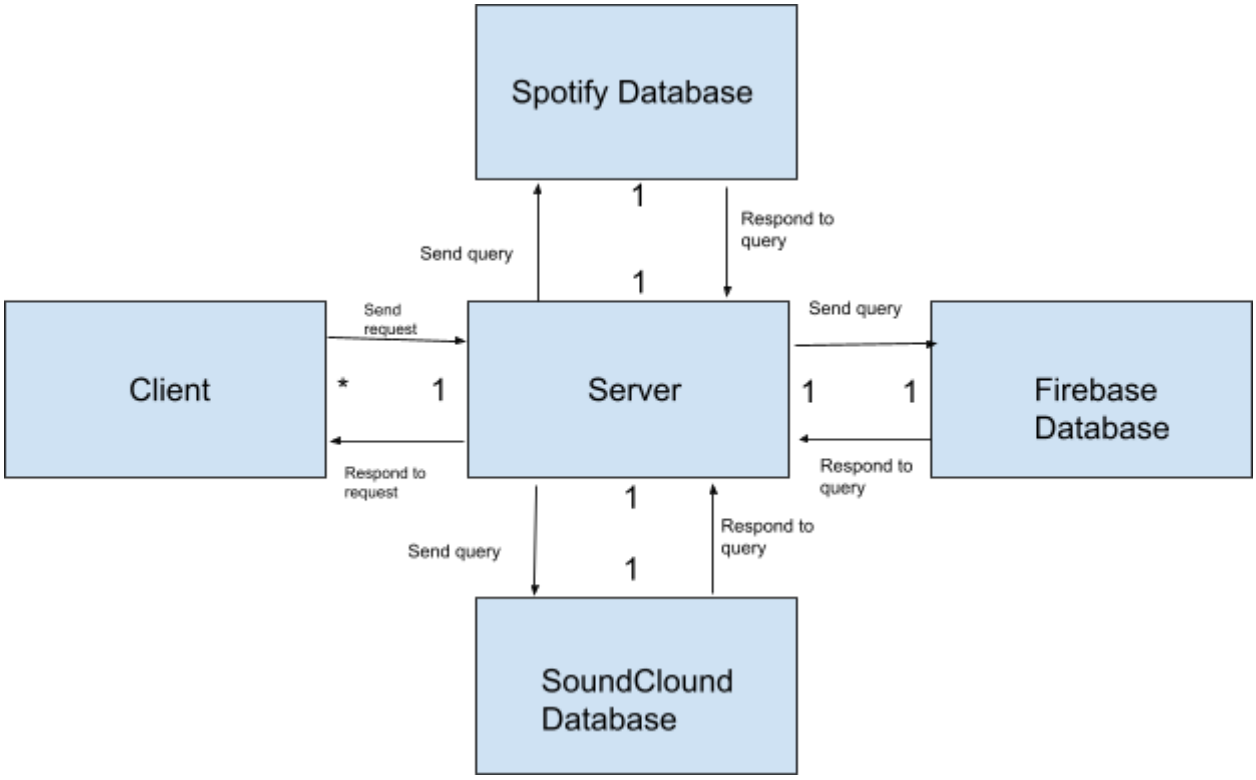
With the rise in subscription-based business models in the modern world, consumers are finding it increasingly difficult to get all of their needs in one place. Take the music streaming industry. Spotify and Soundcloud are two of the largest music streaming platforms in the world. However, they are both constrained by a major issue. The main limitation of these music streaming services is that many artists are licensed to either Spotify or Soundcloud, restricting the access to many songs across both platforms. Further, Soundcloud has a massive pool of unofficial song releases, remixes, and set recordings that Spotify does not have.

The purpose of our project is to create a new music listening web application. Although our group does not have the scope to alleviate issues in every field, we know we can make a difference in one that matters to us most: music. Since there are many different music tastes and a vast number of existing tracks, we've decided to combine two of the largest music streaming platforms into one for streamlined music access. The web app will also contain other quality of life features, such as group listening sessions and bookmarkable skips for parts of songs you don't wish to hear. We also plan on implementing an aesthetically appealing music visualizer.

Design Outline

High Level Overview:

The project will be a web based application that makes use of the client server model. The server will handle requests from many clients querying information from the firebase database along with getting information from the Spotify and SoundCloud databases.



React Frontend

- This will be our front end user interface that allows users access to our system.
- It will utilize http requests to communicate JSON files with our backend and database

- Our frontend will be hosted using github pages

Firebase Backend Server

- Our backend server will act as an intermediary layer that handles requests from the frontend
- Communicate with the database, making queries and entries.
- Handle both the Spotify and SoundCloud API requests
- We will be using firebase to host our backend

Firebase Database

- A relational database to store user info, playlists, bookmarks, and other information.
- Stores data independent of our other systems, which allows us to manipulate our other components without a risk of losing data

Functional Requirements

1. User Account
   As a user,
   a. I would like to be able to register an account at Pineapple music.
   b. I would like to be able to logout of my account for Pineapple music.
   c. I would like to be able to stay logged in after leaving the page.
   d. I would like to be able to delete my account for Pineapple music.
   e. I would like to be able to add a profile picture.
   f. I would like to be able to change my profile picture.
   g. I would like to be able to reset my password if I forget my login information.
2. Spotify-Soundcloud Integration
   As a user,
   a. I would like to be able to use Pineapple music by logging in through my Spotify account.
   b. I would like to be able to change which Spotify account I use.
   c. I would like to be able to use Pineapple music by logging in through my SoundCloud account.
   d. I would like to be able to change which SoundCloud account I use.
   e. I would like to be able to access Spotify tracks through Pineapple music.
   f. I would like to be able to access SoundCloud tracks through Pineapple music.
3. Basic Functionality
   As a user,
   a. I would like to be taken to the home menu when I access the site.
   b. I would like to be able to navigate to a playlists page from the home menu.
   c. I would like to be able to navigate to an account management page from the home menu.
   d. I would like to be able to access a search page from the home menu.
   e. I would like to be able to search through both SoundCloud and Spotfy's music selection simultaneously. (If time allows)
   f. I would like to have the song title be displayed on search results.
   g. I would like to have the song cover be displayed on search results.
   h. I would like to be able to play a song from the search results directly.
   i. I would like to be able to see the current song being played.
   j. I would like to be able to access a page where song specific information is displayed.
   k. I would like to be able to see whether the currently playing song is from Spotify or SoundCloud.

l. I would like to be able to have music playing in the background while I continue to use the website.
m. As a user I would like to be able to switch between SoundCloud music and Spotify music seamlessly.
n. I would like to be able to skip to any part of the song. (If time allows)
o. I would like to be able to manually queue a song to be played next. (If time allows)

4. Playlists

As a user,

a. I would like to be able to create a Pineapple Music playlist.
b. I would like to be able to create multiple playlists.
c. I would like to be able to select a song from the search results to add to a playlist.
d. I would like to be able to select which playlist to add a song to.
e. I would like to be able to see all the songs in a playlist in a scrollable menu.
f. I would like to be able to reorder the songs in my playlist.
g. I would like to be able to delete an entire playlist.
h. I would like to be able to name a playlist.
i. I would like to be able to add a playlist photo. (If time allows)
j. I would like to be able to add Spotify music to my Pineapple Music playlist.
k. I would like to be able to add SoundCloud music to my Pineapple Music playlist.
l. I would like to be able to add my Spotify playlists to Pineapple Music.
m. I would like to be able to add my SoundCloud playlists to Pineapple Music.
n. I would like to be able to merge Spotify and SoundCloud playlists together. (If time allows)
o. I would like to be able to remove songs from my playlist.
p. I would like to be able to queue songs from my playlists.
q. I would like to be able to play a song in my playlist and the rest of the playlist be queued in order.
r. I would like to be able to start a new playlist queue when playing a new playlist.
s. I would like to be able to shuffle songs from my playlist. (If time allows)
t. I would like to be able to repeat songs from my playlist. (If time allows)
u. I would like to be able to skip the current song that is playing.
v. I would like to be able to go back to the previous song that is playing.
w. I would like to be able to pause my Pineapple Music playlist.
x. I would like to be able to resume my Pineapple Music playlist.
y. I would like for my playlists to be saved between instances.

    z. I would like my playlists to dynamically suggest songs according to the overall mood of the songs contained.

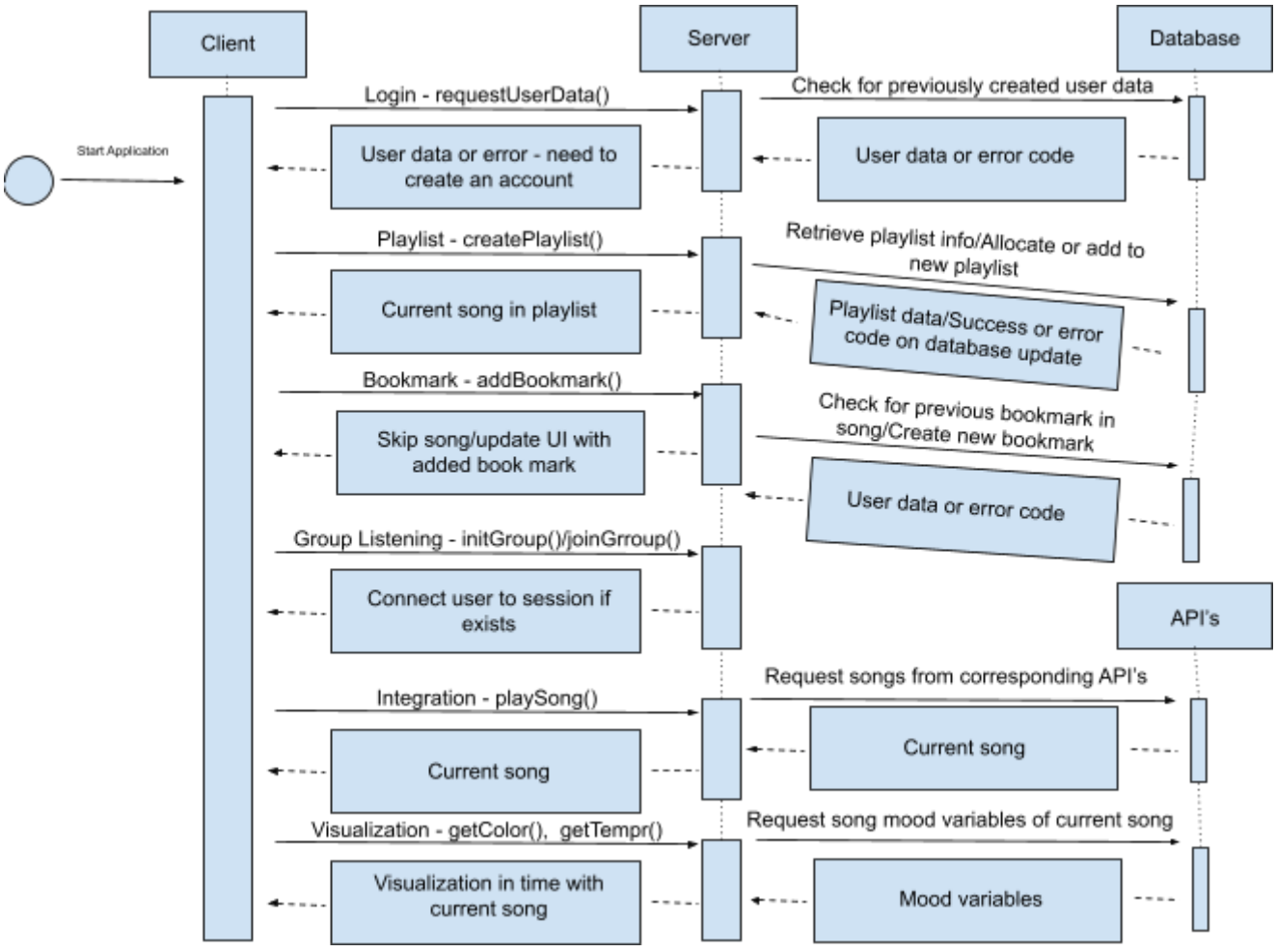
5. Bookmarks
   As a user,
    a. I would like to be able to place a bookmark at any point in a song in order to skip the song after that point when streaming the song.
    b. I would like to be able to edit a bookmark for a song.
    c. I would like to be able to delete a bookmark for a song.
    d. I would like bookmarks to be saved between instances.
    e. I would like to be able to view if a song in my playlist has a bookmark set.
6. Group Listening
   As a user,
    a. I would like to be able to create a group session room.
    b. I would like to be able to decide the group session room code.
    c. I would like to be able to join a group session with a room code.
    d. I would like to be able to leave a group session.
    e. I would like to be able to enqueue a playlist in a group session.
    f. I would like to be able to enqueue a single song to a group session.
    g. I would like to be able to skip songs in the queue.
    h. I would like to call a vote to skip a song in the group session queue.
    i. I would like to be able to pause the group playlist.
    j. I would like to be able to resume the group playlist.
    k. I would like to be able to react in real time while in a group session.
    l. I would like my reaction to show up to the whole group.
7. Music Visualizer
   As a user,
    a. I would like to be able to view a music visualizer for the current playing song.
    b. I would like to be able to view a music visualizer that responds to the beat of a song.
    c. I would like to be able to view a music visualizer that responds with color to the dominating frequency of the music.
    d. I would like to be able to manage the music visualizer color settings.
    e. I would like to be able to manage the music visualizer animation settings.

Non-Functional Requirements:

1. Architecture
   As a developer,
   a. I would like our application to run 24 hours a day.
   b. I would like our application to deliver at least 5GB of data to users
   c. I would like our application to allow up to 100 simultaneous connections
2. Security:
   As a developer,
   a. I would like our application to support user authentication
   b. I would like our application to be secured against SQL injections and other common exploits
3. Usability:
   As a developer,
   a. I would like our user interface to be visually pleasing
   b. I would like our user interface to be structured similarly to other music streaming platforms
   c. I would like our user interface to have access to important features on the same page as the home menu

Sequence Diagram

This sequence diagram depicts the interactions between the client, server, database, and API's. Requests related to login, playlists, and bookmarks are all sent through the client to the server, to the database and back. Group listening requests are simply handled by the server as there is no information stored in the database for these sessions. Requests related to playing songs through the integrated Spotify and SoundCloud platforms and visualization are sent through the server to the API and back again to retrieve the current song and the needed mood variables respectively.



Design Issues

Functional Issues

1. What information do users need to login to use our service?
    - Option 1: No login id creation
    - Option 2: Allow the creation of login using Spotify and SoundCloud
    - Option 3: Allow the creation of username and password that is unique to our service using Spotify and SoundCloud logins

Choice: Option 3

Justification: We chose to allow the creation of username and password that is unique to our service. We should be able to provide all the features and services that are distinct to each user. For instance, each user will have their own profile and will be able to save and edit their playlist, song tags, and more.

2. How will we allow users to communicate in their group listening session?
    - Option 1: Post reactions
    - Option 2: Real time chat room

Choice: Option 2

Justification: We show real time reactions to the songs of each user in the group session. Having a real time chat room will make the group listening session too clustered. Implementing a real time reaction will be easier and put less burden on the server than a real time chat room.

3. How do we save the bookmark of the songs?
    - Option 1: Stored under user data
    - Option 2: Stored under song data

Choice: Option 1

Justification: The bookmark data will be saved under each user. This will allow more efficient retrieval of the bookmark data that can be applied in real time. It will also avoid each song's data being too clustered.

4. How do we create playlists?
    - Option 1: Create playlists using raw data from Spotify and SoundCloud API
    - Option 2: Create playlists using our own Song class

Choice: Option 2

Justification: When using any of Pineapple music’s features, it is necessary to decide which songs are coming from which platform. In order for users to have the smoothest experience, it will be best to have our own Song class for users to enjoy all the features,

including creating playlists, queuing up the songs, and group listening sessions.

5. How will we allow users to join a group listening session?
    - Option 1: Join by code
    - Option 2: Join by invite link

Choice: Option 1
Justification: Users will be able to share the group listening session code, which will be generated when a user creates a new group session. This will be usable as many times as possible and users will be able to join the session directly and efficiently.

Nonfunctional Issues

1. What database should we use?
    - Option 1: MySQL
    - Option 2: Firebase (Cloud Firestore)
    - Option 3: Oracle XE

Choice: Firebase (Cloud Firestore)
Justification: We will use Firebase as our database because we want a NoSQL database that can handle user data dynamically. It will be appropriate for our data sets as it will be more scalable as well.

2. What frontend language/framework should we use?
    - Option 1: raw HTML + Vanilla JavaScript
    - Option 2: React
    - Option 3: Angular

Choice: React
Justification: React has an easier learning curve compared to others. Since the components are self-contained, it has easy maintenance and improvement. Huge variety of UI components are offered. Moreover, The virtual DOM trees do not overload the browser making it have relatively faster performance.

3. What backend language/framework should we use?
    - Option 1: PHP
    - Option 2: Django (Python)
    - Option 3: Node.js (JavaScript)

Choice: Node.js (JavaScript)
Justification: Node.js offers fast data processing and client-server interaction, and speed of development. It is suitable for event-driven two-way connections. With Node.js, code can be shared in both the front and backend parts of the application. There are a lot of available documents to accelerate the development process.

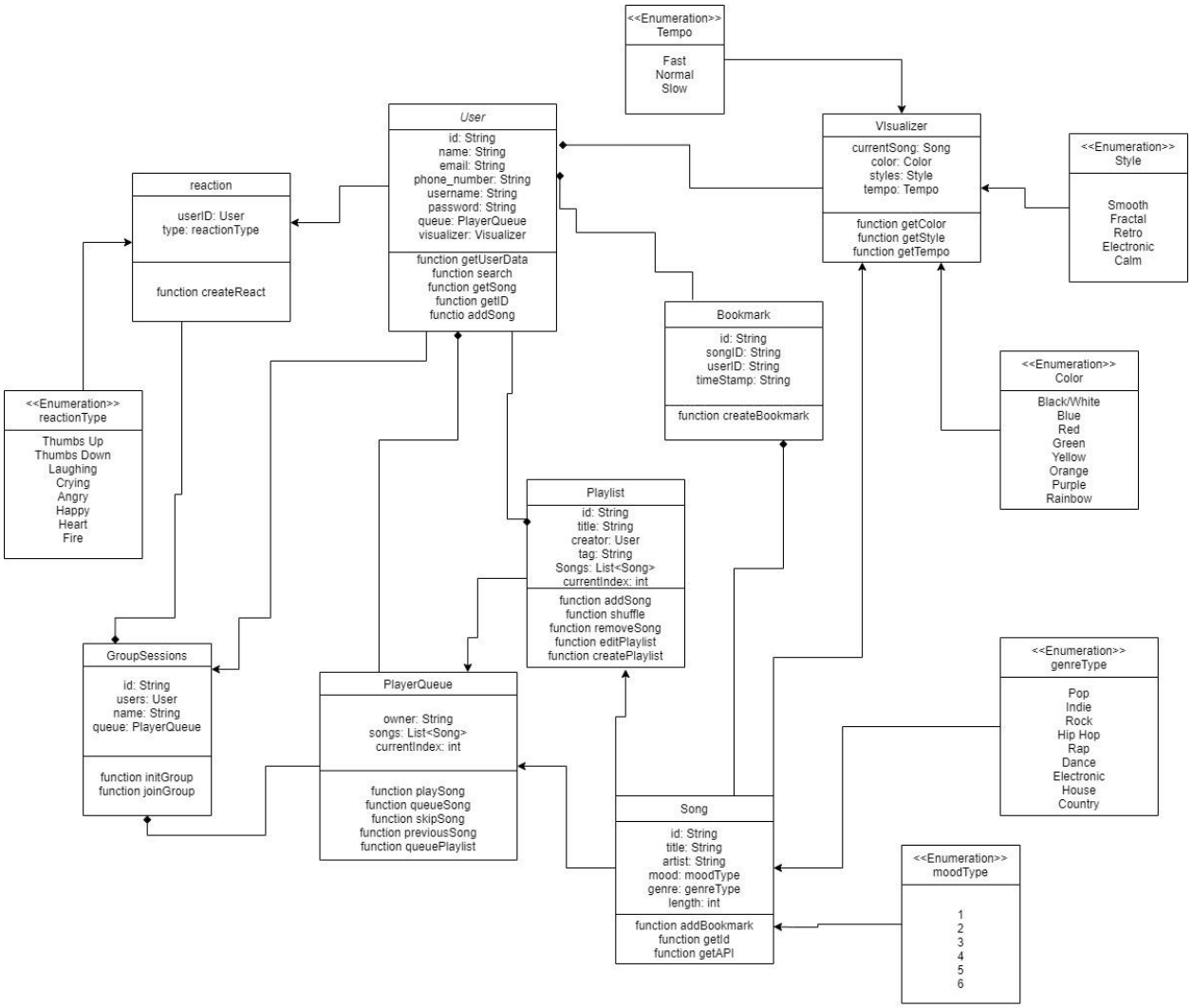
4. Which API should we use to access user related data?
    - Option 1: Spotify
    - Option 2: SoundCloud
    - Option 3: Spotify & SoundCloud

Choice: Spotify & SoundCloud
The main feature of the application allows users to create a playlist from both Spotify and SoundCloud. The Spotify API uses appropriate HTTP verbs to provide actions of GET, POST, PUT, and DELETE. The Spotify Web API endpoints return JSON metadata about user and music-related data. The SoundCloud API works in a very similar way.

The access tokens will be retrieved from both APIs and will periodically refresh the access using refresh tokens.

Design Details

Class Design



Description of Classes and Interaction Between Classes

The classes that we use in our application define an object built by our object oriented java backend.

These objects are connected to each other through our application and we show this visually using our Class Design Diagram. Defined below are the classes we build in our program.

User

- User objects are instantiated each time a new account is created.
- User objects contain its own unique id, which we create using a String
- User objects contain a name, which is the User's first and last name. We create this using a String
- User objects contain an email, which is the email the User used to create the new account. This value is represented using a string.
- User objects contain an optional phone number, which is the phone number the User used to register the account. This value is represented using a string.
- User objects contain a username, which is the username the User created when registering the new account. This value is represented using a string.
- User objects contain a password, which is the password the User created when registering the new account. This value is represented using a string.
- User objects contain a queue, which is inherited from the playerQueue class. This corresponds to the queueing function of the music player.
- User objects contain a visualizer, which is inherited from the Visualizer class. This represents the user's visualizer settings and the visualizer itself.

Playlist

- Playlist objects are instantiated upon request of the user each time they wish to create a new playlist.
- Playlist objects contain a unique id which will be randomly generated as a u_uid.
- Playlists objects contain a string name as specified by the user
- Playlist objects contain a tag…*** (Ask Alan)
- Playlist objects contain a list of song objects as the user specifies
- Playlists will be inserted onto the player/queue at the user's request

Song

- Song objects are instantiated each time a user utilizes one of the custom functions of Pineapple Music, including queuing a song, adding a song to a playlist, and creating a bookmark.

- Song objects contain an id, represented by a string, that will be used to identify the song object in the database.
- Song objects contain a title, represented by a string, that will be used by the player.
- Song objects contain an artist, represented by a string, that will be used by the player.
- Song objects contain a mood, represented by a moodType, that will be used by the visualizer.
- Song objects contain a genre, represented by a genreType, that will be used by the visualizer.
- Song objects contain a length, represented by an int, that will be used by the player.

Bookmark

- Bookmark objects are instantiated every time a User creates a new bookmark for a specific timestamp on a song.
- Bookmark objects contain an id, represented by a string, that we will use to identify the song the bookmark was created on.
- Bookmark objects contain a timeStamp, which represents the place in the song that the User created a bookmark for.

PlayerQueue

- PlayerQueue objects are instantiated a single time for every user. This object represents each user's unique queue.
- PlayerQueue objects contain an owner, which is represented as a String. This represents the User's id of the User this PlayerQueue corresponds to.
- PlayerQueue objects contain a list of Song objects that represent the song the Users add to their queue.

Visualizer

- Visualizer objects are instantiated a single time for every user. This object represents each user's unique visualizer.
- Visualizer objects contain a currentSong, represented by a String. This represents the current song that the visualizer is playing.

GroupSessions

- GroupSessions objects are instantiated when a user chooses to begin a new group listening session.
- GroupSessions objects contain users, represented by a List of User objects, which represent the users currently connected and change as users join and leave.
- GroupSessions objects contain a name, represented by a String, which will be used by the player.

- GroupSessions objects contain a song queue, represented by a PlayerQueue, which will be used by the server to determine the next song to play in the session.

Reaction

<<Enumeration>> Tempo

- The tempo Enum class defines all possible tempos.

<<Enumeration>> Style

- The style Enum class defines all possible styles.

<<Enumeration>> Color

- The color Enum class defines all possible colors.

<<Enumeration>> moodType

- The moodType Enum class defines all possible moods.

<<Enumeration>> reactionType

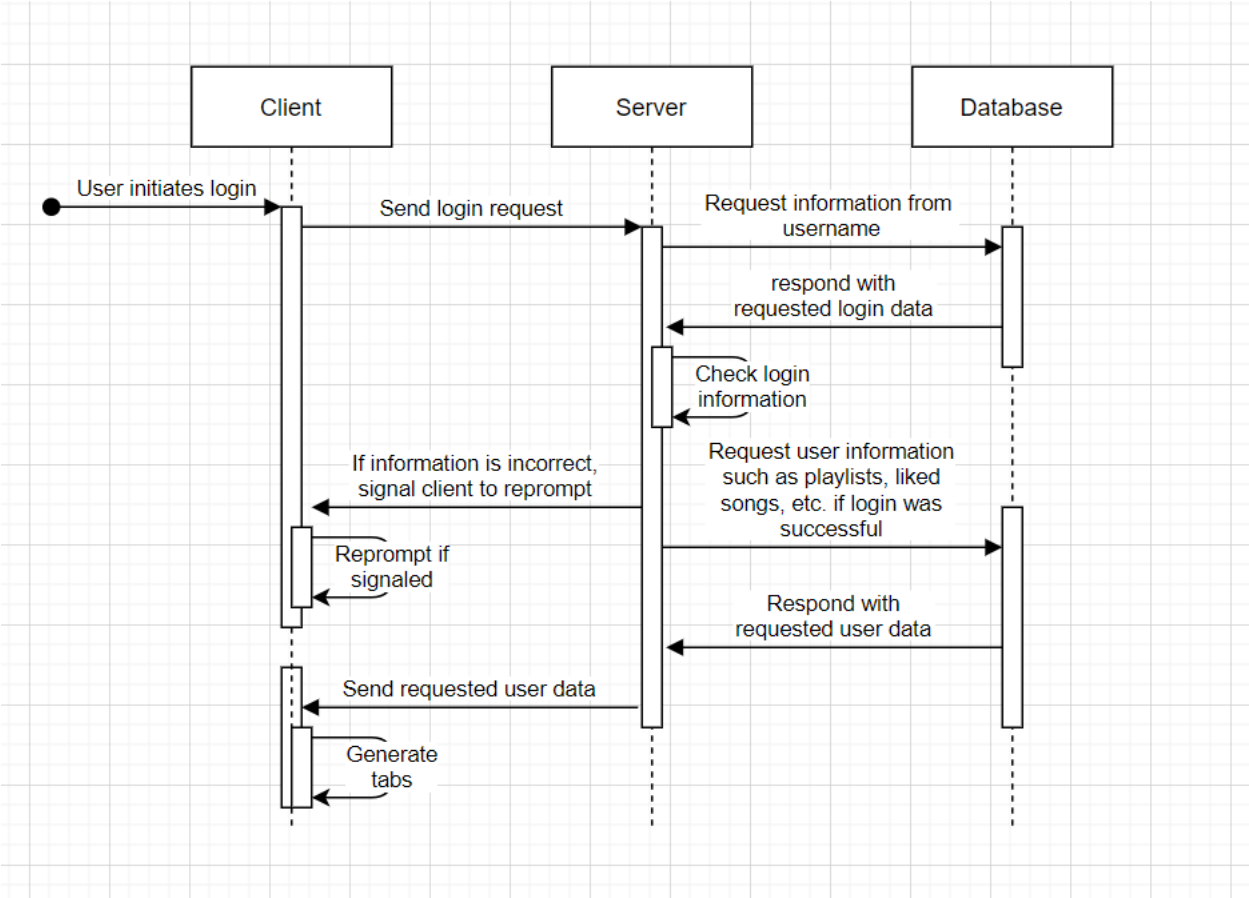
- The reactionType Enum class defines all possible reactions.

<<Enumeration>> genreType

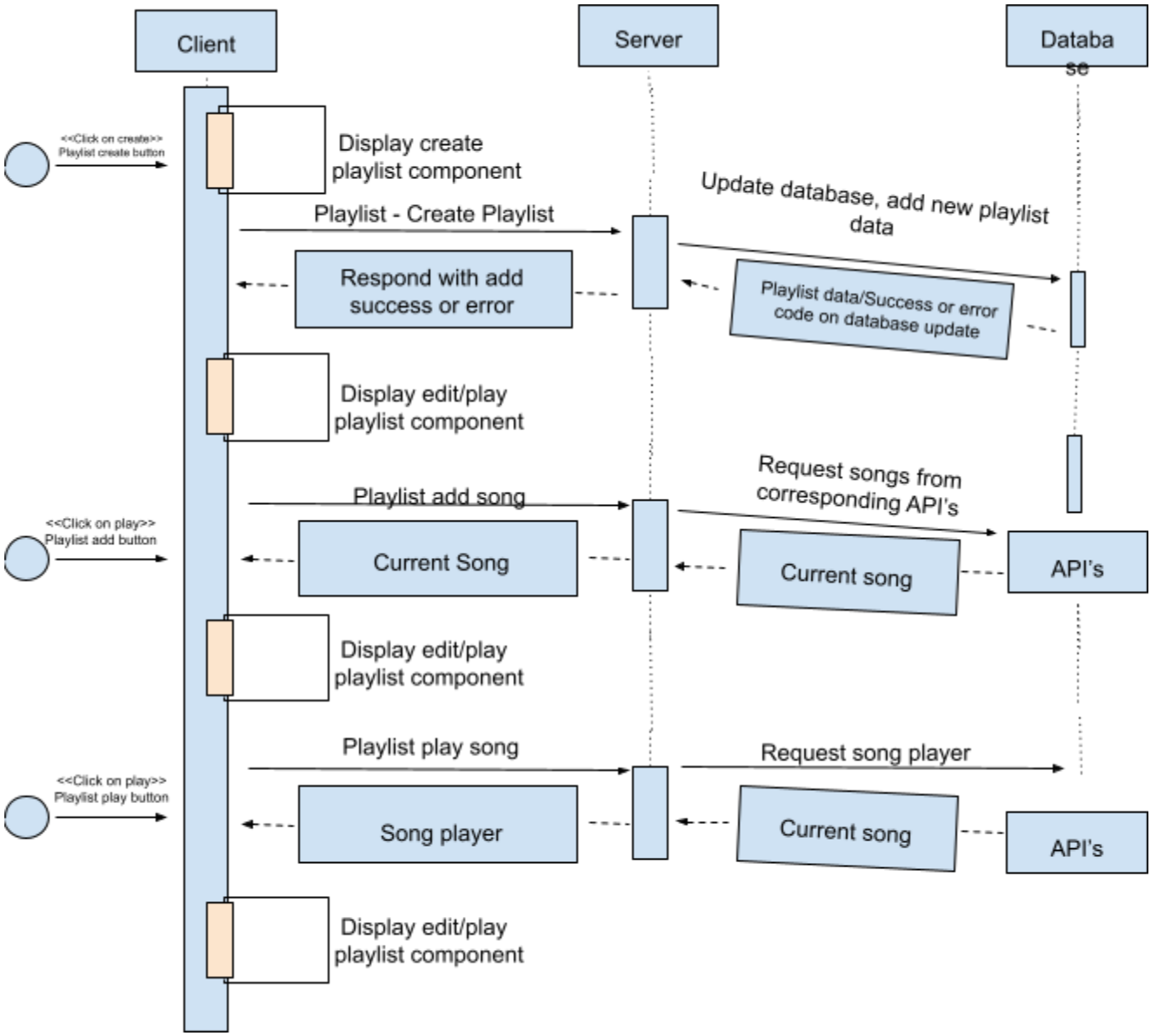
- The genreType Enum class defines all possible genres.

Sequence Diagram

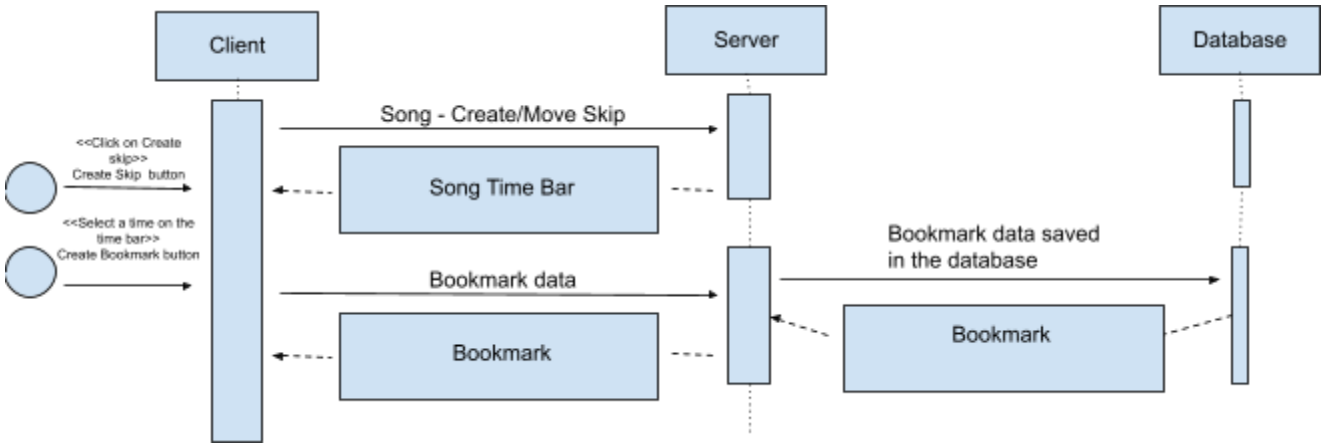
1. Sequence of events when users login/register



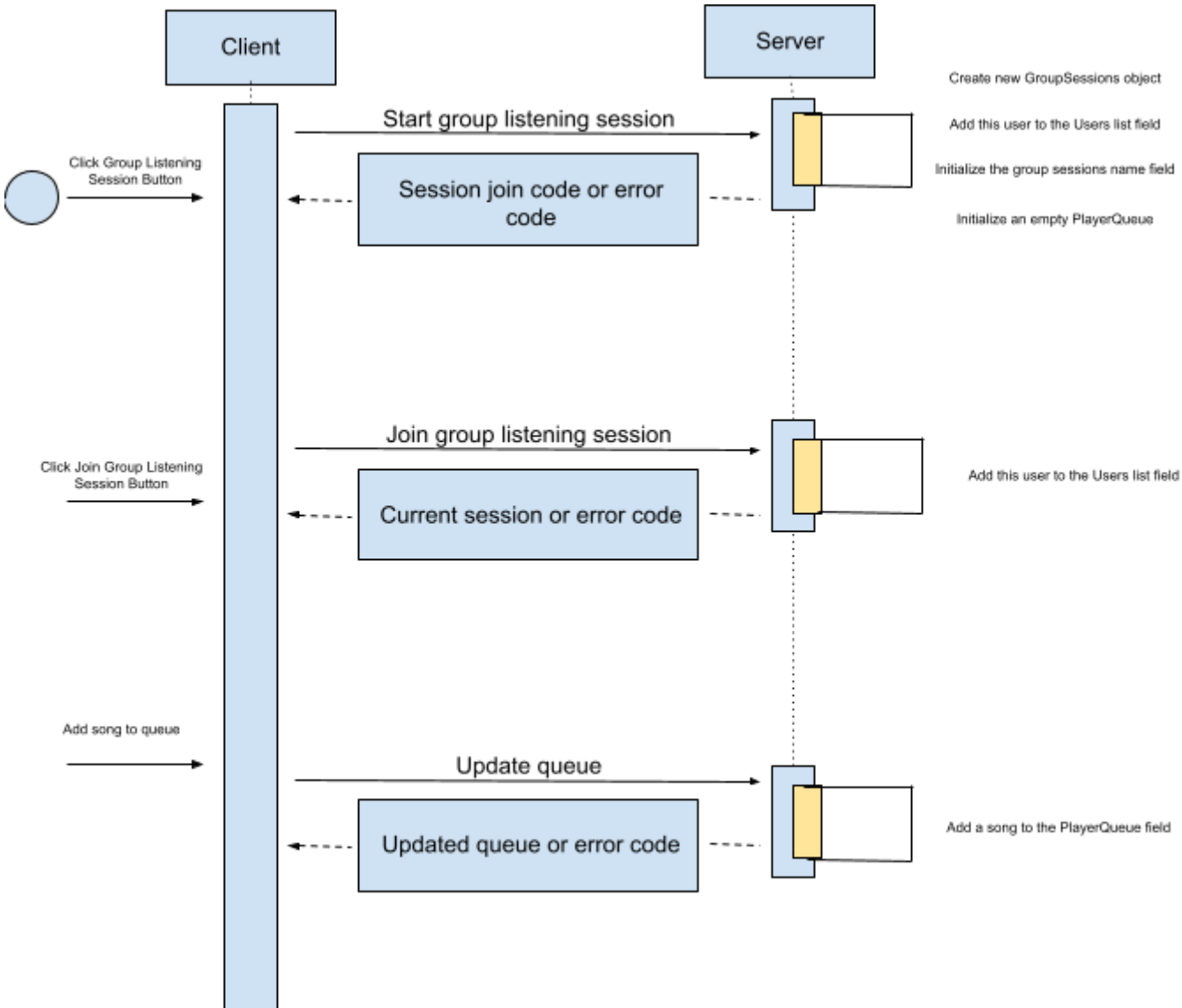
2. Sequence of events when users create a playlist



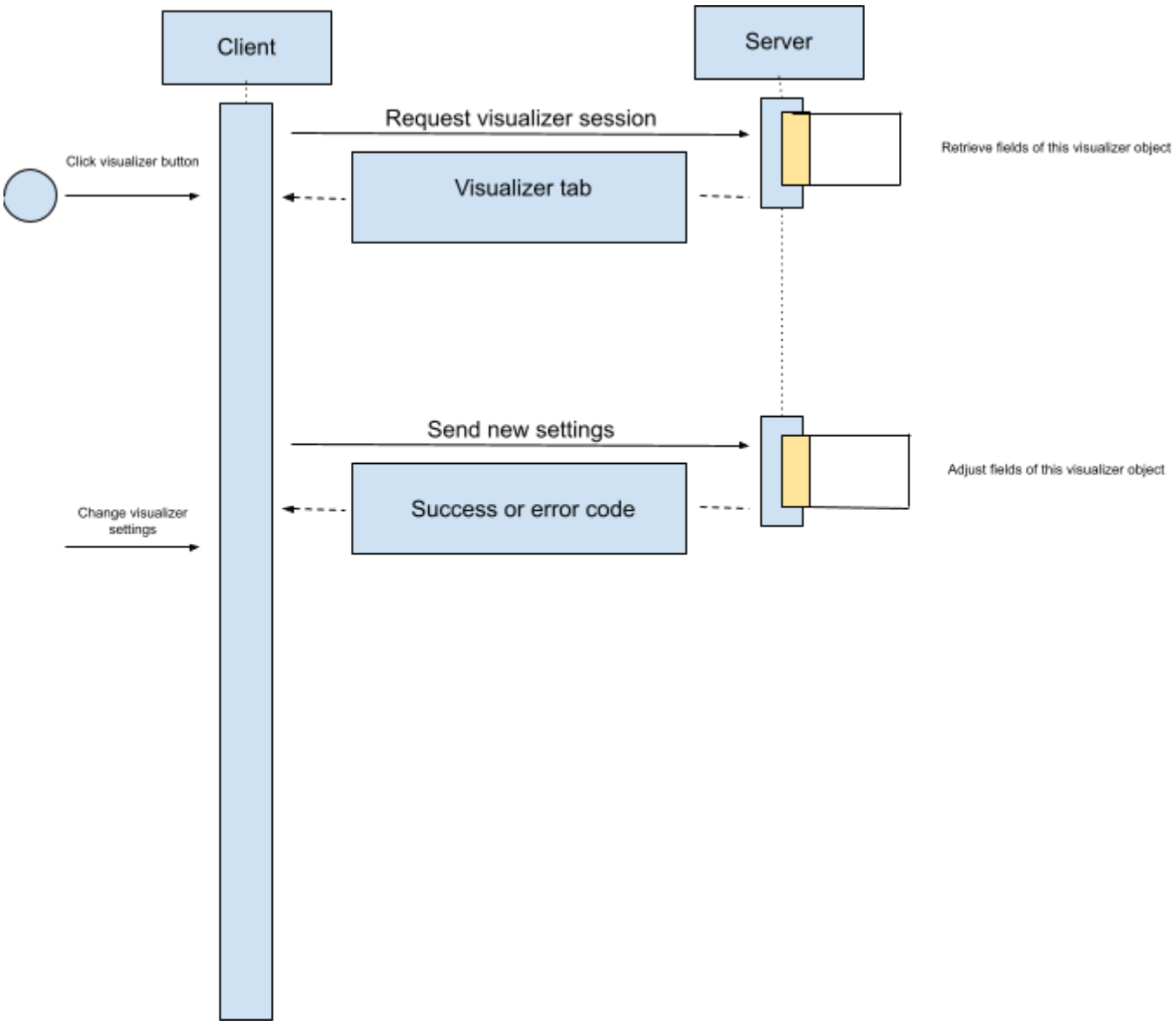
3. Sequence of events when users create a bookmark



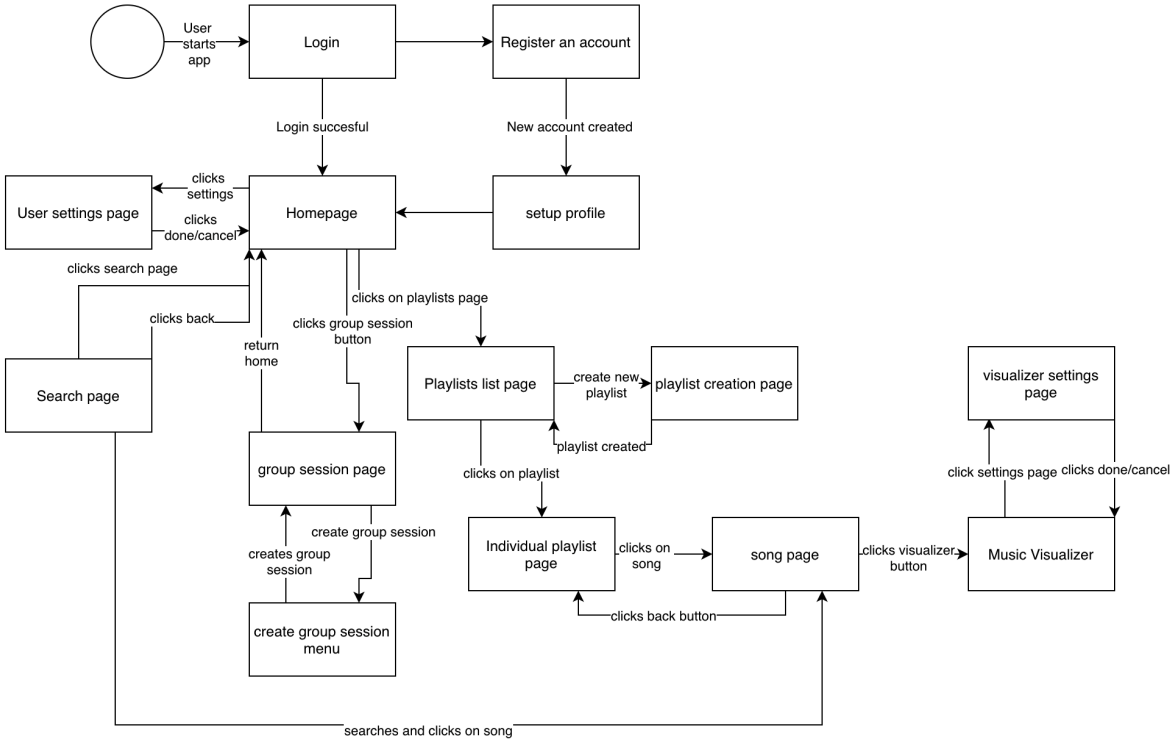
4. Sequence of events when users join a group session



5. Sequence of events when users view the visualizer



Navigation Flow Map



Our navigation is centered around the homepage which is accessible through the login page. A constantly displayed navigation bar will allow for navigation to the homepage from any page on the website even though not displayed on the flow map.

UI Mockups

- Login page

- Home


Home Page
Search
Group Listening
Visualizer
Playlist
Bookmark
Track Cover Art

-

- Visualizer