JudgeMe

Design Document

09/23/2022

Team 6

Aarushi Singh

John Robinson

Kenny Park

Nick Pancione

Sujal Timilsina

Index

1. Purpose

Spotify is a webapp that allows users to stream music to their phone, create playlists, and find new songs based on the characteristics of a song, playlist, or artist.

While it has some useful features for curating and comparing playlists and music taste between users, this sort of functionality is lacking in regards to groups; music taste can vary greatly in friend groups, and it can be hard to find music to play in the car that everyone can vibe with. There is no feature that compares the music taste of a user with the music taste of their group of friends, and there is also no feature that generates a playlist based on the interest of a group of people.

JudgeMe was created with the intent to give Spotify listeners more insight into how their music taste aligns (or varies) with their friends. The domain of the application is music and social media, aimed at college-aged users who would be interested in using this sort of data.

Spotify has some similar features for finding similar through its new Blend feature, but it is limited in simply listing a couple shared songs and giving a vague percentage of how closely users' music taste matches. There are also many webapps out there to analyze a user's own music taste, but not to compare with others.

Our solution gives more control and elaboration regarding how closely your taste matches with your friends, which also extends into friend groups. Additionally, JudgeMe will give more options for playlist creation, allowing you to mix between your friends' playlists as well as make remixes of their existing playlists.

Functional Requirements

1) Users can connect to the service with their Spotify account.

   As a user

   a. I want to easily share my listening information with judgeme

   b. I want to get listening statistics on my listening habits

   c. I want a comparable breakdown of my music tastes

2) Users can connect with their friends through their Spotify accounts.

   As a user,

   a. I want be able to invite friends to JudgeMe using any social media

   b. I want to see my friends represented in JudgeMe with their spotify profile picture

   c. I want to be able to keep my listening private to people who are not my friends

3) Users can compare their own music tastes to their friends.

   As a user,

   a. I want my friend rankings to be displayed as gold, silver, and bronze rankings when I first log onto JudgeMe

   b. I want to see graphical representations of my music tastes

   c. I want to see graphical comparisons between my friends music tastes and my own

   d. I want to send my friends insulting, or complementing gifs based on our music comparisons

   e. I want to be able to compare specific playlists with each other

4) Users can generate new playlists based on their friends' listening habits and my own.

   As a user,

   a. I want to create playlists with songs based on multiple artists that I choose
   b. I want created playlists to be supported by my friends listening habits
   c. I want to be able to mix multiple playlists from my friends
   d. I want to be able to make playlists based on my friends playlists, from specific artists, genres, or moods
   e. I want to select multiple people and make playlists that blend our music tastes

5) Users can compare their tastes to an artist or genre.

   As a user,

   a. I want to be able to get artist recommendations based on my previous listening habits
   b. I want to be able to find the most compatible songs within an artist or genre

Non-Functional Requirements

1) Client Requirements

   As a developer,

   a. I want the application to be compatible with any common web browser

2) Server Requirements

   As a developer,

a. I want the server to be able to save some user data to a local database, so that comparisons can be made when all users are not logged in
b. I want both a production and a developer database
c. I want both a production and a developer server

3) Appearance Requirements

As a developer,

a. I want the application to be aesthetically pleasing
b. I want the layout to be intuitive and easy to use

4) Security Requirements

As a developer,

a. I want to store the minimum for user data
b. I do not want to store any login information for Spotify
c. I want to encrypt, protect, or anonymize any data that is stored

5) Performance Requirements

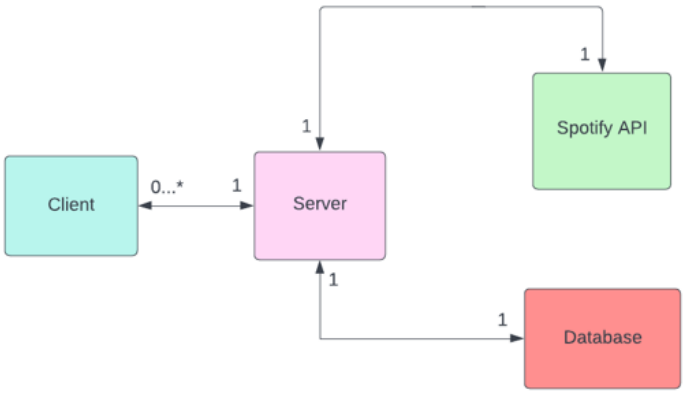
As a developer,

a. I want the app to run quickly and seamlessly
b. I want to load partial accounts only, to avoid long loading periods scanning through entire spotify libraries
c. I want to limit Spotify API calls, to avoid slowdowns

2. Design Outline

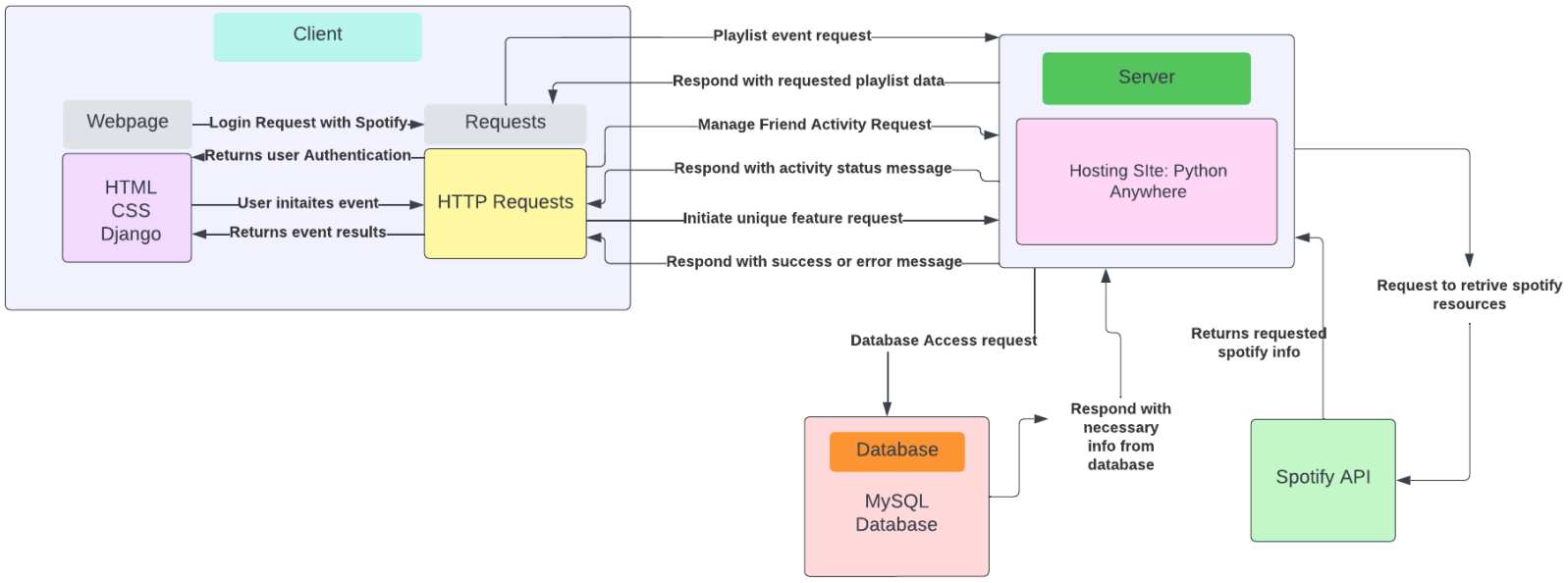
2.1 High Level Overview of the System

The structure of the project follows a simple fullstack framework using Django where the user sends in some request as a client through a webpage. At a given time, we can have multiple users sending requests as a client to the server. Those requests can consist of anything from searching for a song to managing friend requests within JudgeMe. Once the request is sent, the server handles the request and depending on the request, accesses our MySQL database or uses the API layer provided by Spotify.If the request goes to our database, we will execute queries in the database to isolate fields of interests and return those results to our server. On the other hand, if the server requires the use of spotify specific data, we would use our API layer to access that information and return it to our server. The server will then perform the necessary actions on the event type and send those results to the webpage in an appropriate format.



2.1 System Architecture

For the specific system architecture, we will develop our frontend using CSS and HTML to provide an appealing interface for user interactions. Our server will be able to handle multiple clients or “users" at a given time allowing for social interaction amongst the users. For events of any type, we will send a HTTP request object created by Django which will be processed and handled by the server and the server will use PythonAnywhere to develop and host our website. The HTTP request will be processed through our server and depending on the event type will retrieve data through the Spotify API using methods like GET, POST, PUT or DELETE. These would return the song id, genre, song url and other unique information. If the request is directly related to our database, we would access our database through SQL queries, isolate the relevant information and retrieve it to the server. Furthermore, if there are any actions that are to occur between our Spotify API, our API would pass it to the server which would then pass it to the database. Upon finishing the necessary actions, our server would return the results of either our database access or Spotify API results, through packets to the client. The client would handle the results and format the results in an appropriate web format for the user’s convenience.



3. Design Issues

Functional Issues

How do we determine similarity of music taste?

- Option 1: By associated personality traits e.g. conscientiousness
- Option 2: Similarity of fan base
- Option 3: Group opinion with genre spectrum

Choice: By associated personality traits

Justification: When finding similarity, we needed to put the genres of music on a spectrum based on some sort of metric to see how "far off" they are for each other. This will make graphical representations much easier and allow us to determine if someone's music taste is more or less similar to another's. Spotify actually has put some research and development into finding this spectrum, and found that music taste is "best able to determine emotional stability and conscientiousness." Using this spectrum provided by Spotify, we can save on computing power from determining similarity of fan bases and have a more scientific approach than forming a spectrum from group opinion. This also provides a basis for possible future features - like telling users what their music data says about their personality.

How do we compare music taste?

- Option 1: Determine most played artists and then similarity between artists
- Option 2: Determine the most played genres and then decide similarity between genres

Choice: Determine the most played genres and determine similarity

Justification: When comparing overall music taste, we have two relatively simple approaches. The first approach is to see how much of each artist they listen to, and then find out if the other person listened to the same artists. When the two people inevitably have interests in different artists, we then find the similarity between them (hence comparing fan bases). The other approach is to do the exact same thing but with genres, creating fewer variables and removing the requirement of comparing artists' fan bases.

How should we allow users to interact with each other?

- Option 1: Third party platforms e.g Spotify, SMS, Instagram
- Option 2: JudgeMe friend implementation

Choice: Third Party

Justification: Allowing users to interact in a way that they are used to is important for the user's comfort when using JudgeMe. There are already countless platforms that allow users to chat and follow others that they know, including Spotify. Using Spotify's social implementations will make for less of a barrier to using JudgeMe while simultaneously keeping JudgeMe simple and easy to use.

How should we communicate differences in music taste?

- Option 1: Graphical representations
- Option 2: Percentages/numerical representations

Choice: Graphical representations

Justification: When communicating the differences in music taste, we want the user to have something interactive and something that they'd want to share with others for more exposure to the platform. Having a visual representation will be a much better design decision for the future success of JudgeMe.

How should we communicate the user's listening statistics?

- Option 1: Numerically whenever user requests stats
- Option 2: Calendar-based and non-numerical e.g. "You listened to [artist] the most this week"

Choice: Calendar-based and non-numerical

Justification: When communicating listening statistics, we wanted something unique that will engage the user. If we use numerical statistics, it will allow the user to know their music habits really well, however, if everything on JudgeMe is done per user action, JudgeMe will quickly become a one-time use application. By pushing occasional notifications per the user's preferred amount of notifications, JudgeMe can send meaningful notifications about the user's listening habits, engaging them with the platform more regularly.

Non-Functional Issues

What front-end and back-end framework should we use?

- Option 1: Flutter
- Option 2: Django
- Option 3: React

Choice: Django

Justification: When it comes to front-end frameworks, we are looking for something that is secure, fast, easy to use, and can rapidly develop a web application. React's performance is lacking compared to Django, and Flutter does not have the same built-in security features of Django. JudgeMe may also need a machine learning implementation, making Python, the language used for Django, the best option out there. We also wanted to pick a framework that at least one of our group members had some experience in, and that narrowed things down to Django and React.

What CSS framework will we use?

- Option 1: Bootstrap
- Option 2: Tailwind
- Option 3: Bulma

Choice: Tailwind

Justification: When choosing a CSS framework, we want something that is mobile-first, has an easier learning curve, but still gives us a lot of design freedom. While Bootstrap is very performant and Bulma is very easy to use, Tailwind is good enough in both of those categories but still allows us the most design freedom. JudgeMe is not a web application oriented around a workflow or a necessarily functional toolset, it is centered around the entertainment and social experience for the user. Because of this, it is important we have the design freedom to give the user the best visual experience possible.

What database should we use?

- Option 1: MySQL

- Option 2: SQLite
- Option 3: PostgreSQL

Choice: SQLite

Justification: Knowing that we will be using Django as our front-end framework, it was important that we picked a database that worked the most seamlessly with it. This brings us to three options that increase in power and featureset, yet decrease in ease of use. JudgeMe will not be storing an immense amount of information because of the use of the Spotify API, bringing us to use the working out-of-the-box with Django: SQLite. However, if we run into a situation where a more advanced database is needed, MySQL and PostgreSQL are both good options for working with Django.

What web service should we use?

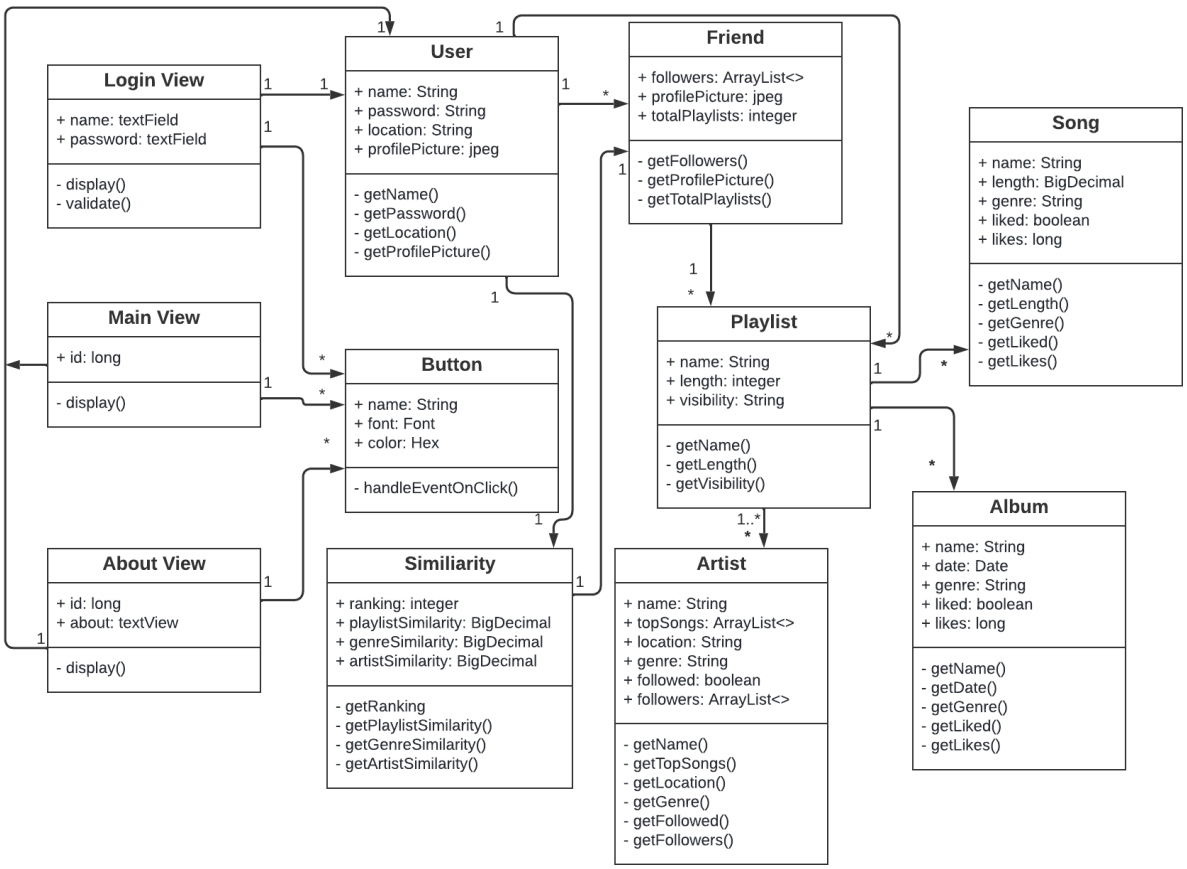
- Option 1: Firebase
- Option 2: PythonAnywhere
- Option 3: AWS

Choice: PythonAnywhere

Justification: When our group was looking for a web service to use, we wanted something that is affordable, easy to use, and can interact with Django seamlessly. Firebase comes in as being the easiest to set up and AWS being the most advanced, feature-rich service, with both being affordable for our use case. PythonAnywhere, however, has a good feature set that is still easy-to-use and is all done in Python. In regards to connecting with Django, PythonAnywhere has a quick setup guide specifically for Django, which keeps our development cycle in one programming language, and allows for rapid development.

4. Design Details

Class Diagram



Description of Classes and Class Interactions

User

This is the class that is used to access the Spotify API. This has the Spotify Link. This class contains the user's Spoify username, location, and profile picture. The user is also unique for every user that uses JudgeMe.

Login View

This is the first page that shows up when a user visits the web application. The user can enter their Spotify username and password. Then the view makes a call to the server, which then pulls from the Spotify API to update the database if the user is not in the database or if the user's information needs to be updated.

Main View

Once information about the user is received, the database sends the server the information for the requested user. Then the server sends it to display. The main view is the home page that gives options for a user to generate playlists. It also displays the ranking of the friends of the user. There is also the about button at the top.

About View

When the about button is clicked, the user is directed to this page. This page essentially displays text that describes what JudgeMe is about. It explains the features, the uses, and also how to navigate the website.

Button

This class is used by the pages of the website to activate an event. Each button has a name and serves a different function. When the button is clicked, the event is performed. Each button also has a color and font if it contains text.

Similarity

This class contains how similar a friend is to a user. This is done through artist, genre, and song similarity. Then a ranking is made. The top 50 songs for each friend would be compared. A percentage that compares how many songs are the same is made. A percentage for artists that are

the same is made. A percentage that compares how many genres are the same. The genre is going to be weighted in the calculations to determine this. The same is done for playlists. Then a overall ranking is made.

Friend

This is the people the user follows on Spotify. The friend has their own playlists. They also have a name, profile picture, and total number of playlists.

Playlist

A playlist is composed of different songs. It has the length of all the songs added up, a name, and visibility. This can be between friends, public, or private (visible to the user only).

Artist

The artist class represents an artist on Spotify. Each artist has a name, genre, and 5 top songs. An artist can also be followed by a user. The artist class also has the total number of followers for the artist.

Song

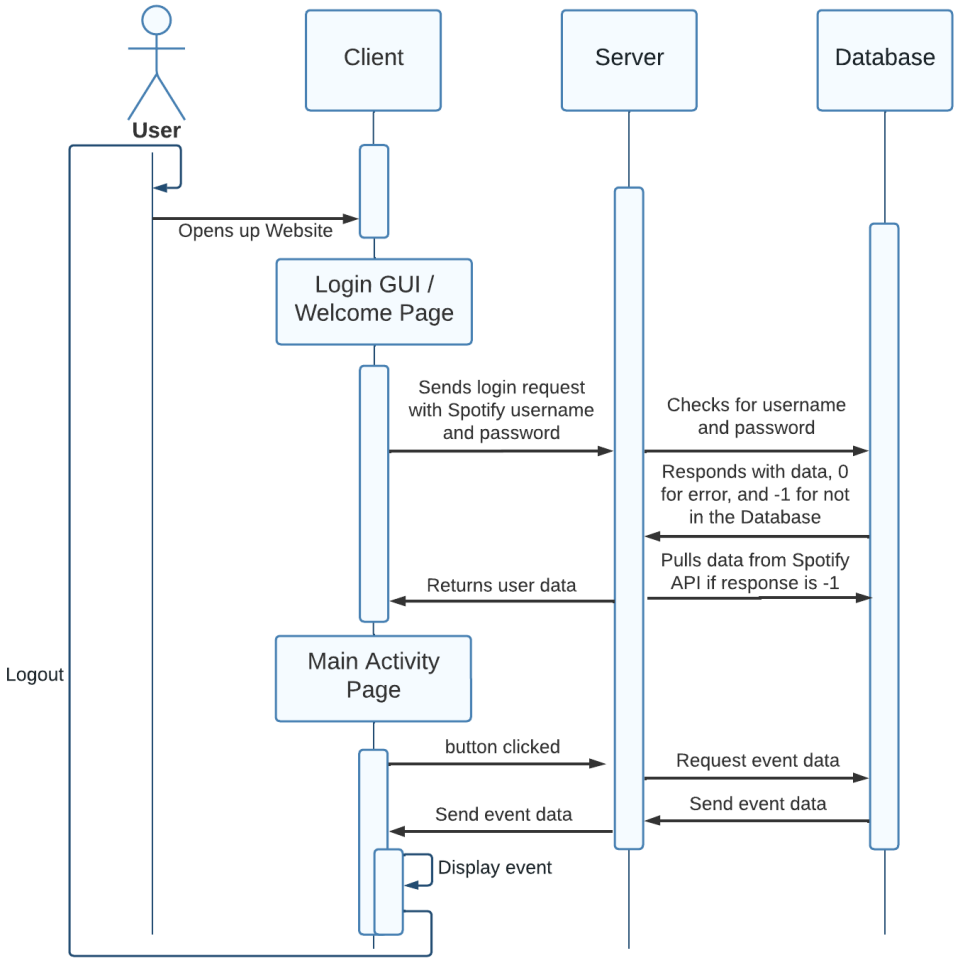
Each song has a genre, length, and name. It also is connected to an artist. The song has a total number of likes, and it can be liked by the user.

Album

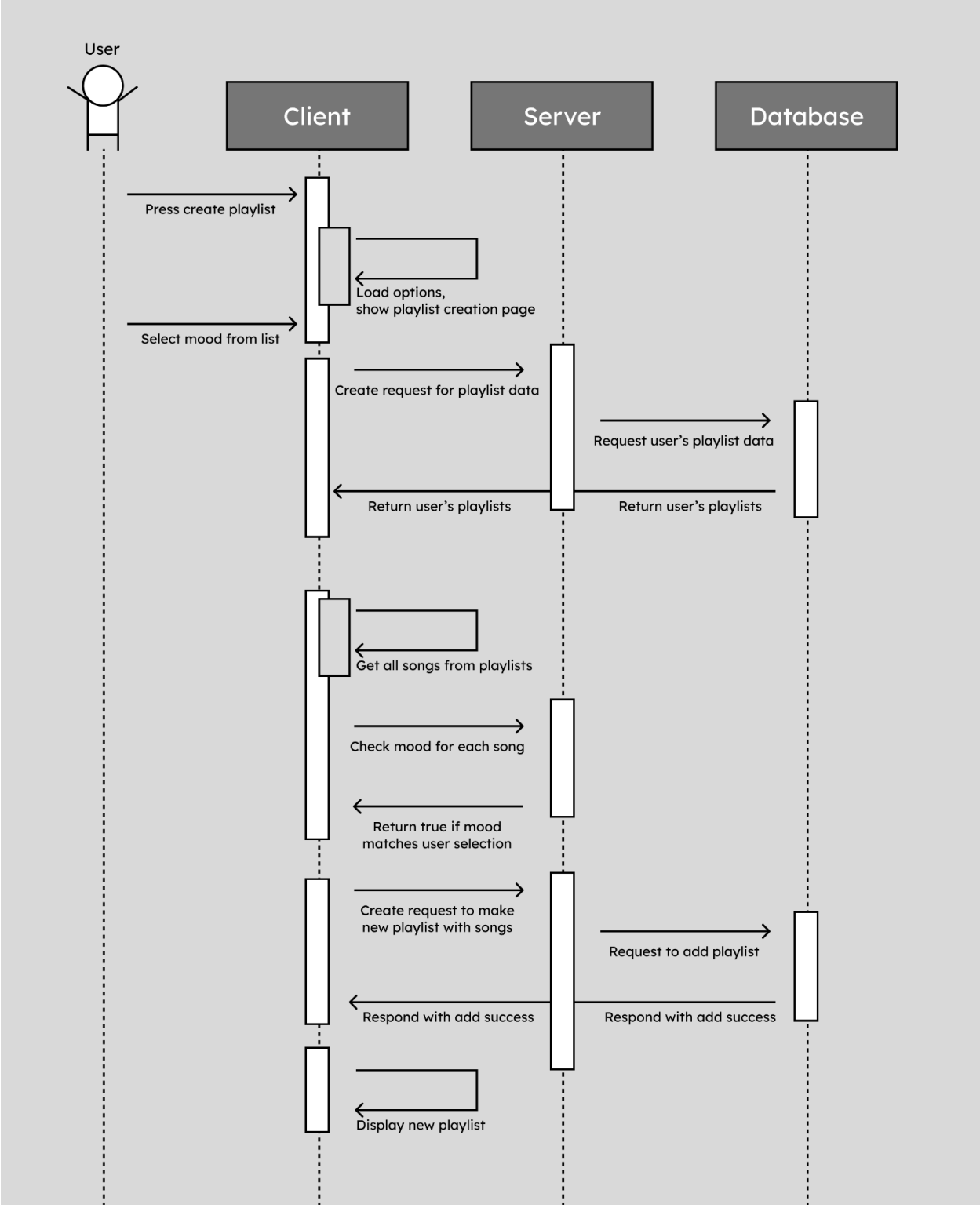
Each album has a set number of songs, likes, name, release date, and total likes. An album can also be liked by the user.

Sequence Diagram

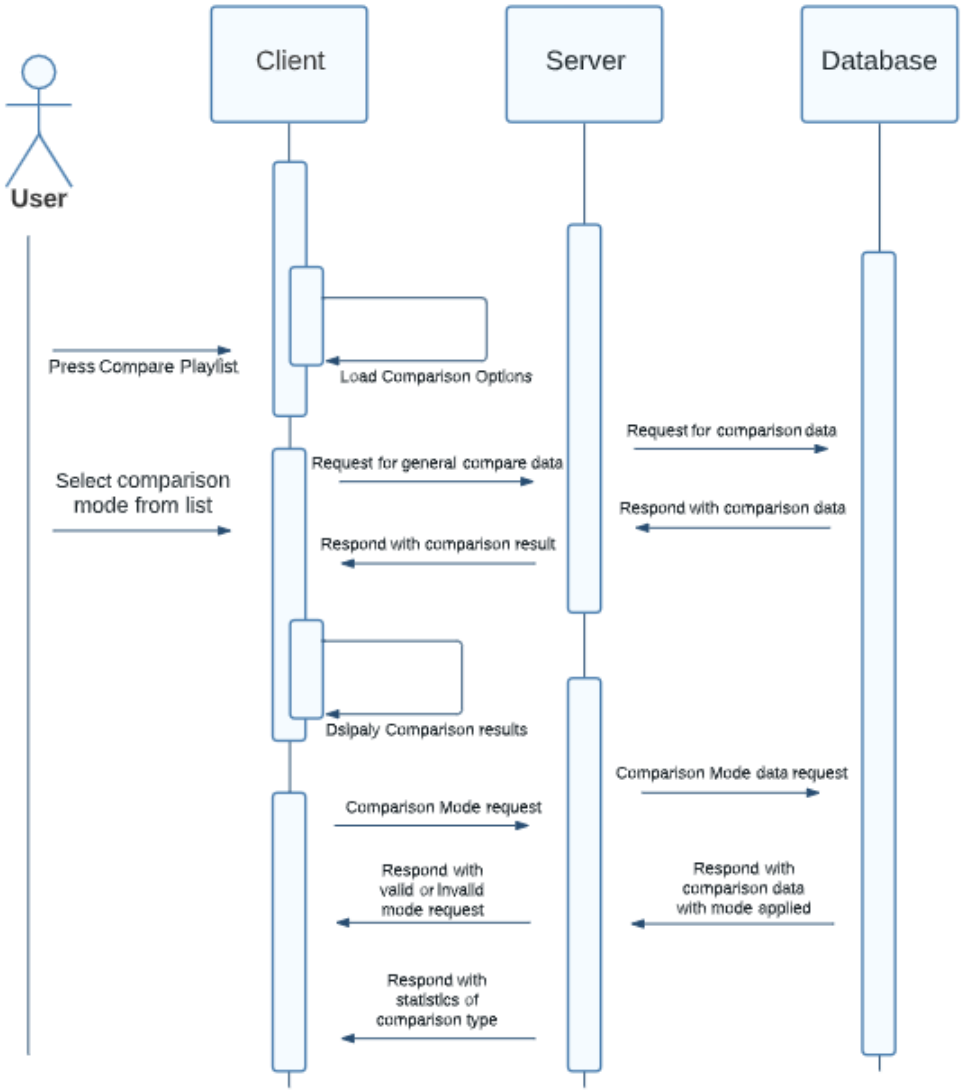
1. User logs in to JudgeMe



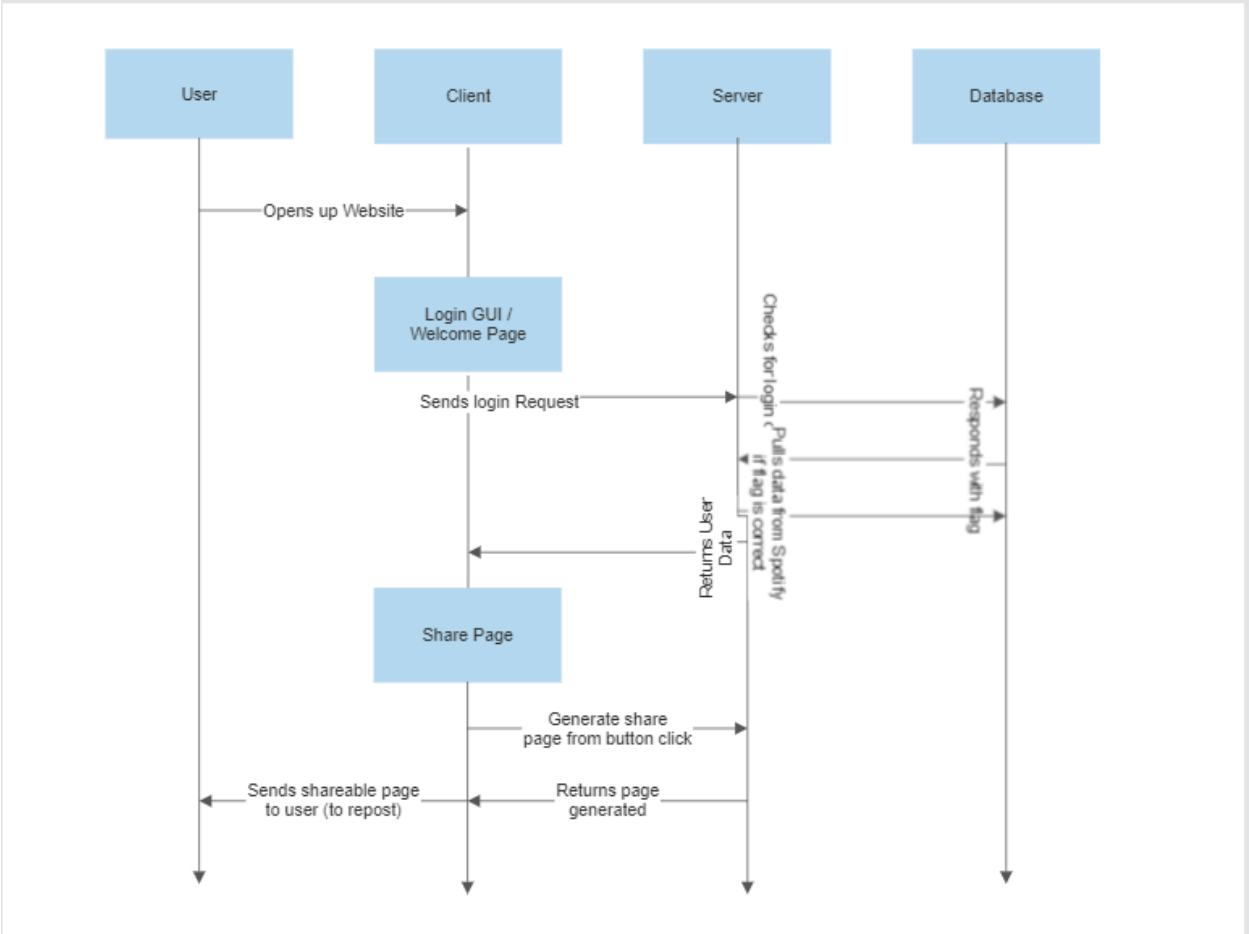
2. User creates a playlist based on mood from songs in their existing playlists.


User
Client
Server
Database
Press create playlist
Load options,
show playlist creation page
Select mood from list
Create request for playlist data
Request user's playlist data
Return user's playlists
Return user's playlists
Get all songs from playlists
Check mood for each song
Return true if mood
matches user selection
Create request to make
new playlist with songs
Request to add playlist
Respond with add success
Respond with add success
Display new playlist

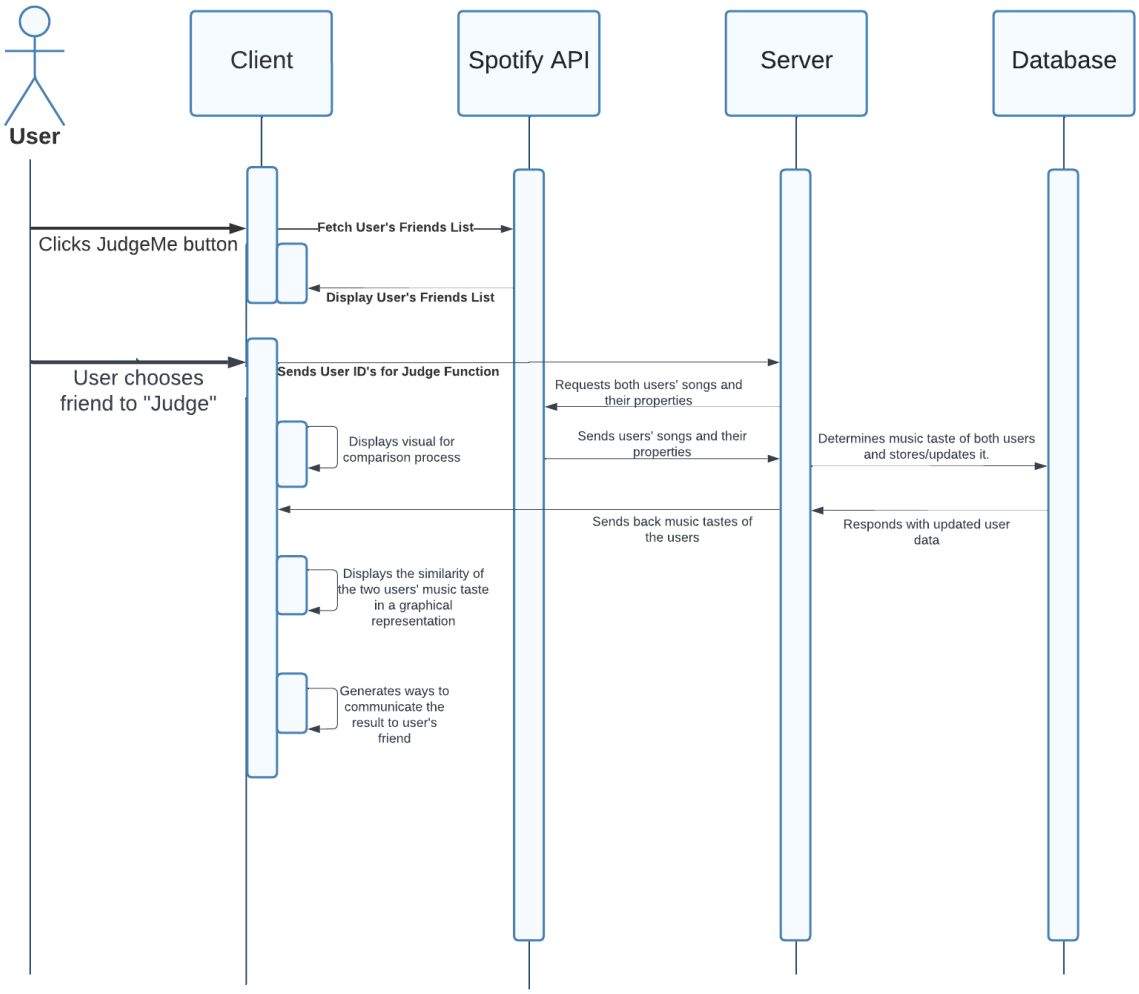
3. User compares a set of playlists with one of their Spotify friendss



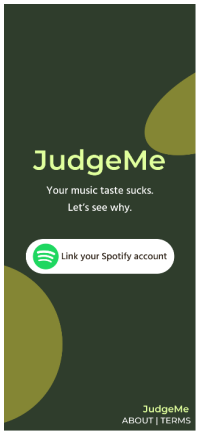
4. User shares their JudgeMe account with others

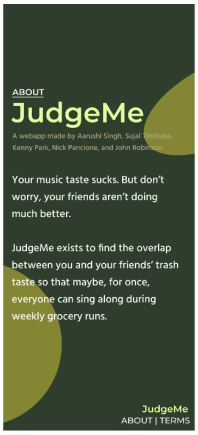


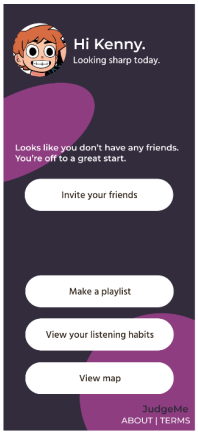
5. User runs a JudgeMe process with one of their friends

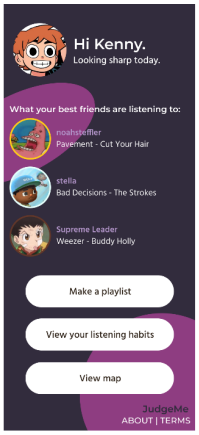

User
Client
Spotify API
Server
Database
Clicks JudgeMe button
Fetch User's Friends List
Display User's Friends List
User chooses friend to "Judge"
Sends User ID's for Judge Function
Requests both users' songs and their properties
Displays visual for comparison process
Sends users' songs and their properties
Determines music taste of both users and stores/updates it.
Sends back music tastes of the users
Responds with updated user data
Displays the similarity of the two users' music taste in a graphical representation
Generates ways to communicate the result to user's friend

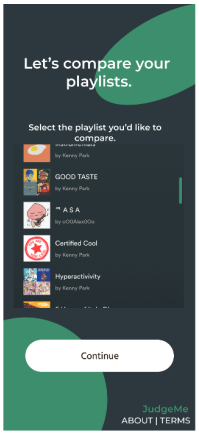
UI Mockups

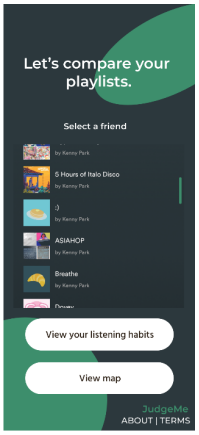

JudgeMe
Your music taste sucks.
Let's see why.
Link your Spotify account
JudgeMe
ABOUT | TERMS

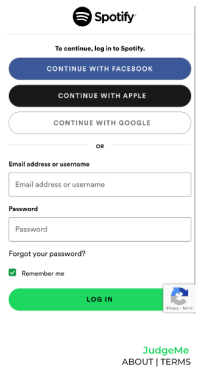

ABOUT
JudgeMe
Your music taste sucks. But don't worry, your friends aren't doing much better.
JudgeMe exists to find the overlap between you and your friends' trash taste so that maybe, for once, everyone can sing along during weekly grocery runs.
JudgeMe
ABOUT | TERMS

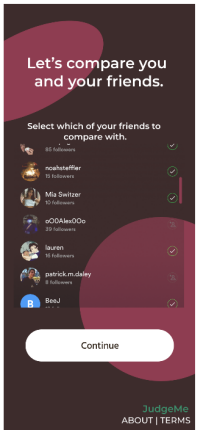

Hi Kenny.
Looking sharp today.
Looks like you don't have any friends. You're off to a great start.
Invite your friends
Make a playlist
View your listening habits
View map
JudgeMe
ABOUT | TERMS

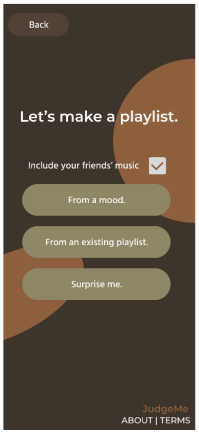

Hi Kenny.
Looking sharp today.
What your best friends are listening to:
noahsteffler
Pavement - Cut Your Hair
stella
Bad Decisions - The Strokes
Supreme Leader
Weezer - Buddy Holly
Make a playlist
View your listening habits
View map
JudgeMe
ABOUT | TERMS

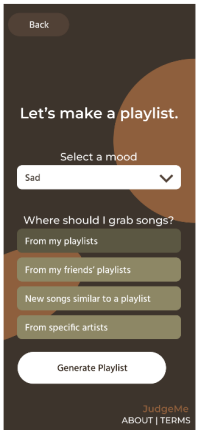

Let's compare your playlists.
Select the playlist you'd like to compare.
GOOD TASTE
Certified Cool
Hyperactivity
Continue
JudgeMe
ABOUT | TERMS


Let's compare your playlists.
Select a friend
5 Hours of Italo Disco
ASIAHOP
Breathe
View your listening habits
View map
JudgeMe
ABOUT | TERMS


Spotify
To continue, log in to Spotify.
CONTINUE WITH FACEBOOK
CONTINUE WITH APPLE
CONTINUE WITH GOOGLE
OR
Email address or username
Email address or username
Password
Password
Forgot your password?
Remember me
LOG IN
JudgeMe
ABOUT | TERMS


Let's compare you and your friends.
Select which of your friends to compare with.
noahsteffler
Mia Switzer
oO0AlexOOo
laurdn
patrick.m.daley
BenJ
Continue
JudgeMe
ABOUT | TERMS


Back
Let's make a playlist.
Include your friends' music
From a mood.
From an existing playlist.
Surprise me.
JudgeMe
ABOUT | TERMS


Back
Let's make a playlist.
Select a mood
Sad
Where should I grab songs?
From my playlists
From my friends' playlists
New songs similar to a playlist
From specific artists
Generate Playlist
JudgeMe
ABOUT | TERMS

Database Diagram