MusicReal

Summary: MusicReal is an application that allows users to post the current song they are listening to at a random given moment and see/listen to their friends' song as well. If a user is not listening to a song at the random given moment, they can provide a song recommendation instead. Additionally, you can see a user's song listening history on their profile.

Detailed description: Users will receive a notification at any random time of the day that prompts them to post the current song they are listening to. After posting, they will be able to see their friends who have posted their songs as well. Each post will include a short preview that the user can listen to and how popular it is among users. Furthermore, if a user is not currently listening to a song, they are able to recommend a song to their friends instead or they can wait till they are listening to a song and post at a later time. In addition to the songs on the home page, there will also be other suggested users through mutuals. On each user's profile, it will display all past songs, current friends, and settings.

Detailed Usefulness: Our application allows users to share and connect through music, as well as inspire users to explore songs their peers like. People are more likely to listen to a song if their friends are listening to it and/or recommend the song. Our application is similar to the app BeReal, where users can capture pictures at a random given moment to share with their friends and community. MusicReal, on the other hand, shares music instead of pictures, and includes more features such as song popularity among the entire community. Some other applications that are similar are Spotify and Apple Music, where users can stream songs, build and share playlists, and more. What differentiates MusicReal from Spotify and Apple Music is that users can recommend songs to other users, and get a more personalized and engaging experience from using MusicReal.

Detailed Realness: The user's data will be pulled using the Spotify API where the current song that the user is listening to is pulled when the random given moment occurs for the user to post their song. In the case that the user is not listening to a song and they opt for posting their song at a later time, the data is pulled in the same way as described above. The second way in which the data is captured through manual user input for the song recommendation at that point of time.

Detailed Functionality:
When the user logs into the application, they are presented with a feed that shows the songs that their friends have most recently listened to in a scroll-able format similar to social media applications. The user can click on their friend's profile to check their friend's past music history. They can click on their own profile to view their own past music history and their list of friends. If the user has opted to recommend a song instead of our application pulling their current song data at the random given moment, there will be a pop-up where the user can recommend a song.

We will have four database schemas in order for our core features to work.
They are:

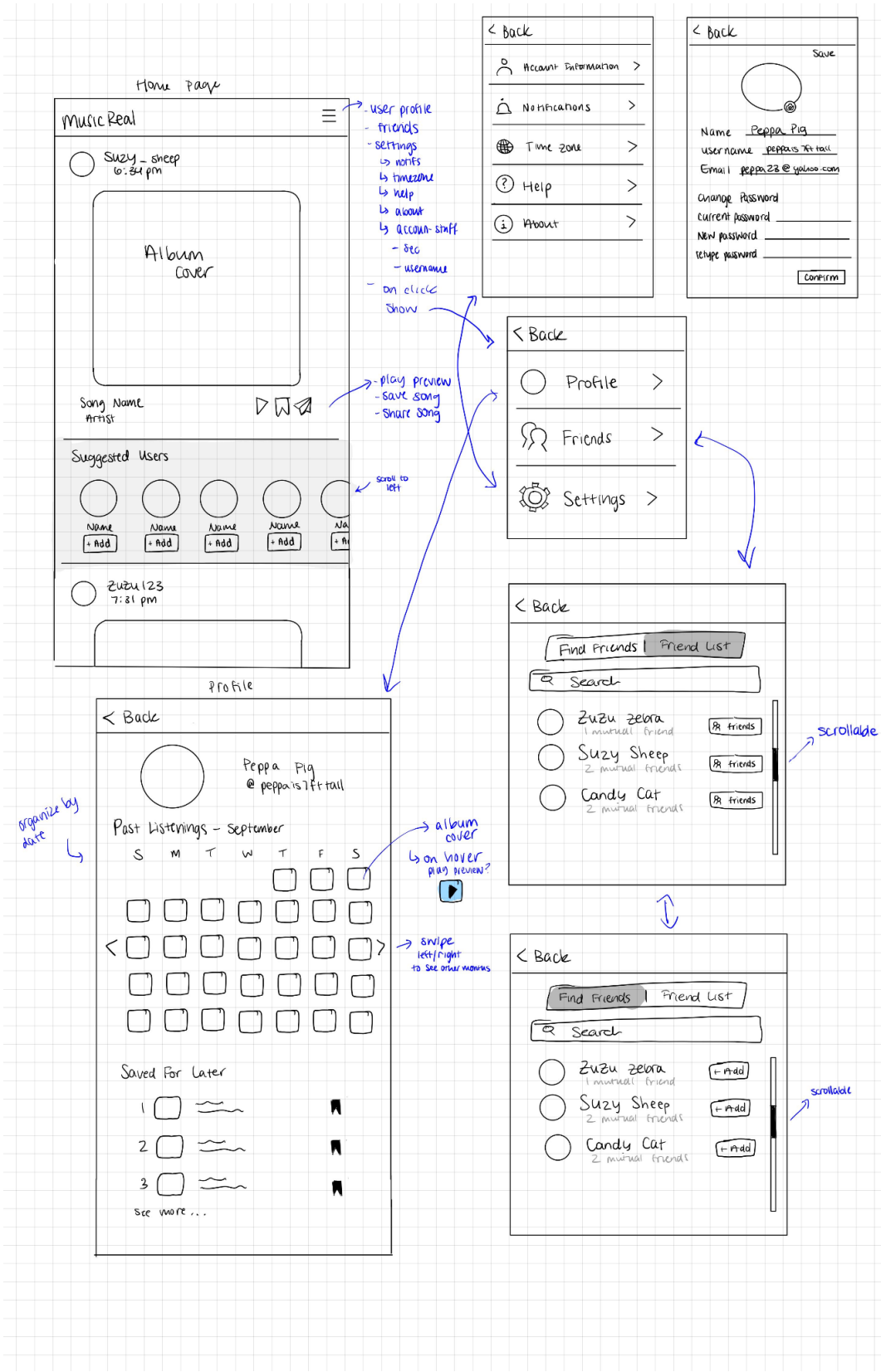
1. History(UserId, [PostIds])
2. FriendsRelationships(UserId, [UserId])
3. Posts(PostId, SongId, Time Late, date)
4. SongsCache(SongId, Song, Hits)

We have described before that the data about songs will be from the Spotify Web API, and the information about friends and users will be retrieved from the app depending on the users behavior in the app.

A creative component that we aim to implement is a monthly calendar on the user's profile where you can see the songs that the user has listened to in the past on a specific day. The song/album's cover will be shown on the calendar, and hovering over the calendar would show a preview of the song.

A reach creative component goal would be a year end analysis of the songs you listened to for the past year. It could be really interesting to see the genres you like the most, how similar your tastes are to your friends, and other facts. This improves our app because our app acts like a social media platform for music listeners, and they would appreciate seeing song listening analysis at the end of the year so that they can share it on social media. More discussion would have to take place in order to figure out what kinds of analysis and how we would want to implement this. Because this is a reach goal, we will implement this if we finish our core components early.

Detailed Low Fidelity UI Mockup:



Project Work Distribution:
In the frontend side of the app, there are two broad scopes of work. The first is the core UI in the app. The second is the handling of communication that comes from and to the backend server.

Anj - Handles profile's interface and retrieves data from backend to display on the profile's calendar
Rachel - Handles home page's interface and retrieves data from backend to present friend's music posts

For backend work, there will be code to retrieve data from the database, code that retrieves data from the Spotify web api, and code that communicates with the app itself.

Leo - Handles code to retrieve data from the Spotify web api and code that sends data to the app
Aumkar - Handles code to retrieve data from database and code that retrieves requests from the app