BeGroovy

Overview

Music is a universal language that brings people together. As a society, we all love listening to music and it is a part of who we are. Sharing songs with friends can allow an individual to not only express themselves but also create a sense of community among their friend groups. Sometimes we get sick of listening to the same music over and over and having daily recommendations of songs/genres of music to listen to would be very suitable. Our app is tailored toward these ideas as users are able to share a song to their friends daily. This app is a social media platform that allows users to share their music tastes with each other and discover new music.

When launching the app, users have the option to create an account by signing up, or logging in to a preexisting account. They are then directed to the main interface of the app, which is where their feed will be. Our app is a spin-off of the app BeReal, where users receive a push notification at a random time daily, motivating the user to shave what they are up to at that exact moment. The app promotes to “Be real”, promoting being authentic and giving an realistic glimpse in one's daily life and stepping away from cultivated and polished content posted on other social media platforms. Similar to the app BeReal, users are not able to see their friends’ posts until they post themselves. Once they upload their song of the day they can see their feed, which consists of a scrollable interface of blocks of song previews for each of their friends who have posted. Users can listen to the song previews on their mobile device as well as like their friends’ posts.

Since this is a social media platform, friend functionality is implemented as well. In the toolbar of the main interface there are three icons, which are for managing friendships (viewing incoming friend requests and current friends), searching for users (with the ability to send a friend request), and viewing account information such as the user’s email address and username associated with their account.

Goals

The overarching goal of this app is to allow users to expand their music interests directly through the app by discovering new music shared from their friends.

User Authentication

User authentication was implemented through Firebase Authentication. The sign-in method that we chose to enable within our app was email and password, although there are multiple other

sign-in methods we could have integrated such as Apple, Google, Facebook, Twitter etc. We have an AuthenticationManager class in our project code that handles all the functionality for authentication, such as signing up, logging in, validating credentials, logging out, and more. We were not able to successfully integrate each of these functionalities within our code, but the main functionalities of logging in and signing up play large roles in our app. Furthermore, whenever we need to access the current user's data (such as the ID that gets assigned to their account when they sign up), we are able to use Firebase Authentication to extract this information.

Pulling Song Data From Streaming Services and Displaying the Data

Using Firestore, we store users' posts in a collection. Each post has the following fields: link, username, userID, genre, UUID, link type, email, text description, and like count. The LPMetadata provider allowed us to fetch the provided URL's metadata, and the LPLinkView class from the LinkPresentation framework allowed us to create rich URL previews. Metadata is fetched from the provided post URLs using asynchronous functions, and the list of posts is refreshed upon refresh of the post list.

Implementing a Friend Network

Much of the data relating to our friend network was stored within the Firebase Firestore database. When each user creates an account, a document consisting of their email, username, and random ID are added to the "users" collection within the database. When a user sends a friend request to another user (which is to be discussed in more depth later), a document in the "friendRequests" collection is created, consisting of the ID of the sender, the ID of the recipient, and the status of the request. When the request is initially sent, the status is "pending", but when the recipient responds to the request the status is changed to either "accepted" or "rejected" depending on how they responded. If the recipient user accepts the request, a new document under the collection "friendships" is created, which just stores the IDs of the two users that are now friends. This is how we are able to keep track of the list of friends a current user has as well as incoming friend requests for a user.

Notifications

A big feature of our app is sending push notifications to users to let users know it's time to share new music with their friend network. The app chooses a random time a day to send the notification, and the user's feed is locked until the user has posted a song for that day. This is implemented in the HomeView with a boolean state variable called "hasPosted", set to false when the push notification is sent, indicating that the user has not posted their music contribution that day. After the user has added their post for the day with the popup screen (more about this below), the boolean is set to true and their feed is shown.

We have created the app with the “hasPosted” boolean that currently is manually initialized to false, until the user shares their post for the day for, since unfortunately we weren’t able to implement user notifications. The notifications feature needs to be configured with an apple developers account, which is $99 a year. We determined that for our product demo we can simulate the “add post” pop-up without using a notification, to keep the cost of our app low. Investing in an apple developer account could be interesting if we ever consider monetizing our app.

Animations & Social Features

A social feature we implemented is liking a friend's post in their feed. A post has a heart button that a user can click on to like a post. The heart button is connected to a like post function that increases the post likes count with one. Then the view will keep track of the user’s liked posts, adding friends username to an array and check if the user has already liked that friend’s post. The likes button will then be filled in with red or not depending if the user has liked the post already. This could’ve been implemented with adding to the post object in the database which users have liked a specific post, and based on a database call it can establish if a user has liked a post or not, yet based on the scale of this project we decided to keep that information local. An additional feature could be showing the total number of likes that post has. Due to the complicity and time constraint, I decided to omit implementing a comment and reply discussion thread for posts.

Under the user account view, we added the feature to add a user profile picture. A user can select the person symbol, which will show a photo picker. A user can select their profile picture in the photos picker and the photo will populate where the button used to be.

User Interactions

When the app is loaded, the user is in the authentication view. This is where they choose to either login with a preexisting account or create a new account. By default the user is taken to the login page but they can navigate to the signup page by clicking the purple “Sign up here!” text. Similarly, if a user wants to navigate from the signup page back to the login page they can click the purple “Sign in here!” text. The two views are shown in the following figures:

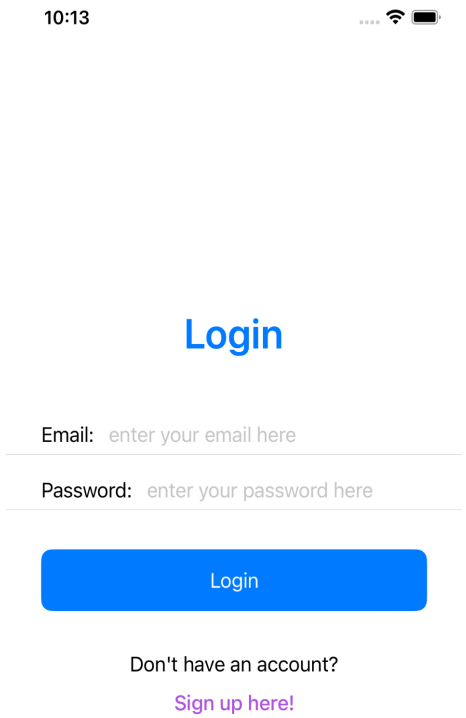

Figure 1. Login View

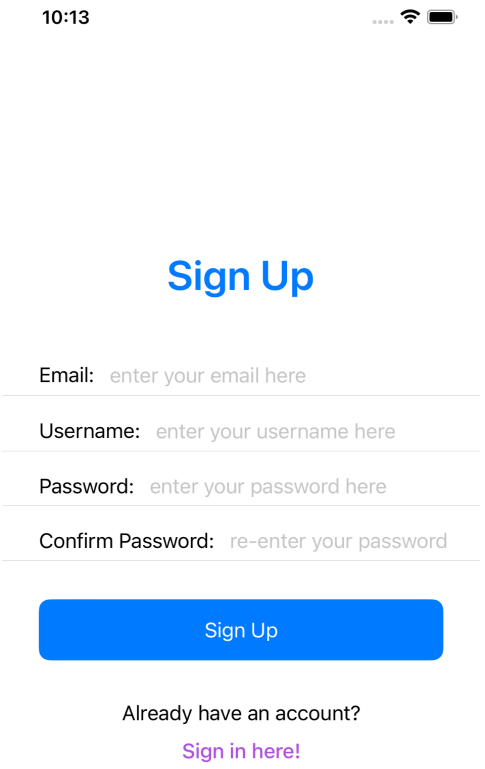

Figure 2. Signup View

There are multiple validation steps that take place during the login and signup processes. For example, if the user leaves any of the fields blank or if the user attempts to login with an invalid email/password, an error message will be displayed.

Once the user has successfully logged in to their account or created a new account, they will be redirected to the main interface of the application. Since they have not posted their song of the day, they will not yet be able to see their friends' posts. Instead they will see a button in the middle of the screen with the message "Add your song!". This view is shown in figure 3 below. When the user clicks this button they will be redirected to a popup page where they enter information about the song they are posting. This information consists of the link of the song from their streaming service, the genre of the song, and a message about the song. This is shown in figure 4.

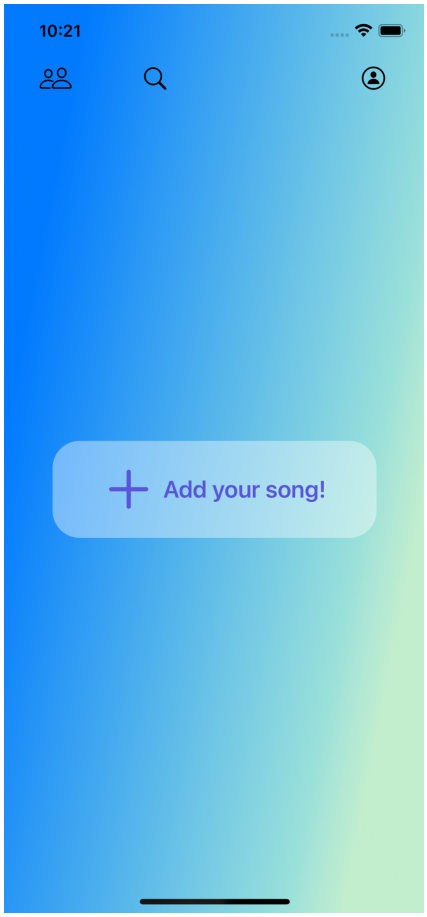

Figure 3: Initial Interface

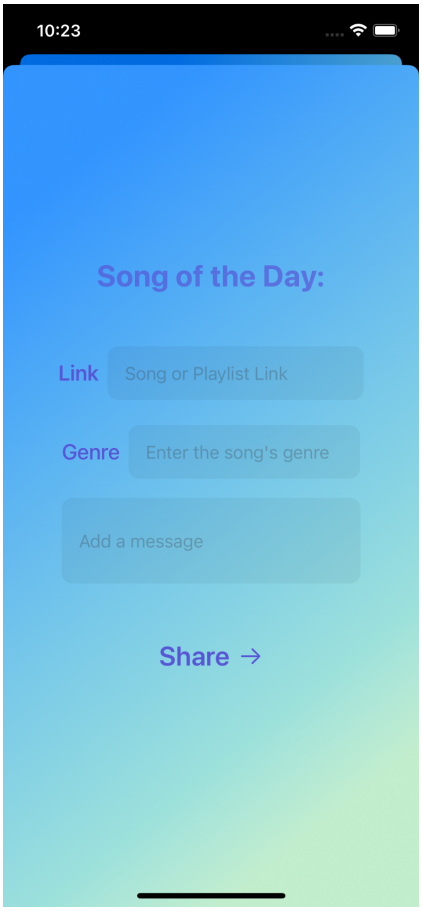

Figure 4: Sharing the Song

After clicking the “Share” button the user is then redirected once again to the home screen, which will now display their feed. Their post, along with posts from their friends (who have posted that day) will be displayed by a scrollable feed. Each post within the feed is represented as a block and contains the post’s account username, a preview of the post (which includes the album cover image for the song, the song’s title, and a clickable play button which stimulates a preview of the song), the user’s message for the post, and a heart button which allows the user to like their friend’s post. An example of the feed a user will see is shown in Figure 5 below.

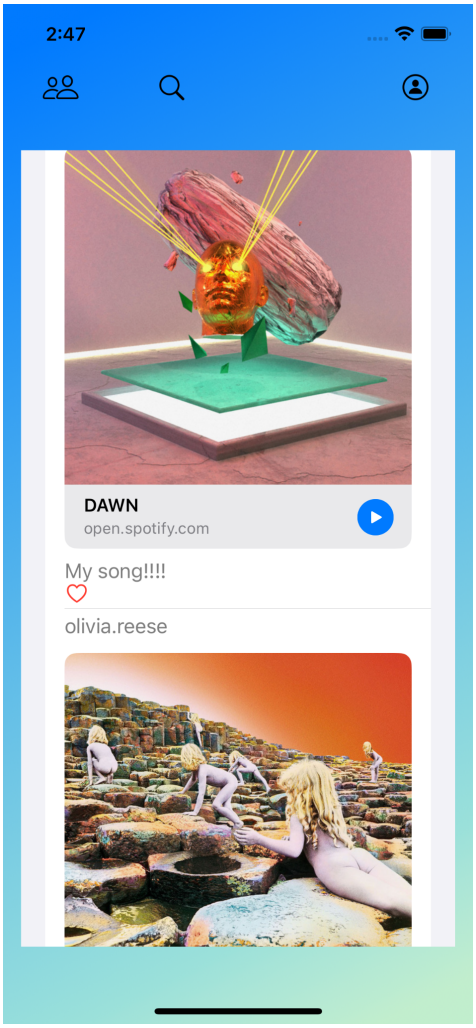

Figure 5: Example Feed

The toolbar at the top of the screen contains three buttons, which redirect users to different views. The leftmost button brings the user to the Manage Friendships view, where there are two tabs for users to select. The first tab is for users to view their current friends list, and this displays a list of usernames that the user has a friendship with. The other tab is for users to view their current incoming friend requests, which is also a list of usernames. For each username in the incoming friends list, the user has the option to either accept or reject the friend request. If accepted, a friendship is created between the two users and they will then be added to each other's friends list in the other tab.

The second button brings users to a textfield bar, where they are able to search for other users in the app. The search criteria is by username, so if the user searches for a username that is registered in the app, their name will show up below. Next to the username is a button that allows the user to add the account with that username as a friend.

The rightmost button is the Account View, where users are able to see their account information such as their username and email address they used to register for the app. A logout feature was implemented and this view intended to have a logout button, where the app would essentially restart and redirect the user back to the login screen. However, due to the time restrictions and scope of the project, we did not add this button into our app, even though the functionality is contained within our source code.

Development Process

The first stage in the development process consisted of the brainstorming phase. Here, our team decided that we wanted to design a music sharing app, since music is something that is significant to each of us. We then came up with the idea to create a social media app similar to the popular app BeReal, but instead of sharing pictures daily we would be sharing music daily. We discussed using Firebase as the database where we would hold information about the data within our application, such as authentication data, friend network data, and post data.

The first step in our project was to set up Firebase Authentication and implement this into our app. We designed login and signup pages that use Firebase Authentication to register a user within our app. This was the first task in our project since we needed to have a way of registering users within the app before implementing other social media features.

We then discussed the basic view design and navigation within the app, and where we wanted certain features of our app to be on the main interface. From here, we started implementing these views, even though there wasn't anything necessarily contained within them since we had not implemented friend functionality yet.

The next step was to implement the friend functionality, which was probably the most important and time consuming part of the app. We were able to create a collection of users that each contained a network of friendships with other users, and we successfully integrated this into the app. This step alone took us a few weeks and as a result we had to reschedule our milestone timelines and reassess how much we would be able to implement in the app because of the significant amount of time it took to implement these features and connect them to the functionality of the app.

Next we worked on the posting aspect of our application, which relies heavily on the Firebase Firestore database. When a user posts a song, the song data is saved to the database and is later fetched from the database when displaying the user's feed, along with the posts from users they are friends with on the app. Although at first we did not think this would be achievable, we were able to get data from streaming services (like Spotify) and create a preview of the song, which would be displayed within each user's post. This functionality takes the sharing url of the song

from the streaming service and generates a preview, which contains the album cover image, the song title, and a play button which would actually play a preview of the song on the device. This is something that was originally a stretch goal but we were able to complete it within one of the milestones.

The final step of our project was to implement notifications in order to integrate the random times of users being alerted to post. Similar to BeReal, we wanted to alert users once a day at a random time to post their song of the day. Their previous day's post would then be cleared as well as their feeds. However, due to needing an Apple Developer account to implement notifications is not something that we are able to do, so unfortunately notifications aren't a feature of our app.

Overall, the project took us around 2 months to complete and involved a large commitment in order to finish. However, we all gained a lot of knowledge and experience from the project, especially with learning how to use and getting comfortable with Firebase Authentication and Firebase Firestore.

Future Directions

One of our initial stretch goals, which we were not able to implement due to complexity was connecting our app with other social media platforms. This is something that a lot of other social media platforms integrate into their apps, and would be interesting for future implementations of our app. Possibly having authentication through other platforms such as Facebook or Twitter could have connected other platforms with ours. Additionally, investing in an apple developer account to implement user notifications could be a great future addition.

There are also certain aspects of our app that we would like to improve/enhance. For example, we would like to improve various elements about our UI, such as the view of the main feed. The posts are displayed using a List, which makes them have the default white background blocks that are developed into the List view. There are also subtle bugs within our app that we would like to improve if we were given more time and resources toward the project. For example, with our friend request functionality, our app would technically allow users to send multiple friend requests to the same person. This is a fix that we would implement if we had more time to work on the project.