Harmonize - Music Dating App

Kashish Goel
Computer Science
Columbia University
New York, USA
kg3044@columbia.edu

Swati Bararia
Computer Science
Columbia University
New York, USA
sb4700@columbia.edu

Aryan Jalali
Computer Science
Columbia University
New York, USA
aj3085@columbia.edu

Nicholas Anason
Computer Science
Columbia University
New York, USA
nda2125@columbia.edu

***Abstract*—Harmonize is a modern and cutting-edge music dating app that leverages the power of Amazon's cloud services to provide a scalable and dynamic platform for music lovers to find their perfect match. Our innovative app provides a concise overview of potential matches, allowing users to quickly and easily connect with people who share similar musical interests.**

Harmonize's algorithm analyze user preferences and listening history to match users with potential partners who share similar music tastes. The app also features a range of interactive features, including the ability to chat, share music, and attend virtual concerts together, allowing users to build deeper connections with their matches.

With its intuitive user interface and powerful features, Harmonize is a unique and valuable addition to the world of music dating apps. Whether users are looking for new friends, casual relationships, or serious partnerships, Harmonize provides a personalized and engaging platform that connects people based on their shared love of music.

Overall, Harmonize represents a new standard for music dating apps, providing users with a fun, dynamic, and effective way to discover new music and build meaningful relationships with like-minded individuals.

***Index Terms*—AWS, music, dating, match, interest.**

I. Introduction

Traditional dating apps have become increasingly popular in recent years, but they often fail to capture the true essence of a person's personality and interests beyond their profile picture and bio. Music dating apps aim to provide a solution to this problem by matching individuals based on their music preferences and interests. By using music as a way to connect people, music dating apps can provide a more meaningful way for individuals to find compatible partners and form lasting relationships.

A. Motivation

Music dating apps have the potential to work because music is a powerful cultural and emotional force that can be used to connect people who share similar tastes and interests. Music has the ability to express emotions and experiences that are difficult to put into words, and people often feel a strong sense of identity and community based on the music they listen to. By matching individuals based on their music preferences and interests, music dating apps can provide a more authentic way for people to connect with each other.

Identify applicable funding agency here. If none, delete this.

Additionally, music dating apps can help break the ice and facilitate conversation between individuals who may not have otherwise connected. The shared love for a particular band or genre can provide a starting point for conversation and potentially lead to a deeper connection. Music can also be used as a way to explore shared interests and values beyond just physical attraction. Furthermore, music dating apps can offer a more enjoyable and engaging user experience compared to traditional dating apps.

B. Target Audience

The target audience for a music dating app would likely be individuals who have a strong interest in music and see it as a key aspect of their lives. This could include music enthusiasts, musicians, DJs, and other professionals working in the music industry. The app may also appeal to individuals who have a specific taste in music and are looking to connect with others who share their musical interests.

The major target audience for this product would be students of similar age groups, or anyone in similar age categories. Overall, the target audience for a music dating app would be individuals who place a high value on music in their lives and see it as an important aspect of their identity and social connections

C. Existing Solutions

Popular Dating Apps: Many of the top dating apps on the app store (Tinder, Bumble, Plenty of Fish, etc) already allow users to incorporate their liked songs into their profile, indicating that there is at least some value towards music taste in matchmaking. None of the popular apps solely use it, which may mean that other factors are more important in romantic relationships.

Vinylly: This is an existing meeting app (either romantic or platonic "concert buddy") for people with similar music preferences. It allows users to import their spotify, then has a similar interface to tinder to allow them to sort through people around their age that listen to the same music. They

have been around for 3 years; however, they underwent significant UI improvements just a few months ago. They have a small user base, only 174 reviews on the app store (compared with Tinder's 440,000).

Marriage Pact: This is a light-hearted matchmaking service that mostly targets campuses. Whereas most dating apps allow for users to iterate through potential partners, this collects many users and then assigns and releases just a single matching. It's quite popular across many college campuses, despite its browser-and-email-based questionnaire UI. It doesn't have any UI for chatting, rather just provides contact information to each match.

Partnership Research: There's quite a bit of evidence that people with similar tastes and interests tend to form stronger relationships. Whether this is because the interests represent inherent personality traits that drive us or the more egotistical view that we like ourselves and therefore like people that are like us, this is well established in relationship psychology.

D. Our Application

We have built an app that takes the user's music data from Spotify, apply recommendation algorithms to that data and other users, and show them the users with similar music taste.

The app requires using Spotify APIs to gather music data for each user on the app, their location and a recommendation model that would work on this collected music data. The users with similar tastes can get in touch with each other.

Key Points:

- User-friendly interface: The app is extremely easy to navigate, with a simple and attractive design that appeals to music lovers.
- Compatibility matching: The app uses matching algorithm and user data to match users with compatible partners based on their music preferences, interests, and personality traits.
- User verification and safety measures: The app includes user verification and safety measures, such as email verification, to ensure that users are who they say they are and to prevent fake profiles and fraud.

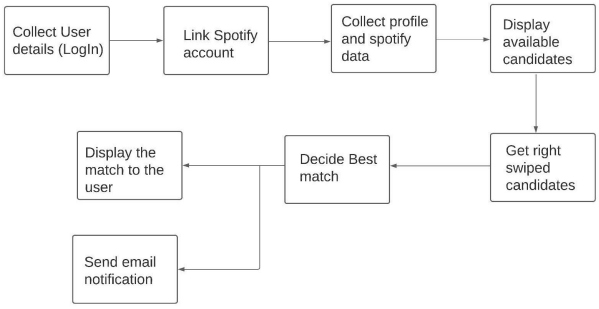


Fig. 1. High Level Design Diagram

II. SOLUTION ARCHITECTURE

A. AWS COMPONENTS USED:

1) AWS Amplify: The Frontend of the dating app is hosted on AWS Amplify, which provides a fast and secure way to deploy web and mobile applications. The user profile details, including preferences and interests, are collected from the frontend and used to connect to the backend of the application through API Gateway.

2) API Gateway: API Gateway integrates with other AWS services, such as AWS Lambda, Amazon DynamoDB, and Amazon S3, to provide a complete serverless architecture for building and deploying our dating app. It provides a scalable and secure way to access backend services, such as user profiles, matches, and messaging functionality. By creating APIs that act as a "front door" to these services, we can easily control access to them and protect against unauthorized access or traffic spikes.

3) Spotify API:

- Spotify uses different APIs for getting albums, tracks, artists and genres.
- We use Spotify Auth for authentication of our users prior to giving access to our service. The way this works is a new user registration service is made using their email and once received a verification code is sent for it to get activated. This is done to avoid any spam mail being created. Once authenticated it redirects to the profile page where users can create their profile.

4) Lambda Functions: The app uses 8 Lambda functions to handle various functionalities related to user management and matching. These functions enable the app to create user profiles, retrieve user details, generate a list of potential match candidates based on user preferences, decide matches, and determine ideal matches for users. By leveraging AWS Lambda and API Gateway, the app is able to provide a fast, scalable, and secure solution for matching people based on their musical preferences.

5) DynamoDB: A database is a software system used to store and manage data efficiently. In the case of our application, we have designed a database with two tables to store the candidate profile details and their matches. The candidate profile table contains information such as their name, age, location, musical preferences, and other relevant details. The matches table stores the information about the matches between the candidates, such as the candidate IDs, and other related information.

6) Amazon SQS and SES: The functionality to send email notifications to users when they receive a right swipe from someone they have previously right swiped is a common feature in dating apps. This feature can be implemented using various email service providers, such

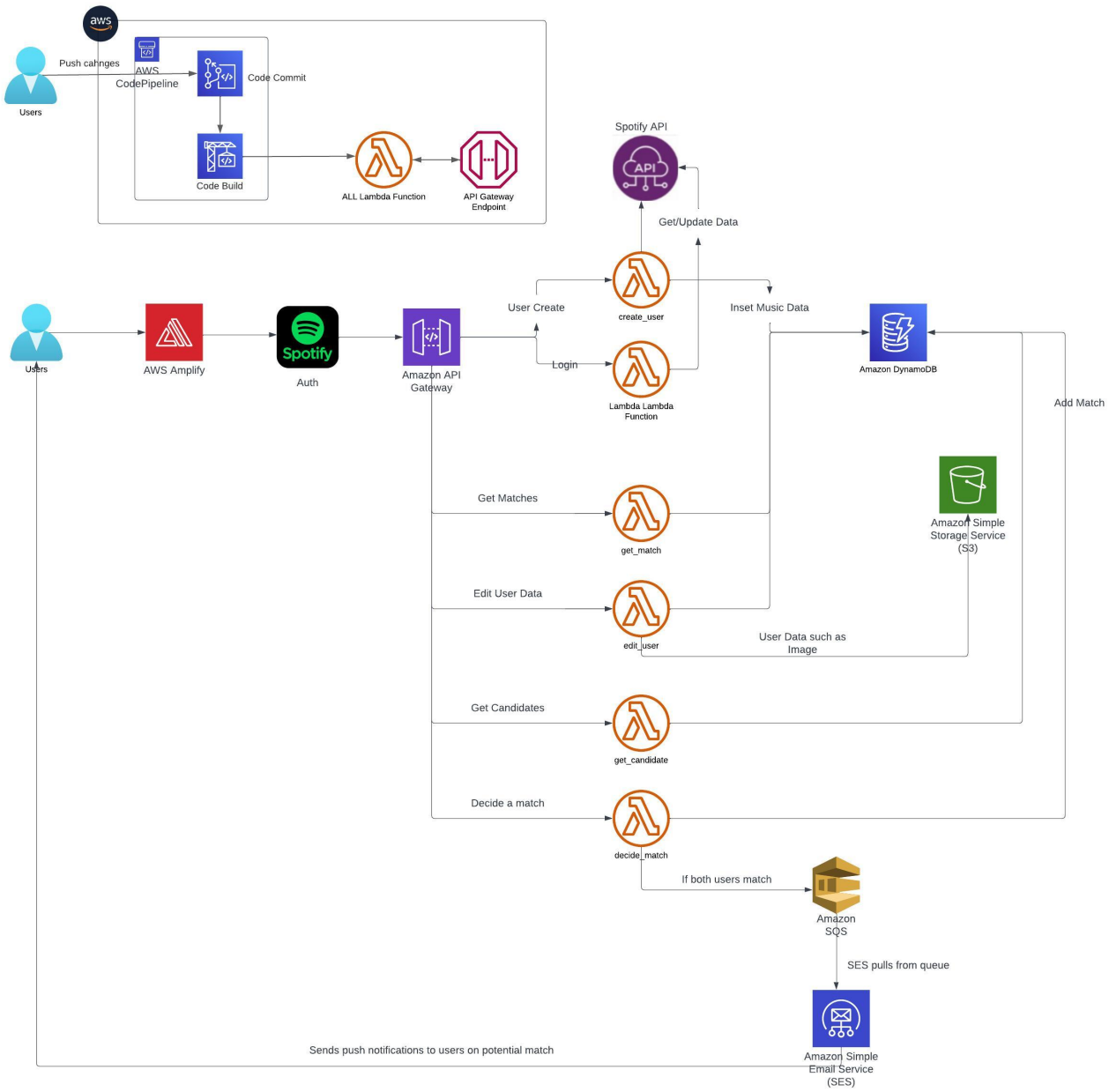


Fig. 2. Architecture

as Amazon Simple Email Service (SES), which provides a scalable and reliable way to send email notifications to users in real-time. By leveraging the power of SES, dating apps can improve the user experience by keeping users informed about their matches and increasing user engagement.

7) AWS CodePipeline: AWS CodePipeline is a continuous delivery service that can be used to build, test, and deploy Lambda functions for an application. By using CodePipeline, we created a pipeline that automatically builds and deploys Lambda functions whenever changes are made to the source code. This can help to ensure that the latest code is always deployed and that any bugs or issues are caught early in the development cycle.

B. DESIGN WORKFLOW

Music dating apps are online dating platforms that focus specifically on connecting people who share a passion for music. These apps allow users to create a profile, browse other user profiles, and connect with potential matches based on their musical interests and preferences. The input to this applicaton is in form of music data and text data (user profile details).

Our overall design/workflow can be broken into 4 parts:

Fig. 3. Frontend Design

1) Frontend: The frontend of application accepts user profile details such as gender, name, contact information, etc from the user along with their Spotify account. The frontend triggers our API gateway and calls the Spotify API to link the Spotify account of the user with our application.

2) Spotify API and Data Extraction: First time user is asked to link their Spotify account at the beginning of the registration where they are asked for certain permissions to access their music data from Spotify (mainly music history, top artists, etc). Spotify Authentication is used to verify their accounts and this is done to ensure security and correct authorization to the user data connected to a real account.

The user profile data and the music data is stored in the database and extracted using lambda functions for getting ideal candidate matching. Only those accounts are used which have a music listening history of atleast 50 songs.

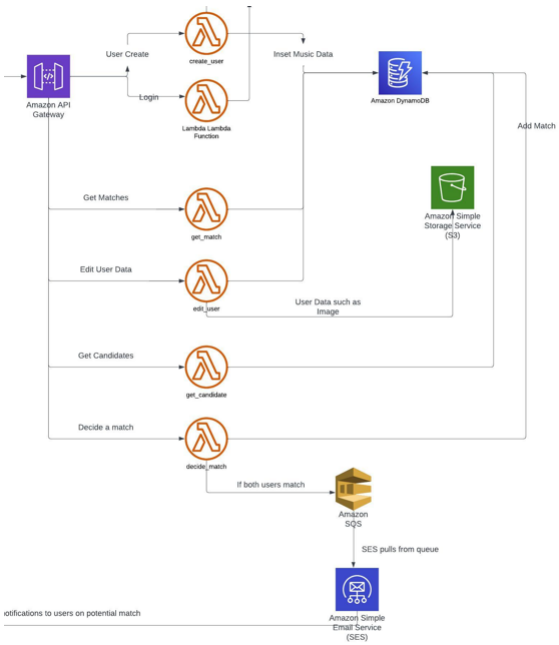


Fig. 4. Backend Design

3) Matching Algorithm: We have developed a custom algorithm that takes into account the music preferences and history of the users to provide them with the best possible matches. The algorithm is used in conjunction with the getcandidates lambda function, which returns a list of potential matches for the current user.

To implement this algorithm, we first developed a custom implementation of the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm. This algorithm is commonly used in information retrieval and natural language processing to measure the relevance of a document to a particular search query. In our case, we used it to measure the relevance of a user's music preferences to the music preferences of other users in the app.

Once we had calculated the TF-IDF values for each user, we then used the cosine similarity algorithm to compare the preferences of the current user with those of the other users in the app. The cosine similarity algorithm measures the similarity between two vectors by calculating the cosine of the angle between them. In our case, we used it to measure the similarity between the music preferences of the current user and those of the potential matches.

The result of this algorithm is a list of potential matches that are ranked by compatibility, with the most compatible matches appearing at the top of the list. The current user can then choose to right swipe on the potential matches that they are interested in. If two users right swipe each other, they are matched and both receive an email notification.

4) AWS SES and SQS: After a user has right swiped on a potential match in our music dating app, the information about the swipe is stored in our database. This allows us to keep track of the user's preferences and to provide them with better matches in the future.

In the event that the user they had right swiped also swipes right on them in the future, we use the Amazon Simple Queue Service (SQS) to send a notification to our backend. The SQS is a fully managed message queue service that enables decoupling and scaling of microservices, distributed systems, and serverless applications.

Once the notification is received, we use the Amazon Simple Email Service (SES) to send an email notification to the current user about the match. SES is a cloud-based email sending service that enables us to send high-volume email messages with high deliverability rates. The email contains information about the match and a link to the user's profile, allowing them to easily connect with their new match.

The use of SQS and SES has allowed us to ensure that notifications are sent in a timely manner and that users are quickly alerted about their matches.

III. SOFTWARE DESIGN

A. Frontend Code

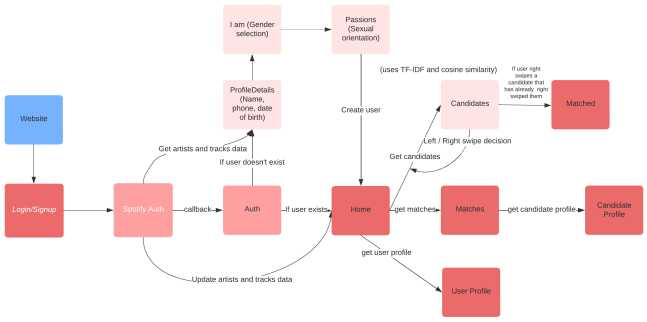


Fig. 5. Frontend Software Design

The landing page of our app welcomes the user and presents two options: sign in or sign up. Clicking on either of these options redirects the user to the Spotify login screen, where they can enter their Spotify credentials. Once authenticated, Spotify returns their email, Spotify ID, and their top 50 tracks and artists. If the user already exists in our database, we update their Spotify data in our backend and log them into the app. However, if the user is new, they proceed with the profile creation steps. This involves providing personal details such as their name, phone number, date of birth, gender, and sexual orientation. The email, Spotify ID, and top 50 tracks and artists of the user are pulled from Spotify Auth integration. A user can enter multiple sexual orientations. After collecting this information, we call the create-user lambda to create the user. Our home page uses the get-candidates lambda function to fetch the best candidates using TF-IDF and cosine similarity, and displays them for the user. The user can swipe right or left on a candidate to indicate interest or disinterest, respectively. Both swipe options call the decide-match lambda, which checks if the two users are a match or not. If the user swipes right, but the candidate has not yet swiped right on them, the candidate is added to the user's potential matches. If the candidate later swipes right, the user receives an email via AWS SES functionality to alert them of the match. If the candidate the user swipes right on has already swiped right on them, they see the match on the "Matched" screen. The user can view their matched users on the "Matches" screen and view their profiles, as well as their own profile if they click on the "Profile" option in the bottom navigation. All profiles, whether of candidates or users, include their top 5 tracks/artists.

Limitations - Spotify limit of 25 users because our app hasn't been approved for their API's production access.

B. Backend Code

- *common* libraries:
  A lot of functionality is needed across many lambdas. These are divided into a couple of utility files deployed to all of our lambdas.
  - *users*: This library handles all functions touching the users table. It provides a useful API to check whether a user exists, various APIs to get information associated with a user(s), and various APIs to modify users.
  - *decisions*: Similarly, this library handles all functions touching the decisions table. It provides a useful API to check whether a decision exists as well as various APIs to get information associated with a decision.
  - *CORS*: This provides a single point of maintenance for handling CORS in lambda returns. It provides a useful wrapper used in all lambdas that adds CORS headers to whatever the lambda is return.
- Lambda Functions:
  - decide_match - Takes a decision (whether a user liked a candidate), and put it into the decisions table. If this decision triggers a match, return as such, modify the users table, and dispatch an email to the counterparty.
  - unmatch - Removes a match between two people for both users. Updates the entry in the decisions table to be liked=False. Currently not used.
  - get_candidates - This creates a user with given information. It runs some validation checks on the submitted data, then validates that the user doesn't already exist, then adds the user to the users table.
  - get_profile - This gets the information associated with a single profile, like picture, name, and email
  - get_profiles - like get_profile but for a batch
  - get_match - gets the matches for a user
  - create_user - Creates a user with the given fields. Performs some validation of the fields and that the user doesn't already exist. Also sends a verification email if the user isn't already verified.
  - edit_user - Updates given fields for a user. Performs validation if necessary.
  - delete_user - Deletes a user with given id. Checks that the user existed before this call. Not currently used.
- DynamoDB:
  - *users*: keeps track of all per-user information. Each item has all metadata (user id, name, phone, email), as well as spotify data (profile pic, listening histories), and finally a record of the user's current matches, if they have any.
  - *decisions*: keeps track of all prior decisions. Mainly, whether a user liked a candidate.

C. Workflow

We used a multi-repo structure to best parallelize our work. We have one for the frontend and one for the backend. The frontend is hosted on amplify, which automatically pulls changes to main and deploys them. Then we use codepipeline for our backend to package and deploy all of the lambdas. At the beginning, we had admin database utilities which cleaned

our data for testing purposes, but we stopped using those since adopting real user data. In the future, we'd like to have a proper test and prod environment.

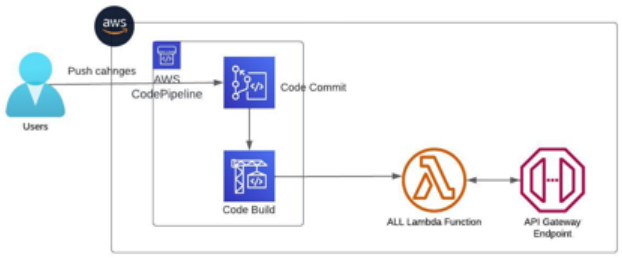
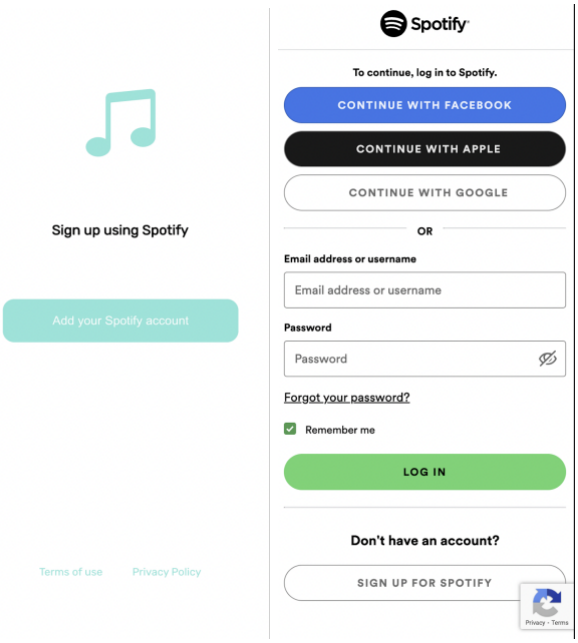


Fig. 6. CodePipeline

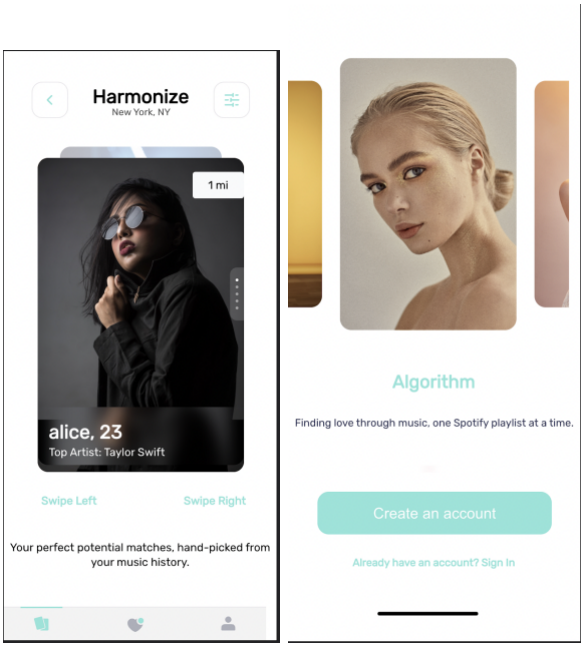
IV. RESULTS

Harmonize, our modern and innovative music dating app, has performed well during beta testing. As this is a school project, we understand that users may approach the app differently than they would a real dating app. However, we are pleased to report that our app has handled beta testers well.



The app has been designed with a mobile-focused userbase in mind and supports a minimalist feature set while providing a fun core feature set to users. The app deploys well, with fast load times and smooth navigation, making it easy for users to quickly create an account using their existing spotify data.

Our app's innovative matchmaking algorithm analyzes user preferences and listening history to match users with potential partners who share similar music tastes. This feature has been well received by beta testers, who have reported high levels of satisfaction with the app's ability to pair them with compatible matches.



In addition to its powerful matchmaking algorithm, Harmonize also offers a range of interactive features, including the ability to chat, share music, and attend virtual concerts together. These features have been popular among beta testers, who have found them to be engaging and fun.

Overall, our app has performed well during beta testing and has received positive feedback from users. While we recognize that this is a school project and may not have the same level of user adoption as a real dating app, we are proud of what we have accomplished and are excited to see where Harmonize goes in the future. With its mobile-focused design, innovative matchmaking algorithm, and range of interactive features, Harmonize has the potential to be a valuable addition to the world of music dating apps.

V. HOSTED APPLICATION AND DEMO

Links to Access the Hosted Application, Video Demo on Youtube, Code on Github and the Presentation -

- Hosted Application
- Demo
- Presentation
- Backend Code
- Frontend Code

VI. CONCLUSION

Harmonize aims to provide a more meaningful way for individuals to find compatible partners and form lasting relationships based on their music preferences and interests. Through a thorough literature review and analysis of existing solutions, we have identified a gap in the market for a music dating app that exclusively uses music as a way to connect people.

In today's world people look for genuine connection when dating and music is an integral part of their lives for a lot of people. Having common interests is something many seek in

a relationship and this app would help those find people with similar interests in music.

The web app utilizes Spotify APIs to gather user music data, apply recommendation algorithms, and match users with compatible partners based on their music and preferences. The app provides user-friendly interface, compatibility matching, advanced search filters, and user verification and safety measures.

Overall, Harmonize has the potential to provide a more authentic way for people to connect with each other and break the ice through shared love for music. With careful execution and attention to technical details, Harmonize could become a popular and successful music dating app in the market.