The Vibrations App: Connecting via Music

David Clark, Sophia Rubsamen, Harditya Sarvaiya, Atharva Sardar, and Lokesh Satpute

[1] Virginia Tech, National Capital Region
[2] CS 5704, Software Engineering

Abstract. In today's world, applications on our phones have become an integral part of our life and our relationships with others. Many of the applications which foster connection between strangers focus exclusively on connecting individuals romantically and only show them matches based on their location with no regard for actual connection via interests. These apps do not promote actual intimate connection and have a low success rate in terms of matches to actual friendships/relationships. We are proposing an application called Vibrations which will help people connect with each other both romantically and platonically based on their music preferences. This will be a medium for people to be matched with those in their location radius who have similar musical interests and will give them a space to connect with each other.

1 Definition of Problem

Many dating applications fall into the trap of prioritizing physical attractiveness over emotional connections. For example, dating applications like Tinder or Bumble capitalize on "swiping", where users are encouraged to swipe through as many people as quickly as possible without reading each other's profiles. According to Tinder statistics, it takes on average 57 matches before matches agree to go on a date and more than 285 matches for a relationship, despite the fact that 80% of users claim to be looking for a relationship. On average, women receive 7 matches a day, and men receive 5 matches a day [3]. This means that the vast majority of matches on Tinder do not result in any connection or even communication between the two matches. Our project, Vibrations, aims to address this problem. Vibrations is an application where users can connect by sharing their music interests, using Spotify. While both Tinder and Bumble allow their users to connect their Spotify accounts, this feature on both applications is severely limited. Spotify data is stored on the bottom of the user's profiles and only displays five songs. Vibrations, in contrast, makes Spotify data the entire focus of the user profiles. Users can show off not only their favorite songs, but also their favorite artists. Music not only provides an immediate shared interest between two individuals, providing incentive for interaction and potential meetups, but its inherent structure, familiarity, and added sensory stimulation also play a crucial role in fostering a more intimate connection [9]. When two individuals listen to their favorite music together, they synchronize with the same rhythm, creating a harmonious experience that enhances their bond. For example, if both

matches like Taylor Swift, and she happens to be performing at a concert in their immediate area, they could meet up together and find comfort in each other as they listen to her music together. Additionally, we decided to downplay the dating aspect of our application. Evidence shows that many young people are now even using dating apps to find new friends, especially post lockdown [6]. While users can and are encouraged to connect romantically, we also encourage platonic relationships. We believe that this distinguishes our application from Bumble and Tinder and solves the problem that we have identified.

2 Solution Description

The vibrations application allows users to create an account, create a profile that features a profile picture, a short bio, and some of their favorite songs, artists, and top genres. They can connect to their Spotify account to pull their actual data that features their top 5 favorite songs and artists or they can manually search the Spotify API for them. They can also share their location with the app and select a specific distance radius to view matches in. Once they have uploaded this data, a matchmaking algorithm will find other individuals in their area with similar music interests and display them to the user. They can "like " another user's profile and if the other user also "likes" their profile, they will be able to connect and chat in the app. This gives users a way to meet and connect with each other via music. Vibrations is an Amazon AWS cloud application. We chose a cloud based solution because the cloud provides SaaS; it provides us features that are difficult to implement manually that we can plug into our application. For example, we would not have the time to implement our own signup login service, and it could potentially take us the majority of our development to implement this one feature alone. However, with AWS, we can simply plug Cognito, Amazon's user registration service, into our application. Additionally, the cloud allows us to easily host our application on the Internet. We considered using Microsoft Azure as our cloud service provider. However, we opted for AWS for two reasons. Firstly, AWS has the largest market share for any cloud provider, and we saw value in learning more about and developing with AWS for this reason. Secondly, AWS provides an application called Lambda, which we believed would prove useful when implementing the matching algorithm for our application. Our user interface is written in React.JS, which we chose because it is a popular front end framework that most of us had significant experience in. Additionally, our frontend also connects to two external APIs. First, it connects to Spotify's API [8] to pull data from users' Spotify accounts. Secondly, it connects to a GPS api [5] for determining their location for the matchmaking algorithm. This ensures that a user based out of Virginia will not connect with a user based out of Liverpool for example. Our backend is a Spring Boot application. We chose Spring Boot because we knew for a fact that it could do exactly what we needed. Our backend needed to be able to easily fetch user information from a potentially large pool of users, and we knew for a fact that Spring Boot could do this efficiently. Additionally, many of us had prior experience in Spring Boot.

Our matchmaking algorithm is a Python program hosted in Amazon Lambda [1] which is invoked by the Spring Boot application when performing matchmaking. We chose Lambda because it allowed us to host a program written in a separate programming language which could be stored outside of the server and invoked only at specific times. Python works well as a scripting language for matchmaking algorithms, as it can easily perform distance formulas in short amounts of time.

3 A Motivating Example

Let us consider a person who is very fond of music. He has just moved to a different country where he has no friends. He is feeling lonely and wants to meet new people. But he has no idea how to meet people. In this scenario he comes across the application "Vibrations" which helps people connect with each other based on their music interest. He visits the web application and creates his profile on the portal where he is prompted to give his name, email, mobile no., list of favorite songs, artists and genres and the maximum distance in which he wants to meet people. Once he has signed up on the application, the matches are generated based on the algorithm. He is matched with 10 different people. He connects with them and due to their same taste for music they click and become very good friends. This helps him come out of his phase of loneliness. He also meets a girl among the matches. They connect romantically. It shows that the application can be used for both platonic and romantic connection.

4 Background

4.1 Online Dating

Online dating is the practice of using dating websites to find short-term or long-term romantic relationships. Dating websites work by asking users to create self-descriptive profiles and then matching them with databases of potential matches. Online dating services differ in how they match users with partners. Websites like Match.com enable users to find suitable companions on their own by doing keyword searches for desired relationship attributes. System-selection websites like Bumble and Tinder use mathematical algorithms to find compatible possible matches. Hybrid Web sites (for example, OkCupid.com) use compatibility algorithms to generate "suggested matches," but also allow users to choose their own partners [10].

4.2 Online Dating Algorithm

Online dating platforms use various matchmaking algorithms to connect users based on compatibility and preferences. The specific algorithms vary across platforms, but they generally involve a combination of user-provided information, behavioral data, and machine learning techniques. While recommender systems,

widely used in platforms like Amazon and Netflix, leverage collaborative filtering (CF) algorithms, the specific focus on dating applications is underexplored in existing literature. The Random Algorithm, which generates recommendations based on equally distributed random values within the rating scale, is one of the many algorithms intended for matchmaking. The Mean Algorithm predicts based on the mean value of all non-zero ratings for a given profile, establishing a baseline that ignores individual user input. The User-User Algorithm investigates user similarities by evaluating ratings and making use of the well-known Pearson's correlation coefficient. Similarly, the Item-Item Algorithm uses an adjusted Pearson correlation to measure profile similarity. Parameters such as minimum common ratings and maximum neighbors have an impact on algorithmic performance. Despite claims made by dating services about proprietary collaborative algorithms, the literature on collaborative filtering in the context of dating preferences is limited [2].

5 Related Work

Online dating is a massive market with over 44 million online dating users in the United States and more than 240 million users worldwide. Currently there are few apps which match people based on their music interest and even fewer that give the option to connect platonically as well as romantically. One of them is Vinylly, which takes the user's streaming data and music habits, then asks questions about the role music plays in your life in order to create matches, but this app only intends to connect users romantically[4]. Spotify is one of the most popular music streaming services with over 515 million active users worldwide as of 2023 [7]. Spotify users can search for music, add it to public or private playlists on their profile, and share their playlists with others. Dating apps like tinder allow you to connect a couple songs from your Spotify to your tinder profile but the music makes no difference in the tinder algorithm of connecting users.

5.1 How Our Project Differs

Our project differs from existing work in several ways. First, our application will allow users to connect platonically as well as romantically. Second, our application will use a more sophisticated matchmaking algorithm that takes into account a variety of factors, including the user's musical preferences, gender preferences, and physical distance. Third, our application will be written in React and JavaScript, which will make it more scalable and maintainable than existing applications. Finally, our application will be hosted on AWS cloud, which will make it more reliable and available than existing applications.

6 Implementation

Vibrations was implemented as a cloud application, specifically using Amazon Web Services. We chose the cloud primarily because the cloud would allow us

to host our application on the Internet easily and because it provides us Software as a Service tools, which would allow us to implement many features that would take a substantial amount of time and manpower to develop easily. For example, we used Amazon Cognito for handling logging in and registering accounts. User registration is a large endeavor that would take several months to properly develop. We also used Amazon S3 for hosting files, specifically for user profile pictures. We also used Elastic Container Repository for storing frontend images, and Elastic Beanstalk for hosting the front and backend applications on the Internet. We finally used Amazon RDS to host our database. When choosing cloud services, we considered AWS and Microsoft Azure. We chose Amazon Web Services for two reasons. Firstly, AWS has the largest market share of the two, and we saw value in learning AWS for improving our resumes. Secondly, AWS has a component called Lambda. Lambda is an example of "serverless code," meaning that Amazon will both store the code and run it when we need it. This turned out to be a very useful component for Vibrations, particularly for the matchmaking algorithm.

The frontend, user interface application was written in React.JS. We chose React because it is a highly popular and useful framework for JavaScript, particularly for user interface development. Some members of the group had prior experience with it. It allows us to render pages in real time, which is useful when continuously sending and pulling information from the API. Additionally, the front end React application was stored in a NGINX image. NGINX is an open-source web server that we chose because we had familiarity with configuration. The backend API was written in Spring Boot, which was chosen for a couple of reasons. Firstly, everyone in the group knew Java, and many of the group members had experience with Spring Boot specifically. Secondly, it can retrieve specific data from a large data pool in a small amount of time. This would be useful for Vibrations, because there could potentially be thousands of users if the application was publicly available. Finally, Spring Boot automatically handles wiring of components, from security settings, to Amazon AWS components, to REST controller endpoints. Considering the limited development time, using Spring Boot to automatically wire our components made sense to the team. However, the one problem with Spring Boot and Java is that they are ineffective tools for large scale data processing. Vibrations needed the ability to process large scale data at once for the matchmaking algorithm. The Python programming language can robustly process large amounts of data in a short amount of time. Using AWS Lambda, we could host the matchmaking algorithm, written in Python, and access it from the Spring Boot application. This would allow us to perform matchmaking separately but still be able to store the algorithm's results using Spring Boot.

Testing was largely performed during development. The unit tests seen in the backend repository were written towards the end of the project's development. Instead, most of the testing was done by developers as they developed. Our code is stored in GitHub repositories. We configured the repositories to prevent code from being merged to the main branch unless they passed a verification test.

These verification tests essentially ensured that the code could compile and were used in both the front and backend repositories. Towards the end of development, the front and backend developers were working closely together. As soon as a backend endpoint was ready, we would immediately connect it with the frontend. This method served as the way we tested our code, as backend developers were present when the frontend connected to the backend and would provide on-the-fly changes, improvements, and bug fixes. It is difficult to write tests for frontend React applications. We largely tested our frontend code by simply using it. We would use it, look for problems, identify them when we found them, and then fix them.

7 Deployment Plan

The first steps for deploying Vibrations into a production setting have already been started. At the present, Vibrations is hosted in the Amazon AWS cloud. Specifically, the user interface and API applications are both hosted in Elastic Beanstalk and can be accessed by anyone with the URL. However, there are still many steps necessary before the application is ready for the public.

The website needs more robust security. At the present, the user interface is stored in a NGINX image, which is then hosted in Elastic Beanstalk. The NGINX configurations do not meet CIS Benchmark standards for security. Right now, the API application is accessible to anyone who has access to the URL to the application. However, for deployment, only the user interface application should have access to the API.

Additionally, final deployment would require registering and configuring a domain for the user interface and maybe for the API application. At the present, both applications use the default, Elastic Beanstalk provided domains. These domains are both insecure, as they are self-signed, and very long and difficult for a user to remember. We would need to configure and register a custom domain, such as vibrationsapp.com. Vibrations.com is already registered by an ultrasound company. Finally, we would need to pay for a larger database and more Elastic Beanstalk instances, as the current cloud settings would not be able to handle a large user base.

8 Discussion

8.1 Opportunities for Growth

In the future development of the Vibrations app, several key features and improvements will be essential to enhance user experience and security. Firstly, the implementation of the "send message" feature or a chat room functionality will foster greater social interaction among users, allowing them to connect and discuss their shared musical preferences. To complement this, a significant refinement or overhaul of the UI design is imperative to ensure a visually appealing and intuitive interface that aligns with the dynamic nature of the app.

Introducing two-factor authentication (2FA) will add an extra layer of security, safeguarding user accounts and personal data. Additionally, enabling password reset via email and incorporating email validation mechanisms will enhance the app's security infrastructure and user account management. Persistent user likes will ensure that users can effortlessly track and revisit their selected matched users over time. Lastly, the implementation of an "edit profile" feature will empower users to personalize and update their profiles, creating a more dynamic user experience. We also would implement a feature that ensures user privacy by facilitating contact between users only when there is mutual interest, requiring both parties to have liked each other's profiles before initiating communication. This approach enhances user control and fosters meaningful connections within a more secure and privacy-focused environment. We would also want to transition Vibrations from a web app to a mobile app. The shift would allow for a more immersive experience, giving users features like touch gestures and push notifications and it also aligns more closely with the current landscape of social connection apps. Finally, we would want to transition our Spotify app from developer mode to extended quota mode. While the app is in developer mode on the Spotify developer platform, only 25 predesignated users can access their personal Spotify data for their profiles. In extended quota mode, any user with the app would be able to give Spotify permission to share their data with our app so we can provide an even more personalized user experience.These future developments collectively aim to elevate the overall functionality, security, and user engagement within the Vibrations app, making it a more robust and enjoyable platform for music enthusiasts to connect and share their passion.

8.2 Limitations

During development, we faced limitations with the AWS free usage tier. Relying on AWS services brought challenges due to usage constraints which affected our deployment and the quantity of users we could store. Addressing these limitations involved having to postpone deployment to the cloud and doing regular database maintenance (deleting useless or broken user objects). Another limitation we encountered was the need to learn certain aspects on the fly during app development. While this adaptive approach allowed us to swiftly address challenges, it also posed constraints on the depth of our initial understanding. Learning on the fly meant that some aspects may not have been optimized or implemented to their full potential, underscoring the importance of ongoing learning and refinement as the project evolves. Finally, we also had the constraint of time in regards to usage of the spotify development platform. As mentioned in the discussion of opportunities for growth, it can take up to 6 weeks to receive approval for extended quota mode which would allow for mass dissemination of the app and the ability to receive larger quantities of data for acceptance testing.

9 Conclusion

Vibrations app offers a unique solution to the problem of superficial connections in online dating applications. By focusing on music preferences and fostering platonic connections alongside romantic ones, our web application sets itself apart from existing applications. The use of cloud-based technology and a well-defined matchmaking algorithm ensures scalability and efficiency. Our web application has the potential to revolutionize online dating by creating meaningful connections based on shared passions. With continued development and a focus on user needs, Vibrations can become the go-to platform for music lovers seeking genuine connection, both platonic and romantic.

References

1. Amazon Web Services. *AWS Lambda Developer Guide*. https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html.
2. L. Brozovsky and V. Petrıcek. Recommender systems for online dating service. 2007.
3. B. Elad. Tinder statistics – by users, demographic, match rate, country, usage and social media traffic, 05 2023. https://www.enterpriseappstoday.com/stats/tinder-statistics.html.
4. J. Minor. Vinylly review, 11 2021. https://www.pcmag.com/reviews/vinylly.
5. Mozilla Developer Network. *Geolocation API*. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation$_API$.
6. B. W. Noted. Why young people are using dating apps to find friends—without benefits, 07 2021. https://www.wsj.com/articles/why-young-people-are-using-dating-apps-to-find-friendswithout-benefits-11626101537.
7. J. Shepherd. 23 essential spotify statistics you need to know in 2023, 06.
8. Spotify for Developers. *Spotify API - Web API Reference*. https://developer.spotify.com/documentation/web-api.
9. J. Stevenson. Music: A tool for connection.
10. C. L. Toma. Online dating. *The International Encyclopedia of Interpersonal Communication*, page 1–5, 2015.