Ubitune: ubiquitous, crowdsourced, public music

Final project report for CS7470 Ubiquitous Computing, Fall 2018

Khushman Patel
Georgia Institute of Technology
khushman@gatech.edu

Patrick Friedrich
Georgia Institute of Technology
pfriedrich@gatech.edu

Benjamin Du Preez
Georgia Institute of Technology
dupreez@gatech.edu

F. McKenzie Murphy
Georgia Institute of Technology
fmurphy7@gatech.edu

1 ABSTRACT

We introduce Ubitune: a novel mobile application to create real-time, ubiquitous, location-based, automatically crowdsourced collaborative music playlists for wherever groups of people are gathered. We are primarily interested in solving the problem of playing music in public spaces that cater to the music tastes of everyone present. Ubitune works by drawing music preferences from users' Spotify accounts and performing continuous background location tracking. It then automatically adds or removes the music preferences of users to the music playing in public environments, e.g. Parties, Cafes, Gyms as they enter or leave the space. After thoroughly putting the problem and our proposed solution into context by citing related work on the problem of public music selection and previous attempts to solve this problem, we present the results our own survey on the topic. The results of this survey and our background research indicated that we have identified a clear need in the space and that our solution is both unique and desired. Next, we introduce the Ubitune solution in detail: we present the system architecture (a React Native mobile application, a Node.js server for music selection, and a location database to track user positions), a discussion of each of these components, and we explain the user flow through the app. Finally we present the results of our evaluation. The results were positive, and indicated that Ubitune might revolutionize the way people enjoy public music in the future.

2 INTRODUCTION

Currently, people have little to no control over the music that plays in public, but research shows that they would like to have their preferences taken into account [6]. It is incredibly frustrating to hear the same pop song five times during a shopping trip. The overarching goal of this project was to build an app that can create real-time, ubiquitous, location-based, automatically crowdsourced collaborative music playlists for wherever groups of people are gathered, tailored to the context and the collective music taste of the users present. This will give people control over the music without forcing any action, and improve the public experience. Our first objective was a formative study. We sent out questionnaires to learn how people listen to music, and what their preferences and thoughts are on public music. Our goal was to see how much people actually would care about getting aggregated music recommendations. Our next objective was technical- React Native was used to create Android and iOS compatible versions of an app that aggregates musical preferences into a shared playlist that location hosts can then access. We used Spotify to collect user data on what music they like, and to create the hub playlists– the app is a way of creating and maintaining crowdsourced playlists on a Spotify account. A functional prototype was made that was used in the user engagement study, and we will be updating the app based on user feedback. Our last objective was evaluating user engagement. We evaluated what users thought of the app and the experience of Ubitune, and determined how we should alter it to better fit their needs. We conducted a user experience study in which we showed the prototype and asked users questions. Then, they completed an activity that compared a group using Ubitune versus a group manually creating a playlist. After this was done, as previously stated, we analyzed the feedback to incorporate into a new version of the app.

All the broad objectives stated in this introduction were achieved. After an encouraging formative study, we successfully created a cross-platform mobile application. The app is as ubiquitous as we envisioned: it runs completely hidden in the background and adds the favorite songs of users to the Spotify playlists of nearby hosts around them, without requiring any action from the user. The Android version of the app is currently publicly available on the Google Play store. Feedback was positive overall: we confirmed that the public music listening experience created by Ubitune is much quicker and easier than manually creating a collaborative playlist as a group.

3 BACKGROUND

Crowdsourcing for music is an idea that has been researched for a while [9]. While there have been many attempts at collaborative creation of music, curation of music in public spaces is a problem that has not been solved yet.

A 2012 paper introduced "CheckinDJ": a location-based check-in system to allow crowds to curate music at an establishment [8]. Their approach is heavily hardware dependent. The establishment has to own a specially designed jukebox and users have to physically "check-in" with NFC-enabled objects at the jukebox to have their preferences considered. In addition the creation of the crowdsourced playlist is very coarse: it only takes genre preferences into account.

Most other work done on this problem has been done in the commercial sector. Spotify has a collaborative playlist feature which allows a user's friends to edit their playlists [1]. Flo Music offers a similar service but focuses on real-time playlist creation, i.e. manually editing the music that will play next in the moment and not beforehand like Spotify Collaborate [2].

While both these approaches are an attempt to solve a subset of the problem which Ubitune aims to solve, it has two major shortcomings: it only works when the users collaborating know each other and have shared the playlist between each other (i.e. it won't work in public spaces) and it requires manual input (i.e. it does not disappear into the background). The ubiquitous and automatic real-time nature of the proposed Ubitune solution with no extra hardware required seems to be a novel approach and will provide a complete effortless solution to the problem.

Our project goal is to alleviate what we feel is an issue– people being forced to listen to music they don't like in public. But is this something the general public even cares about? Krause, North, and Hewitt would argue that on some level they do. The researchers investigated how people interact with music in their daily lives, and found that "feeling lethargic associated with recorded music broadcasted in public, in contrast personal music collections promoted contentment" [10]. They also found that having control or input on what music was played was linked to a positive mood response. Thus, the current space looks to be one of people who dislike the current lack of control of public music, and who should react positively to having their preferences taken into account when music is played.

Ubitune relies heavily on Spotify in the backend: we get song recommendations, create playlists on the usersĂŹ Spotify account, and play music all by using the Spotify API. Spotify is a leading music streaming service, known for fitting recommendations to its users. They provide an easy to use API with a broad spectrum of endpoints. We use the Spotify authentication service for our users to log in and gain their approval for gaining access to their Spotify accounts. We then use the following of their endpoints: get a user's top tracks[3], get recommendations[4], create playlist[5], add tracks to a playlist[6] and start a user's playback[7]. In addition, we are using the Spotify authentication service for our users to login and gain their approval for gaining access to their Spotify accounts.

4 THE UBITUNE APPLICATION

4.1 Online Survey

Before creating the app, we wanted to see what people thought of the Ubitune concept and the direction that we wanted to take it in. The idea was that, if people tended to favor a music experience that differed from our vision, we could modify it accordingly. An online questionnaire was made that consisted of 23 statements with answers on a Likert scale that ranged between "Strongly Disagree", "Disagree", "Neutral", "Agree", and "Strongly Agree". Example statements are "I listen to music regularly on my mobile device", and "The music playing in a public place affects my decision to stay there". There was then the start of a statement that was finished by selecting from a multiple choice option. Based on answers to previous questions, participants would be shown up to four open-ended questions to further probe their answers. An example open ended question is "Since you do not listen to music regularly on your mobile device, what device do you primarily listen to music on?". We had 71 responses to our survey, with 64 people completing the survey until the end.The participants were located around the world (but not Europe) and ages from under 18 to over 50. There was an exactly equal gender spread, with 32 men and 32 women. The questionnaire supported our idea that people would like to influence music around them without having to actively do something. Specifically, our multiple choice statement made this clear as shown below.

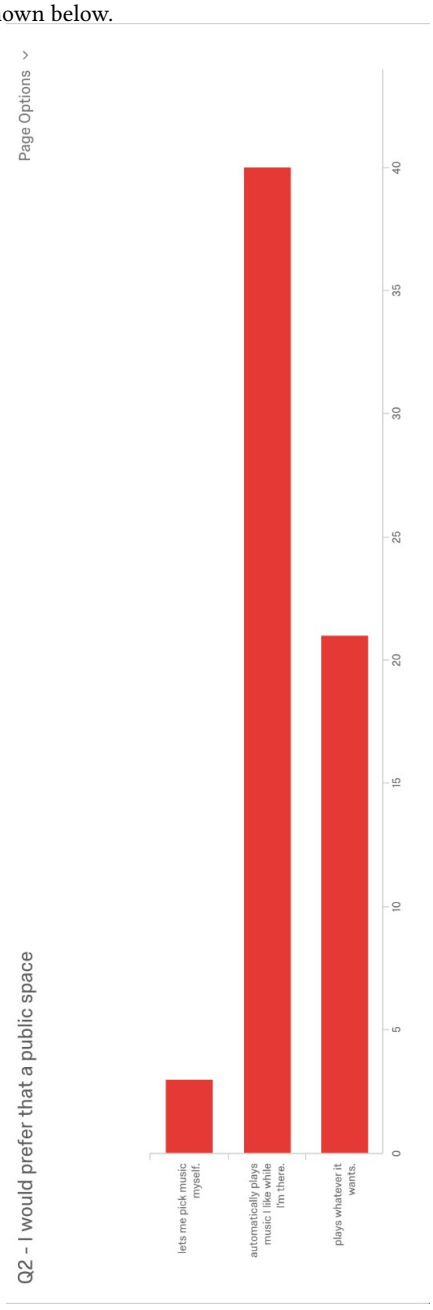

Figure 1: Responses to Question 2 in our online survey

Only 3 participants wanted to pick music themselves, 40 wanted their music to play automatically, and 21 did not care what music was played at all. This implies that people generally did want their music to factor into public playlists, but almost no one wanted to have to take action and pick the songs. In fact, more people wanted to have no impact at all than have to be forced to pick

the music themselves. As the entire goal of our app is to play music ubiquitously without any action from the user, this was validation that we had an idea that people cared about. The Likert scale statements strengthened this, with the majority of people caring about music, playlists, and music playing in public. It is interesting to note that most people had not used a crowdsourcing music service before, and the majority of people did not seem to be very interested in music trends.

4.2 Application Architecture

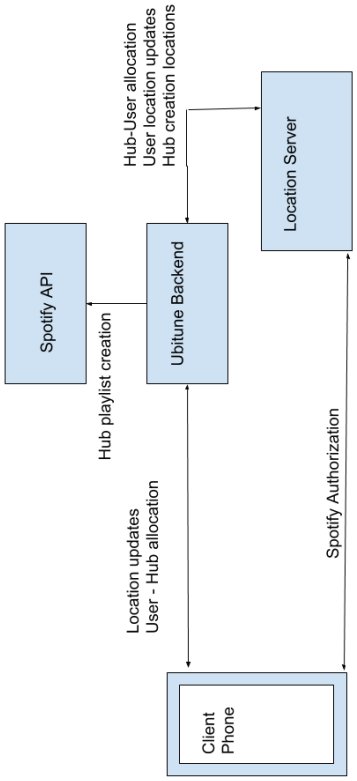

Figure 2: High-level block diagram of the Ubitune system architecture

The different components of the system are based on a microservice architecture, with each handling their own functions. They communicate to each other using a JSON REST API.

4.3 Development Environment

We have used multiple technologies for the various parts of the architecture, depending on the tools available for the pertinent tasks. The system can be broken down into three main components: a frontend client app, location server, and the main backend.

- The client app was developed in React Native, a cross-platform framework for mobile apps, and it runs on both the Android and iOS platforms. Users log into the app with their Spotify account.
- The location database was written in Python using the Django framework, using PostgreSQL + PostGIS for geographic calculations, and is hosted on a Google Cloud server.
- The main backend was written in Typescript, and runs on a Node.js server with a MongoDB NoSQL database, which is hosted on a cloud hosting solution on AWS. This server performs all Spotify API related tasks.

4.4 Software Components

4.4.1 Frontend client app. The frontend of our system is a React Native mobile application. The app provides a gateway to the Ubitune experience and is designed to require minimal user interaction. It has two distinct modes: user mode and hub mode. In hub mode, Ubitune creates a playlist on the Spotify account connected to the app. If any devices are actively playing music using that Spotify account, the playlist created by Ubitune will also start playing on those devices. An example of someone that would make use of hub mode is a coffee shop owner. In user mode no music is being played on the connected Spotify account - instead music preferences is drawn from the connected Spotify account and used to influence the playlists of nearby hubs. As a whole the frontend facilitates three important tasks: Spotify authentication, background location tracking / user hub assignment, and the creation of a new hub.
Spotify authentication only requires input from the user on first launch: the user gets prompted to enter their Spotify credentials. After a successful login, an authentication token is stored and the user will not need to manually log in again on the second launch of the app.
The second very important function performed by the mobile app is continuous background location tracking. Every time the user has moved more than 50 meters from their previously recorded position a new precise GPS measurement is taken and the result is sent to the location server. This automatic background tracking happens regardless of which mode the app is currently in, but it is especially important for user mode: based on this piece of context, the backend can then assign the user's music tastes to a different hub. This key function is what allows Ubitune to be a truly ubiquitous mobile app, since it allows the user to enjoy the full value that Ubitune offers without physically having to open the mobile app. To ensure that the hub assignment is accurate, the app was designed to allow a manual override: the user is automatically assigned to the hub nearest to their current position, but a list of other nearby hubs is also shown in the frontend and the user is afforded the option to switch to one of them if they desire to do so.

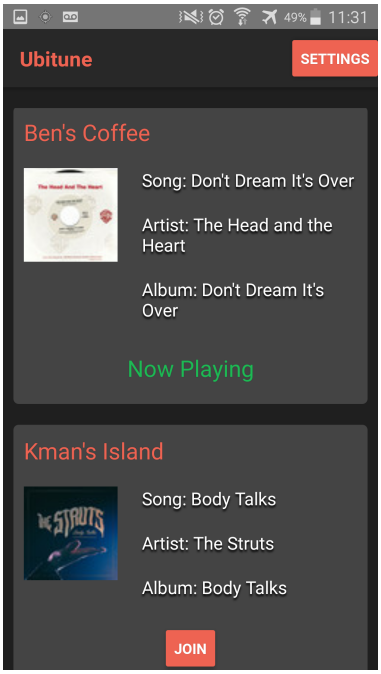

Figure 3: A screenshot of the Ubitune app in user mode. Two nearby hubs are displayed. Ben's Coffee is the closest hub to the user and their song preferences were therefore automatically assigned there. The user also has the option to switch their contribution to Kman's Island if desired.

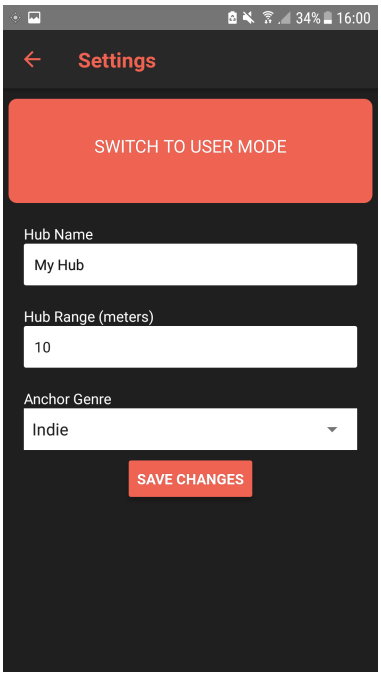

Figure 4: A screenshot of the Ubitune app in the in hub mode settings menu. Hub preferences can be changed here, or the app can be switched to user mode.

Lastly, the mobile app allows the creation of a new hub. This task can only be performed from hub mode. The user can specify a few hub preferences: hub name, anchor genre (the default genre of music with which the created Spotify playlist is pre-populated before any users have joined the hub), hub range (the maximum radius that users can be away from a hub and still be assigned to it). The user's preferences are stored on our server, as well as in persistent local storage so they don't need to re-enter it every time they create a new hub. After saving their settings, the user can activate a new hub and nearby people with their Ubitune app in user mode will be able to join.

4.4.2 Location server. The location server uses PostGIS for most of its geographical calculations. It provides various endpoints that performs location-related tasks, most importantly: updating the usersś location, retrieving the nearest hubs to the user's current location, and the creation of a hub.
The endpoint to update user location uses the locations of active hubs and the latest recorded user location to reassign the user to a different hub, if needed. We used the PostGIS distance_sphere function to calculate accurate geographic distances between the two points and do the assignment.
The endpoint to retrieve nearby hubs is used when a new user is added to the system, and we need to newly assign them to a hub. Cached distance spheres from the previous operations are used to optimize this calculation against all hubs.
The endpoint to create a new hub is used when a user wishes to switch to hub mode, and needs to know all the people around them in a radius. This operation is slightly computationally intensive because of the need to check all the users and their distances. It can, however, be optimized using region assignments in the future.

4.4.3 Backend. The backend resides between the frontend and the location server: it both offers endpoints to the React Native mobile app and consumes the location server's REST API. Three of the eight offered endpoints offered to the mobile app are for the user mode, and the other five are for hub mode. Firstly, /storeUserData sends a request to the location server to get the user information collected during the authorization process (e.g. spotifyID, favorite songs) and then stores that data as a new entry in the MongoDB collection. The endpoint /findNearbyHubs returns a list of the active hubs around the user, with one hub marked as the one being influenced. The raw list of hubs around is received from the location server. The user may also change the hub she is influencing (via endpoint /changeActiveHub).
The first and most important functionality needed for the hub mode is wrapped in the endpoint /activateHub; this creates a new hub at the user's current location. While doing so, the backend uses the Spotify API to get recommendations for the anchor genre specified by the user and then creates a new Spotify playlist with a certain number of the recommended tracks (e.g. 10). The location server is notified to store the new hub as well. The hub owner can /deactivateHub and change the settings of their hub with the endpoints /setHubRange, /setHubAnchorGenre and /setHubName. Every user assigned to a hub (when detected as being within the hub's range) influences the playlist. This is done automatically within the endpoint /findNearbyHubs. Users are assigned to only one hub at a time, which the server marks as their "active" hub.

For the hub marked as the active one for the user (from the list of nearby hubs), two of the influencer's favorite songs are added to the hub's playlist (via the Spotify API).
The MongoDB instance running on the same server as the Node application allows easy and fast data fetching. We use three collections to keep the application data organized: one for the user data, one to assign users to hubs and one to perform live updates on active hubs. When a hub is activated the user's settings are loaded from the first collection and a new document is created in the collection for active hubs. If the user changes the settings of their hub after activation, those changes are persisted in the collection for active hubs. After the hub is deactivated (and deleted from the active hubs collection) any changes made are then written back to the collection with the user data.

5 USER FLOWS

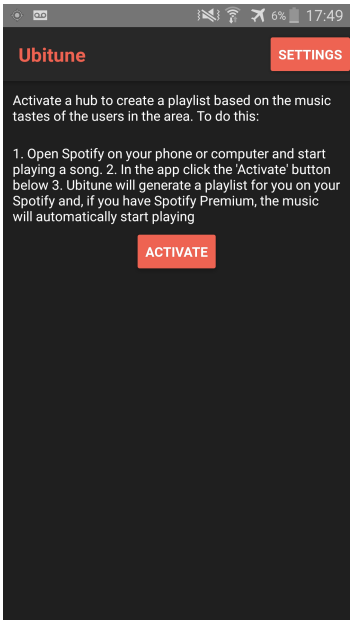

Figure 5: A screenshot of the Ubitune app in hub mode. Pressing activate will create a new hub to which nearby people with their Ubitune app in user mode can contribute their music preferences. This initiates the flow in the next figure.

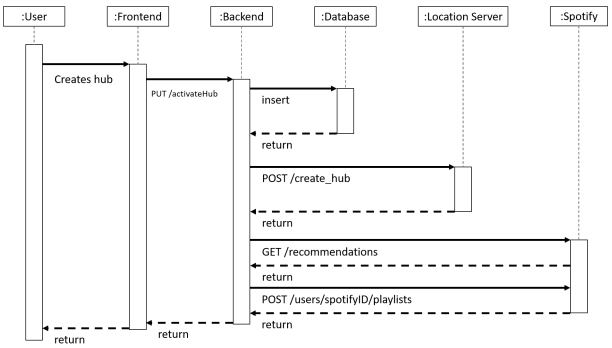

Figure 6: The flow of the hub activation process

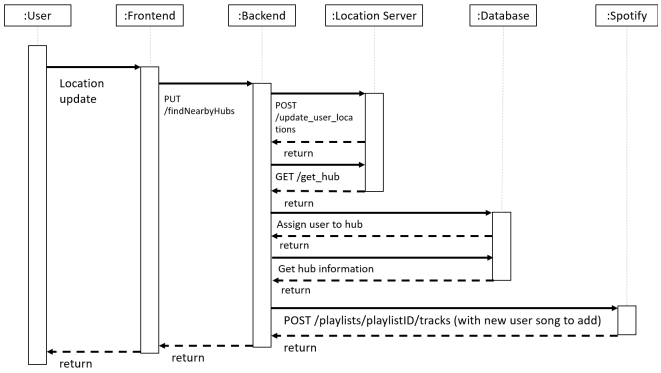

Figure 7: The flow of the process by which a user and their music tastes are assigned to a nearby update after a location update

6 PROPOSAL ALIGNMENT

For our project, we had four main objectives. These are reiterated here and we explain how we achieved them. First, we formulated the general goal of this project, the vision: to automate the process of creating collaborative music playlists. These playlists should fulfill certain requirements. Ubitune creates and updates in real-time when a hub is created and a new user joins a hub respectively. A hub's playlist is location-based by only incorporating the music preferences of users in the hub's range. There is no need for any user to choose songs manually. All songs are added in the background by incorporating the user's Spotify preferences. The playlists are thus automatically crowdsourced. In addition, to honor the main theme of this class, Ubitune works ubiquitously; everywhere, at anytime and woven into the user's everyday experience. To individualize the user experience further, the incorporation of activity recognition is on our roadmap. This falls into future work though. Ubitune currently uses Spotify to create playlists. This has the positive side-effect of persisting the playlists, such that the hub owner may go back to it later and analyze or enjoy it. In terms of market research, we wanted to conduct two user studies during the semester. For the first one, we opted for the online survey where we managed to get the feedback of 71 participants on how they consume music and the way they want to influence the music played around them (see dedicated sections for details). Moreover, we tested the prototype of the Ubitune application built with eight study participants in a user engagement study and with five people in a less formal setting to observe how potential users interact with the mobile app and prioritize next steps. Foremost, we wanted to build a functioning Android mobile application for this project. As we built it with React Native, we were not only able to deliver a working Android app (currently available on the Google Play Store) but also a corresponding application for iOS.
While achieving all of our initial ambitions turned out be more challenging than anticipated, we were still able to achieve most of our goals and follow through on all our core objectives as stated above. Nevertheless, this also means that we are aware of important future steps for improving Ubitune.

7 RESULTS

After creating the app, we knew that we needed to test it with real potential users. Before creating the evaluation protocol, we had a discussion with Udaya Lakshmi, a PhD student in the Ubiquitous Computing Lab at Georgia Tech, to fully understand usability. She clarified for us that since our app is so ubiquitous, we would not really be testing usability for our users– in an ideal scenario, the user does not need to look at their phone much at all. Instead, we would be testing an experience. Does the user see the value in our service? Do they like what the app does? Is it easier than doing the same task manually? Based on this feedback, we created an evaluation protocol with two main tasks. We had two groups of participants, to test two different use cases. Each group consisted of four people. The participants were all in their twenties, and all but one are currently students at Georgia Tech. Both men and women represented. The first task was a one-on-one walkthrough with the participant to explore the usability of the hub mode. While both app modes center on experience, the hub mode requires more interaction and specific ordered tasks, and is therefore appropriate for usability evaluation. The second task was a group task; it compared creating a playlist in a group and playing it manually versus using Ubitune to create the app. Participants were first asked to create a Spotify playlist on the computer as a group, and were timed to see how long they took. Next, they were asked to create a playlist as a group using Ubitune, and were again timed. The time differences were compared to see how Ubitune changed the base case of how users normally make playlists. As previously mentioned, we had two sets of participants for two different use cases. We foresee that our app will be used either in private situations, like for parties or ambient music in homes, or it will be used in public, like at a coffee shop, or a sponsored event. To capture use in these cases, in our second task our first set of participants (Group 1) were instructed to create a playlist for themselves, that they would then listen to. The second set of participants (Group 2) were instructed to create a playlist for a coffee shop owner that he would then play in his shop. This engagement study provided valuable feedback. Our users told us what they thought was intuitive and what needs work– specifically, setting up a hub is fairly confusing and needs more direction in an engaging way. There was also feedback on what should be factored into creating the hub playlist, as hub owners may want more options than just genre. We plan to implement these changes. However, at its core, Ubitune worked, as the following graph shows:

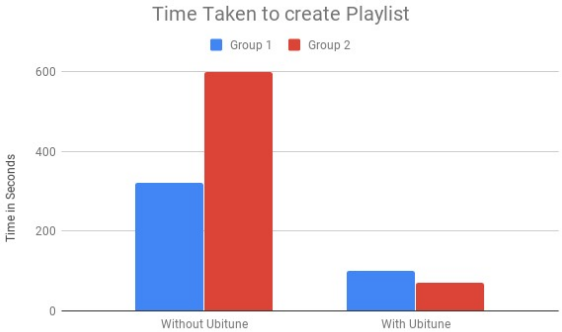

Figure 8: The time duration each group spent to make a collaborative playlist with and without Ubitune

In our second task, the participants took over 300% longer to make a playlist manually than using Ubitune. The target audience of the playlist also had an effect, with Group 1 taking 5 minutes and 22 seconds to create the playlist and Group 2 getting cut off after 10 minutes of working on their playlist. Group 2 did have some network connectivity issues (poor WiFi connection in the testing room), but still took drastically longer. Thus, while Ubitune is helpful in creating playlists for both use cases, it seems that it will be even more helpful in some situations than others, specifically creating playlists for the public. The difference in time between taking 5 minutes to do a task versus 1 minute is huge, giving Ubitune high appeal to various audiences. Lastly, we asked participants how likely they were to recommend Ubitune to a friend to assess its Net Promoter Score.

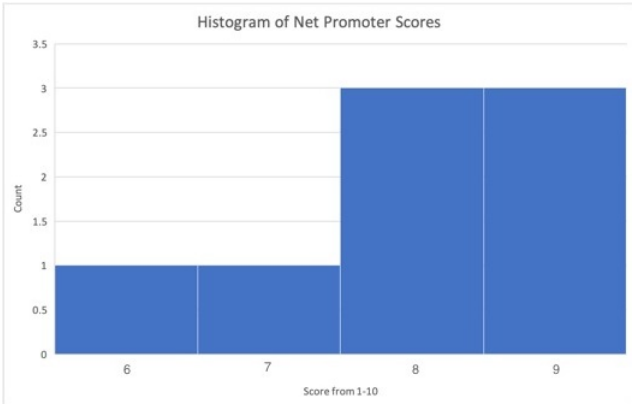

Figure 9: Histogram of Net Promoter Scores

The mean score was an 8, with a low of 6.5 and a high of 10; this signifies that people generally like the app enough to recommend it to someone else. Sharing, or lack thereof, is the ultimate factor in app success. Based on the Net Promoter Scores received, it therefore appears that Ubitune is likely to be shared and gain popularity.

8 CONCLUSION AND REFLECTION

This project achieved a new way of listening to music; specifically, we created a ubiquitous listening experience that takes almost no effort on the part of the user. The app connects to a user's Spotify account and pulls their favorite songs, adding them to a hub playlist if they are in range. All of the project objectives that were set out have been achieved – we used a formative study of questionnaires to plan exactly how the app would work, and tested it with a user engagement study near the end. We also completed development on the app and it is now available on the Google Play Store. Users generally enjoyed using Ubitune, and gave valuable feedback. They showed that Ubitune is indeed faster and much easier than manually making playlists, and said that they would be likely to recommend the app to their friends. Overall, the project process went very well. We achieved all of the goals we had set, and created a functional prototype. We also worked well together and split tasks evenly. However, there were some aspects of the project that were not as smooth. We did not have time to do multiple rounds of user testing,

as the app development was a much longer process than anticipated. Instead of several sprints, we ultimately had one– we were not as Agile as we thought. We would also like to improve our playlist song selection and how playlists are originally seeded to further improve the user experience. This project could revolutionize how people interact with music in public. Instead of being forced to listen to Justin Bieber five times in a day, people could instead hear songs they like everywhere they go, and discover new music that may not make the radio top 20. This might spread throughout public spaces – coffee shops, stores, restaurants, gyms – all of these spaces could benefit from Ubitune. Other services could use our inspiration to create similar services, such as creating playlists from people around a user that depend on the user's activity and play personally on that user's phone. Future work could also improve the music selection for playlists; perhaps a seed genre could simply be "coffee shop music" or "x beats per minute" instead of genres like "pop". The public music experience could become much more personal and intimate compared to the blaring noise that exists now.

9 ACKNOWLEDGEMENTS

We would like to thank Chrisjan Wüst, an electrical and electronics engineering student from Stellenbosch University (South Africa), who played a big role in inventing the Ubitune concept and kindly encouraged us to use it for this project. We also want to thank Thomas Plötz, our mentor, who was always very helpful throughout the duration of the project.

REFERENCES

[1] [n. d.]. Collaborative playlists. https://support.spotify.com/us/using_spotify/playlists/create-playlists-with-your-friends/. Accessed: 2018-09-09.
[2] [n. d.]. Flo Music. https://www.forbes.com/sites/nelsongranados/2016/05/27/flo-finally-an-app-to-crowdsource-live-the-partys-playlist/. Accessed: 2018-09-09.
[3] [n. d.]. Spotify API. https://developer.spotify.com/documentation/web-api/reference/personalization/get-users-top-artists-and-tracks/.
[4] [n. d.]. Spotify API. https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/.
[5] [n. d.]. Spotify API. https://developer.spotify.com/documentation/web-api/reference/playlists/create-playlist/.
[6] [n. d.]. Spotify API. https://developer.spotify.com/documentation/web-api/reference/playlists/add-tracks-to-playlist/.
[7] [n. d.]. Spotify API. https://developer.spotify.com/documentation/web-api/reference/player/start-a-users-playback/.
[8] Daniel Burnett, Mark Lochrie, and Paul Coulton. 2012. CheckinDJ using check-ins to crowdsource music preferences. (10 2012), 51–54.
[9] Carlos Gomes, Daniel Schneider, Katia Moraes, and Jano De Souza. 2012. Crowdsourcing for music: Survey and taxonomy. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. IEEE, 832–839.
[10] Amanda E. Krause, Adrian C. North, and Lauren Y. Hewitt. 2015. Music-listening in everyday life: Devices and choice. *Psychology of Music* 43, 2 (2015), 155–170. https://doi.org/10.1177/0305735613496860