CROWD CONTROL

SOCIAL COLLABORATIVE MUSIC SELECTION

LASSE HEDEGÅRD RASMUSSEN, 202005931
HENNING LIN, 202007040
ANDREAS GRAMVAD JENSEN, 202004592

UBI 9

BACHELOR'S THESIS
June 2023
Advisor: Niels Olof Bouvin

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CROWD CONTROL

Social Collaborative Music Selection

UBI 9

Bachelor's Thesis

Department of Computer Science
Faculty of Natural Sciences
Aarhus University

June 2023

ABSTRACT

This thesis presents the development and evaluation of Crowd Control, a web application designed to enhance user engagement and enjoyment in social collaborative music systems. The application allows users to create and join collaborative playlist sessions, influencing the music through song requests and voting.

The objectives of the study were to investigate the impact of Crowd Control on user engagement, explore its influence on the music experience in social gatherings, and assess the convergence of genre suggestions among users over time. Two user studies were conducted, focusing on technical implementation and qualitative evaluation through questionnaires. Our findings indicated that Crowd Control increased user engagement and provided a more enjoyable music experience. Although our original hypothesis assumed that the system would result in a convergence of genre suggestions, findings indicate that user suggestions remain varied across multiple genres.

The limitations of the study, including sample size and controlled environments, are acknowledged, and suggestions for future research are proposed. Overall, this research contributes to the understanding of user engagement in collaborative music systems and offers insights into the challenges and opportunities of promoting inclusivity in the social music experience.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

With the proliferation of music-capable devices, the landscape surrounding music listening has undergone significant changes. Originally, music was primarily consumed in the public sphere, in social gatherings such as bars, concerts, and festivals. If listeners wanted any control over the music, the only option was record players, or tune into radio stations that matches their tastes. Today, a majority of people own one or more devices capable of playing music, and some have multiple speakers, all placed in separate rooms. Music streaming services like Spotify and Apple Music offer on-demand access to a vast array of music, ranging from high-fidelity studio releases by popular artists to music uploaded and sold by amateur musicians. This accessibility and variety have allowed listeners to develop diverse and individualized tastes in music, enabling them to listen to their preferred genres and artists on their own terms.

However, in the public sphere, little has changed regarding how we listen to music. Whether at bars or radio stations, music is typically curated by a single person, often a DJ or radio host, who may take requests from the audience. In smaller social gatherings, controlling the music selection usually rests with the owner of the speaker or is distributed to whoever expresses the most interest in controlling it.

A single person or small group controlling the music for every other participant might leave a majority of people feeling unheard and unsatisfied with the music being played.

Collaborative playlists, such as Spotify's[1] shared playlists, are commonly used to ensure that all participants have the opportunity to include songs in the playlist. This allows users to have their song wishes played at the event, ensuring that all participants feel included. However, it does not guarantee that other participants will agree with or enjoy the music selected.

In this thesis, we explore ways to improve the music experience through a system that allows users to engage with and collaboratively influence music selection in social settings. This system will focus on using weighted voting, a voting system that grants greater influence to some users. We examine whether a collaborative playlist system utilizing weighted voting on song suggestions can guide users toward creating a better music experience for all. Additionally, we wish to test how social dynamics and user experience is affected by a system where users have unequal influence.

1 https:/spotify.com/

1.1 CROWD CONTROL

To facilitate our investigation, we introduce Crowd Control — an interactive web app developed with the FERN stack. It consists of a landing page where multiple users can create or join music sessions. Once a session is established and populated, the owner assumes the role of a central speaker, while other participants can request songs and actively participate by upvoting and downvoting tracks. Ratings for a user's song suggestion directly affect their status in the system, affording greater influence to users that suggest songs liked by the general user base. Our aim with this system is to reward popular and well-liked song suggestions, potentially leading to overall increased satisfaction with the playlist.

In this thesis, we conducted experiments to evaluate how user experience and engagement are affected by the Crowd Control application. Combining quantitative analysis of user interactions with qualitative insights from user opinions, we sought to get a complete understanding of the system's performance and users' perceptions.

By exploring the interplay between weighted user ratings and collaborative playlists, we hope to provide valuable insights for the design and development of interactive music platforms that foster active user engagement and enhance the social music experience.

The Gitlab code for Crowd Control can be found in Section A.2.

A video demonstration of the system can be found here: `https://www.youtube.com/watch?v=lW1Ebe9gevY`

1.2 CENTRAL QUESTIONS

To guide our work in this exploration, we formulated three hypotheses:

- **Hypothesis 1:** Over time, the influence of the Crowd Control web app will result in a convergence of song choices among users, aligning with session genre trends.
- **Hypothesis 2:** The implementation of the Crowd Control web app, will result in a more enjoyable music experience overall.
- **Hypothesis 3:** The implementation of the Crowd Control web app will increase user engagement among the users and with the music.

1.3 STRUCTURE OF THE THESIS

The thesis is organized into several chapters. Chapter 2 provides an overview of related work in fields such as recommendation systems, collaborative playlists, and technical implementations. Building upon

this foundation, Chapter 3 discusses the design implementations and how they were influenced by the related works. In Chapter 4, we present the intended structure and implementation of the Crowd Control app, based on analysis conducted in Chapter 3. Chapter 5 details the current implementation of Crowd Control, which served as the testing tool for our user studies. In Chapter 6, we present and evaluate the results of our user tests, highlighting key findings from the collected data. Finally, in Chapter 7, we draw conclusions regarding our hypotheses and the state of Crowd Control, discussing future work and areas that require further exploration.

2 RELATED WORK

Several studies have concerned themselves with developing and studying ways to improve the music listening experience of both individuals and groups. A considerable amount of research has been dedicated to developing context aware systems, meaning systems that select songs to play based on the context of the user group. Another focus of research is collaborative playlists. A collaborative playlist is understood as a music-playing system where some or all of the users have an influence on the music being played. Some collaborative systems also implement voting as a way of democratizing the music selection process. Related research has explored the implementation of weighted voting, where some users have a greater influence on the system.

2.1 CONTEXT

The concept of context is in itself obscure as it can be taken to mean all information relevant to a setting or user at a given time. Seeking to gain an objective understanding of user context can prove to be a never-ending endeavor. Regardless, several attempts have been made to explore and analyze many different factors that form context as a whole. Kaminskas and Ricci [8] explored the use of location tags in relation to places of interest (POI). The study took users' assigned tags for songs in their playlists and tags given to POI. The system then groups songs and locations together based on the tags to create context based on user location. Schmidt et al. [26] argued that "ultra-mobile" computing devices would allow us to determine context from more than just location, outlining use cases for different sensors and the context they can provide. Ning-Han and Shu-Ju [12] present a music recommendation system that infers context based on the time of day and users' interests. The users' ratings would be timestamped, and further song recommendations would be based on this time context. Giri and Harjoko [6], further expanded on time as a context, creating a system where mood, time of year, weather, and more were used as contexts to create groups of songs with tags matching the context. The system would then play songs based on the user's current context, matching them with the song tags. Reflektor[1] is a recommendation system that uses the context of users' chat logs to create playlists. Focused on events or meetings, users will converse about music, and the system will then create a playlist for the event based on the sentiments of these chats. One of their findings was the need

for *mutual understanding* of context, for context-based recommendation systems to work.

The rise of context-based recommendation systems led to many other researchers attempting to describe and define context in terms of music selection. Schedl and Knees [25], states that context in recommendations systems exist beyond just the user, but also context around the artist and music itself, thus making context multimodal. Music context in this regard can include geographical origins of the artist, artist and users political stance, their gender, etc.

Scheld et al. [24] outlines the importance of users in recommender systems, but also states that users' context is largely neglected in research and development. On the other hand, they also remark that user context is particularly volatile and difficult to determine. Besides the user context and music context, Scheld [23] later added how music content also is important to the recommendation system. Music content refers to information directly related to the sound being played, genre, beat, tempo, etc.

2.2 COLLABORATIVE PLAYLISTS

As described in Section 2.1 trying to infer user context is a difficult task. An alternative to trying to infer user context is to let users decide it for themselves. People often intrinsically grasp their own context at any given time. Meaning that, unless the goal is discovering new music, users are often more capable of selecting appropriate music than any context-inferring algorithm could hope to be. With users having an intuitive understanding of their own context, allowing them to influence the systems themselves is often a valid choice. However, in group settings allowing several users to influence the music poses several challenges both technical and social.

Systems like *What a Juke* [3] focus on facilitating the streaming of users' songs from a technical standpoint. Seeking to facilitate collaborative music listening by emulating a jukebox providing a central unit from which music is played. Everyone within range of the jukebox can connect to the system and stream locally stored songs from their phones to the playlist playing. By allowing users to stream music directly from their own devices to the central unit, all users can contribute to the playlist. Queue[14] removes the requirement for a specific artifact in favor of allowing users to use their own device as the central host. Hosts create a lobby that guests can join, from here users can add tracks to a playlist played by the host device. Notably the streaming service Spotify also recently released its own collaborative playlist feature [1], where users can add and remove songs from a shared playlist that they all have access to.

1 https://explore.spotify.com/us/pages/mobile-feature-share-collaborative-playlists

While both What a Juke and Queue focus on architecture for implementing peer-to-peer systems allowing users to connect and share song data, other research is ethnographic in nature. Here researchers are concerned with understanding the use cases and outcomes of using collaborative playlists. In their studies, So Yeon Park et al. [17–21], conducted several interviews and user studies with users and non-users of collaborative playlists. The result of their studies outlines who, what, when, where, why, and how collaborative playlists are made and expanded. Their findings suggest that playlists often are made for a practical goal, often an event of some sort, road trips, parties, or other social gatherings. Their research also looked into the emotions of users of collaborative playlists. Highlighting how communication between users is critical as some changes to a playlist can affect people negatively, leading to conflict.

Lenz et al. developed Mo [11], a portable music player, seeking to address the social and emotional aspects of music sharing. From their research, they outline several social factors in music sharing. Firstly they found that users generally do not want to have the sole responsibility of selecting music, mostly for fear of complaints and rejection over their choices. Further, they found that having a centralized point of interaction for controlling the music disincentives certain users from interacting with the music. This occurred because having a central object of interaction requires you to leave existing social situations to engage with the music and because a central artifact draws attention to anyone interacting with it. Lastly, they argued that the ability to skip songs from other users could lead to negative social interaction. Skipping a song in essence is a blunt rejection of a suggester song which could cause feelings of anger or sadness in the suggester.

2.3 USER INFLUENCE AND VOTING

As mentioned in Section 2.2 systems often seeks to allow users to influence the playlist directly or indirectly. UbiRockMachine [10] is a system for use in public urban settings. The system allows musicians and music producers to upload their music to a central speaker. Other users of the space can then vote on the songs in the system and play them on the public speaker. Overall feedback from the study suggested that allowing users to vote on music played in public settings made for a more enjoyable listening experience. MusicFX [13] is a system made to arbitrate and appeal to the broadest audience possible in a fitness center. The system works by recording every user's preferred music genre and rating them on a number scale. When people would use the fitness center, the system would aggregate all the genre ratings of users currently in the center, and play music from a radio station playing the most liked genres. The study found that

appealing to the broadest audience maximized the number of people that were content with the music being played. Flytrap [4] is a system for group listening. The system records and registers what songs and genres users listen to on their computers and phones. When a user enters a group setting, the system uses RFID chips to register who's using the system and implements a voting system based on the context of each user's music history. Each user rates genres and songs they listen to numerically, and the system plays songs with the highest overall ratings. As seen in the last two examples, the use of numerical ratings is often employed, to infer user preferences. Another tool for researchers is to make us of democratic systems when designing systems with user influence. By giving each user a vote to weigh in on the decision made for their group all users are taken into consideration and their preferences are heard. R-Music[28] is a system of collaborative listening. Users stream songs from their own devices to a server, responsible for playing the music. The system makes use of a voting system, where users call referendums on songs in the playlist, affecting when and if the song will be played. The system also implements mechanisms for balancing votes, making sure that some users do not control the system, or that some users are isolated from the system. Jukola [16] is another system that implements voting while making the voting process more of a game. The process of voting consists of tables in a bar having a handheld system from where they can vote for songs. The system allows users to walk around from table to table, discuss songs and obtain critical mass of votes for a song.

2.4 WEIGHTED VOTING

To the best of our knowledge vote-based collaborative playlists have only been implemented using an unweighted voting system, meaning all votes from all users are equal. However, collaborative systems making use of weighted voting have been explored within other fields. Gjorgji Kjosev [9] has experimented with the development and testing of a weighted voting system for web interaction. Kjosevs system is based on ideas of meta-moderation.

In the system users all start off with the same level of influence. As users upload content their influence increases or diminishes relative to the reception of the uploaded content. A user can achieve up to twice the influence of a baseline user. When a user casts a vote on a piece of content, that user's influence score will amplify the votes weight, and the score of both the content and its creator is then adjusted accordingly. Though not implemented, Kjosev also discusses having the system calculate an appropriate level of influence to assign user joining late. This ensures that new users are not locked out of the system immediately upon joining.

For more reading on rating systems and meta-moderation, we recommend the book The Reputation Society [7], with each chapter written by researchers and their findings related to rating systems.

2.5 TECHNOLOGY

There are countless ways for developers to implement functional and robust web applications. The FERN stack, consisting of Firebase, Express.js, React.js, and Node.js, is a powerful combination of tools used for building modern web applications. Each component plays a specific role in the stack, providing distinct benefits.

2.5.1 *Firebase*

Firebase [2] is a comprehensive backend-as-a-service platform provided by Google. It offers a range of cloud-based services, including real-time database, authentication, cloud storage, hosting, and more. Firebase simplifies the backend development process by abstracting away server management and providing ready-to-use APIs, enabling developers to focus on building features rather than infrastructure.

2.5.2 *Express*

Express.js [3] is a fast and minimalist web application framework for Node.js. It provides a robust set of tools and features for building web servers and APIs. With Express.js, developers can handle HTTP requests, define routes, and implement middleware easily. It offers flexibility, allowing developers to structure their applications as per their requirements and integrate additional libraries for enhanced functionality.

2.5.3 *React*

React.js [4] is a popular JavaScript library for building user interfaces. It allows developers to create reusable UI components that efficiently update based on changes in system data. React.js follows a component-based architecture, promoting code reusability, maintainability, and a modular approach to building user interfaces.

2 https://firebase.google.com/
3 https://expressjs.com/
4 https://react.dev/

2.5.4 *Node*

Node.js [5] is a backend JavaScript runtime environment built on Chrome's V8 engine. It allows executing JavaScript code server-side, enabling developers to build scalable network applications with JavaScript backend. Node.js follows an event-driven, non-blocking I/O model, making it efficient for handling concurrent connections and processing requests asynchronously. It also boasts a vast ecosystem of packages available through npm, providing access to numerous libraries and tools for building server-side applications.

5 https://nodejs.org/en/about

3 ANALYSIS

In this chapter, we will analyze the related work from Chapter 2 and discuss how our work relates to and builds on previous work in the area. Lastly, we describe the technical details of how we plan on implementing Crowd Control.

3.1 ANALYSIS OF RELATED WORK

As mentioned in Chapter 2, the concept of collaborative playlists arises from the necessity to resolve conflicts among users' musical preferences. In response, various systems have been developed to ensure that all users have equal access to contribute to the music selection.

Equal access, in this case, means that all users have the ability to add their preferred songs to the playlist. Given equal access to the playlist, users can ensure that they enjoy at least the songs they suggested, but they have no effect on the rest of the session's suggestions, leaving them only slightly better off, than systems with a single individual choosing every song. Consequently, the potential for dissatisfaction still exists within the system.

This situation highlights the need to address the broader context within a group, rather than solely focusing on equal access. Consequently, influence-based collaborative playlists have been implemented, where each user can influence the music being played by expressing their preferences. These preferences can be indirectly indicated through user data [13] or directly through voting on songs [28]. To ensure fairness, influence-based solutions allocate equal influence to all users.

However, some users may be better able to read the overall context of an event and can suggest songs that align with a larger majority. These individuals should be regarded as more reliable sources of good song suggestions, as their ability to accurately gauge the collective context makes their input more valuable. Conversely, users who consistently propose unpopular songs may be perceived as lacking the ability to assess context effectively, and their suggestions may carry less weight in the decision-making process. By acknowledging these differences in reading the crowd's preferences, collaborative playlist systems can prioritize suggestions from users with a proven track record of understanding context, thereby enhancing the overall satisfaction of the group.

Through the implementation of Crowd Control, we wish to explore how a weighted voting system affects user interaction, experience, and satisfaction. Implementing a weighted voting system might serve to improve the listening experience for more users by providing a bigger influence to users who have grasped the general context of the user base. We believe there is value in exploring Collaborative Playlists through the lens of weighted voting.

3.2 IMPLEMENTATION

The implementation of Crowd Control will be based on the FERN stack as described in Chapter 2

3.2.1 *Firebase*

The core concept behind Crowd Control relies on robust real-time data updates sent to all users of a session. Firebase allows us to largely abstract our backend development unto Firebase. This suits the scope of our project perfectly as simplified backend development allows us to focus on developing for testing our hypothesis. Primarily, Firebase provides a real-time noSQL database, Firestore, for our application. This makes it ideal for managing our data across multiple users in real time, as clients simply react to changes in the database and update the user's view as new data is entered. This also allows us to forego a layer of middleware that manages and enters inputs from all users. Instead, all clients are simply allowed to enter new data directly into the relevant parts of Firestore as needed. Firebase also provides secure authentication of users through a variety of identity providers such as Google, Facebook, GitHub, etc. meaning we can easily and safely verify user identity. User authentication is mainly used to ensure that hosting sessions cannot be abused by bad-faith actors. It also enables us to grant different access levels based on whether a user is acting as a host or a client, as well as allowing us to store user-specific information for future use.

3.2.2 *Spotify*

For our application to be able to play music we need a library to draw music from. Spotify has one of the largest music libraries in the world and is also the most widely used music streaming service. This means that users are very likely to already have a Spotify account and to have experience with their search engine and library selection. Spotify also has an extensive API allowing us to integrate the functionality of our application with Spotify's capabilities. Spotify's API is extensive and offers many different functionalities however we will primarily use it to interact with their search engine and song player. It

is important to note that using Spotify to play music requires the session host to have access to a Spotify premium account to play songs in the desired order.

4 DESIGN OF ARCHITECTURE AND EVALUATION

4.1 APPLICATION CONCEPT

The goal of our system is to create a robust and responsive web application that allows users to easily create and join various party sessions. When in a sessions, users will be able to suggest songs to be played and react with their preference, on the suggestions of others. We hope to use this system to explore how feedback from other users affects the use and experience of the system, as well as its influence on the satisfaction of the user base. An overview of the user interface concept for Crowd Control can be seen in Figure 1.

4.1.1 *Landing Page*

The landing page of our application simply presents a user with 2 choices, to host a session or join a session. This allows users to quickly join an established session without needing to go through a cumbersome login process. Alternatively, they can create their own session if they desire. The landing page can be seen in Figure 1(a).

4.1.2 *Hosting*

Hosting a session requires more verification than simply joining one. If a user wishes to host, they must confirm their Google credentials to Firebase, see Figure 1(b), Afterwards they must log in to their Spotify account to enable the host device to play music through Spotify's API, see Figure 1(c). Users are then asked to provide a name for the session, see Figure 1(d), this name is then connected to a unique identifier in Firebase. Once a session is created, the host' view displays both a PIN and a QR code that other users can use to easily join the session, see Figure 1(e). The sequence diagram for creating a session can be seen in Figure 2. The host device will oversee communication with Spotify's API, making search requests and queuing songs.

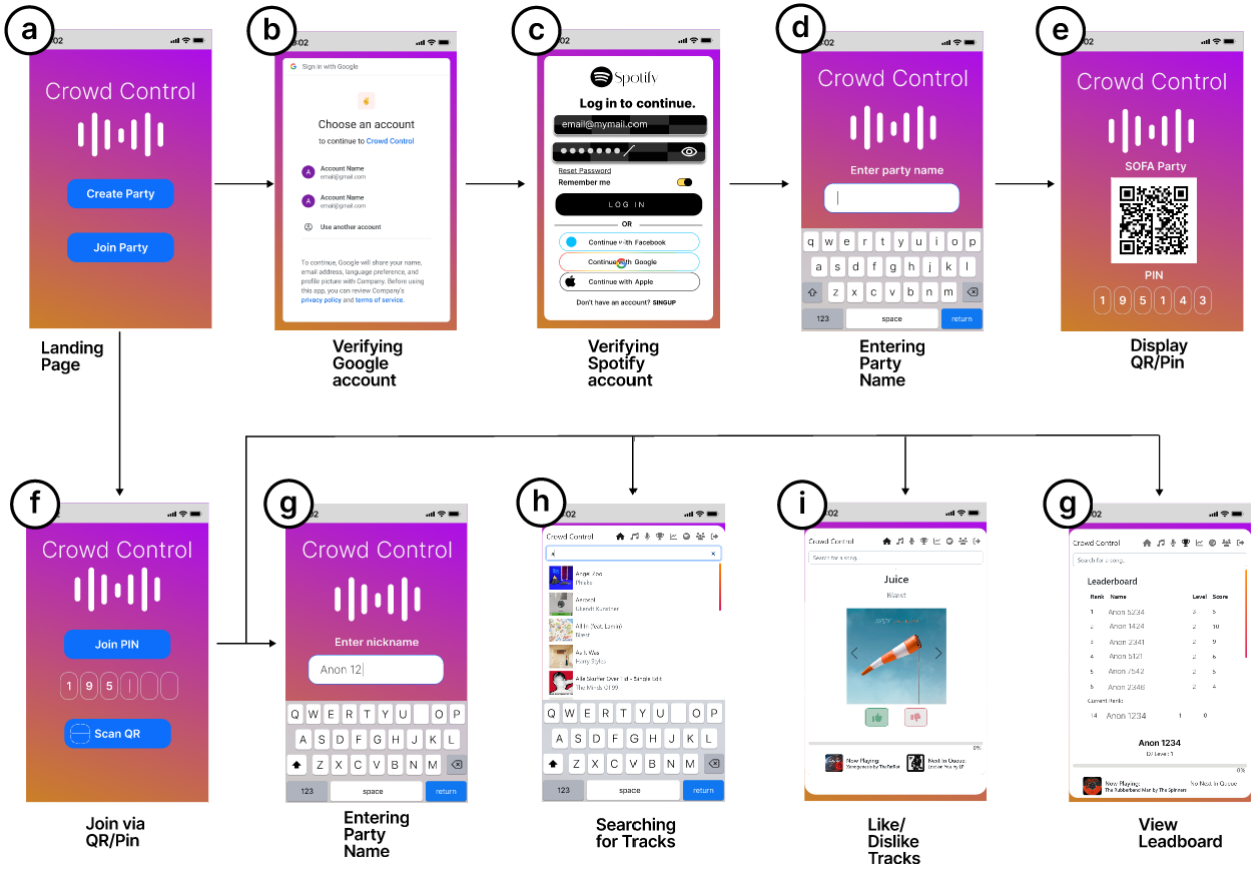

Figure 1: An overview of key design elements of the user interface: **a.** The landing page presents the option to either create or join a session **b.** If the user chooses to create a session, they must verify their Google account **c.** Next, they must sign in with their Spotify account **d.** They are then prompted to choose a session name **e.** After the session is created the host screen displays PIN/QR codes to enable other users to join the party. **f.** Users joining are presented the option to join either via QR/Pin **g.** When joined users can search for and request a track to be played **h.** They may also like/dislike suggested songs **i.** Several alternate views are also available, here we see the leader board view.

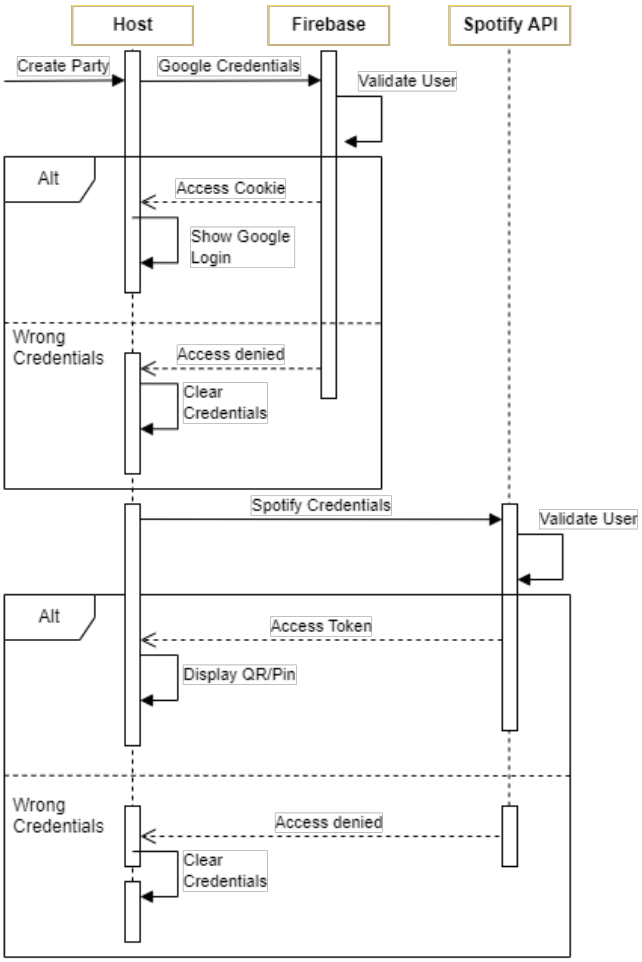

Figure 2: A Sequence diagram for hosting a session

4.1.3 *Joining*

When joining a session, the user enters the PIN associated with the desired session. Alternatively, if the user's device has access to a camera, they can scan a QR code to join, see Figure 1(f). Users are then prompted to enter a nickname that will be used for the session, this provides users with the ability to be anonymous in the system. After a user joins an access cookie is stored in their browser, allowing the system to track a given user. The sequence diagram for joining a session can be seen in Figure 3.

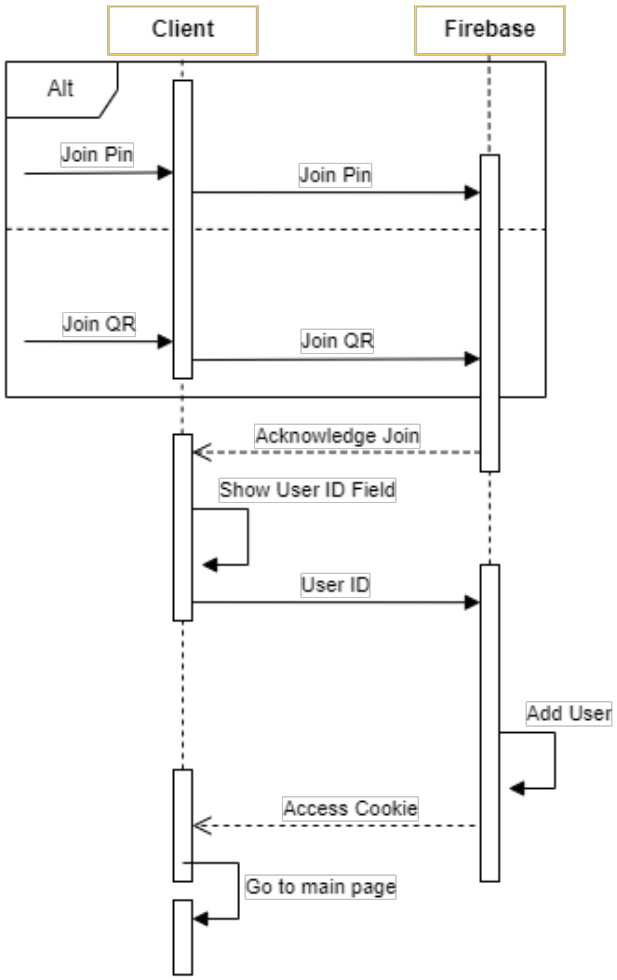

Figure 3: A Sequence diagram for joining a session

4.1.4 *Interaction*

When the user has joined a session, they will be able to interact with the system in several ways. One of the primary ways of interacting with the system is through the search and suggest function, see Figure 1 1(g). Here users can search for their desired tracks and add them as their suggestions. With the goal of keeping track of suggested songs easier for the users, users are limited to one song suggestion at a time. Users are also able to like or dislike the song suggestions of others, see Figure 1 1(h). Songs are represented as cards containing song/artist names, album covers, and like/dislike buttons. The cards disappear when liked/disliked and users can swipe to navi-

gate through the cards. Users also have access to several sub-tabs that display various information, such as a leaderboard, genre statistics, or song lyrics, see Figure 1 1(i). A sequence diagram describing the use of the application can be seen in Figure 4.

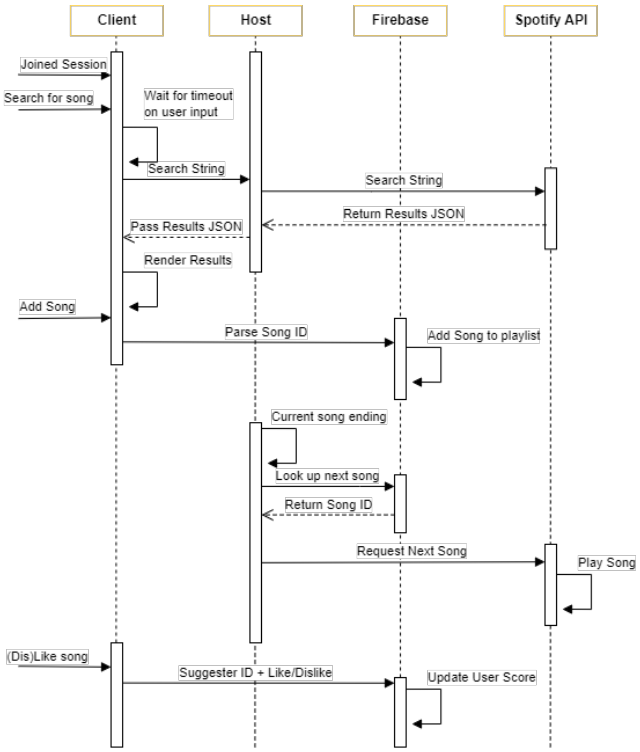

Figure 4: A Sequence diagram of key parts of the main program

4.2 USER INTERFACE (UI)

A key focus in the development of Crowd Control's user interface has been readability. The primary goal is to convey to users that they possess a significant degree of agency over the system and its playlist. This involves effectively communicating the interplay between the system's mechanisms and how they relate to the user. To accomplish this, we have introduced a rating system called the "DJ-score" within the UI. This numeric value reflects the user's standing and level of influence within the system. It is determined by the likes and dislikes received by songs suggested by the user. Highlighting changes to the DJ score is crucial in keeping the user informed about their evolving status and level of power within the system. Additionally, we have implemented a leaderboard feature that allows users to compare their DJ scores with those of other participants, see Figure 1(g). This leaderboard serves as a means for users to assess their relative position and influence within the session. Importantly, it also illustrates how the leaders on the leaderboard exert control over the playlist. This connection between the leaderboard and the music being listened to

underscores the impact and significance of a user's standing within the system. Furthermore, since the like and dislike function serves as the primary means for users to provide feedback to the system, it is essential that these actions are visually supported. Visual cues and indicators are incorporated into the UI to ensure that users can easily understand and perform these actions, reinforcing their ability to actively engage with the system.

Other than making the functionality of Crowd Control visible to users the UI should also be easily readable and usable for our users. In general, the UI design should adhere to Schneiderman's 8 golden rules [27]:

1. **Strive for consistency** - When designing the UI, we have sought to keep consistency throughout Crowd Control, using a simple color scheme and consistent element placement. Because Crowd Control is not too crowded with functionality keeping the interface consistent is relatively easy.

2. **Seek universal usability** - Developing interfaces with universal usability requires extensive user testing with diverse subjects, as such we have tried our best to keep the design universal usability, however much more can be done on this topic.

3. **Offer informative feedback** - Crowd Control mainly provides feedback to users through small animations such as the like button turning green when pressed, songs being emphasized when selected, and changes to DJ Score, etc.

4. **Design dialogues to yield closure** - Closure essentially means clear communication to the user, this is very visible in our primary concern of communicating the system's functionality to the users. We do not want users to be confused about how the system controls what songs are played or what the meaning of liking/disliking songs is. The inclusion of the leaderboard is based mainly on communication with other users on who has the most control over the system.

5. **Prevent errors** - Preventing errors is key to the usability of any application in Crowd Control it takes the form of preventing certain actions, such as adding the same song twice. Likewise providing confirmation dialogue on big actions such as leaving a session or removing a song suggestion to prevent these from happening accidentally.

6. **Permit easy reversal of actions** – Reversal of user actions relates largely to error prevention in Crowd Control. Users should be able to change their decisions at any time, changing their song suggestion or their likes/dislikes if they change their minds.

7. **Keep users in control** – The design of Crowd Control invariably contrasts with keeping users in control. While users should obviously feel in control of their own suggestions and likes, the music is controlled by no individual person. This relates directly to our research question of how users experience the loss of direct control of the output of a system.

8. **Reduce short-term memory load** – Reducing is primarily important in systems with complex interactions and functionality. As mentioned above Crowd Control is relatively simple in its implementation and does not provide many overly complex features. Additionally, the design resembles many similar applications drawing on users pre-existing knowledge.

4.3 FUNCTIONALITY CONSIDERATIONS

This section outlines some key functionality considerations we had in designing Crowd Control both technical and user experience related.

4.3.1 *API Calls*

Most APIs, including Spotify's, impose a limit on the rate of calls that can be made. If this limit is exceeded the API simply returns an error message rather than the requested data limiting the amount of data you have access to at any given time. When working with external API this is a key consideration as it can impact functionality and especially system scalability. If the system is structured in a way that allows users to, intentionally or not, make rapid API calls, it is likely that they at some point will be timed out from the API. Worse yet, they might lock the entire system for all users if API calls are handled from a central unit like the design of Crowd Control. We can limit the number of calls made to the API by first storing data in our system as it is received. For example, if a user wants to see the lyrics for the song currently playing an API call will be made to fetch those lyrics, these are then saved locally on the host unit. If a second user then wishes to do the same the lyrics are sent directly from the host, avoiding unnecessary API calls. By structuring the search functionality to only make calls once users are done inputting, we can also limit the amount of API calls made from searching. This can be done in different ways such as implementing a search button that must be pressed to send a search request, or by waiting until an appropriate amount of time has passed since the last user input before sending the request.

4.3.2 *Anonymity*

Providing the possibility of anonymity is important knowing who is participating can be fun for a tight-knit group of friends, but having to reveal your identity to people you might not know is daunting. Additionally, anonymity allows for separating the rating of song suggestions from the other social dynamics present in a group. In a similar vein, it is not possible to see the rating of individual songs, to prevent people from evaluating songs based on the preferences of others.

4.3.3 *Skipping Songs*

Whether or not users should be able to skip a suggested song if a majority agrees to it is a valid consideration. On one hand, if a majority feel that a song is not fitting into the current context, it would be wasted to keep playing it instead of moving on. On the other hand, having your song suggestion actively skipped can leave the suggester feeling discouraged [11], potentially disengaging from suggesting songs entirely. Instead, Crowd Control nudges users towards changing their suggestions based on feedback from others. If a user sees that their song is poorly received and unlikely to be played before the session ends, they might switch their suggestion out of their own volition.

4.4 DESIGN OF EVALUATION

In order to best mimic the intended use scenario of Crowd Control we will be conducting our experiments with large groups, ideally 15 or more. The larger the test size the more likely we are to have several diverging musical tastes, allowing us to better test our hypothesis. Additionally, we will conduct a pilot study with the same test size in order to ensure the functionality of the system. This will also allow us to evaluate the effectiveness of our UI as discussed in Section 4.2. We will then be able to test our hypothesis without system errors obscuring our results. The main user test will consist of participants being asked to join a session hosted on our device. Users will then be able to suggest and vote on songs as described earlier. Throughout the session, we will gather data related to user behavior, likes, dislikes, suggestions, etc. Lastly, users will be asked to fill out a questionnaire Section A.1 giving us qualitative data regarding the user experience of Crowd Control.

5 IMPLEMENTATION

This chapter presents the implementation details of the Crowd Control app. In this section, we will delve into the technical aspects of how the app was developed and provide an overview of its system architecture and system features.

5.1 SYSTEM ARCHITECTURE

The system architecture of Crowd Control follows a client-server model, where the server-side component handles the Spotify API including authentication, getting access tokens, and displaying the lyrics from the currently playing song. Meanwhile the client-side provides an intuitive user interface for interaction that connects the users to the Firebase database. This is primarily done through the "Dashboard" component which interacts with the main functionality of Crowd Control, such as choosing a song to play, viewing the song queue, and liking/disliking songs. The "Dashboard" component is divided into a lot of smaller components such as "LikeDislike", "Leaderboard", and "Level" components that each independently work in isolation. An overview of these components has been made in Figure 5.

As mentioned in Chapter 3 the web development stack of Crowd Control consists of four technologies:

- Firebase
- Express
- React
- Node.js

The system uses Firebase's database to store user information, song information, and session information. Express provides a server-side framework and is fast for building APIs. React with Bootstrap library handles the front-end user interface and Node.js serves as a runtime environment that allows us to write back-end code with additional libraries.

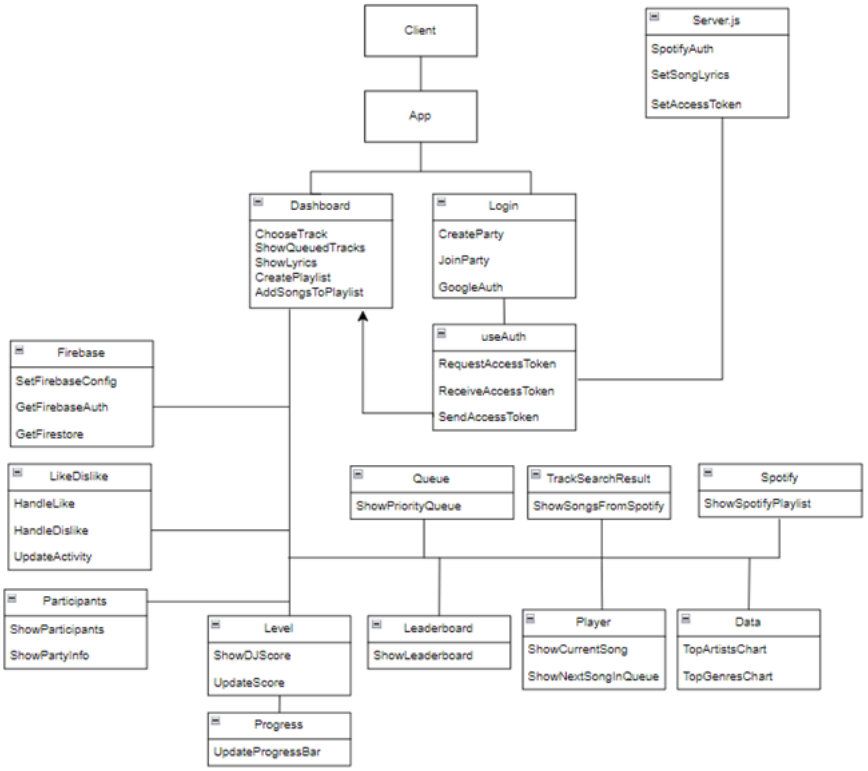

Figure 5: Notice that this overview is unlike a classic UML diagram, as React components have dynamic properties that can change state or props based on user interaction. The Dashboard component will only be shown when it has received the AccessToken from useAuth component.

5.1.1 *Server side*

As mentioned in Chapter 4, the implementation of our system was heavily dependent on a centralized server that would interact with the Spotify API so that each client would not need to login to Spotify at all. By implementing this approach, the clients can send search queries to the server, which acts as an intermediary between the clients and the Spotify API. The server processes the queries, communicates with the Spotify API to retrieve relevant song data, and returns the song suggestions to the clients. Although this structure was technically possible, we faced a few challenges with this. The first challenge was that this approach would require a robust server that could handle multiple requests at the same time from multiple clients. If the server experiences downtime or issues, it could disrupt the client's ability to interact with the Spotify API and thus we would have a single point of failure. Another challenge we didn't consider at the beginning was if this approach was a breach of Spotify's developer terms of service. One might argue that this approach is considered to be account sharing as the system gives users access to Spotify

Premium's functionality which users wouldn't be able to use otherwise. Although we could not find any clarification about this issue, it was still uncertain if this was the right approach. With this uncertainty in mind and adding an extra layer of complexity to the architecture, we decided not to continue with this approach after several failed attempts.

The approach we would end up implementing was the straightforward idea of the clients connecting individually to Spotify's API and then fetching that data themselves to the server. Although this required the clients to connect to their own Spotify account, it reduced the dependency on the server significantly. As searching for songs would not be done on the server side, the only thing we needed the server to handle was the authentication process of Spotify's API.

The server-side implementation for the Spotify API interaction is based on Node.js and is implemented as an Express application with Axios library which provides a convenient and efficient way to handle HTTP requests from a client to a server or an API. The interaction with Spotify API is done through the "spotify-web-api-node" library, which provides a simple interface for making API requests, handling authentication, and managing access tokens. The "Lyrics-finder" library fetches the song lyrics based on the artist and song name and returns the lyrics as string if found.

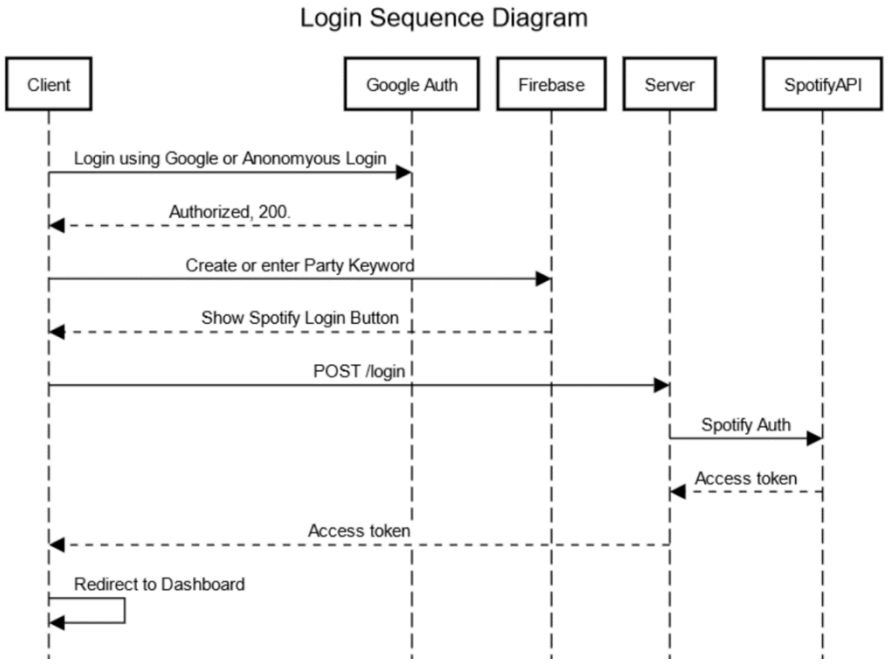

Figure 6: Sequence diagram illustrating the communication that occurs when the clients are logging into the system. It requires three steps to complete. 1. Google authentication. 2. Choose a party to join using Firebase. 3. Receive access token.

5.2 SYSTEM FEATURES

The main features of Crowd Control are user-driven music selection and as mentioned in Section 4.1 the system empowers users to actively participate in the music selection process, providing a sense of ownership and influence over the music.

5.2.1 *Client-side*

The client side of the application is built primarily using Firebase. The first possible interaction that users engage with when entering the website is either to create a party or join an existing party. This feature is done to improve security and scalability, to secure the party from intruders as well as have the ability to create multiple parties on different occasions. Once the user enters an existing keyword a Spotify Button will appear, and the user will get redirected to the dashboard once they have been authorized from the server with Spotify's API.

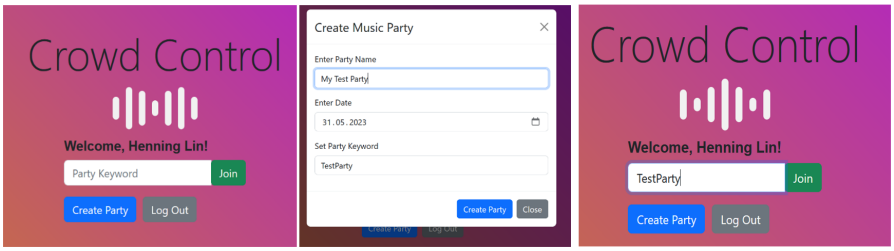

Figure 7: Screenshots of the login process. When creating a party, the host will need to set a party keyword that is shared. Users will need to know the party keyword in order to join the party session.

The given keyword is stored in localStorage, which the system will fetch in the dashboard component. By implementing it this way the dashboard receives all the up-to-date party information that has been stored in Firebase for that specific party, like how many participants there are, what songs have been requested, and a breakdown of the different genres and artists. See more in Figure 9.

5.2.2 *User Interface Design*

In the dashboard as seen in Figure 8, the user will be presented with the home page. It consists of the navigation bar at the top, a carousel/slideshow component that displays each queued song with a corresponding like and dislike button, the scoring area where the users can see the current DJ Level and score progression, and the player component at the bottom that displays the current playing song and what song is playing next. The styling for the UI compo-

nents such as the navigation bar, slideshow, and buttons are from Bootstrap CSS Library. The icons for each component are imported from FontAwesome.

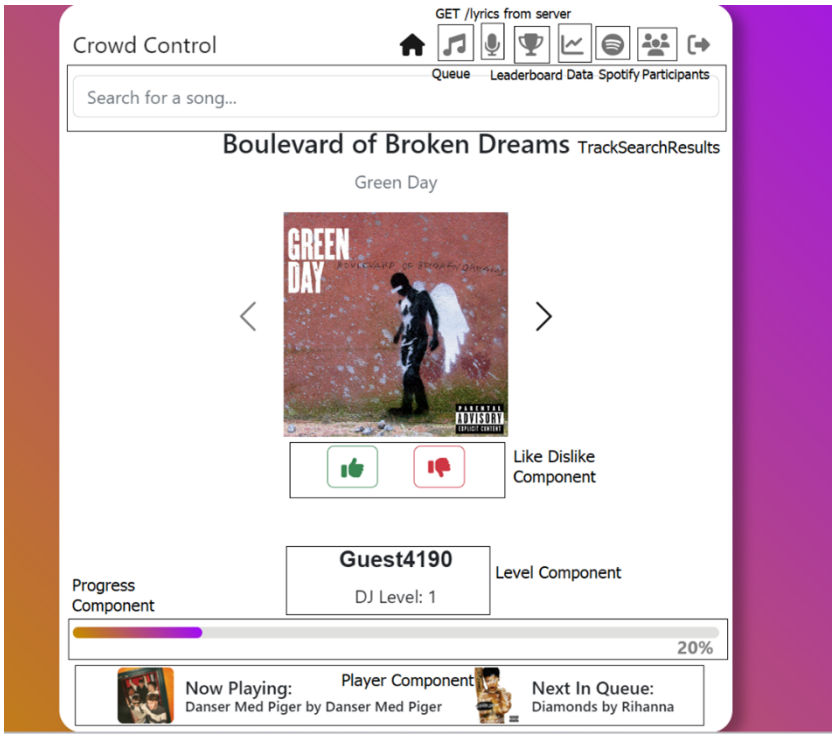

Figure 8: Dashboard component that is divided into smaller components each with its own functionality. See video in Section 1.1 for more.

5.2.3 *Database*

As shown in Figure 9 Firebase stores all the variables needed and they will be automatically updated when changed.

Once a party has been created, the party keyword and the user who initiated the party are stored. The user who creates the party becomes the party host and holds a special role. As the host, they have the privilege of having the Spotify Playlist created within their account. Additionally, the host is responsible for connecting their Spotify account to a central speaker and sharing the party keyword with other users.

In our system, each user starts with a DJ level of 1 and a DJ score of 0. When a user selects a song they would like to be played, other users have the option to express their preference by liking or disliking the chosen song. These likes and dislikes directly impact the DJ score of the user who made the selection. When a song receives a like, the DJ score of the user who chose the song increases by 1. Conversely, when a song receives a dislike, the DJ score of the user decreases by 1. As users accumulate DJ scores, they have the opportunity to progress to higher DJ levels. Once a user's DJ score exceeds 10, they

will advance to the next DJ level. However, it is important to note that users can lose a DJ level if their DJ score reaches zero and they receive an additional dislike, causing their score to become negative.

DJ level serves as a variable that determines the priority of songs. As users increase their DJ level, their song selections gain higher priority within the party's playlist. This means that users with higher DJ levels will have their chosen songs played earlier than lower DJ levels. In Figure 10 a sequence diagram is shown to visualize this.

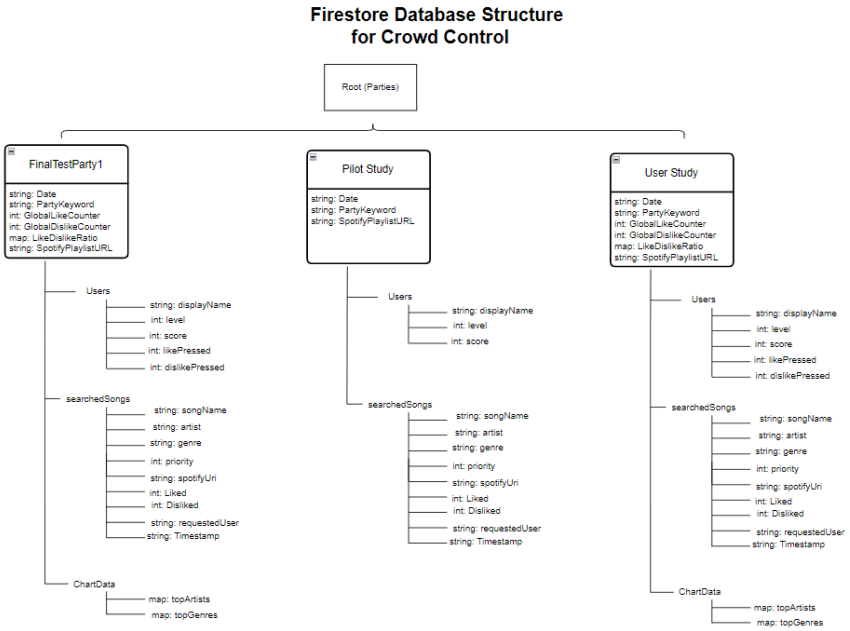

Figure 9: Overview of Firebase Database Structure

When a user makes a song request, the relevant details such as the song name, artist, genre, Spotify URI, priority, and the user who made the request are stored in Firebase. The name, artist, and genre variables are stored within the ChartData component, which enables the display of the top 5 artists and 5 genres currently popular in the session. The URI serves as the unique identifier required to add the requested song to Spotify. Finally, the user is stored to track the reception of each song, quantifying the number of likes and dislikes received, which is added to or subtracted from the user's DJ score.

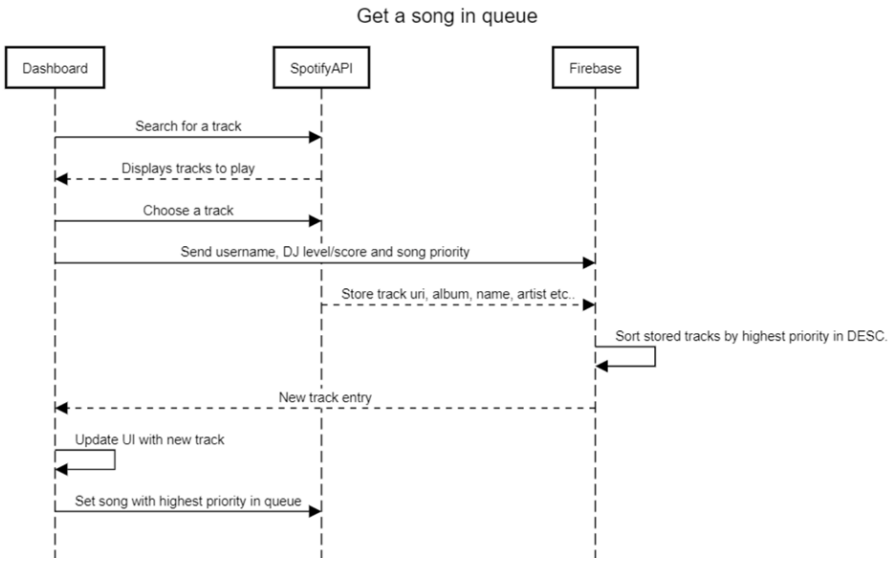

Figure 10: Sequence diagram showing the process of adding a new song and playing the highest-rated song

One may notice in Figure 9 that the Pilot Study has fewer variables compared to User Study. This is caused by a lack of data collection in the pilot study which we haven't originally thought about. This is further explained in Section 6.1.

6 EVALUATION

In this chapter, we present the evaluation conducted on the Crowd Control web app. The goal of this evaluation is to highlight relevant data and validate the hypothesis outlined Chapter 1.

6.1 METHODOLOGY

This section provides an overview of the methodology employed, which consisted of a pilot study and a user study. The primary objective of the pilot study was to stress test the system as well as gather information regarding system defects and potential architectural improvements for better data collection.

In both the pilot study and the user study, all participants were asked to fill out a GDPR document, indicating their consent to have their data stored. Subsequently, their Spotify email addresses were added to the Spotify developer kit, enabling them to access the Spotify API from their own accounts. At the end of the user study, participants were asked to fill out a questionnaire, the questionnaire can be found in Section A.1.

6.1.1 *Pilot study*

The pilot study consisted of 22 participants, who all got a brief introduction to the system. The 22 participants were all computer science students, and the test took place over a time span of around 1.5 hours. The study took place in a shared workspace where the music served as background music while the students worked on other tasks.

From our pilot study, we learned that the data we intended to gather was not sufficient to draw conclusive insights on their own, needing more data. The only data gathered was what songs users played and the DJ-score of each participant, this was still useful data for providing certain insights. However, it was not enough to distinguish what genre trends the songs had, and what general influence the system had on the users. Another issue we identified was that the scoring system did not work as intended and therefore the songs from all users were prioritized the same. Finally, the UI in the dashboard component featured a rotating carousel that displayed all songs in the queue. However, this required users to scroll through the carousel, containing every song suggested during the session, to find newly suggested songs, which was cumbersome and time-consuming.

Based on the findings in our pilot study we changed the system to store more data related to likes and dislikes from users, as well as the score each song got. We also updated the way the carousel worked, removing songs that had already been interacted with to make finding new song suggestions easier. Further improvements to the system include updating DJ-level and score visualization, making it more dynamic, while also making sure users can rank up and down. To give a better overview of the progression of data throughout a session, we implemented a snapshot feature, that timestamps users' activity, artists- and genre preferences every 5 minutes. Lastly, we implemented a global like-to-dislike ratio, providing data on the overall progression of user sentiment over the course of a session. These improvements allow us to keep track of how user interactions change over time and help us understand how they act within the system.

6.1.2 *User study*

With an updated version of the app, we carried out a user study involving 15 participants over a period of approximately 2 hours. The participants were computer science students that had not been a part of the pilot study. Similar to the pilot study, we maintained the setup where participants focused on a primary task while the music played in the background. Although we encouraged participants to actively engage with the system, we also acknowledged and valued passive usage as an important aspect of the study.

6.2 EVALUATION METRICS

For the evaluation of our crowd control system, we collected data in the form of system data, and with the use of a questionnaire, giving us both quantitative and qualitative data for our evaluation.

6.2.1 *Questionnaire*

Measuring user satisfaction was done through the use of a questionnaire. In the questionnaire, the users were asked to rank their experiences with the system on a Likert scale from 1 to 5, 1 being strongly disagree, and 5 being strongly agree. The questions included topics like their behavior, the fairness of the system, and their enjoyment and satisfaction with the system. The data from the questionnaire provided a lot of insights both in regard to the goals of the study, but also in enhancing some of the data points from the system data.

6.2.2 *Quantitative Data*

The system collected data from user interactions, namely, it recorded when a user liked and disliked a song. Likewise, each song request was recorded and connected to the user requesting it. Each song also collected the amount of likes and dislikes it received, along with its priority in the queue. For each song, the system also recorded the relevant genre tags for each artist pulled from Spotify. Each interaction was time-stamped, providing the tool to gauge interactions over time.

6.3 EVALUATION RESULTS

In this section, we will outline the preliminary results from the pilot and user studies.

6.3.1 *System Fairness*

In relation to system fairness, our objective was to examine user engagement and its correlation with rewards and punishments. The data obtained from the system revealed that when users experienced their song being heavily disliked, their inclination to engage with the system diminished. Specifically, among the five users with the lowest scores for their first song requests, four chose not to request another song. For the single remaining participant who did make additional requests, their participation ceased after encountering another song that received significant dislikes. Conversely, among the five users with the highest scores for their initial song requests, a total of 33 song requests were made throughout the testing period.

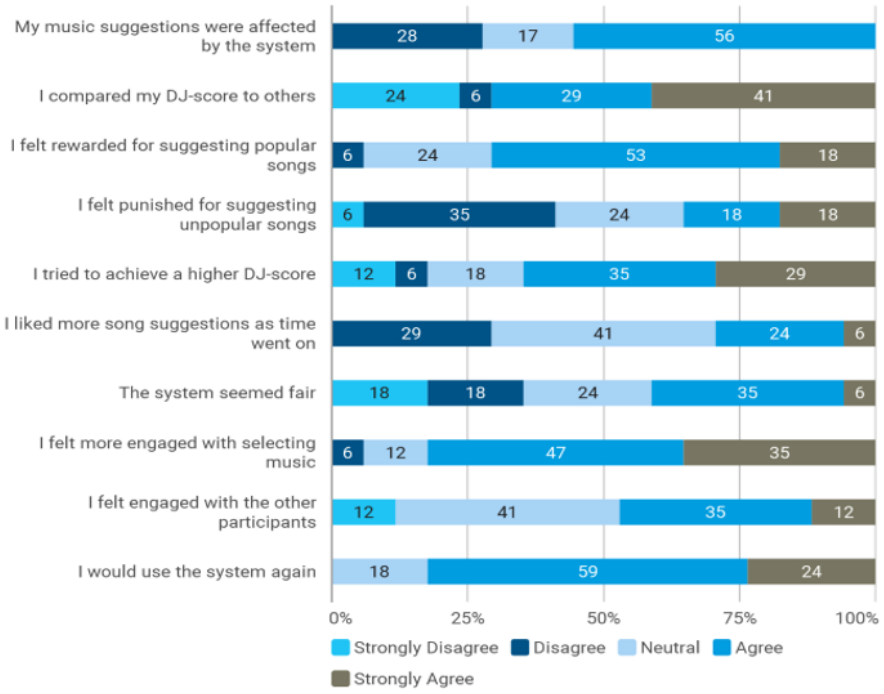

Figure 11: Answer distribution across the questionnaire

Looking at the questionnaire responses provided valuable insights into how users perceived the rewards and punishments within the system. The responses can be seen in Figure 11. Regarding reward, 53% of users rated their experience with a 4 on the Likert scale, indicating a sense of being rewarded by popular song suggestions. Additionally, 18% of users responded with a 5, expressing a stronger feeling of reward. On the topic of punishment, users were divided almost equally, with approximately half feeling punished and the other half not. Among those who felt punished, 50% provided a rating of 05, indicating a strong perception of being punished, while users who did not feel punished predominantly chose a more neutral rating of 2. Notably, a single user strongly believed they were not subject to punishment.

Regarding the fairness of the system, opinions were split among users. Approximately 50% considered the system to be unfair, with a notable proportion strongly perceiving it as such. In contrast, the remaining users overwhelmingly agreed with the fairness of the system, with a single user strongly advocating for its fairness.

In psychology, the use of doctor gray BIS/BAS scale is often used to measure and evaluate participants' responses to expected rewards and punishment[2]. BIS is an acronym for behavioral inhibition system and is a framework to measure motivations to avoid aversive outcomes, BAS is an acronym for behavioral activation system, with the goal of measuring motivation towards goal-oriented outcomes. Using this as a framework George, J. M[5] studied the effects of rewards

and punishment, and how they can affect "social loafing". George defines social loafing as users not partaking in the system, taking advantage of the users who do engage, therefore doing none of the work, but still getting some reward. The study concludes that feeling non-contingent punishment would lead to social loafing or disregard from the system, while contingent rewards would enforce more interaction and better performance. Supporting these findings, Sava and Sperneac [22] conducted a study of sensitivity to rewards and punishment, confirming both doctor gray's BIS/BAS scale, for the use of rating systems. In relation to our work, and data from our tests, it seems that these effects also play out in relation to collaborative playlists. The users that feel punished, either by being rated, but might also include other punishments like skipping songs, tend to disengage with the system. Conversely, when people feel rewarded, they would be more likely to engage with the system.

6.3.2 *User Engagement*

For each of the two studies, we stored and evaluated the interactions with the system. The data from the interactions between the two studies can be seen in figures 12, 13, 14, 15.

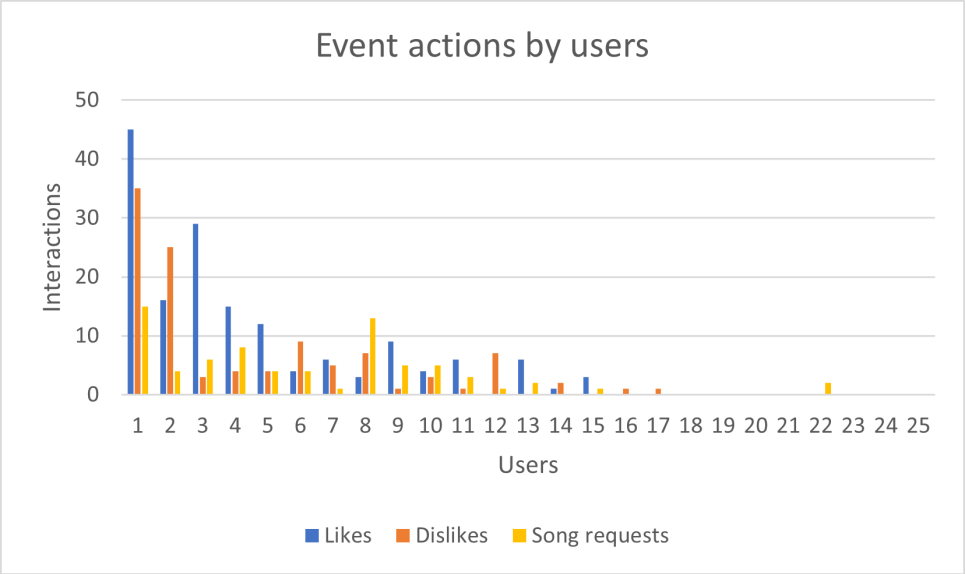

Figure 12: Bar chart showing the amount of different interaction types from the pilot study

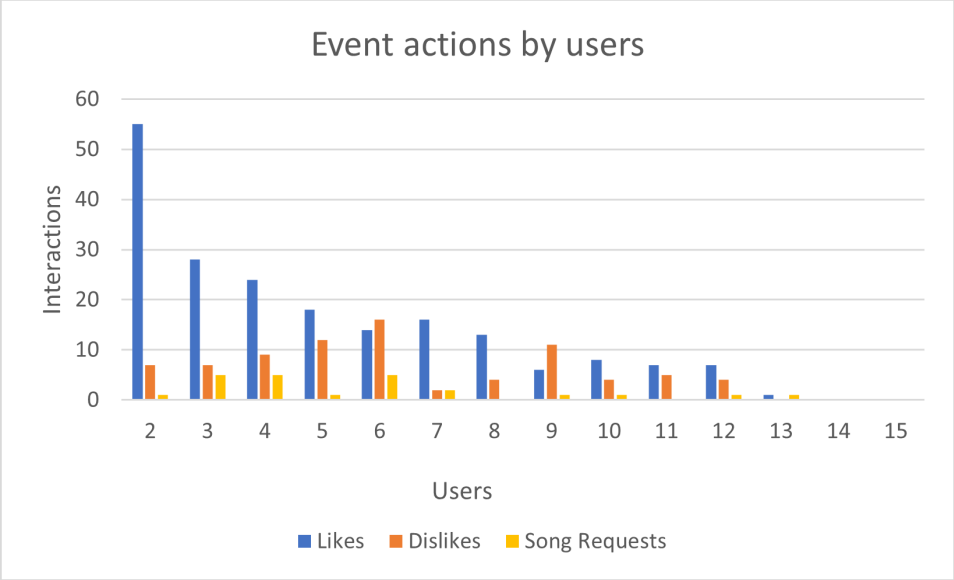

Figure 13: Bar chart showing the amount of different interaction types from the user study

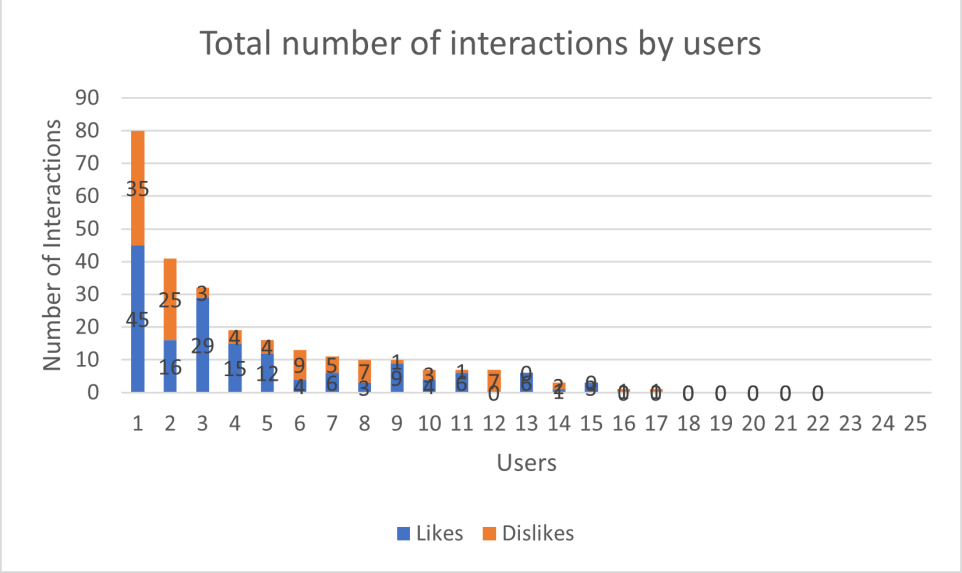

Figure 14: Bar chart showing the total interactions from each user id for the pilot study

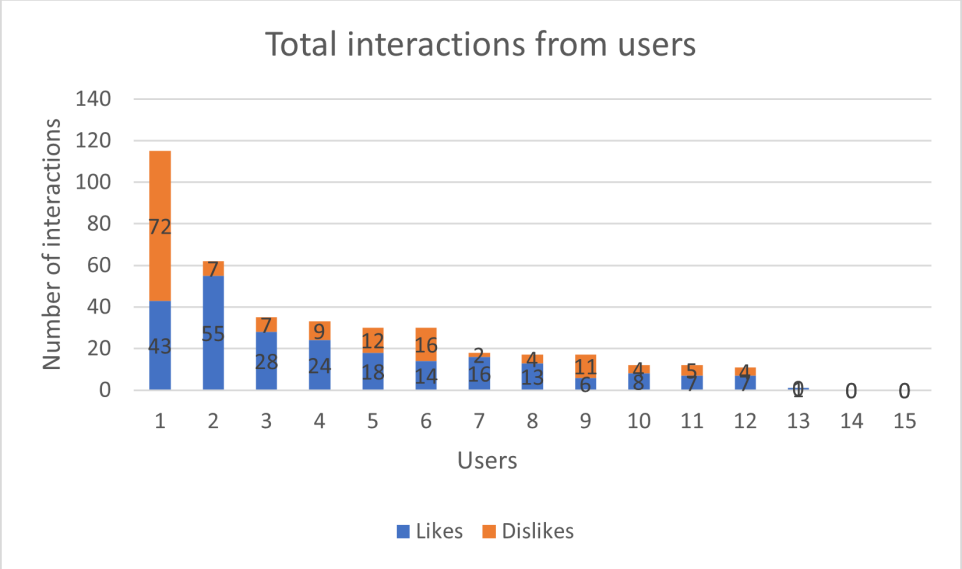

Figure 15: Bar chart showing the total interactions from each user id for the user study

From figures 12, 13, 14, 15 we can see each interaction broken down into different types of interactions. In their study, Morrison and Hayes [15] explored the website Reddit [1], with the intention of categorizing users into different roles. These roles included contributor, ignored, lurker, and casual commentator. The contributor role was categorized as the backbone of the website, containing the users who contribute most of the content. Contributors would also have a medium to high level of engagement with the system like upvoting and commenting. Contributors, who were often distinguished by their high amount of karma, were also known for their popularity and active participation in the sub-forums they engaged with. High karma scores indicated their valuable contributions and established a positive reputation within the community. The ignored users included mainly spam and reposts users, as such, this is not relevant to our system. The lurker role has very low both engagement and reciprocity, meaning they would not really engage with the system. Lastly, the casual commentators, who have a medium level of engagement, but would often have a high reciprocity, meaning they would comment a lot on a few topics. The data set from our tests likewise shows that our system contains a few contributors, who requested a lot of songs and played a very active part in the session. In the pilot study the users who requested the most songs, namely 15 and 13 songs were way above the average of 3,3 songs for that session. Likewise in the user study, 3 users would request 5 songs each, well above the average of 1,66 songs. The data also highlights how most users were of the casual commentator role. We can see that users that request around the average amount of songs, often have more engagement with the rating

1 https://www.reddit.com/

aspect of the system, rating more songs than they suggest. Most of the casual commentator users however did not engage as much with the rating system as the contributors, as the casual commentators mainly have between 10-20 interactions, with the contributors having up to 80 interactions in total for a single user. Likewise, the test also confirmed that the system has lurkers, people who do not engage with the system aside from either liking or disliking a few songs or requesting a single song.

6.3.3 *Influencing Song Requests*

Our hypothesis 1 assumes that the system has an impact on users' song choices, shaping a general trend or genre among the users in a session. However, responses from the questionnaire (see figure Figure 11) and our gathered data, suggest this is not the case. Users only indicate a mild agreement in the questionnaire that the system influences their song choice, while our data shows that users mostly stick to their own preferences without paying much attention to other users' suggestions.

Determining whether or not user suggestions change over time based on data alone will first require more data. Secondly, it would most likely require a larger variety of data. Spotify's API does not provide metadata that is attached to a specific song, rather all song data is based on the attached artist. This works to give a rough overview of genre preferences. However, a lot of the finer details are lost, as songs can vary wildly in content even if they are made by the same artist. It is possible that users' personal preferences and subjective interpretations of genres are not fully captured through this method.

We also suspect that the impact of DJ-score, and by extension other people liking your song suggestions, was unclear to some participants. This mainly stems from a lack of clear communication through the UI. Short-term this could be rectified by providing a more in-depth explanation to participants. For future studies, updating the UI to even more clearly visualize the impact of rating on song selection might change our results. We feel that the quantity of data we collected is not sufficient to completely enlighten this topic, demanding further studies and exploration with different methods.

6.3.4 *User Satisfaction*

In Section 1.2, we outlined hypothesis 2, in relation to the overall user satisfaction of the system.

For each session, we recorded the overall progression of the like-to-dislike ratio. From this, we wanted to see if the overall satisfaction of the users increased, decreased, or stayed around the same. Each session started out with a hard-coded 50% ratio, and as the sessions

progressed, the system updated this ratio based on user actions. The results of the ratio can be seen in Figure 16.

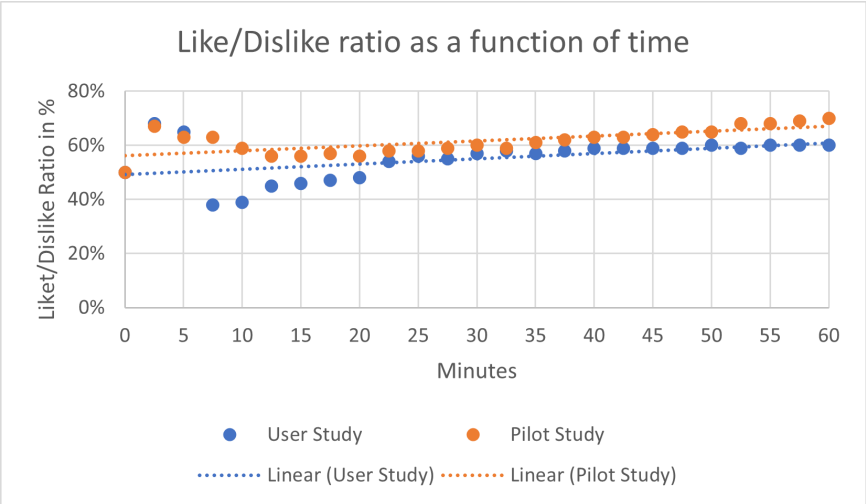

Figure 16: Graph displaying the like-to-dislike ratio over time for both pilot and user study

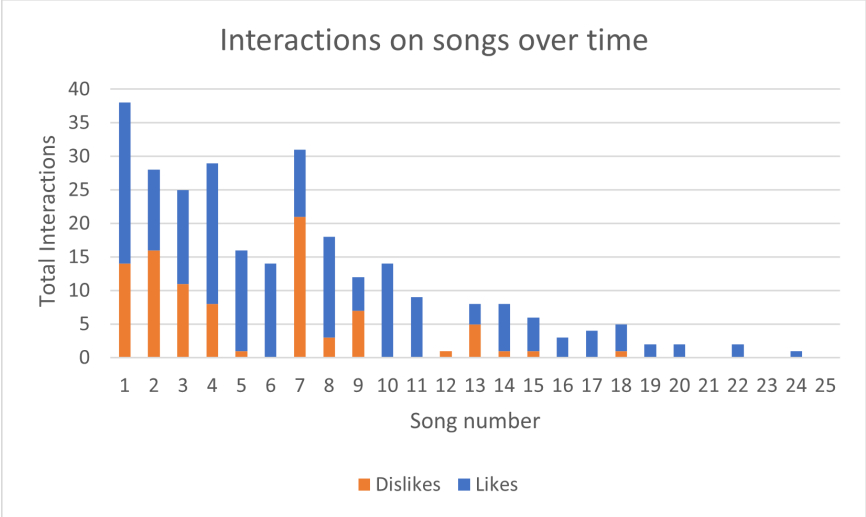

Figure 17: Graph over total interactions on songs over time. Showing both likes and dislikes

As seen in the figure, both sessions started out with a big spike in the ratio, as the first couple of songs suggested, followed by a fall, with one session falling below 50%. As the session went on, we see the like ratio steadily increase, and a trend line shows it to be rising. Results from the questionnaire (See figure Figure 11), show that on topics of user satisfaction, most users felt more engaged with the music selection process. Of the 17 participants in the questionnaire, 35% said they strongly agree feeling engaged, while 47% said they agree. On the topic of engagement between users, the questionnaire leans

towards users feeling more engaged, with about 50% saying they felt more engaged, while only 12% said they did not feel more engaged. The last question on the questionnaire was if the users would use the system again, with an overwhelming 82% agreeing and strongly agreeing they would use the system again. From the highlighted data, as well as spoken feedback from users and others who have been shown the system, the overall feedback seems to suggest that users like the concept. One thing the spoken feedback seems to suggest is that the idea of democratizing the playlist, and tools through voting to reward good behavior and punish bad behavior resonated with the users. The data recorded from the tests are only a preliminary exploration. Much more data is required to make a definite conclusion on whether weighted voting in collaborative playlists provides a better user experience. However, the data indicates that users are at least excited by the concept.

Another indication of increased user satisfaction can be seen in figure Figure 17. In this figure, we can see the amount of interaction on a given song, as a function of when in the session it was requested. We can see a clear downward trend in the number of interactions, meaning users stopped engaging with the system, especially the rating system, after a while. It is noteworthy that the number of dislikes, with the exception of a few spikes, fell off a lot faster than the amount of likes a song would get. This may indicate that as the session progresses, user suggestions move towards collectively more liked songs. This could indicate that most users were satisfied with the overall music selection and the end of the session.

Likewise, as mentioned in Section 1.2, our aim with hypothesis 3 is to promote user-to-user interactions to enhance the music experience through social dynamics. With this in mind, one might raise the question of whether this decrease in interaction supports the desired social dynamic or hinders user engagement. The decline in interactions may indicate a positive outcome, suggesting that users naturally gravitate towards collectively liked songs as the session progresses. In this case, the system may successfully align users' preferences and foster a cohesive music experience. However, it is also possible that the decline reflects a challenge in sustaining user interest and engagement with the system, particularly if users find it difficult to actively interact over time.

To better understand the underlying factors contributing to the decrease in interactions, further analysis, user feedback, and additional studies may be necessary. This would provide insights into whether the decline is intentional, supporting the desired social dynamic, or if adjustments are needed to enhance user engagement and maintain a vibrant music experience.

7 CONCLUSION

This thesis presented the development and evaluation of the Crowd Control web application, designed to enhance user engagement and enjoyment in social collaborative music systems.

Our evaluation suggests that the application successfully increased user engagement by enabling active participation and song requests, granting users a sense of control over the playlist. Users also reported greater engagement with music selection and increased engagement with other users. The implementation of weighted voting seemed to contribute to a more enjoyable music experience overall, Our data shows rising user satisfaction as a session went on, supported by the sentiment in the questionnaire. Findings also indicate that usage of the application did not lead to a convergence of genre choices among users as initially hypothesized.

By definition, a weighted voting system is not fair. Certain users are prioritized over others, which can lead users to feel alienated from the decision-making process. As mentioned in Section 6.3.1 users tend to disengage with the system if they feel punished. This might explain our observations that user participation stagnates over longer periods of time. There are several ways the voting system could be tweaked to feel fairer and less punishing to all users, we will discuss this further in Section 7.1. Weighted voting shows the potential for improving user satisfaction with group decisions. However, it is worth considering whether this type of voting is well-suited for a social setting where the goal is to have an enjoyable experience with others. If overall music satisfaction increases but leaves a large number of users feeling frustrated or neglected, the tradeoff might not be worthwhile.

7.1 LIMITATIONS AND FUTURE WORK

It is important to acknowledge the limitations of this study, such as the small sample size and lack of a controlled environment, which may have influenced the outcomes and generalizability of the findings. Future research should focus on conducting larger-scale studies in replicable environments to further validate and refine the system's effectiveness. In the next few sections, we will highlight further limitations of the system and also highlight potential future work, that solves these limitations.

7.1.1 *Logic and Fairness*

The algorithms and logic employed within the system can be further refined and optimized. This includes improving the way we prioritize songs in the queue and improving the DJ scoring and leveling system to feel fairer for all users. It is important to keep in mind the disparity in work and benefits for all users of the system. In relation to this, one challenge that needs to be addressed is the potential imbalance caused by top-ranked users having the ability to constantly set new songs at the top of the queue. This results in other users having to wait excessively long periods to hear their suggested songs. To mitigate this issue, several approaches can be considered.

One possible solution is to introduce an aging variable that increases over time and is correlated with the priority assigned to songs. By incorporating this variable, older suggestions gradually increase in priority, allowing older suggestions from other users to have a chance of being played. This aging mechanism can help balance the selection of songs and prevent a small subset of users from dominating the playlist.

Another approach is to limit the frequency of song suggestions that users can make. By implementing a cooldown period between consecutive suggestions, users are encouraged to make thoughtful and selective choices rather than constantly flooding the queue with their preferred songs. By refining the algorithms used to prioritize songs in the queue and incorporating measures such as an aging variable or cooldown periods for suggestions, the Crowd Control system can achieve a more balanced and inclusive music-playing experience. These enhancements will promote fairness and ensure that all users have a chance to enjoy their suggested songs without excessive wait times or dominance from a select few.

Another challenge involves the lack of influence of users who join the system later or use it at a later stage. These users encounter difficulties in getting their desired songs played as they start at DJ level 1, while those who have been using the system for a longer duration already have higher priority. Addressing this issue and incorporating a feature to accommodate later participants is an important aspect to consider in future iterations of the system.

7.1.2 *Social Features and Gamification*

Expanding the social features of the Crowd Control system can significantly enhance the interactive and engaging nature of the user experience. By introducing additional elements that encourage user participation and interaction, the system can foster a stronger sense of community and enjoyment. One potential feature to consider is incorporating a gesture-based interaction, where users can shake their

mobile devices to express their liking for a song. This interactive gesture can serve as a fun and intuitive way for users to show their appreciation and actively engage with the music system.

In addition to gesture-based interactions, integrating mini-games or quizzes into the system could further enhance user engagement. These interactive activities can be designed to promote entertaining interaction with the system and between users. Users who actively participate and perform well in these mini-games or quizzes can be rewarded with increased DJ scores, creating an incentive for users to engage and have fun with the system. To promote a sense of achievement and friendly competition, the implementation of user profiles, badges, and achievements can be beneficial. User profiles allow individuals to showcase their preferences, music tastes, and DJ scores. Badges and achievements serve as virtual rewards for reaching milestones or accomplishing specific tasks within the system. This not only adds a competitive element but also encourages users to explore different aspects of the system and strive for recognition within the user community.

7.1.3 *Data collection and Analysis*

The current implementation of data collection within the Crowd Control system has several limitations and areas for future improvement. While the system collects relevant data on user interactions and song preferences, there is room for improvement in terms of the scope and depth of data collected. Additionally, the system can be enhanced by providing users with more comprehensive and interactive data visualizations. Currently, the Data component offers basic charts and visual representations. Future improvements should consider incorporating more advanced visualization techniques, allowing users to explore and analyze data in a more intuitive and informative manner. This will enable users to gain deeper insights into popular songs, trends, and user engagement.

7.1.4 *Testing in other social contexts*

While our evaluation primarily focused on testing the system in the background of the student's activities, it is important to acknowledge that this setup may not fully capture the real-world usage scenarios of the system. In future work, we aim to expand the testing methodology to include diverse social contexts, such as parties or other social gatherings, and thereby obtain a more comprehensive understanding of how the system performs and is perceived in other social contexts. By exploring these contexts, we can gain further insights into how the system compares to other traditional setups where a single person controls the music from their phone, allowing us to assess the

advantages and disadvantages of the Crowd Control system in comparison to existing systems.

By addressing these limitations and pursuing future work, the Crowd Control system can evolve into a more inclusive, engaging, and enjoyable platform, fostering a sense of community and user satisfaction in social collaborative music systems.

Part I

APPENDIX

A

APPENDIX

A.1 QUESTIONAIRE

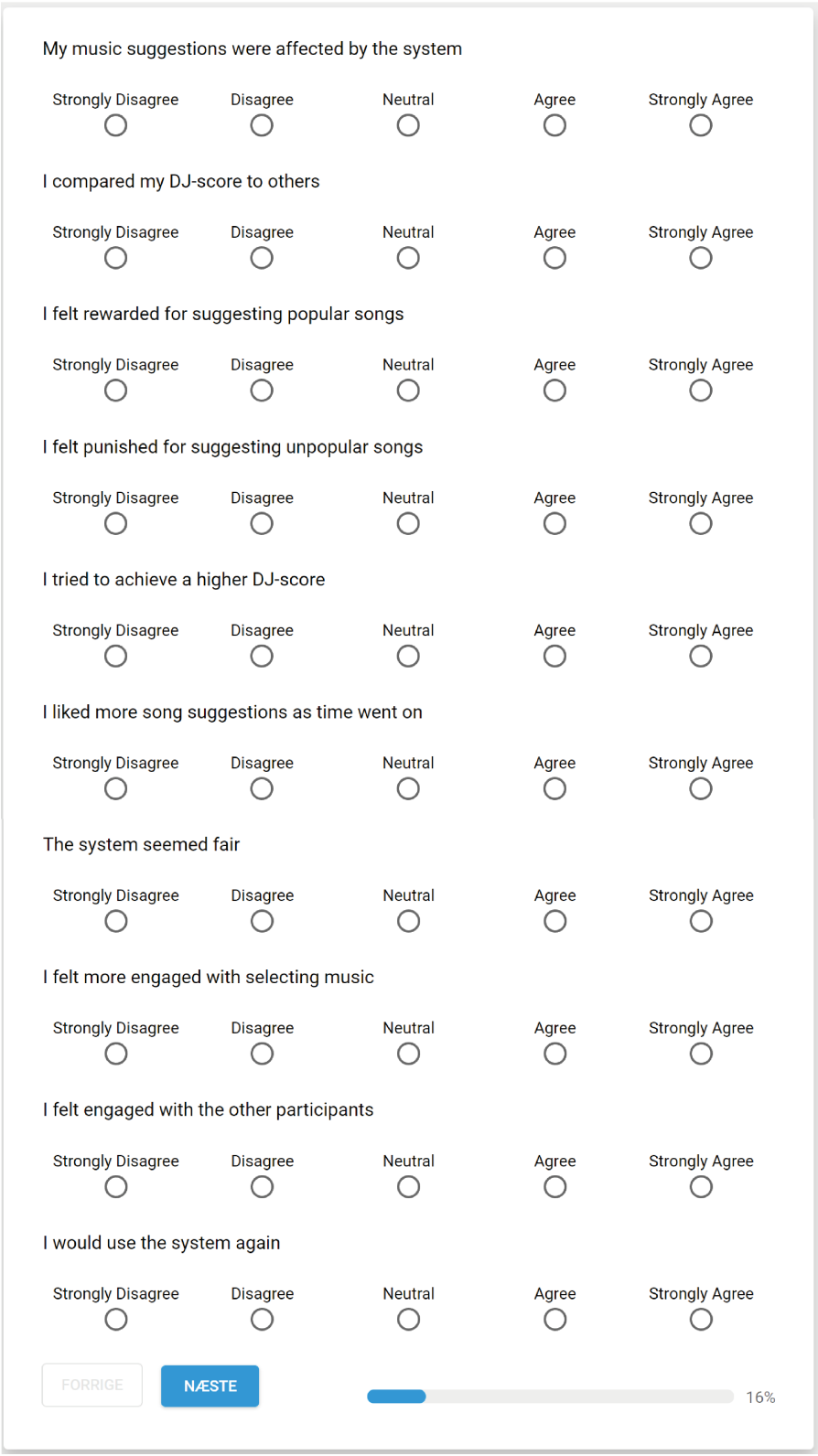

Figure 18: The questions in the survey provided to users after the test

A.2 GITLAB

https://gitlab.au.dk/au667259/bachelor-music

BIBLIOGRAPHY

[1] Jared S. Bauer, Aubury L. Jellenek, and Julie A. Kientz. "Reflektor: An Exploration of Collaborative Music Playlist Creation for Social Context." In: *Proceedings of the 2018 ACM International Conference on Supporting Group Work*. GROUP '18. Sanibel Island, Florida, USA: Association for Computing Machinery, 2018, 27–38. ISBN: 9781450355629. DOI: 10.1145/3148330.3148331. URL: https://doi.org/10.1145/3148330.3148331.

[2] Charles Carver and Teri White. "Behavioral Inhibition, Behavioral Activation, and Affective Responses to Impending Reward and Punishment: The BIS/BAS Scales." In: *Journal of Personality and Social Psychology* 67 (Aug. 1994), pp. 319–333. DOI: 10.1037/0022-3514.67.2.319.

[3] Zhaofei Chen, Emre A. Yavuz, and Gunnar Karlsson. "What a juke! A collaborative music sharing system." In: *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2012, pp. 1–6. DOI: 10.1109/WoWMoM.2012.6263751.

[4] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. "Flytrap: Intelligent Group Music Recommendation." In: *Proceedings of the 7th International Conference on Intelligent User Interfaces*. IUI '02. San Francisco, California, USA: Association for Computing Machinery, 2002, 184–185. ISBN: 1581134592. DOI: 10.1145/502716.502748. URL: https://doi.org/10.1145/502716.502748.

[5] Jennifer George. "Asymmetrical Effects of Rewards and Punishments: The Case of Social Loafing." In: *Journal of Occupational and Organizational Psychology* 68 (Aug. 2011), pp. 327 –338. DOI: 10.1111/j.2044-8325.1995.tb00591.x.

[6] Gst Giri and Agus Harjoko. "Music Recommendation System Based on Context Using Case-Based Reasoning and Self Organizing Map." In: *Indonesian Journal of Electrical Engineering and Computer Science* 4 (Nov. 2016), p. 459. DOI: 10.11591/ijeecs.v4.i2.pp459-464.

[7] Eds. H. Masum M. Tovey. *The Reputation Society: How Online Opinions Are Reshaping the Offline World*.

[8] Marius Kaminskas and Francesco Ricci. "Location-Adapted Music Recommendation Using Tags." In: Jan. 2011, pp. 183–194. ISBN: 978-3-642-22361-7. DOI: 10.1007/978-3-642-22362-4_16.

[9] Gjorgji Kjosev. "Weighted collaborative rating system for the interactive Web." In: (2010).

[10] Hannu Kukka, Rodolfo Patino, and Timo Ojala. "UbiRockMachine: A Multimodal Music Voting Service for Shared Urban Spaces." In: *Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia*. MUM '09. Cambridge, United Kingdom: Association for Computing Machinery, 2009. ISBN: 9781605588469. DOI: 10.1145/1658550.1658559. URL: https://doi.org/10.1145/1658550.1658559.

[11] Eva Lenz, Sarah Diefenbach, Marc Hassenzahl, and Sébastien Lienhard. "Mo. Shared music, shared moment." In: Oct. 2012, pp. 736–741. ISBN: 9781450314824. DOI: 10.1145/2399016.2399129.

[12] Ning-Han Liu and Shu Hsieh. "Intelligent Music Playlist Recommendation Based on User Daily Behavior and Music Content." In: vol. 5879. Dec. 2009, pp. 671–683. ISBN: 978-3-642-10466-4. DOI: 10.1007/978-3-642-10467-1_59.

[13] Joseph F. McCarthy and Theodore D. Anagnost. "MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts." In: *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*. CSCW '98. Seattle, Washington, USA: Association for Computing Machinery, 1998, 363–372. ISBN: 1581130090. DOI: 10.1145/289444.289511. URL: https://doi.org/10.1145/289444.289511.

[14] Vlad Mirea. "Queue: A Mobile Application for Collaborative Music Playlists." In: *Williams Honors College, Honors Research Projects. 1009*. (2019). URL: https://ideaexchange.uakron.edu/honors_research_projects/1009.

[15] Donn Morrison and Conor Hayes. "Here, have an upvote: communication behaviour and karma on Reddit." In: *GI-Jahrestagung*. 2013.

[16] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. "Jukola: Democratic Music Choice in a Public Space." In: *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. DIS '04. Cambridge, MA, USA: Association for Computing Machinery, 2004, 145–154. ISBN: 1581137877. DOI: 10.1145/1013115.1013136. URL: https://doi.org/10.1145/1013115.1013136.

[17] So Yeon Park and Blair Kaneshiro. "An Analysis of User Behavior in Co-Curation of Music Through Collaborative Playlists." In: *Extended abstracts for the Late-Breaking Demo Session of the 18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017*. 2017.

[18] So Yeon Park and Blair Kaneshiro. "Social Music Curation That Works: Insights from Successful Collaborative Playlists." In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW1 (2021). DOI: 10.1145/3449191. URL: https://doi.org/10.1145/3449191.

[19] So Yeon Park, Audrey Laplante, Jin Ha Lee, and Blair Kaneshiro. "Tunes Together: Perception and Experience of Collaborative Playlists." In: *International Society for Music Information Retrieval Conference*. 2019.

[20] So Yeon Park and Sang Won Lee. "Lost in Co-Curation: Uncomfortable Interactions and the Role of Communication in Collaborative Music Playlists." In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW1 (2021). DOI: 10.1145/3449137. URL: https://doi.org/10.1145/3449137.

[21] So Yeon Park, Emily Redmond, Jonathan Berger, and Blair Kaneshiro. "Hitting Pause: How User Perceptions of Collaborative Playlists Evolved in the United States During the COVID-19 Pandemic." In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: 10.1145/3491102.3517604. URL: https://doi.org/10.1145/3491102.3517604.

[22] Florin Sava and Ana-Maria Sperneac. "Sensitivity to Reward and Sensitivity to Punishment rating scales: a validation study on the Romanian population." In: *Personality and Individual Differences* 41 (Dec. 2006), pp. 1445–1456. DOI: 10.1016/j.paid.2006.04.024.

[23] Markus Schedl. "Ameliorating Music Recommendation: Integrating Music Content, Music Context, and User Context for Improved Music Retrieval and Recommendation." In: *Proceedings of International Conference on Advances in Mobile Computing amp; Multimedia*. MoMM '13. Vienna, Austria: Association for Computing Machinery, 2013, 3–9. ISBN: 9781450321068. DOI: 10.1145/2536853.2536856. URL: https://doi.org/10.1145/2536853.2536856.

[24] Markus Schedl, Arthur Flexer, and Julián Urbano. "The Neglected User in Music Information Retrieval Research." In: *Journal of Intelligent Information Systems* 41 (2013), 523–539. DOI: 10.1007/s10844-013-0247-6.

[25] Markus Schedl and Peter Knees. "Personalization in Multimodal Music Retrieval." In: *Adaptive Multimedia Retrieval*. 2011.

[26] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen. "There is more to context than location." In: *Comput. Graph.* 23 (1999), pp. 893–901.

[27] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 3rd. USA: Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0201694972.

[28] U. Wolz, M. Massimi, and E. Tarn. "r-MUSIC, a collaborative music DJ for ad hoc networks." In: *Proceedings of the Fourth International Conference onWeb Delivering of Music, 2004. EDELMUSIC 2004.* 2004, pp. 144–150. DOI: 10.1109/WDM.2004.1358111.