JamNSync: A User-Friendly, Latency-Agnostic Virtual Rehearsal Platform for Music Ensembles

by

Nanette Wu

B.S. Computer Science and Engineering and in Music
Massachusetts Institute of Technology, 2020

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . .
Department of Electrical Engineering and Computer Science
May 27, 2021

Certified by . . . .
Eran Egozy
Professor of the Practice, Music Technology
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

JamNSync: A User-Friendly, Latency-Agnostic Virtual Rehearsal Platform for Music Ensembles

by

Nanette Wu

Submitted to the Department of Electrical Engineering and Computer Science
on May 27, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Existing technologies that support virtual rehearsals are complex and unintuitive. Network music performance platforms promise real-time interactions between physically separated musicians, but they demand hard-to-achieve network conditions, lack rehearsal-specific features, and require extensive knowledge of network jargon. JamNSync, designed with musicians in mind, proposes a hassle-free alternative. The JamNSync web application offers a recording-based, synchronous rehearsal experience using a novel consensus protocol. Each musician only needs a basic understanding of playback systems and common audio devices. To account for non-deterministic audio latency in the browser, JamNSync provides a user-friendly audio alignment tool that efficiently automates audio processing and produces a group mix ready for immediate feedback. Three rounds of user testing show that JamNSync holds significant advantages over current virtual rehearsal solutions. By providing an intuitive platform for physically distanced groups to rehearse, JamNSync enables musicians around the world to remotely make music together. This is especially valuable during times of social isolation.

Thesis Supervisor: Eran Egozy
Title: Professor of the Practice, Music Technology

Acknowledgments

First and foremost, I would like to express immense gratitude to Eran Egozy, who has been a mentor in every aspect of my college experience. Under his unparalleled guidance, I've taken on more projects that I could have ever imagined, from annotating Lincolnshire Posy for NoteStream, to developing guitar-learning software for IMS, and even traveling to Miami to run ConcertCue with the New World Symphony. Seeing Eran's passion for coding and music has taught me to not only care immensely about what I do, but also have high expectations to work hard to exceed. And that nothing is impossible (like a 115-page thesis written in one month), give or take some sleep. There are so few people who share our interest in music technology, clarinet playing, and stuffed animal Zoom show-and-tells, and I am extremely fortunate to have been mentored by Eran throughout my five years at MIT.

To my friends at the Music Technology Lab, Julia Fiksinski, Madi Wong, Crystal Wang, and Emily Hu: at the end of a stressful week, I knew I could count on memes, sleep-deprived conversations, and shared commiseration over classes to make everything feel better.

To the music faculty at MIT, Evan Ziporyn, Natalie Lin Douglas, Eileen Huang, Kristian Baverstam: thank you for being extremely supportive of my growth in music, especially during the virtual pandemic 2020-21 school year. My world view of music has changed so much during college. If it weren't for how much I enjoyed the eclectic and fruitful MIT music experience, I wouldn't have wanted to commit an extra year to write this thesis in music technology.

To my CMS group, Thomas Xiong (the stats and Z-Center king), Sylvia Biscoveanu, and Amy Jin: thank you for playing Messiaen with me this year and helping me user test time after time. Not only did you guys remind me to sleep, but you guys also boosted my mood every time we got to rehearse (in-person!) together.

Finally, to my amazing friends, An Jimenez, Priscilla Wu, Emily Zhang, and Diana Flores, and the best mom in the world, Gangzhu Chen: thank you for being immensely supportive in every step of this thesis. You guys are the best! (>^^)>

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Ensemble music-making is deeply rooted in the empathy and humility of collaboration. Working together with a small group to perform classical music, in particular, is an intimate experience that fosters a sense of community among both musicians and audiences.

However, the recent COVID-19 pandemic abruptly severed connections within these communities. While many working environments like schools and offices quickly adapted to remote forms of communication, music groups struggled to find their place in the virtual world. What once was a straightforward procedure to coordinate a rehearsal became more of a logistical nightmare, wrangling uncooperative technology to make recordings online and fighting bureaucratic processes to reserve spaces. Even if groups found ways to get together in person, the act of rehearsing became a risky activity that required extensive safety measures, especially for musicians that utilized air in their performance (e.g. wind instrumentalists, vocalists).

Without accessible options to rehearse and perform, opportunities to play music dwindled. However, at the start of the pandemic, there was an urgent need to devise alternative methods to rehearse despite physical isolation. As such, people started investigating tools that would enable virtual music collaboration.

In the early 2000s, network music performance (NMP) was developed to allow a group of physically isolated musicians to interact over a network and perform as if they were in the same room [23]. While the concept sounded nice in theory, applications of network-based musical interaction never made big breakthroughs in practice. Due to the time-sensitive nature of ensemble playing, it was difficult to design rehearsal platforms that were both efficient and user-friendly. Many intrinsic characteristics of the Internet, like best-effort delivery, were well-suited for most use cases of the web, but not for low-latency audio streaming [10].

Initial NMP implementations attempted to work around limitations in network infrastructure. Some solutions sacrificed audio quality to reduce data transfer latency. Others removed the visual component of music performances altogether to preserve network bandwidth [17]. However, these compromises resulted in partial solutions on platforms that lacked usability for musicians, who are less familiar with technical jargon. People who actually tried these NMP solutions ended up devoting much of their rehearsal time to wrestling uncooperative technology and tolerating unintuitive methods of rehearsing. Figure 1-1 shows one such example [11].

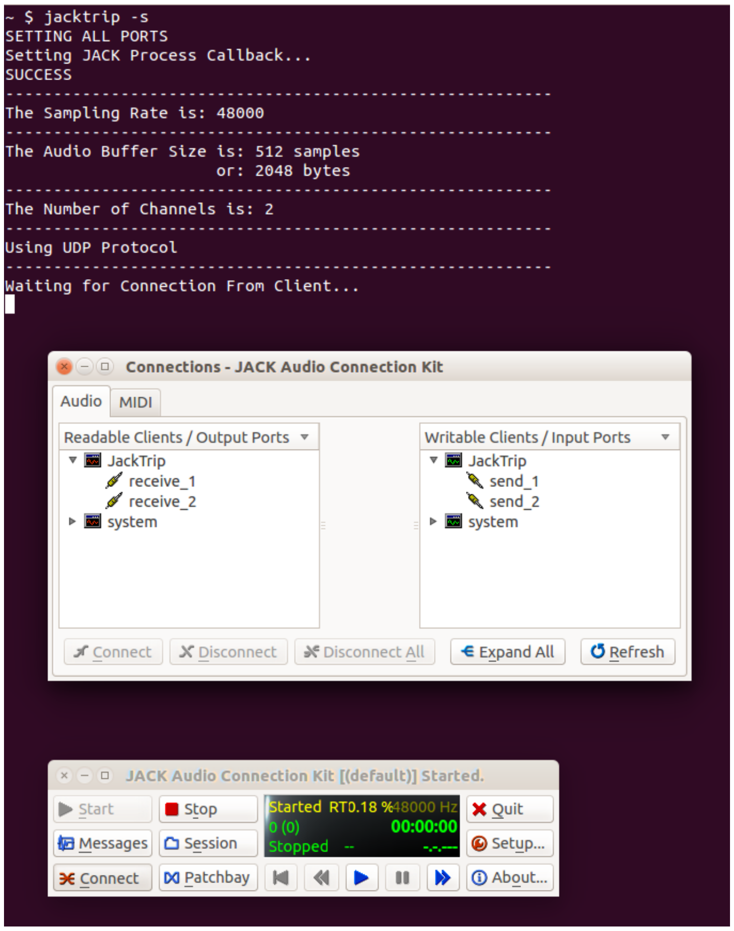

```
~ $ jacktrip -s
SETTING ALL PORTS
Setting JACK Process Callback...
SUCCESS
--------------------------------------------------------
The Sampling Rate is: 48000
--------------------------------------------------------
The Audio Buffer Size is: 512 samples
                      or: 2048 bytes
--------------------------------------------------------
The Number of Channels is: 2
--------------------------------------------------------
Using UDP Protocol
--------------------------------------------------------
Waiting for Connection From Client...
```

Figure 1-1: A screen capture of interacting with Jacktrip, a system that supports bidirectional audio streaming using a command line interface. While functional, this is inherently complicated for musicians.

Despite technological difficulties, there are still benefits to rehearsing virtually. The Bay Berry String Quartet, a professional ensemble that relies heavily on remote collaboration, is a strong proponent of the flexibility of virtual rehearsals as well as the greater attention to detail resulting from the recording process [9]. Technology-based rehearsals enable group members—at a time that best fits their schedule—to parse out individual parts for better score comprehension and understanding of musical decisions. This motivates a musician to take strong ownership of their parts by paying closer attention to rhythm, intonation, and dynamics while recording. For these reasons, there was sufficient interest to build better online rehearsal solutions.

The proliferation of commodity software and hardware enabled improved NMP implementations with fewer trade-offs, grouped in four categories: generic video conferencing systems, real-time music rehearsal systems, online digital audio workstations, and home-spun solutions (detailed in Chapter 2). Video conferencing solutions have a simple set up process but do not prioritize music-making, resulting in extensive audio latency and bandwidth issues. Real-time rehearsal systems offer music-specific features but use low-latency connections, requiring specialized hardware and confusing network configurations. Online digital audio workstations (DAWs) provide an interface for users to record and edit audio directly in the browser, but lose the immediacy to iterate and discuss with a group. Home-spun solutions are conveniently asynchronous but require extensive post-processing by a single person to mix recordings.

While these four categories of solutions have evolved the NMP landscape, they demand particular skill-sets and significant resources from the user. Steep learning curves, unintuitive user interfaces, and poor audio quality all contribute to the growing pains of adopting a single NMP solution. As such, a clear winner has yet to emerge among existing tools, creating a gap between what currently exists and what a musician wants. This poses the research problem of investigating the possibility of both effective and user-friendly virtual rehearsal systems.

This thesis presents JamNSync, an accessible, comprehensive solution that encapsulates key aspects of an in-person rehearsal (Figure 1-2). By abstracting away underlying complexities of the technology, JamNSync allows musicians to rehearse in real-time using a collaborative recording session in the browser. This system bridges the aforementioned gap by focusing on usability — a tool created for classical musicians by classical musicians.

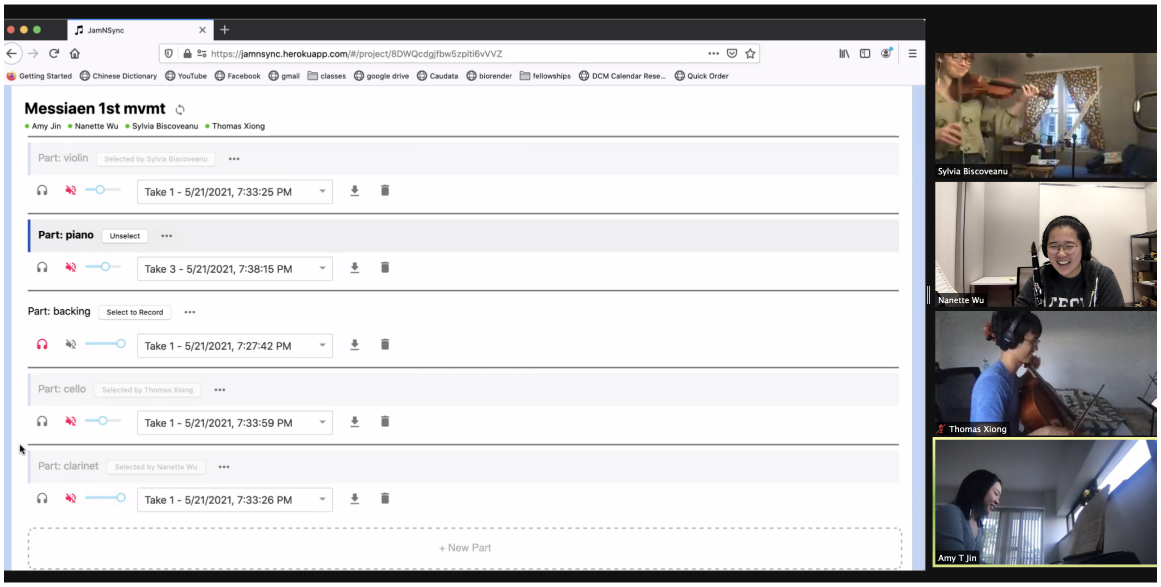


Figure 1-2: A screen capture of a quartet rehearsing with JamNSync.

JamNSync is a web application that streamlines the virtual music-making experience for small music groups (i.e. 2 to 5 people). The user interface is a React-based website hosted on a Flask server that facilitates storing, processing, and aligning of tracks in real-time. To use the application, a musician just needs a microphone and headphones to record and playback their part. They can meet their rehearsal group on a project-specific page that resembles a lightweight DAW. There, group members can coordinate logistics, synchronously make recordings, playback the ensemble mix in a built-in media player, and iterate on their recordings.

JamNSync makes the following contributions to the broader discipline of NMP:

- A **user-friendly, latency-agnostic** virtual rehearsal tool for groups to record and playback their piece with simple setup, all **without sending real-time audio** over the network.
- The **Real-Time Rehearsal protocol**, a distributed consensus algorithm to synchronize group recording and playback using WebSockets.
- An **audio alignment tool** that compensates for inconsistent browser latency by allowing users to visually align (i.e. drag) their recording to match a backing mix.
- A **DAW-like**, **versioned recording system** that allows a group to easily upload, download, and share audio files.
- Built-in **per-group project organization**, which gives a group easy access to manage the same set of pieces.
- A virtual rehearsal solution **resembling an in-person rehearsal** that allows musicians to see their group members over a video stream and immediately discuss what they played together.

The rest of this thesis explains the details of the JamNSync system. Chapter 2 provides background on music rehearsals and computer networks. Chapter 3 discusses related work on NMP applications. Chapter 4 outlines the system design in terms of frontend, backend, and data. Chapter 5 describes the system's web-based implementation. Chapter 6 evaluates the system through user tests. Chapter 7 contains the conclusion and future work.

Chapter 2 Background

This chapter provides an overview of small group music rehearsals (in-person and virtual) and computer network terminology.

2.1 Small Group Music Rehearsals

2.1.1 In-Person

In a traditional in-person context, ensembles meet regularly to prepare for performances, ranging from weekly meetings to a few rehearsals the week before a concert. Rehearsal lengths can vary between one and three hours. The process begins with communication over email or group chat to coordinate a time and place to meet. On the day of the rehearsal, group members travel to their predetermined location. They often arrive five to ten minutes early to arrange chairs and warm-up. Warm-up includes tuning among string and wind instrumentalists, playing technical exercises, and last-minute practicing of tricky excerpts.

After everyone has arrived, the group selects the subset of pieces to rehearse and where to begin rehearsing. Everyone then flips their sheet music to the agreed starting point. One group member visually cues to synchronize the start, much like how a conductor cues the beginning of an orchestral piece. The first run-through usually involves playing from start to finish, unless major issues required restarting. Playing a piece from beginning to end allows musicians to identify trouble spots, which are isolated for detailed work. Variations of rehearsing include playing slower with a metronome (match tempo and stylistic choices), louder (check intonation and blend of sound), softer (gauge balance and overall volume), and in pairs (work on strict alignment with a subset of instruments). Once a group makes corrections, they usually play more run-throughs, each of which is followed by a discussion to share suggestions to improve the ensemble work. In some cases, musicians use their phones to make audio recordings and track the progress of their piece.

Once a group finishes practicing their first piece, they repeat the process with the next, discussing where and what to play and proceed to rehearse again. At the conclusion of the rehearsal, group members discuss what to work on, clean the room, and travel home.

2.1.2 Virtual

Much like the process of coordinating in-person rehearsals, the process of coordinating virtual rehearsals begins with communication over email or group chat. Groups pick a time and virtual "meeting place" like an audio or video call. For synchronous rehearsals, each group member must find a quiet space with sufficient Internet bandwidth to participate. They typically require fifteen to thirty minutes to set up their technology by testing audio levels, configuring network cables, and connecting external input devices like cameras. As before, musicians will individually warm up, but tuning occurs relative to a fixed pitch (e.g. mobile app tuner) rather than to other group members.

After everyone arrives in the virtual space, the group verifies that each member can be heard properly, diagnosing potential issues like echoing and microphone distortion. The group then selects a subset of pieces to rehearse and the starting location in the first piece. Everyone then flips their sheet music to the agreed starting point.

If the rehearsal occurs on a real-time platform (detailed in Section 3.2), a group member might verbally count off or breathe sharply to provide an aural cue. Without in-person body language, this helps a group start the piece together. If the rehearsal occurs on an asynchronous platform (detailed in Section 3.4), a group needs to clearly articulate and agree on a set of logistical steps. First, a group must agree on and distribute a backing track (click track created from a tempo map) to ensure that individually recorded parts are aligned. Then, they choose the order of recording (e.g. piano records to the backing track, cello records to the piano part, violin records to piano and cello). Finally, the group needs to decide how to promptly distribute their audio file to the group (e.g. using a cloud file system).

Once group members listen to each other play, they discuss musical decisions about their pieces. The primary form of communication is aural (e.g. voice) and sometimes visual (e.g. text, video), which becomes challenging due to a lack of body language and eye contact. Once the group reaches a consensus about points of improvement, they will rehearse the piece again.

This process repeats for each piece the group works on. After a group finishes rehearsing, they reconnect in their virtual space, discuss their next meeting agenda, then disconnect their technology setup.

2.1.3 Characteristics of Effective Rehearsals

At each meeting, groups aim to have productive rehearsals by making significant progress on their repertoire. Effective in-person and virtual rehearsals have these desirable characteristics:

- Rehearsal groups spend most of their time making music: setup time should be minimal and straightforward, so musicians can focus on making music.
- Everyone is a conductor: group members should actively influence each other during rehearsals. Although one person cues the start, rehearsals should not be constrained to having a single leader. Instead, rehearsals are a collaborative effort, in which musicians react to each other in real-time.
- Iteration cycles are fast: a group immediately addresses points of improvement after identifying them. The time between rehearsing run-throughs of the same piece should be minimized.
- Rehearsing fosters a sense of community: it is important for each group member to feel connected to their group. A group rehearsal is just as much a social experience as it is a musical one.

2.2 Computer Networks

A computer network shares information by adhering to an agreed set of rules. The following section provides an overview of network concepts relevant to NMP.

2.2.1 Best Effort Internet

The Internet is a system architecture designed to connect networks in geographically separated locations. Properties of the Internet's original design still hold today, including the notion of a "best-effort" network [14]. When transporting data, the network ensures fairness among users by offering no guarantees for reliability or quality of service. As such, the Internet does not bias certain users or data types. In the context of NMP, the notion of best-effort poses challenges in providing high quality audio streaming. Unpredictable network conditions cause latency and packet loss, which are difficult to counteract in real-time [17].

2.2.2 Transport Layer Protocols

The Open Systems Interconnection (OSI) model describes a computer network using seven abstract layers. The fourth layer is the transport layer, responsible for creating, managing, and closing end-to-end communication between devices. Two transport layer protocols accommodate different application needs.

Transmission Control Protocol (TCP)

TCP supports reliable data delivery. Key characteristics include the handshake (i.e. establishes initial connection), congestion control (i.e. uses acknowledgments to ensure in-order delivery), and data streams split into small packets. Packets are deemed lost when acknowledgments are stale, requiring immediate retransmission before the next packet can be sent. However, unnecessarily retransmission can clog a network to the point that no connection can make useful progress [18]. Since high latency inhibits real-time virtual rehearsals, a more rapid method of data delivery is needed.

User Datagram Protocol (UDP)

UDP establishes low latency for connections that can tolerate loss. UDP avoids much of the performance overhead in TCP by not guaranteeing reliable delivery. No handshakes are needed, as no long-running connections are used to send messages between devices. No acknowledgments are needed either, since UDP does not require a response from the receiving party before sending data. Without needing to strictly monitor data transfer, messages can be sent quickly but be lost or out-of-order. Since real-time audio streaming prioritizes speed over quality, it is often better to use UDP over TCP in NMP applications.

2.2.3 Port Forwarding

In a residential network, devices can access the Internet through a cable modem connected to a router, a network component that sends and receives data. Modern home routers are based on network address translation (NAT), which uses the router as an agent between the Internet (public network) and the local home network (private network). When public network requests arrive at a router, NAT redirects them to a specific device within the private network. This makes private hosts publicly available by remapping the destination address to the internal host.

To use UDP in virtual rehearsals, port forwarding must be enabled on a router. To modify traffic flows, NAT needs to set up the appropriate forwarding state for a group to reach each other's private networks. Each group member sets aside a port number on the router that can exclusively access a service in the private network. Given that port number and the router's public IP address, external hosts can communicate with the internal service, allowing group members to connect in real-time and stream low-latency audio.

2.2.4 Internet Connection Types

Two common ways to connect to the Internet use wired and wireless connections.

Wired (Ethernet)

Ethernet connections use physical cables between the router and the device. Isolated wired connections provide reliability and stability in data transmission, as well as high-quality data transfer. For NMP, Ethernet can support low-latency audio streaming, but is less common in typical households. Since hardware is needed to make an Ethernet connection, wired setups also tend to limit user mobility.

Wireless (Wi-Fi)

Instead of using wires, Wi-Fi uses radio frequency signals to connect devices to the Internet. The main advantage of using Wi-Fi over Ethernet is convenience. Most households have routers that, by default, support Wi-Fi connections. However, wireless connections are less reliable and have unpredictable latency characteristics. They are also prone to interference by physical barriers (e.g. walls) and connections from other devices. In the NMP context, Wi-Fi is suitable for asynchronous platforms, but cannot support low-latency audio streams between users.

2.2.5 Audio Device Connection Types

In NMP, audio devices are needed to record input (microphone) and playback output (speaker). Audio devices can connect to an Internet-enabled device in two ways.

Wired (Audio Jack)

Traditionally, audio devices are connected by plugging a physical cable into to an audio jack. Wired devices generate high quality recordings by sending analog signals, which are uncompressed and have very low latency.

Wireless (Bluetooth)

Wireless audio devices, which have seen a recent rise in popularity, use radio transmission technology like Bluetooth to transport audio signals. People prefer wireless to wired headphones for their portability and convenience of unrestricted movement.

However, wireless headphones are unsuited for NMP due to high latency. As such, wired connections are typically necessary to participate in virtual rehearsals.

2.2.6 Network Architectures

Two popular network architectures establish connections between users: the client-server model and peer-to-peer model. For NMP, one is not necessarily better than the other; trade-offs are discussed below.

Client-Server Model

The client-server model is a traditional network model with a centralized manager (server) that provides resources to a user (client). A server has a one-to-many relationship with clients, allowing it to simultaneously communicate with multiple users.

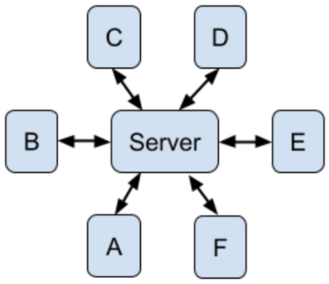


Figure 2-1: The client-server model. Clients A-F connect to the centralized server [15].

Once a client successfully connects to the server, the connection is held open for subsequent communication. The client-server model works well for virtual rehearsals when a group lives in close proximity (i.e. in the same city, to reduce packet travel distance). The server provides a virtual rehearsal space that users can share and join. It also handles concurrent updates to enforce consistency and safe access to shared data (e.g. audio files, rehearsal state). The main disadvantage of client-server is increased latency between end-users (i.e. routes must include extra hops to the server to determine the packet destination's physical location). Other disadvantages include higher initial setup cost to configure and maintain a server and a single point of failure (i.e. if the server crashes, the system is entirely unavailable).

Peer-to-Peer (P2P) Model

The P2P model is a distributed application architecture that allows interconnected users (peers) to communicate without a centralized manager. Originally created to form groups and share audio files for Napster, the P2P model is inherently designed for collaborative use [27].

Each peer is an equal participant that makes a portion of their resources available to other network participants. By design, a peer is both a supplier and a consumer, which distributes work more evenly among nodes than the client-server model does.

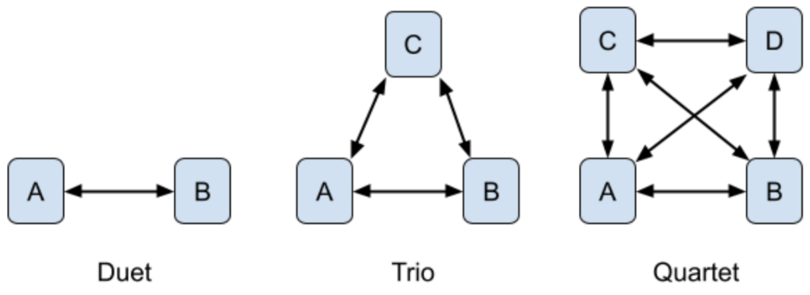


Figure 2-2: The P2P model for virtual rehearsals. Each node (A-D) represents a musician, and each double arrow represents a connection between two musicians [15].

For virtual rehearsals, the main advantage of P2P is a direct connection between users that do not need to pass through the server. Used on top of UDP, P2P direct routing has the potential to transmit audio data very quickly. Another advantage is resiliency: even if one computer goes down, the system remains available and easily recoverable (i.e. other users can continue communicating while the node reconnects). The main disadvantage is that the number of connections increases significantly for each user added to the network. When the number of users becomes too high, sending data across a fully connected network becomes unwieldy, significantly increasing local CPU cost. However, as shown in Figure 2-2, the network complexity is still manageable for small group rehearsals.

Chapter 3 Related Work

This chapter introduces related works that currently implement software-based music-making. Four categories of NMP applications include generic video conferencing systems, real-time music rehearsal systems, online digital audio workstations, and home-spun solutions [34]. The subsections below describe benefits and pain points of each category.

3.1 Generic Video Conferencing Systems

Generic video conferencing systems refer to software initially designed for remote meetings and later adapted for music rehearsals. Real-time video support and simple set-up processes drew musicians to these platforms. However, these solutions do not prioritize music-making use cases. Features that popularize these systems directly conflict with the goals of synchronous music playing, which is unideal for virtual rehearsals.

Zoom is a comprehensive enterprise product that hosts virtual meetings [35]. Musicians flocked to this platform because of its widespread popularity during the pandemic. Unfortunately, many barriers arose from Zoom's non-music specific features like noise cancellation and single-speaker voice optimization. This made synchronous rehearsals, where all musicians should hear each other at once, nearly impossible.

A common workaround is to do a round-robin rehearsal: one group member, the "leader", remains unmuted, while the rest of the group mutes themselves and plays along to the leader. For the ensemble to hear each part, group members take turns being the leader. The key issue in this workaround is the lack of feedback among musicians. The solo player cannot hear any other musician, while other musicians can only react to the soloist. As group sizes get larger, this process requires more iterations, making the rehearsal process more tedious and time-consuming. For groups like jazz combos that could tolerate some degree of latency, an alternative approach was to intentionally ignore the delay and attempt playing synchronously. However, variable latency, compounded with unpredictable network congestion and buffering, also caused audio stream dropouts and artificial tempo variation. Thus, poor audio quality prevented adoption of this idea.

Other audio issues in Zoom exacerbated the limited opportunities of feedback, including audio that was tailored specifically to the human voice due to acoustic filters that erroneously filtered out frequency bands of non-vocal instruments. To compensate for this, Zoom users can turn on “original sound" to apply a less aggressive filter. Not only was lots of manual tweaking required to determine how to reduce sound distortion, but the audio quality was also still clearly lacking. These approaches to work around an enterprise-oriented application are not authentic to an in-person rehearsal.

WebEx is a popular competitor of Zoom that offers a similar set of enterprise features [33]. Like Zoom’s “original sound," WebEx’s “music mode” preserves a microphone’s original feed. A primary difference between WebEx and Zoom is the pricing of premium features: WebEx tends to be more affordable at a cap of 50 participants per meeting. Nonetheless, WebEx’s features uncovering similar issues found in Zoom.

Jitsi is a free platform that provides high video quality and encryption [22]. A key factor that distinguishes Jitsi from other video conferencing systems is the open-source nature of the tool. Unlike Zoom or WebEx, which requires a premium plan for full use, Jitsi provides free meetings with unlimited meeting durations. However, because Jitsi still prioritizes video conferencing specific features, it lacks an option to toggle a “music mode." Instead of using Jitsi as the sole platform for rehearsing, musicians often use Jitsi to complement virtual rehearsals (Section 3.2).

3.2 Real-Time Music Rehearsal Systems

Real-time music rehearsal systems refer to software specifically designed to address audio synchronization for NMP. Current research suggests that a realistic musical interaction between two rhythm-based instruments requires a one-way latency less than 25 milliseconds [10]. Any larger value results in tendencies that awkwardly drag the tempo of the music.

To achieve low latency network conditions, musicians need to use wired audio devices to minimize latency. Some rehearsal systems also require home network routers to use Ethernet and support UDP port forwarding. However, this is often difficult for an average musician to set up and coordinate, especially to achieve optimal network conditions.

Founded by Stanford's CCRMA, Jacktrip is a multi-platform, high-quality P2P solution supported by UDP that supports bidirectional audio streaming [11]. The significant reduction in low latency, ability to work with commodity hardware, and low financial investment make Jacktrip an appealing platform for musicians. Although Jacktrip allows fine-grained control over network settings, like packet size, frames per period, and bandwidth, the terminal-based interface to discover other Jacktrip clients is unintuitive. Using IP addresses to connect to other musicians over the network made Jacktrip difficult to use for the average user [34].

Developed as a reaction to Jacktrip's unfriendly interface, SoundJack is another UDP-based P2P solution that provides low latency communication [29]. A properly configured SoundJack system results in an in-person-like audio experience using an Ethernet cable plugged into a computer network port or an external Fastmusic box [31], a dedicated processor with pre-configured audio and network settings that minimize latency. To improve usability over Jacktrip, SoundJack provides a browser-based "stage" interface to easily connect to other users without needing to know each other's IP addresses. Many musicians also prefer SoundJack to other real-time rehearsal systems because of its ideal balance of features, flexibility, and complexity. However, there is still room for improvement to make the user interface easier to use.

Currently, settings for low-level network parameters (e.g. jitter buffer, sample rates) are still graphically unintuitive, time-consuming to tune, and inherently complicated. SoundJack also disconnects frequently without notifying the user and produces delays that are hard to diagnose. These silent failures can stall the process of synchronous rehearsals, making it difficult to know if an issue is rooted in the equipment, the setting configuration, or the platform itself. Becoming comfortable with this system requires overcoming a steep learning curve, thus preventing widespread adoption.

Jamulus uses a public client-server model to coordinate musicians to jam with their ensemble in real-time over a broadband connection [16]. Because Jamulus uses a client-server model, packet transmission requires more network hops, making it intrinsically harder to achieve the desired 25 ms latency. Instead of a single hop from the source to the destination, two hops are needed to pass through the centralized server, which directs packets to the appropriate target address. Even though an appropriately set up recording environment in Jamulus offers high quality audio and some degree of low latency, the user interface is outdated and much too complicated for the average user to understand, much like SoundJack. As such, Jamulus is difficult to set up and manage during a virtual rehearsal, and its architecture is not conducive to guaranteeing low latency.

Jamkazam is a live music platform that was developed out of efforts to reduce latency in cloud gaming [21]. Unlike Jamulus, Jamkazam allows users to choose between client-server and P2P architectures, depending on which model is faster for where the user is located. Options of using an audio interface and Ethernet cables are recommended but not required to use the application. A distinguishing feature of this platform is the use of JamTracks: fully isolated multi-track backing tracks stored in the cloud. While tools like SoundJack and Jamulus only offer a virtual space for rehearsals, Jamkazam provides a library of material for musicians to work with. The main issue with Jamkazam is its unintuitive front-end, despite claiming to offer a user-friendly interface to mix track. New features randomly pop up in new windows without a clear sense of organization, making application navigation confusing. Options and features are often described with wordy yet vague text description; it is unclear which buttons and modals are actually interactive.

Cleanfeed is a multi-party, browser-based high definition live audio recording tool that uses any Internet connection over TCP [12]. Designed for easy use and high quality, Cleanfeed relieves the user of configuring low-level network parameters such as setting buffer sizes. Cleanfeed is heavily used by broadcasters, podcasters, and producers to allow studio operators to interact with remote guests and co-hosts with few to no audio dropouts. However, Cleanfeed does not operate as a real-time system for synchronous rehearsal; the desired simultaneity of virtual rehearsals cannot be achieved because Cleanfeed does not attempt to obtain a 25 ms latency.

NINJAM is an open-source, centralized system that uses a unique representation for organizing recorded music [24]. Instead of defining tracks in terms of time (e.g. seconds, minutes), NINJAM is based on the idea of looping music in terms of well-defined sections such as 16-beat chunks. Because this system accepts that true real-time synchronization is near impossible, NINJAM records and streams smaller intervals of compressed music that are flexible with larger latency values [34]. Rather than rehearse full-length pieces, musicians play along to previously recorded buffers of their ensemble members. Since NINJAM is also compatible and frequently used with Reaper, a popular DAW application for mixing technology-based music, NINJAM is more suitable for music genres that emphasize repeating sections (e.g. electronic dance music, pop music) and improvisation (e.g. jazz), as opposed to classical music.

3.3 Online Digital Audio Workstations (DAWs)

A DAW is an application used to record and edit audio files. While real-time systems are specifically designed for group rehearsals, DAWs are typically intended for a single person. However, a DAW-based framework for group rehearsals uses recording and producing software in a way that is not its primary purpose.

Online DAWs are browser-based: they create a platform that lends itself well to accessible and asynchronous music playing. In a typical online DAW rehearsal, one musician records first (e.g. piano offers consistent tempo), followed by other group members who asynchronously layer their tracks on the initial recording. Because layered tracks are independent of each other, musicians can individually record at their convenience—a major benefit to asynchronous rehearsals. Once all musicians have recorded their parts, they export their project session as a combined group recording, a representation of what a rehearsal might sound like. To discuss and reflect on the recording, an ensemble might meet on a video conferencing platform like Zoom to pinpoint areas of improvement. However, they are unable to immediately work on identified issues and re-record on the spot. Each musician must instead remember suggested changes for the next time they record, a task both more difficult and unproductive than a synchronous rehearsal. Even though online DAWs provide accessibility and flexibility for the user, major flaws stem from the loss of immediacy of group rehearsals.

Soundtrap is a freemium browser-based cross-platform DAW acquired by Spotify [30]. With video and text chat support, this user-friendly virtual music studio allows musicians to collaborate in real-time with an extensive set of presets, automation, and virtual instruments. The interface to record and playback audio tracks is well-designed, as the visual feedback from navigating the webpage allows processes to be self-explanatory. From an organizational standpoint, Soundtrap excels in allowing users to organize projects into separate folders, allowing rehearsal groups to easily track every project that is a work in progress. A musician can also invite their group to join a session by simply sharing a web URL - no additional download or software is needed. However, the collaborative aspect is limited: people can only see who else is recording, listen to each other's ideas, and discuss them in a video conference style. If a user makes changes to a shared project page, the changes do not immediately populate in their group member's pages; the rest of the group can only view modifications by manually refreshing after being prompted to do so. Synchronous performance and immediate feedback are instrumental to virtual rehearsals but missing in Soundtrap.

Upbeat is a DAW-based platform that allows musicians to record music virtually in real-time [32]. Upbeat offers simple recording features, unfiltered audio feeds, and immediate playback after recording. Key features of Upbeat include a Jitsi-based video feed, high quality audio stream, and an audio alignment tool that allows users to adjust recordings based on a backing track. However, the process to transfer files significantly impedes the rehearsal process. Upbeat requires a platform-specific file type (.uptrk) and local audio storage. Tedious file upload and download add hassle to the non-performing portions of the rehearsal, slowing down the rehearsal iteration cycle. Furthermore, the visual and aural eight-beat countdown before a recording begins (shown as two groups of four, always in a fixed tempo) is confusing for pieces of different tempos or time signatures.

BandLab, an easy-to-use social music platform, offers a lightweight DAW that has just enough features to run a virtual rehearsal [7]. Its interface allows musicians to collaborate on projects with unlimited cloud storage. However, pain points include track organization (i.e. need to make new groups for each piece), track alignment (i.e. workstation does not consider possible delay from hardware and software - not compatible with Bluetooth headphones and frequently need realignment), and lack of features tailored for classical music (i.e. those familiar with more advanced recording software, like Reaper, prefer interfaces with more functionality). Poor experiences in setup and rehearsal turn away some users from continual usage.

3.4 Home-Spun Solutions

The previous three categories offered ways to rehearse synchronously. A home-spun solution, however, is an asynchronous process that gives the illusion of synchronous playing through an edited performance video. A popular example is the Zoom-like grid of musicians who have separately recorded video and audio tracks but appear to be playing in real-time to a listener.

To put together such a video, the group creates and distributes a backing track to each member. Musicians follow the backing track to individually record their part with a recording interface (e.g. Garageband, Reaper, Audacity), but cannot hear their group members as they record. Completed recordings are uploaded to a shared folder (e.g. Dropbox, Google Drive) that collects tracks from the group. A designated member edits the tracks together and distribute the finished product. However, the group can only iterate on their previous recording after reviewing the exported video file. This process is inefficient and unrepresentative of a real rehearsal.

A common method of editing uses built-in computer software like Garageband and iMovie for Mac. After video and audio files are collected, a designated editor will mix the audio first by adjusting volume, balance, and equalizers in Garageband. The audio mix is imported into iMovie along with the raw video files, which are used to create a grid of videos. The completed video is exported and uploaded to a sharing platform like YouTube for all members to stream, find points of improvement, and discuss rehearsal methodologies.

While this process produces a perfectly aligned virtual performance, it is both slow and artificial. Tools with more features like Reaper and Audacity have steep learning curves that make them difficult to use. This also requires a cloud-based file service (ex: Dropbox) to exchange files and a communication medium (ex: Zoom) to review the mastered track, which puts a heavy burden on the designated editor to manage files. Since audio editing can mask mistakes and intonation issues, musicians also frequently overlook areas of improvement in their pieces.

Openshot is an open-source software that combines video and audio editing into a single tool [25]. While a designated editor is still needed, Openshot expedites the process of stitching together a video grid. However, issues of slow video distribution and group discussion feedback are still present, limiting the number of iterations a group can make per piece.

The Acapella App is a mobile split-screen video editor designed to create music collaborations for up to nine people [1]. Unlike Openshot, this app has features specifically designed for classical music projects. To produce a collaboration, all group members need to download the application and decide on a recording order. The first person to record has the most responsibility: they need to set up the project, decide on a tempo, and choose the number of project tracks. These decisions are permanent after project creation. The last person to record becomes the designated editor who decides on the video layout, adjusts audio levels, and exports the completed project to convenient messaging platforms (e.g. SMS). Compared to previous tools, this app provides a simpler distribution process and a centralized space to manage recordings. However, only one collaboration can be created at a time: free users can record only up to one minute, and premium users can record up to ten minutes. The strict limitation in file storage inhibits the possibility of long-term rehearsals, which are also heavily dependent on parameters set by the first person who records.

Regardless of the platform used for media editing, a key issue of home-spun solutions is the dependence on an individual. The manual editing process managed by one person involves tedious data-wrangling to adjust video submissions (i.e. inconsistent file formats, lighting, frame ratios) and audio recordings (i.e. muting wrong notes, pitch correction, balance) [26]. Moreover, a musician's recording process becomes a lonely and repetitive exercise to attempt a flawless, high-quality track [19]. Thus, these home-spun solutions result in virtual music performances that are only perceived to be in real-time and do not effectively replicate in-person rehearsals.

3.5 Summary

Given four partial NMP solutions, JamNSync aims to address the overarching problem of completeness and usability in virtual rehearsal platforms. Video conferencing systems like Zoom lack sufficient features for musicians to rehearse synchronously. Real-time systems like SoundJack depend on strict network conditions that require precise and complicate setups that are inaccessible for most musicians. Online DAW's lack immediacy in feedback and iteration of a group rehearsal. Home-spun solutions are wasteful of both time and storage. As such, there is significant value in investigating solutions to bridge the gap, caused by design decisions that did not fully consider the needs of a musician, between what currently exists and what musicians want.

The approach used by JamNSync to address existing limitations in NMP shifts the focus from complex partial solutions to a simple complete solution. The impact of this approach is rooted in the notion of improved usability for musicians through an accessible interface, with features tailored specifically for the users of the system.

Chapter 4 Design

This chapter introduces JamNSync's virtual rehearsal process, design goals, and a basic system overview. A novel consensus protocol to enable synchronous rehearsals is presented, followed by the organization of data in a relational model and a cloud file system. The rest of the chapter details the system's frontend (i.e. user interface) and backend (i.e. API) design.

4.1 Virtual Rehearsal Process

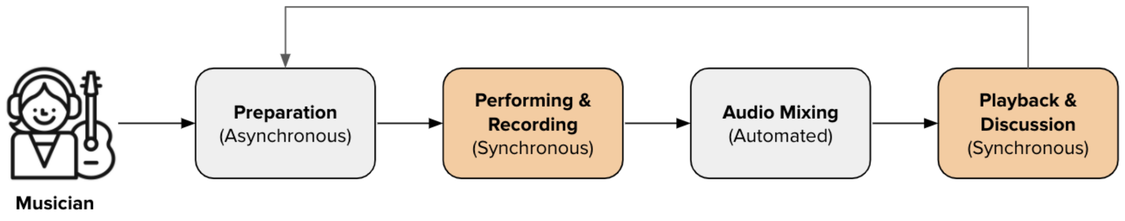

Musician
Preparation (Asynchronous)
Performing & Recording (Synchronous)
Audio Mixing (Automated)
Playback & Discussion (Synchronous)

Figure 4-1: Virtual rehearsal as a "feedback loop".

A musician begins a virtual rehearsal by assembling their computer recording environment and checking audio device levels ("preparation" phase). This is performed asynchronously to provide flexibility in a musician's individual preparation process. Next, the musician virtually meets their ensemble on a preferred communication medium (e.g. Zoom) and logs in to the JamNSync web application. Once all members are present, the "performing and recording" phase begins. The group decides on a backing track—an initial seed for the piece like a pre-made click track or a recording made by one group member—which is then uploaded to the project and shared with the group. Then, the group records a session resulting in n recordings for n musicians. The first session involves everyone recording their part along to the common backing track — this achieves part synchronization. In future recording iterations, the group has the choice to record along to the original backing track again or the just-recorded takes as a backing mix. Following each iteration, recordings are uploaded to the backend server and aligned to create a unified group recording ("audio mixing" phase). The adjusted tracks are then sent back to all group members, who synchronously listen to the mix and discuss areas of improvement ("playback and discussion" phase). Observations from a playback session, such as dynamic and phrasing choices, are used to inform the next recording session. The group's collective feedback initiates a new iteration cycle, looping back to the preparation phase again (Figure 4-1).

4.2 Design Goals

JamNSync is designed with the following goals:

Usability for Musicians. Setting up this application should be as simple as setting up Zoom. Computer and network jargon should be explained with musician-friendly terminology. User interface design choices must prioritize the needs and knowledge of musicians.

High Audio Quality. A musician's recording should accurately reflect what they experienced in-person. Audio data should have high fidelity to an instrument's dynamic range and frequency spectrum. Filtering and compression of audio data should not compromise the quality of sound.

Completeness. The system serves as a "one-stop shop" that encapsulates the fundamental features necessary for a productive virtual rehearsal. Users should find everything they need for recording and playback within the application; additional recording software should not be needed for a successful ensemble playing experience.

Fast Software Performance. The software is designed to enable productive group rehearsals. Users should get through multiple iterations of recording and playback during a virtual rehearsal. As such, file transfer and processing should be fast and seamless; an ensemble should spend rehearsal time primarily rehearsing music.

Reliability. Musicians expect their recordings to be stored safely. Audio files must be replicated to tolerate missing or corrupt files.

As it turns out, some of these goals work against each other (i.e. completeness and usability). Based on feedback and results from user tests, necessary trade-offs were made to complete and deploy a functional application.

4.3 System Architecture

JamNSync uses the client-server model over a standard Internet network to coordinate rehearsals among group members. Since the application does not stream real-time audio, a peer-to-peer system over Ethernet/UDP is not needed. Before any playback begins, audio data from the recording phase is shared amongst users. This allows remote procedure calls to initiate synchronous group recording and playback sessions. Because this architecture is a standard web application in the browser, additional network setup like opening firewalls or port forwarding is unnecessary, thus creating a simpler onboarding process for users.

As shown in Figure 4-2, each musician independently connects to the server using a browser. The server acts as a centralized manager that stores state about client connections and group sessions. As the server processes incoming client requests, it communicates with two external storage services to read and write data. One is a relational database, which stores persistent object metadata and relationships. The other is a cloud storage service, which supports an easily accessible file system that offloads storage management and replication of larger audio files.

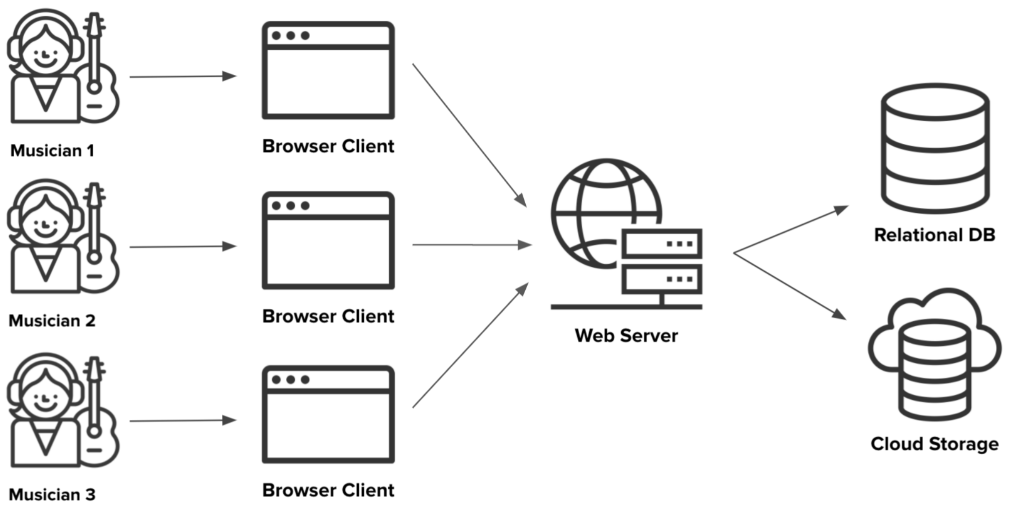


Figure 4-2: Main components of JamNSync system design with three musicians.

4.4 Real-Time Rehearsal Protocol

For the server to facilitate synchronized recording and playback, a consensus protocol is used to coordinate communication between clients. Inspired by the voting phase of Two-Phase Commit [8], the Real-Time Rehearsal (RTR) Protocol was developed for synchronized group recording and playback, to mimic the conventions of an in-person rehearsal as much as possible. The protocol has two variants — first to initiate a group record or play session, then to stop an existing session.

Group Record / Play

Figure 4-3 shows a sequence of network requests needed to initiate a group session. The same sequence of steps is used to initiate group record and group play. Without loss of generality, assume client 1 initiates a REQUEST to the server (1). A wait response is sent from the server to client 1, while a BROADCAST REQUEST is sent from the server to the rest of the group to inform them of client 1's request (2). All users must agree to begin group recording or playback, so each client votes Yes or No to participate in the synchronized session, which gets sent to the server (3). Each group member blocks until all clients have submitted a Yes vote, but their request

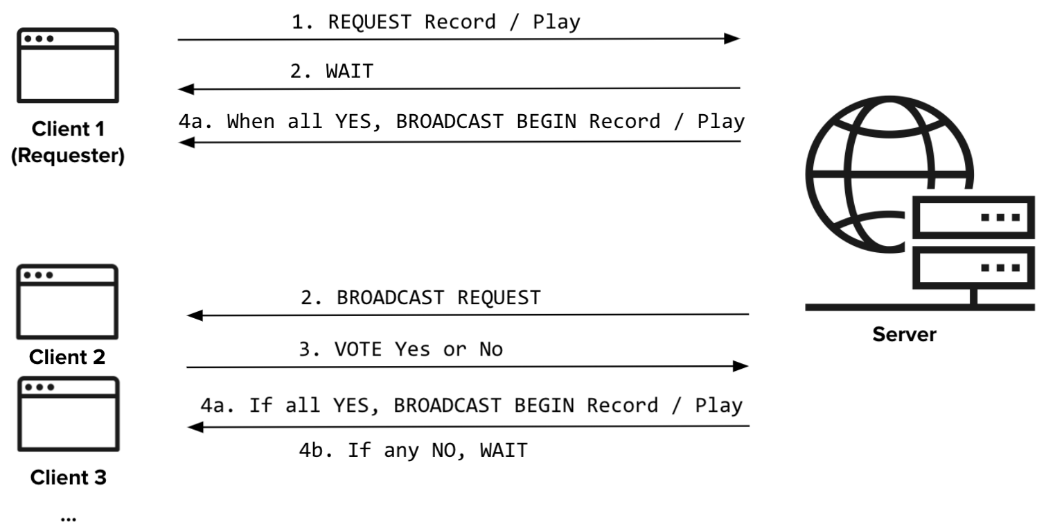


Figure 4-3: Initial request of group record or group play using RTR.

can be cancelled at any time without aborting the entire voting process. Once the server sees a consensus of Yes votes, it will send a BROADCAST BEGIN message to each client, which then immediately begins local recording or playback (4a). If any client votes No, group members continue to block until all clients agree to record (4b).

If a request has already been made and the vote has not reached quorum, another client may make a request to group record or play. When the server receives this request, they will not emit any BROADCAST REQUEST messages to prevent inundating clients with requests. Instead, the server will track the number of "prepared" clients who are waiting to begin, and increment this number for each new request. Once the total number of prepared clients matches the total number of members in the group, the server sends a BROADCAST BEGIN message as before. If not, group members continue to wait.

Group Stop

While a group must collectively decide to begin recording or playback, any single group member can halt the session. The design of the group stop mechanism emulates the stopping of in-person rehearsals (Figure 4-4).

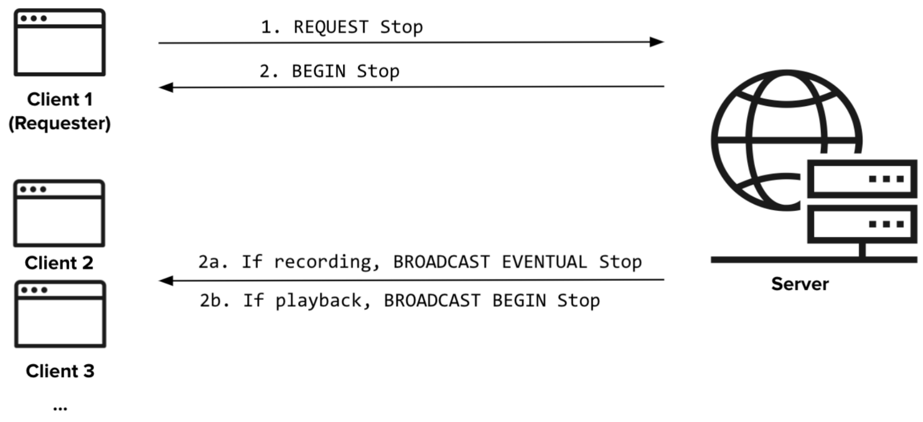

Client 1
(Requester)
1. REQUEST Stop
2. BEGIN Stop
Client 2
Client 3
...
2a. If recording, BROADCAST EVENTUAL Stop
2b. If playback, BROADCAST BEGIN Stop
Server

Figure 4-4: Request of group stop using real-time rehearsal protocol.

To stop a session, client 1 initiates a REQUEST to the server, without loss of generality (1). As soon as the server receives the request, it will send an acknowledgment to client 1 to permit them to BEGIN stopping. Depending on the type of group session, the server broadcasts a specific message to the rest of the group. If the group was recording, the server will BROADCAST EVENTUAL stop, alerting the group that a request to stop has been made, but will not actually stop the recording until the client acknowledges the alert (i.e. clicks OK). This prevents the broadcast from abruptly cutting off a user's recordings, in case there were browser latency or issues causing a delayed start. However, if the group was playing back, the server will BROADCAST BEGIN stop to other clients — the same acknowledgment message sent to client 1. There is less variability during a group play (e.g. no countdown, no need to request microphone permissions), so immediately stopping playback is a safe procedure that does not affect correctness.

4.5 Data Organization

The intrinsic hierarchy from group to project to track to take suggests a relational model to structure project metadata. Figure 4-5 outlines the entity-relationship model for entity sets related to JamNSync.

The corresponding database schema is derived below, where each bullet point represents a relation and each tuple contains the fields of that relation. * indicates a

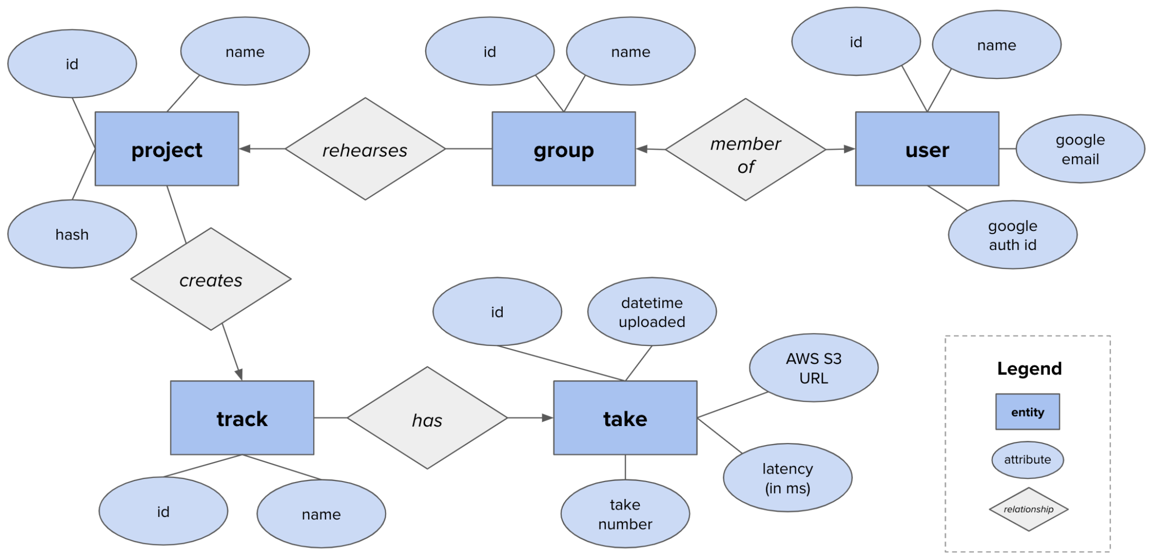


Figure 4-5: Entity-Relationship diagram.

primary key, and $a \to b$ indicates that a is a foreign key reference to b.

- user: (id*, name, google_email, google_auth_id)
- rehearsal_group: (id*, name)
- group_membership: (user_id* → user.id, group_id* → rehearsal_group.id)
- project: (id*, name, hash, group_id → rehearsal_group.id)
- track: (id*, name, hash, project_id → project.id)
- take: (id*, take_num, date_uploaded, s3_info, latency_ms, track_id → track.id)

Some cloud storage services use a hierarchical file system. JamNSync uses a flat hierarchy consisting of two layers: 1) projects (identified by hashes), then 2) takes (identified by track id, take number, and timestamp). Similar to how Google Docs assigns a single ID to a document, using a flat structure rather than a deeply nested one makes audio files easier to access. In particular, each take has the following file format, which provides readability and uniqueness (avoids stale data from caching names, robust to project and track name changes).

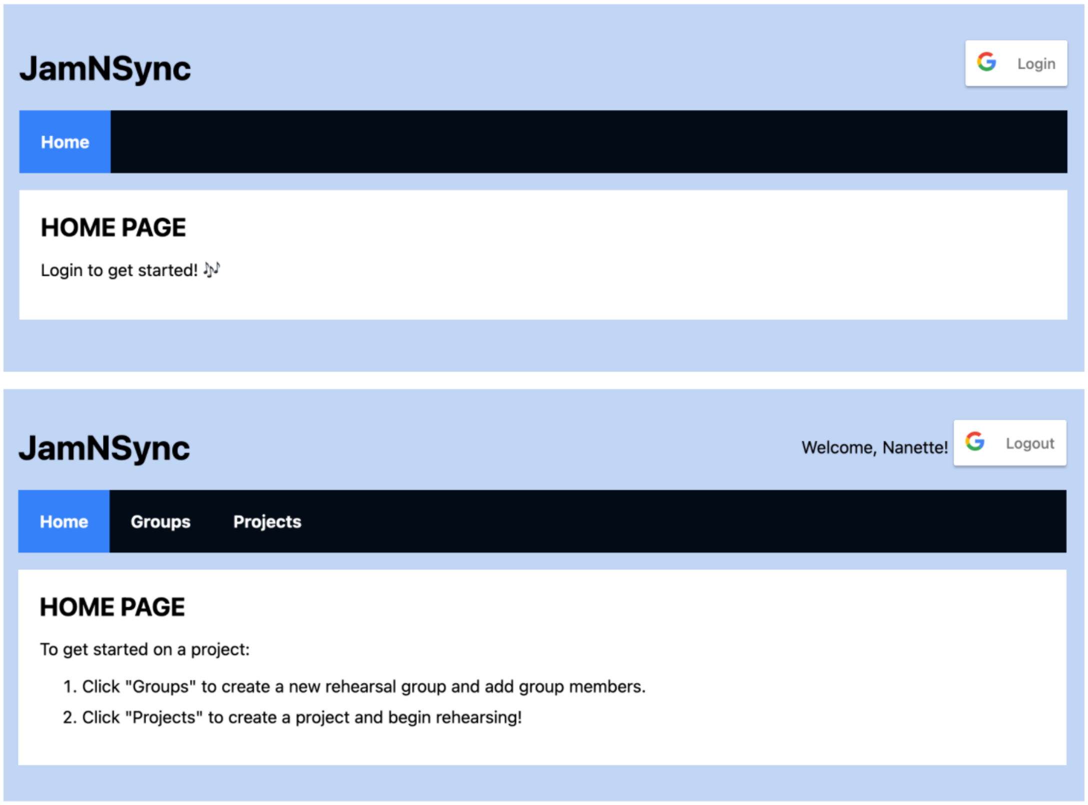
```
track<track_id>_take<take_num>_<datetime>.mp3
```

4.6 Frontend: Client UI

The client interface consists of four pages: 1) Home, 2) Groups, 3) Projects, 4) Project. The first page is publicly accessible, while the other three are only visible to authenticated users.

1) Home

The home page is the website's landing page, allowing a user to login and logout (Figure 4-6). To provide a familiar authentication process, the top-right corner of the page has a button with the symbol of a popular third party API (e.g. Google Sign-In). An external service is preferred over building a login management system from scratch to offload authentication logic to more secure systems [2].



HOME PAGE

Login to get started! 🎶

HOME PAGE

To get started on a project:

1. Click "Groups" to create a new rehearsal group and add group members.
2. Click "Projects" to create a project and begin rehearsing!

Figure 4-6: Home page interface: not logged in (top) and logged in (bottom).

2) Groups

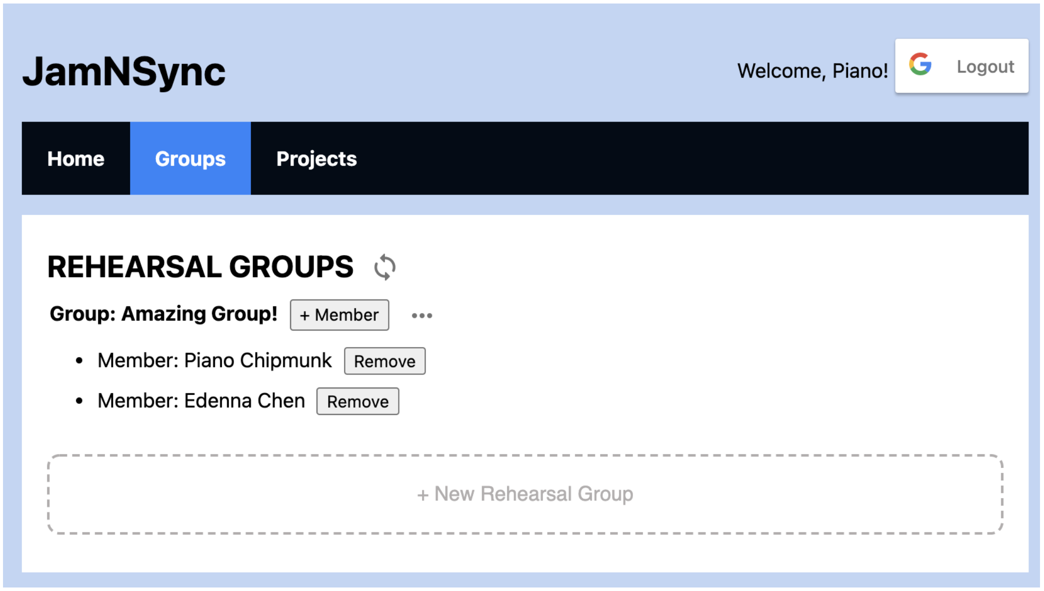


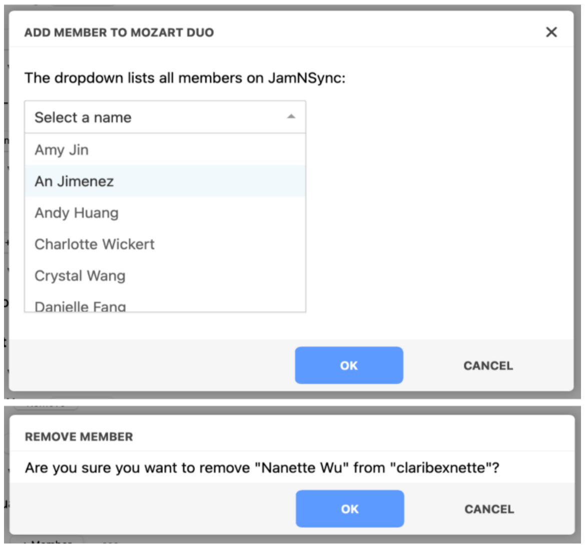
Figure 4-7: Groups page interface.

This page lists every rehearsal group that a logged-in user is a member of (Figure 4-7). For each group, a user can view, add, and remove members, as well as rename and delete the group. When any group member makes an update, the change automatically propagates to the entire group. The refresh button at the top queries the database for new updates. The dashed button at the bottom of the page creates a new rehearsal group.

Figure 4-8 shows the modal UI for adding and removing members. Currently, the add member modal has a dropdown menu containing every member from the database in alphabetical order. This is a temporary solution when the user base is small. An auto-complete search functionality should be considered if the application scales larger.

3) Projects

This page lists every project in groups that a logged-in user is part of (Figure 4-9). Each group section contains a "+ Project" button to add a new project. Each project has a three-dots menu for the user to rename or delete the project, and a "Rehearse!"


ADD MEMBER TO MOZART DUO

The dropdown lists all members on JamNSync:

Select a name

Amy Jin

An Jimenez

Andy Huang

Charlotte Wickert

Crystal Wang

Danielle Fang

OK CANCEL

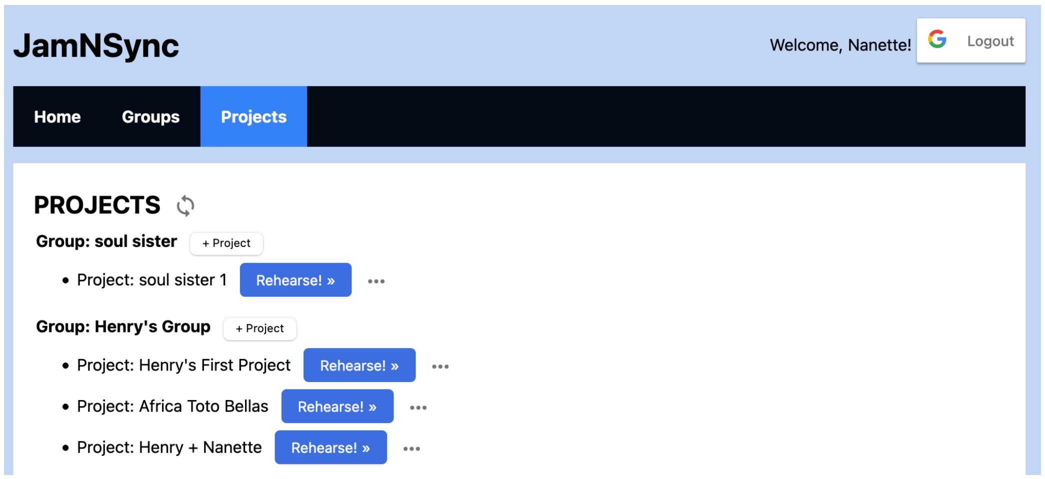
REMOVE MEMBER

Are you sure you want to remove "Nanette Wu" from "claribexnette"?

OK CANCEL

Figure 4-8: Add (top) and remove (bottom) member modals in the Groups page.

button to access the Project page. When any group member makes an update, the change automatically propagates to the entire group. The refresh button at the top queries the database for new updates.



PROJECTS

Group: soul sister + Project

- Project: soul sister 1 Rehearse! »

Group: Henry's Group + Project

- Project: Henry's First Project Rehearse! »
- Project: Africa Toto Bellas Rehearse! »
- Project: Henry + Nanette Rehearse! »

Figure 4-9: Projects page interface.

Inspired by Soundtrap's interface, the design of assigning projects to a group resolves the lack of organization in BandLab—where projects are standalone—that frustrated groups who wanted to work on multiple pieces at the same time.

4) Project

The project page is a simplified DAW that allows a group to synchronously record and playback multiple parts (Figure 4-10). Here, the word "part" is synonymous with "track", chosen to provide more user-friendly terminology. DAWs are typically quite complex, supporting a large set of features. JamNSync borrows the familiar DAW interface for recording but prioritizes usability for musicians by only offering the subset of features necessary for a virtual rehearsal.

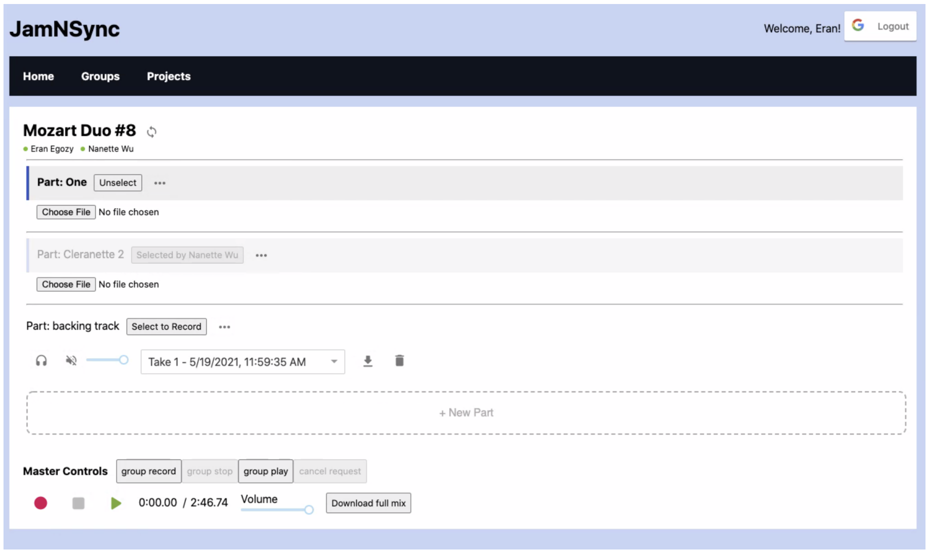


Figure 4-10: DAW-like project page with two connected users.

The page has three horizontal sections: the header, project parts, and master controls. Located under the navigation bar, the header displays the piece title, a refresh button to update the latest project information, and a horizontal list of users who are currently online.

The second section, the project parts, contains a visual list of instrument parts created by the group. The top of each list item contains the part name, a button to select/unselect the part for recording, and a three-dots menu to rename/delete the part and realign takes. When a user selects a part, the list item highlights with a dark blue accent on the left. When another group member selects a part, the list item is semi-highlighted with a light blue accent on the left, and the button to select/unselect becomes disabled with the name of that member. The bottom of each list item contains a file upload button if no prior takes exist. Otherwise, that area contains options to mute, solo, adjust volume, and download/select/delete takes. When a part is soloing, all other tracks automatically mute (Figure 4-11). Multiple parts can be soloed together to listen to any subset of parts. The bottom of the project parts also has a dashed button to create a new part.

Each recording automatically creates a new take on the selected part. When a user records, their part will mute while the other parts play. This simulates the in-person experience of hearing other group members during rehearsals. The volume of each part can also be dynamically adjusted during playback.

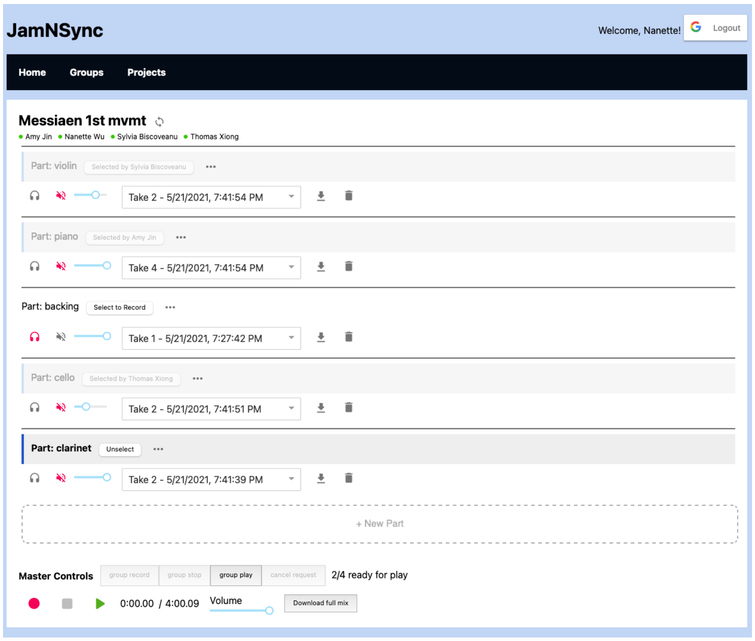


Figure 4-11: Project page for a quartet where a single part is soloing.

The third section houses the master controls for the project. Under the "Master Controls" label, three buttons activate recording, stopping, and playing respectively. The record button creates a new take for a selected part, while the stop and playback buttons apply to all parts. If the user is playing or recording, only the stop button is active. If the user has stopped, only the play and record buttons are active. Before any recording begins, JamNSync requests user microphone permissions in the browser; recording will not be allowed until they are granted. The master controls also show the scrolling time over the total time (determined by the longest part), which stops playback when it reaches maximum time. The scrolling time is bolded black during playback, and bright red during recording. This provides a clear visual indicator of an active session. The controls also have a master volume slider and a button to download a mix of the latest takes.

If more than one user is online, four gray buttons (see Figure 4-12) used for the RTR protocol appear next to "Master Controls". If only one user is online, the buttons are hidden. By default, group stop and cancel request disable until group record or group play is selected. During an active request, a label to the right of the buttons appears to show the number of clients who are ready to record/playback over the number of clients online (e.g. “1/2 ready for record” or "3/4 ready for play").

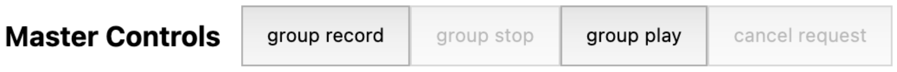
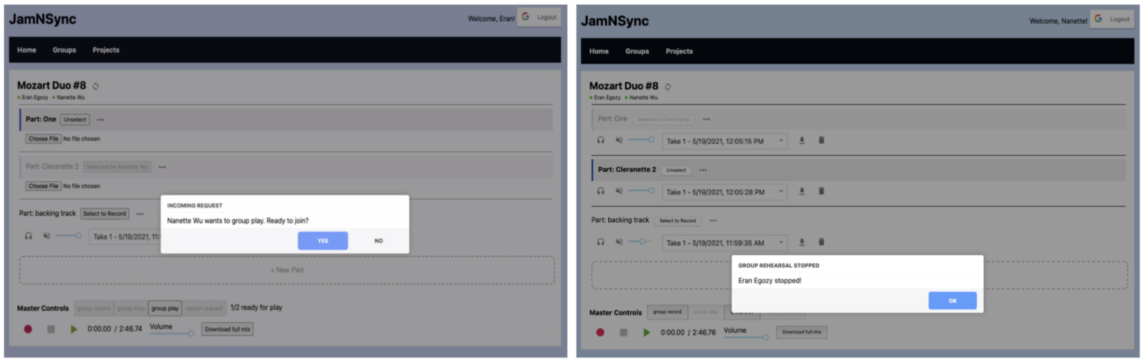


Figure 4-12: Group buttons used to synchronize rehearsals.

Figure 4-13 and 4-14 show the visualization of the group playback and recording variant, respectively, of the RTR protocol discussed above.



INCOMING REQUEST

Nanette Wu wants to group play. Ready to join?

YES NO

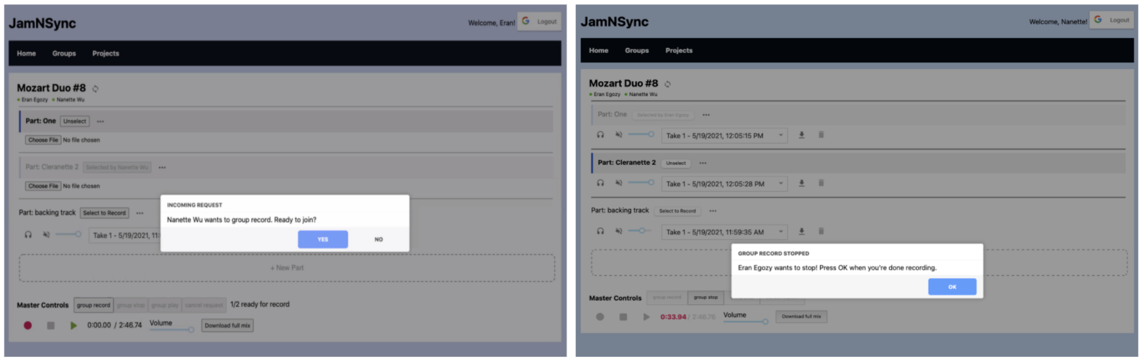
GROUP REHEARSAL STOPPED

Eran Egozy stopped!

OK

Figure 4-13: Group play modals. The left image shows the request for playback that is broadcast from the server, after a client initiates the group play. The right image shows the alert for an immediate stop, after a client initiates the group stop.

To provide the illusion of group rehearsal, a user can make a recording to a backing mix: a subset of project parts played simultaneously as the user records. After recording to a backing mix, an audio alignment tool pops up immediately when the recording stops (Figure 4-15). Due to variable latency in the recording infrastructure


INCOMING REQUEST

Nanette Wu wants to group record. Ready to join?

GROUP RECORD STOPPED

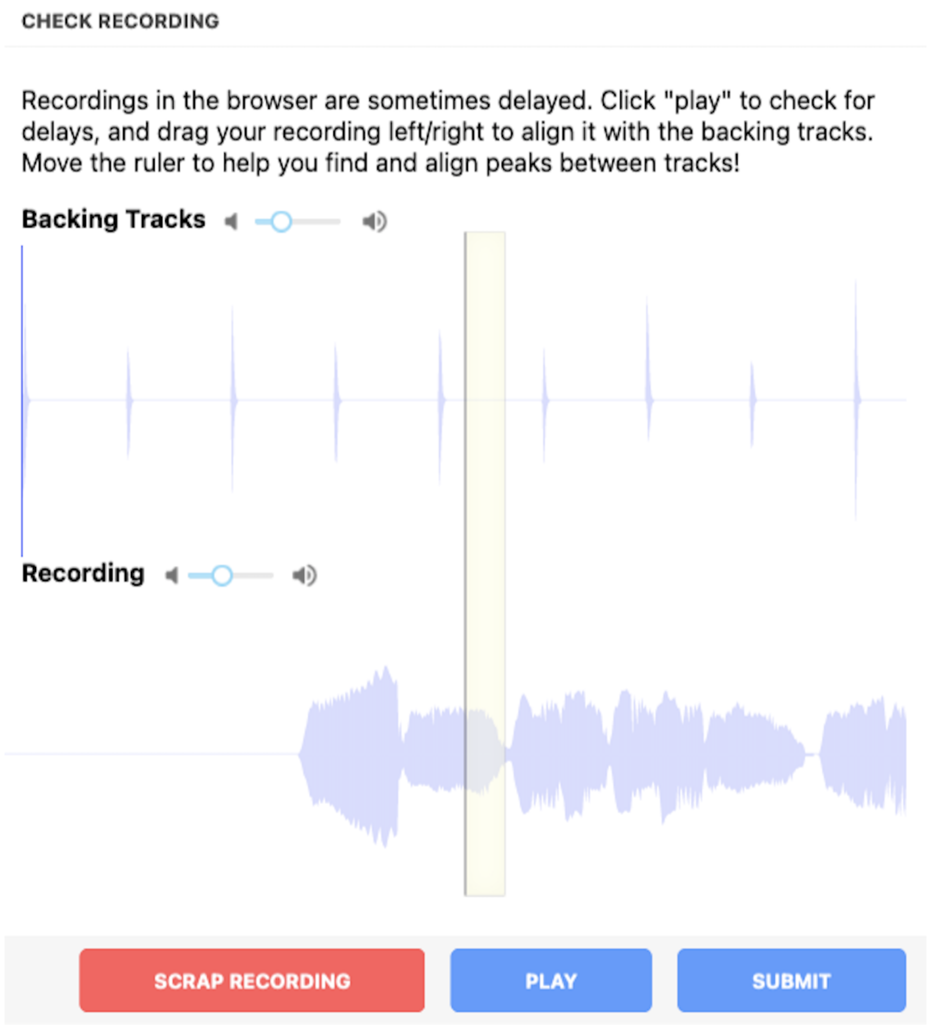
Eran Egozy wants to stop! Press OK when you're done recording.

Figure 4-14: Group record modals. The left image shows the request for recording that is broadcast from the server, after a client initiates the group record. The right image shows the alert for an eventual stop, in which the recording only stops after the client acknowledges the pop-up.

(i.e. browser, audio devices, application logic), it is difficult to predict a single latency value to uniformly adjust all recordings. Instead, a visual tool assists users in properly aligning their recording.

The alignment tool plays the first seven seconds of the recording and a backing mix. The top of the tool provides the reasoning for the tool and a brief overview of its suggested usage. The top waveform visualizes the backing mix of all parts that a user listened to while recording (i.e. unmuted, non-selected). The bottom waveform visualizes the user's recorded track. Each waveform has a volume slider to adjust the balance between tracks while aligning. The pale yellow rectangle with the bolded left edge functions as a visual alignment guide to help align peaks in the waveforms.

To perform the alignment, the user drags the recording waveform left and right. A pixel-to-milliseconds conversion determines the amount of latency adjustment per drag. Independent of the waveform dragging, the yellow rectangle can also be dragged left and right to help with the visual alignment. The cursor indicates which parts are interactive in the alignment tool (shows "not-allowed" cursor when hovering over the backing mix, "ew-resize" cursor when hovering over the recording track, "grab" cursor when hovering over the visual alignment guide). If the user is unable to properly align their recording the first time, they are able to click the three-dots button on their part to realign their take, which pops up this alignment tool once again.


CHECK RECORDING

Recordings in the browser are sometimes delayed. Click "play" to check for delays, and drag your recording left/right to align it with the backing tracks. Move the ruler to help you find and align peaks between tracks!

Figure 4-15: Audio alignment tool.

The bottom of the tool has buttons to scrap the recording, toggle play/stop of the excerpt, and submit the alignment to finalize the take.

4.6.1 Client Usage

Figure 4-16 outlines the client usage flow. A musician accesses the application website through a browser. Using a third-party login API, the musician signs in on the home page, navigates to the "Groups" page to create or select a group, then the "Projects" page to create or select a project within a group. Selecting a project redirects them to the project-specific page, where they can choose to solo or group record/playback.

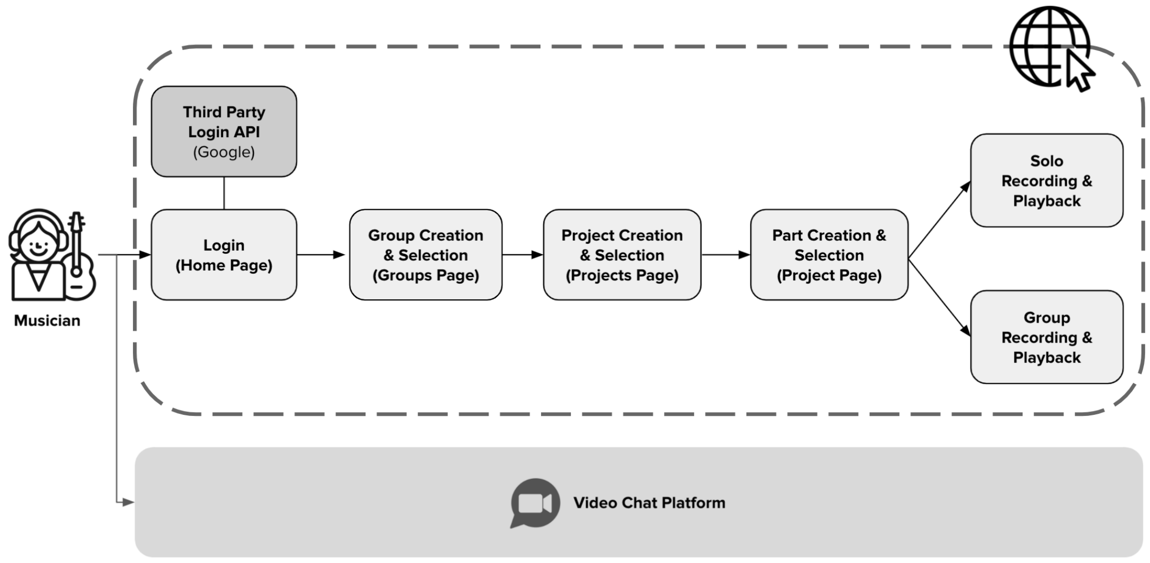


Figure 4-16: Frontend system flow.

Individual use involves creating a part, uploading and recording new takes, and listening to single-part recordings. Group use requires an additional communication tool to facilitate rehearsals. Rather than embed video or audio calls in the page, user tests suggested that people prefer existing mediums to communicate (Zoom, FaceTime), as custom video chat integrations tended to suffer in quality. External conferencing software will not affect application performance since JamNSync does not require low latency audio streaming (no competition for network bandwidth), which also keeps the website lightweight with fewer dependencies.

Once all members have connected on a video chat platform and are online on the project page, they can begin a rehearsal by creating and selecting their own parts. When a group is ready to record, they will mute their video call, record a take, align that take, then unmute the call. They can then listen back to the combined recordings together and discuss the music. If desired, they can repeat the process again. Users are also able to upload and download takes to their liking.

4.7 Backend: API Design

When the client makes requests to fetch and update project data, the server translates them into database and cloud storage queries. JamNSync's API design specifies 1) REST endpoints over HTTP for browsers to read and write to external storage services and 2) event handlers over WebSockets to support the RTR protocol.

REST Endpoints

Endpoints that manipulate entities (listed in Section 4.5 are prefixed with /api and suppors a subset of CRUD (create, read, update, delete) operations. Two additional authentication endpoints are prefixed with /auth (login, logout).

- users: GET all users
- groups: GET all groups
- group: GET/PUT/DELETE group, POST new group
- group membership: GET all members in group, POST/DELETE member
- project: GET/PUT/DELETE project, POST new project
- track: GET/PUT/DELETE track, POST new track
- take: GET/DELETE take, POST new take

Event Handlers

To handle incoming events to synchronize group record and playback, four dictionary data structures maintain state for each group session (where below, a room is equivalent to a group session):

1. client_names: maps a client connection identifier → client name
2. all_clients_by_room: maps a project hash → list of online client connection identifiers

3. prepared_play_by_room: maps a project hash $\rightarrow$ list of client connection identifiers who have requested group play (or voted yes for playback)
4. prepared_record_by_room: maps a project hash $\rightarrow$ list of client connection identifiers who have requested group record (or voted yes for recording)

The following event handlers registered on the server support the real-time rehearsal protocol and enable group record/play functionality:

- connect: add new user to client_names
- join project: add new user to group session state
- leave project: remove user from group session state, emit updates for online users/waiting clients/claimed tracks
- disconnect: remove user from all group session state, emit updates for online users/waiting clients/claimed tracks
- request group record: initiate RTR protocol for group record
- request group play: initiate RTR protocol for group play
- request group stop: initiate RTR protocol for group stop
- cancel request: remove client from prepared_<play/record>_by_room and emit update to group
- get all online users: return sorted list of client names active on the project
- catch up new member: if new member joins after a request to group record/play begins, pick any existing member in the group and ask them to relay the latest information that the server does not store (online users, who selected what track) to the new client

Another set of event handlers synchronize shared data in a group. The server emits updates to a group whenever a member makes an update to the following entities: group, project, track.

Chapter 5 Implementation

JamNSync is a single-page web application built with React.js (JavaScript, HTML, CSS) and Flask (Python), deployed using Gunicorn on Heroku. A PostgreSQL database and Amazon S3 bucket provides data storage and management. The source code used for deployment can be found at the GitHub repo below:

`https://github.com/nanettewu/jamnsync`

5.1 Frontend: React

Class components implement the four pages described in Section 4.6. To navigate between pages, React Router is implemented such that the Home page uses a built-in route, while the Groups, Projects, and Project page use custom Private routes visible after authentication. Common packages utilized in the application are listed below (see Appendix A for full list of npm packages).

- socket.io-client: synchronously updates state and supports group recording/playback
- Material UI: provides pre-built icons with tool tips to expedite development
- react-google-login: manages authentication through Google Sign-In. Uses local storage to persist login information to cache credentials.
- react-st-modal: contains pre-built pop-ups, confirmations, alerts.
- react-dropdown: allows users to select takes and add group members with drop-down menu.

- rc-slider: shows a slider for per-track and master volume adjusters.

Audio

Recording is implemented using the Web-Audio-Recorder package with 320 kbps bit rate, 44.1 kHz sample rate MP3 files. A three-second circular timer (react-countdown-circle-timer) is shown before recording to display a continuous countdown that is both visually and aurally independent of specific tempos.

Playback is implemented using embedded HTML5 audio elements. Each track contains a separate element, which is buffered for immediate playback and loaded with the user-specified take. As such, multiple elements can be played back at the same time to provide the illusion of a mixed track, but kept separate to provide fine-grained play/pause and volume control.

```
<audio
  id={‘audio-file-<trackid>‘}
  src={‘<s3-url>?cacheblock=true‘}
  preload="auto"
  autobuffer="true"
/>
```

The audio alignment tool uses wavesurfer.js to visualize audio waveforms, react-draggable to drag the recording waveform and visual alignment guide, and crunker to mix MP3 tracks into a single file. Crunker frequently used the wrong sample rate, which was automatically detected by a user's browser depending on OS and device. A modified version of crunker was used to force 44.1 kHz sample rates by deterministically setting the AudioContext's sample rate on initialization.

5.2 Backend: Flask

The audio processing backend is written in Python as a Flask service. API endpoints defined in Section 4.7 are implemented as Flask routes (routes.py), protected with login-required decorators. Pydub is used to manipulate uploaded audio files by converting sample rates, appending silence, and trimming extra spaces - enabling latency adjustments for each take. Flask-SocketIO implements event handlers (routes.py) in conjunction with socket.io-client in the frontend. Flask-Login (auth.py) provides user authentication and session maintenance. The backend is also responsible for managing data transfer (audio files, metadata, musician/ensemble descriptors). Appendix A contains a full list of Python packages used to build the Flask server.

5.3 Data: PostgreSQL/AWS

A PostgreSQL database implements the relational model defined in Section 4.5. The backend uses SQLAlchemy to interface with the database to manage CRUD operations. Entities are defined as SQLAlchemy models (models.py), and Flask-Migrate handles alteration to the database schema using Alembic. Any deletes on hierarchical relationships are chained (e.g. delete group will delete all projects and their associated tracks and takes). Database queries are performed at the group level to grab larger chunks of data at a time and avoid repeated smaller calls (i.e. any callback in the React application requests to reload everything).

A public AWS (Amazon Web Services) S3 bucket is used to store audio files. Access methods (aws.py) use awscli and boto3 to upload and download audio files with the bucket. CORS settings allow access from all origins and "?cacheblock=true" is appended to each S3 file URL is used to defeat the browser cache when retrieving files. Using a cloud service guarantees the reliability design goal through replication.

5.4 Deployment: Heroku

The application is deployed to Heroku[1] as a single Flask server. Using a managed platform such as Heroku simplifies the server setup process. Since Google's OAuth requires HTTPS, the application is deployed on a Heroku hobby web dyno, which

[1]At the time of deployment, the application was located at `https://jamnsync.herokuapp.com/`. The website was taken down after user testing.

provides automated SSL certificate management to enable HTTPS. A Heroku add-on (Heroku Postgres) hosts the database server. To start the web server on command, a single command in the Procfile is needed. Using Gunicorn as the WSGI application server and eventlet to support socket.io connections, JamNSync is run with the following command:

```
web: gunicorn --worker-class eventlet -w 1 audio_processing.app:app
```

Because Flask supports static files, a simple deployment strategy uses the Flask project to serve both React and Flask. The production-ready React bundle of `npm run build` is copied into the Flask server, which also contains built-in API endpoints. When the server initializes, Flask is directed to point at the folder containing the build and an index.html to download the entire application. Any change to the frontend code requires a new build that is recopied into the project. Although this strategy is not ideal for permanency, it is convenient and simple for research.

The Flask server also requires manually configured environment variables, including AWS keys, a Google Client ID, and the Flask server's secret key.

5.5 Technical Challenges

5.5.1 Technology Differences and Inconsistencies

Because of differences in browsers, audio devices, and operating systems, platform-specific bugs were hard to diagnose and reproduce.

The Google Chrome browser was initially used for website development but was found to have non-deterministic problems recording and playing MP3 audio. Because recording logic used the default AudioContext initialization, JamNSync was auto-detecting sample rates depending on a client's browser environment, which caused differences in the recording infrastructure among users. As such, MP3 files inconsistently sped up or slowed down, and often contained stuttering interference. During playback, project tracks were periodically out of alignment without clear reason, perhaps due to heavy CPU or memory usage. Since these problems occurred arbitrarily, it was near impossible to pinpoint the cause. Pivoting to Firefox provided more consistency, but the reasoning behind the improved stability is unclear.

Varying grades of equipment (Apple AirPods, studio microphones) also produced recordings of distinctly different quality. Different devices also produced a range of audio sample rates (16 kHz, 44.1 kHz, 48 kHz), causing misalignment in both mixing and playing tracks.

Differences in operating systems also caused problems that were only uncovered in user testing. Although system tests passed in local development, they revealed issues with warped audio, user interface sizing, and concurrency when the app ran on different machines. Even after finding a bug, it could not be reproduced locally without the user who discovered it. Moreover, fixing bugs for one operating system sometimes introduced new bugs in others. With limited development and testing resources on this project, it was difficult to guarantee a consistent experience across multiple environments.

5.5.2 MP3 Web Recording Logic

There are several ways to record and encode MP3 files in the browser. However, it was surprisingly difficult to find a package that offered consistently high audio quality. Packages that were easy to integrate (e.g. pre-built React components) ended up being hard to customize without a thorough understanding of their implementation. Some also had stale dependencies on outdated browser versions, which were confusing to update and replace. Creating implementations from scratch allowed complete control over customization and development, but the process was incredibly tedious with a steep learning curve. Since "do-it-yourself" tools are usually not rigorously tested, they also often break down in edge case usages. The final solution uses WebAudioRecorder.js, an open-source recording library using the Web Audio API, which strikes a balance between convenience and customization.

5.5.3 Aligning Tracks with Browser Latency

Recording audio in a web browser introduces variable latency, which pads a non-deterministic amount of silence at the beginning of a recorded track. If a user played their raw recording at the same time as the backing mix, they would notice their recording consistently lagged. To compensate for variable latency, tracks in the same project need to be aligned to start playing at the same time. This requires finding the numerical latency offset for a user's choice of computer, browser, and audio devices.

The original plan was to automate the alignment process, possibly using a manually tuned value slider (picks a fixed value) or a user calibration system (determines latency value by measuring once before recording) - all to get as close as possible to the actual latency value. However, even with tuning an estimated latency value, latency was not consistent between takes. There was some combination of potential lag from the browser, React application, and audio devices that caused different takes to always be out of sync without clear reason. As such, the solution to automate alignment was rendered infeasible.

Therefore, it was necessary to perform a take-specific alignment in a simple and user-friendly manner. Soundtrap uses a look-up table that maps a client setup (i.e. browser and operating system choice) to its measured latency value, but collecting sufficient data for such a table was out of scope for this thesis. Instead, the visual alignment tool from Upbeat was the inspiration for manually adjusting latency per take [32]. The frontend logic determines the latency value, which is sent to the server to perform audio manipulation using pydub.

Chapter 6 Testing and Evaluation

We tested application features iteratively in three rounds. Round 1 consisted of solo (1 musician) tests, which used formative evaluation to qualitatively assess the usability of browser recording and playback. Round 2 consisted of duet (2 musicians) tests, which evaluated concurrency within the smallest distributed network of clients, who use different recording software and hardware. Round 3 consisted of duet, trio, and quartet (2-4 musicians) tests, which used varying ensemble sizes to gauge the efficacy of real-time collaborative rehearsals. In Round 2 and 3, we used both formative and summative evaluation to determine the overall usability of the system.

We recruited participants who are representative users of the application: MIT undergraduate, graduate, faculty, and alumni musicians covering a spectrum of music technology expertise. All participants had formal music training and experience with creating audio recordings. The final system test involved professional musicians and MIT chamber music students, who actively used Zoom, SoundJack, BandLab, and Audacity/Google Drive to make music during the pandemic. These four tools cover the four categories of virtual rehearsal tools discussed in the related works (Chapter 3) respectively.

During testing, all participants accessed the web application using Firefox on their personal laptops (both Mac and PC). They also used their own recording equipment, ranging from wired professional audio devices to wireless Apple AirPods. We intentionally did not enforce a standardized set of headphones and microphones in order to expose our system to different recording environments. This allowed us to better understand how different grades of audio devices can impact the user experience.

6.1 Round 1: Solo (1 Musician)

In the first round of testing, our main objective was to assess the usability of the web browser as a recording interface. By understanding if the proposed system was a viable method of recording and playback for a single user, unexpected problems with the basic interface could be isolated from more complex issues involving multiple users, such as race conditions and data inconsistency.

We recruited five participants from the MIT EECS community. All users were graduate students or upperclassmen with formal music training (defined by taking private music lessons in childhood) and extensive technology experience (defined by academic experience with web interfaces). Each participant used their built-in computer microphones and speakers in testing. Two participants actively rehearsed in MIT's chamber music ensembles during the pandemic and were familiar with virtual rehearsal software described in the related works (Section 3). With the participants' joint music and technology experience, we anticipated finding bugs more efficiently and receiving actionable suggestions on the overall web design. Our small testing pool size was chosen to quickly identify and solve common issues. Over Zoom, we conducted each test as a 30-minute session, where the participant had access to a video stream and screen-sharing capabilities. We provided them a link to the website hosting the application. Once the participant navigated to the homepage of the website, we asked them to share their screen.

6.1.1 Tasks

Each participant was given two tasks. They were asked to verbalize their thought process as they completed each task.

Task 1: *Record and playback an audio track*

Given minimal instruction, users were tasked with figuring out how to navigate through the application pages. To do so, they needed to log in through the upper-right hand button, navigate to the groups page (Fig. 6-1), create a new group and project, navigate to the project's DAW page (Fig. 6-2), create a new track, then record and playback. If they succeeded in completing the task, we asked them to download and email their recording to us for further review.

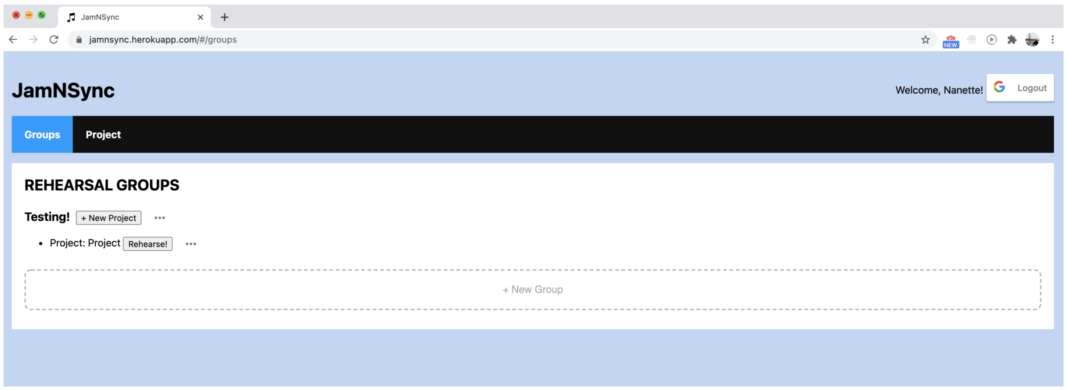


Figure 6-1: Interface of groups listing page for individual test.

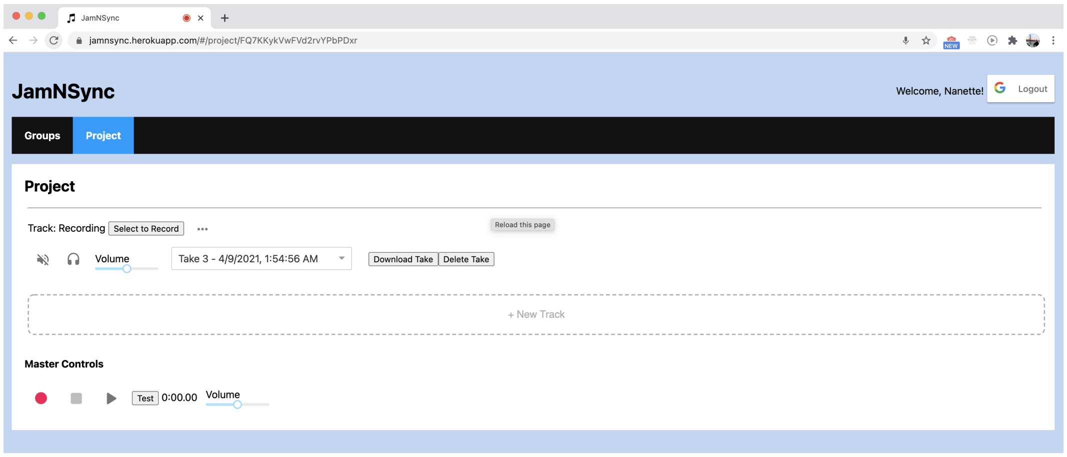
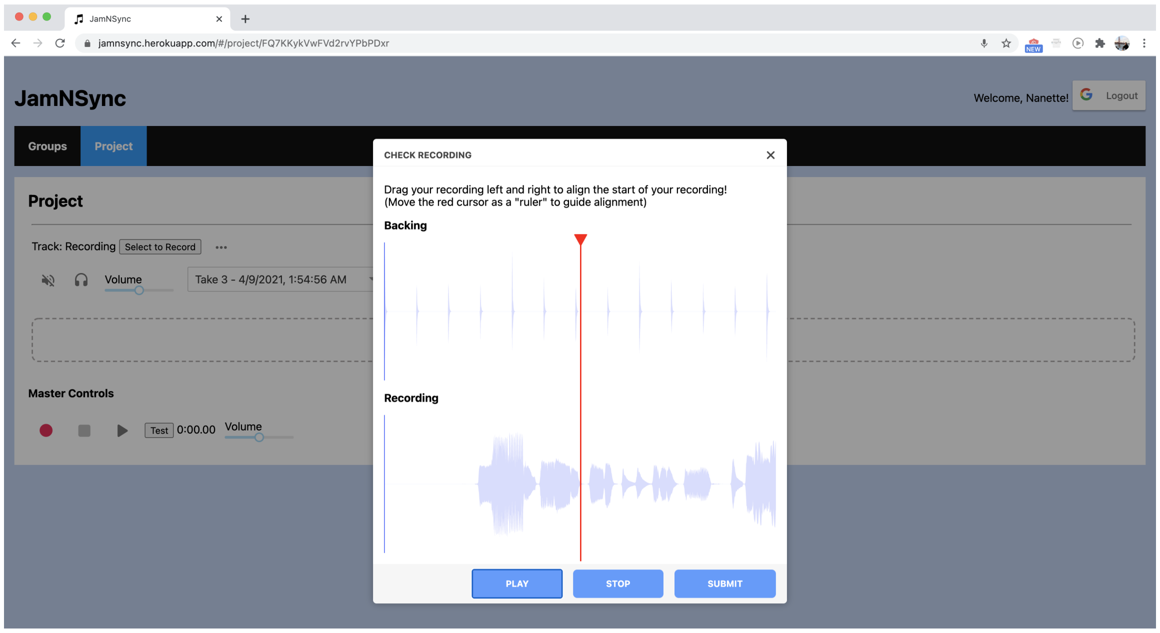


Figure 6-2: Interface of digital audio workstation (DAW) page for individual test.

Task 2: *Figure out how the audio alignment tool works.*

Users were instructed to click the "Test" button on the DAW page and experiment with an initial version of the audio alignment tool (Fig. 6-3). Pre-existing track were provided: a 104 BPM click track as the backing track and a misaligned clarinet recording as the recording track. This task required users to read the instructions at the top of the pop-up modal, click the "Play" button to find that the tracks were out-of-alignment, and drag the recording to align it with the backing track.



Drag your recording left and right to align the start of your recording!
(Move the red cursor as a "ruler" to guide alignment)

Figure 6-3: Interface of audio alignment test for individual test.

After both tasks were attempted, we conducted a post-test interview to determine: 1) which parts of the system were self-explanatory, 2) which parts of the system were hard to use and unintuitive, 3) recommendations for improving the system, and 4) how a multi-user based interface might look.

6.1.2 Results

All five users completed the first task in under one minute. A summary of the user descriptions and feedback is outlined in the table below. Only one user completed the second task without extra guidance and intervention.

Task 1: *Record and playback an audio track*

All users were able to complete the task, by navigating from the home page to the DAW interface, in under a minute. They reported that the UI generally straightforward and intuitive, particularly with the recording controls.

When users played back their audio files, a common observation was the poor audio quality. We immediately noticed that this negatively impacted their testing experience based on their facial reactions when listening to recordings. Upon closer investigation, the audio recording packages used in this design had low bit rate (128

Table 6.1: Summary of Task 1 Results for Round 1

User	Instrument	Chamber Musician	Reported Audio Quality	Recording Length (sec)	OS
1	Guitar	No	3/10	10	Windows
2	Piano	Yes	4/10	10	Mac
3	Cello	Yes	4/10	30	Mac
4	Voice	No	6/10	8	Mac
5	Ukulele	No	6/10	15	Mac

Each participant used their own instrument, laptop, and recording devices. A participant who identified as a chamber musician was actively participating in the MIT Chamber Music Society. The reported quality was provided verbally at the end of a testing session. Participants were not given any constraints on recording length.

Kbps) and buggy recording and encoding logic. Using Google Chrome as the browser of choice also tended to produce interference in output recordings (i.e. clicks, blips, bursts of repeated sound). We asked users to test their audio recordings with an offline recording software, which resulted in high quality recordings and verified that their audio devices were not the source of the issue. To diagnose audio quality in future tests, we added comprehensive console logging in the application's recording logic and kept record of each participant's browser choice.

Furthermore, many users found it hard to find their bearings to get started. Although the learning time was only five minutes, the lack of text-based instructions and labels on certain controls made unlabeled components unintuitive. Some were confused about what the "play" and "stop" buttons on the bottom of the interface controlled. Many people were looking for a button to play each track, when the interface should have stated that the master controls were applied to all tracks.

Users without chamber music experience had difficulty in understanding the relationship between a "group" and a "project." They were confused about why the logical relationship of a "group" was necessary to rehearse music. On the other hand, users with extensive chamber music experience, specifically in the form of virtual rehearsals, expected common functionalities from other DAW interfaces like track seeking and waveform visualization.

Task 2: *Figure out how the audio alignment tool works.*

Four out of five users did not figure out how to use the audio alignment tool. While all users appreciated the visual waveforms and the horizontal dragging for alignment, most did not understand what the red cursor was for. The fact that the cursor could be dragged was not immediately obvious, nor was its intended purpose to guide alignment. Even though instructions were written at the top of the modal to inform this process, the cursor-specific instructions were ignored because they were written in parentheses. Furthermore, the play/stop buttons on the bottom of the modal were unintuitive - users wanted play/pause to have more fine-grained control.

Because the audio alignment tool was necessary for correctness and completeness, we prioritized improving the usability of this feature. Instructions on usage needed to be clarified for users to learn its functionality on first use, and the track alignment process needed to be simple enough for users to want to reuse in future rehearsals.

When we asked each participant about how they might imagine a multi-user interface, three users preferred using existing platforms like Zoom and Facebook Messenger to communicate. All five users wanted to have collaborative editing, such as a log of user activity (automatically updated or manually inputted) or a Google Docs-like cursor indicating each user's status in the DAW. User 3, who was most familiar with audio editing, pointed out that concurrent usage of a playback system needed to be extremely precise; being milliseconds off in an audio track can significantly hamper the playback experience.

Discussion

In general, users had high expectations for website speed and ease of use. While participants tolerated inconveniences, the testing sessions showed that they implicitly held standards from websites they access regularly. For example, users expected to be able to click "enter" to select "OK" on pop-up modals, even though there was no instruction indicating that was possible. Users unfamiliar with audio editing also thought that playback was possible per track, resembling a feature found in the interface of Spotify playlists. However, their expectation of clicking a single track for immediate playback did not fit the intended usage of JamNSync, nor the correct usage of a DAW in general.

Issues were prioritized based on the order of importance of design goals:

- Usability for Musicians: fix UI bugs identified in user tests, include more text descriptions, add more clarity and features to audio alignment tool, add loading icons to offer feedback when waiting for request responses
- High Audio Quality: use higher bit rate, experiment with other recording packages, try different browsers, test various audio file formats
- Completeness: consider adding missing features (e.g. visual waveform, track seeking)

6.2 Round 2: Duets (2 Musicians)

In the second round, ten participants tested concurrent usage in a variety of environments (different laptops and recording devices). They represented a wide range of instrument types, years of study, style of music, and previous experience with DAWs (see Table 6.2 for details). Figure 6-4 shows the breakdown of audio hardware used in this round.

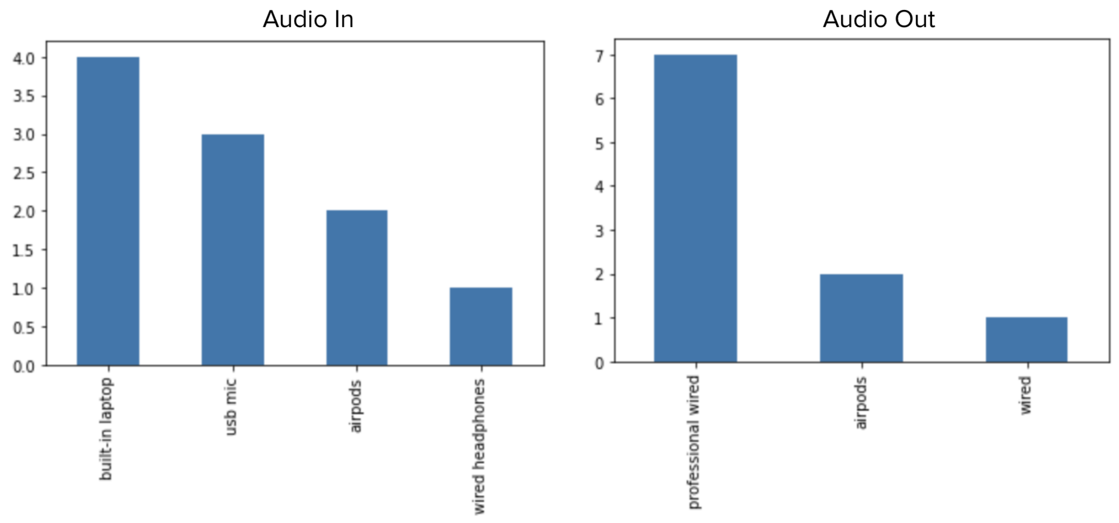
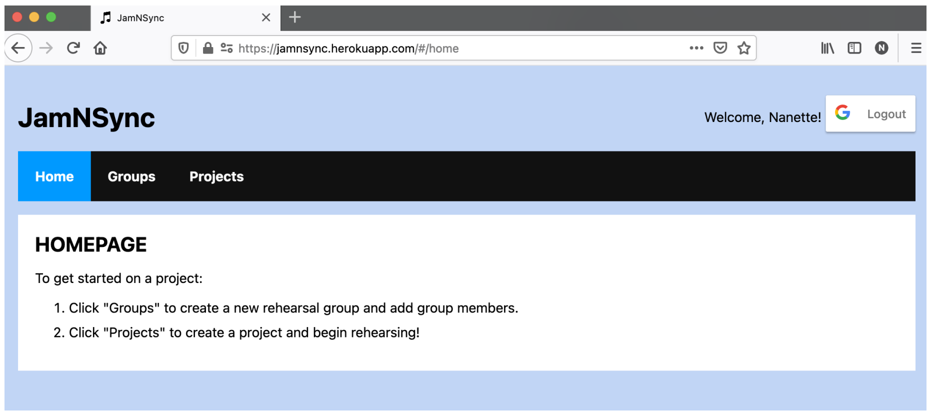


Figure 6-4: Left: number of users who used each type of audio input device. Right: number of users who used each type of audio output device.

Two types of duet tests were performed. The majority were duets between a participant and the user test facilitator, which were easier to schedule and manage. Two tests involved duets between participants. In both cases, we conducted each test in a 1-2 hour Zoom session, where users set up their recording environment, attempted three tasks, and filled out a Google Form to provide feedback on their experience.

First, we asked each user to set up their environment by: 1) downloading the latest version of Firefox and 2) finding headphones to isolate the audio input and output streams. Due to non-deterministic issues in audio quality from Chrome, we decided to uniformly use Firefox during user tests. We then provided them a link to the website hosting the application. After the participant navigated to the home page and logged in (Fig. 6-5), we asked them to share their screen.



HOMEPAGE

To get started on a project:

1. Click "Groups" to create a new rehearsal group and add group members.
2. Click "Projects" to create a project and begin rehearsing!

Figure 6-5: Interface of home page for logged-in user for duet test.

Before we assigned the user test tasks, we gave each user an overview of what JamNSync aims to achieve, but did not offer a formal walk-through of the application. We said this at the beginning of each test in Round 2 and Round 3:

> "Due to the pandemic, we cannot rehearse music together in-person. A handful of existing technologies try to enable people to rehearse together despite being separated, but they're often complicated or require special audio equipment. My thesis aims to get around this by creating a web app in the browser, which is easy to access and use for musicians."

Table 6.2: Summary of Round 2 Users

User	Instrument	Private Lessons (yrs)	Small Group (yrs)	Record. Exp. (1-7)	DAW Exp. (1-7)	Piece Name
1*	Ukulele	8+	2-4	2	2	Hey, Soul Sister
2	Piano	<1	<1	2	3	Disney Medley
3	Guitar	<1	1-2	6	6	Gymnopedie No. 1
4	Clarinet	4-8	1-2	4	4	Mozart Duet
5	Clarinet	4-8	2-4	4	4	Krommer Duet
6	Voice	8+	8+	6	7	Goldberg Variations
7*†	Cello	8+	8+	2	3	Brahms Sonata
8*†	Piano	8+	2-4	4	4	Brahms Sonata
9	Voice	8+	4-8	6	6	Not What I Meant
10	Voice	8+	4-8	6	6	Not What I Meant

* indicates user who participated in Round 1 of testing. † indicates user who used wireless Apple AirPods. The mean rating for recording experience was 4.2 (stdev=1.75), and the mean rating for DAW familiarity was 4.5 (stdev=1.65).

6.2.1 Tasks

Three successively harder tasks were assigned to each duet, culminating in group rehearsal. Participants were encouraged to think out loud, ask questions, and provide real-time verbal feedback about what was working well or what was unclear.

Task 1: *Record and playback an audio track.*

The first task remained the same from Round 1: create a group, create a project, then record and playback a track. Based on user feedback from earlier tests, we found that a simple, unguided task helped users become familiar with the application. If they were able to finish the task, the user was prompted to download their first take, later uploaded to the Google Form questionnaire.

Task 2: *Record, align, and playback an audio track using a backing track.*

For the second task, we asked participants to create and upload a backing track to the DAW interface (Fig. 6-6, left). Duets with two participants were asked to take turns attempting this task and provide commentary while observing their partner. No further guidance was provided after they uploaded the track.

Once the participant finished recording, they were asked to articulate their thought

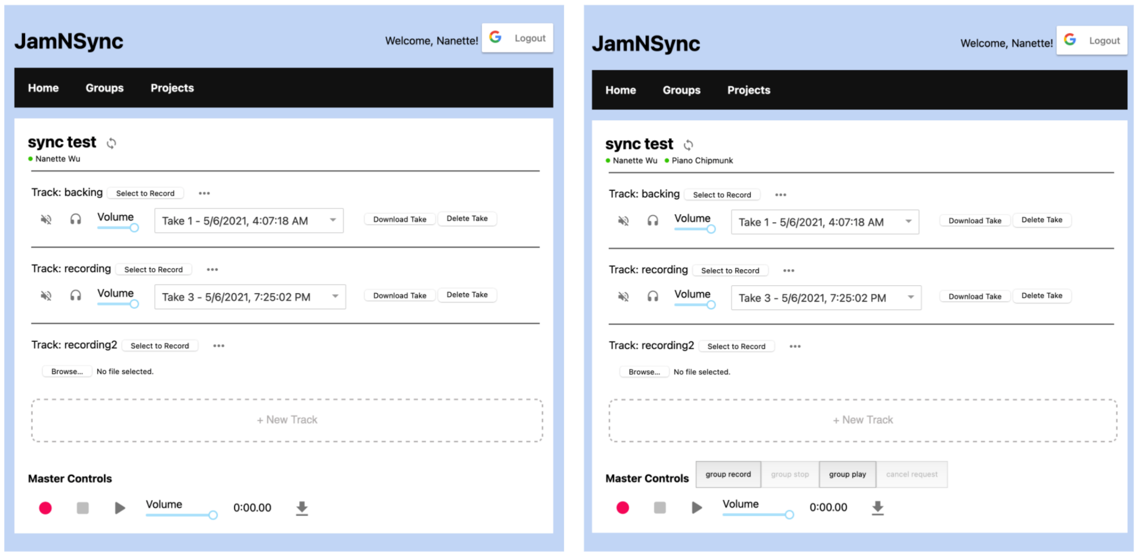


Figure 6-6: Interface of DAW page with one active user (left) and two active users (right).

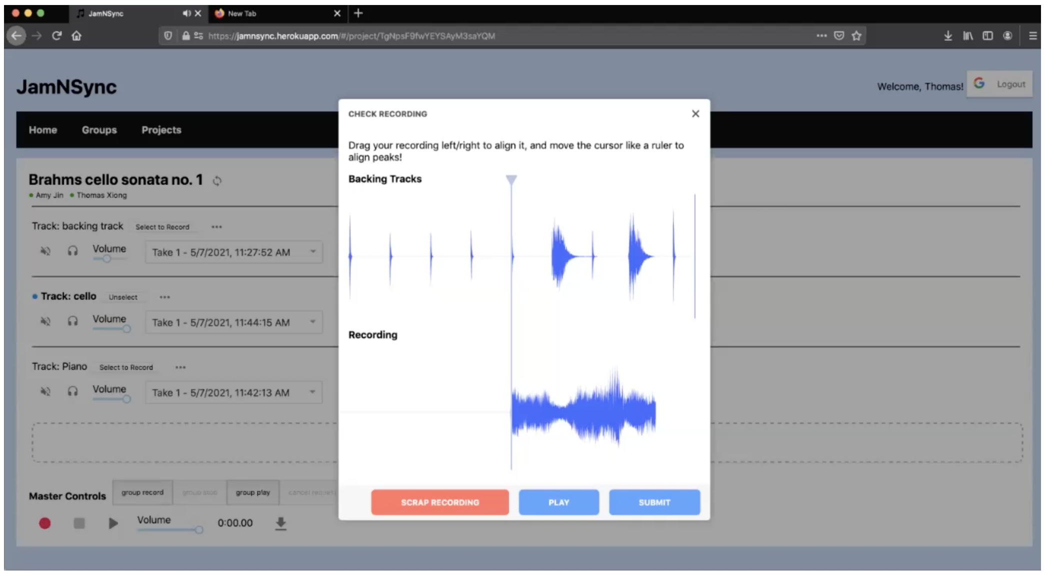
process as they used the alignment tool to compensate for the delay from system latency. The interface of the alignment tool at the time of the test is shown in Figure 6-7. Once the tracks were aligned, users were asked to review and playback the track.

Task 3: *Rehearse a piece with a duet partner.*

Unlike the first two tasks, task 3 was completely guided by the facilitator to make sure both partners were in sync and completed the same process. They were instructed to simultaneously open their project's DAW page (Fig. 6-6, right), and take note of the active users displayed in the heading and the new group buttons on the bottom of the page.

First, the duet was asked to playback ("group play") the existing backing track to acquaint themselves with the functionality of the group buttons. Then, they were instructed to click "group record" and "group stop" (Fig. 6-8), which prompted them to record, align, and upload their track. A few more cycles of group playback and recording were requested, sometimes without the backing track.

After all tasks were attempted, we conducted a short post-test interview to gauge interest in future features, including track seeking and embedded video call. Each user was also asked to fill out a Google Form questionnaire (Appendix B).



Drag your recording left/right to align it, and move the cursor like a ruler to align peaks!

Figure 6-7: Interface of alignment tool to line up a recording to a backing track.

6.2.2 Results

Table 6.3: Summary of Round 2 Google Form Ratings

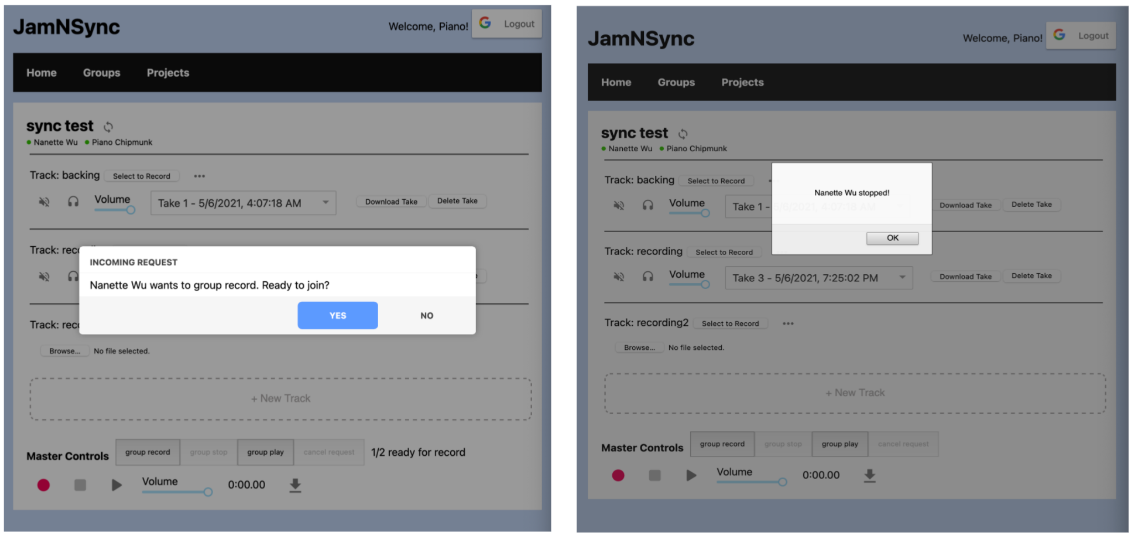
Question	1	2	3	4	5	6	7	Avg	Std
Prior experience making recordings	0	3	0	3	0	4	0	4.2	1.75
DAW familiarity	0	1	2	3	0	3	1	4.5	1.65
Rating: application navigation	0	0	1	0	1	2	6	6.2	1.32
Rating: group creation and project set-up	0	0	0	0	2	4	4	6.2	0.79
Rating: group features	0	0	1	0	4	5	0	5.3	0.95
Rating: alignment tool ease-of-use	0	0	1	2	1	3	3	5.5	1.43
Rating: audio quality	1	0	2	1	3	1	2	4.6	1.90

Each row represents one question, and each table entry indicates the number of participants who responded with the column label for that row's question (1 to 7).

Task 1: *Record and playback an audio track.*

As expected from Round 1 of user testing, all users completed this task in under two minutes. Each recording was under a minute, and they were later downloaded for us to review audio quality.

In terms of what participants liked, users found that application navigation was easy and intuitive: it was clear on the homepage which tabs were clickable and which weren't. Group member creation was also straightforward. The countdown to record



INCOMING REQUEST

Nanette Wu wants to group record. Ready to join?

Nanette Wu stopped!

Figure 6-8: Left: interface of pop-up to request recording for the entire group. A similar interface exists for playback. Right: interface of pop-up alert to notify the user that a group member has stopped recording or playback.

for three seconds was also well-received, as it gave sufficient time even for a pianist to go up to initiate the request and go back to their piano.

In terms of what was frustrating, users found some recordings too loud. They wished they had visual feedback for the volume of their audio stream to understand when they were clicking and needed to adjust sound levels. People were also unwilling to fully read instructions with actionable words (i.e. "Click 'Groups'") since they assume immediate understanding, which caused some confusion during initial set up. There were also some issues regarding the user interface. Some users inquired about "missing" features that they were familiar with in other recording interfaces, like a waveform visualization, clearer (i.e. green) play button, and more text descriptions for controls. Other users were confused by misleading UI: they were confused that the application allowed them to click play even though there were no recordings to play, and they weren't sure which track was actually selected (i.e. looking for a bolder background color and active hover state).

There were two areas of mixed feedback. The first was regarding audio quality, which had improved as noted by participants who also tested in Round 1. However, since users provided their own audio devices, the perceived audio quality ratings were quite varied as shown in Figure 6-9. Users were very aware of their microphone's expected audio quality. Some thought the quality was "accurate for what I expected my laptop to be [and] surpassed my expectations of what I thought the app would be", or that it was "surprisingly nice for my built-in mic and recording through the browser". Others thought "the audio quality was bad, with loud and soft spots", but one user stated it was "certainly because [they] used AirPods, which have terrible mics". The most prominent issue was periodic clicks in the recordings, described as "peaking", "popping", and "crackling". There was also a "tsh" sound at the beginning of some notes, and a high-pitched overtone with others, but these issues occurred inconsistently and did not hamper the overall experience. When we asked some users to test recording in an offline platform, the noisy interference was eliminated, indicating issues with the browser recording or audio processing logic.

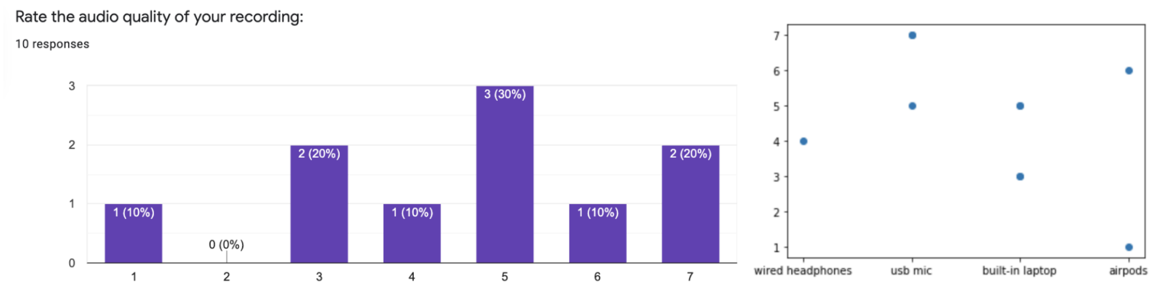


Figure 6-9: Left: ratings of perceived audio quality (mean=4.6, stdev=1.9). Right: audio input type vs. perceived audio quality ratings.

The second area of mixed feedback was for the recording experience. Users found that it was overall smooth, easy to manage, and intuitive; recording on JamNSync was "not more difficult than Audacity or any recording software." However, users were bothered by small inconveniences from hidden failures, such as hanging browser requests for microphone permissions that, if not accepted, still permitted a recording session that ultimately failed due to blocked permissions. Some users also found it troublesome that they could not tell whether they were clipping. The most common issue occurred when they had to select a track before recording. Some thought it was confusing to see an error message (i.e. "Select a track to record!") when there was only one track to record to. We thought this design would teach them how to select a track when the interface was still simple (i.e. before multiple users signed on), and they were able to learn the track selection after the initial hiccup.

An interesting observation was that users tended to only look at their sheet music or microphone when recording. Although they asked for more visual feedback, it would only be necessary at the beginning and end of their recording session. As such, we focused on highlighting key aspects of the interface used right before and after recording and playback (like a bolder color for elapsed time).

Task 2: *Record, align, and playback an audio track using a backing track.*
Nine out of ten users were able to successfully complete this task. The user who was unable to complete the task encountered a bug in which the alignment tool incorrectly played the backing tracks at the wrong playback speed. This bug was fixed prior to subsequent user tests.

Users appreciated being able to version takes and found that the tool was easy to learn: "if I used this a couple of times, I would've understood how to do it well". However, file upload speed varied among users. Faster internet connections uploaded files in less than three seconds, while slower connections took over 15 seconds. Nonetheless, this was more of an annoyance than a roadblock in completing the task.

Most of the feedback from this task came from using the audio alignment tool. Participants who were familiar with audio editing and annotation tools (e.g. Sonic Visualizer) liked how silence was automatically inserted and clipped (rather than having to manually do so on other platforms). For experienced DAW users, the tool was intuitive, and no instruction was necessary for them to figure out how to drag the cursor and align the recorded track (i.e. "Oh, that's totally off, let me fix this"). They also found dragging a loopable, visual waveform to be more precise than using a tool like Audacity, which was more of a "guessing game" to align tracks.

However, the instructions for the tool were too vague: the text at the top of the tool lacked clarity about why the tool was necessary. Nine out of ten users just skimmed the alignment tool instructions. Some participants thought that they had played perfectly with the track, resulting in hesitation from the user about what they needed to fix. As such, the alignment tool caused confusion about the source of the misalignment - from playing incorrectly or technology latency. Furthermore, using the wording of "align peaks" seemed useful in some contexts, but not all (i.e. a singer performing an aria which is smooth and "peak-less").

The audio alignment tool also had issues with usability. Participants wanted to drag both tracks, but there was nothing visually indicating that they could not. They also wanted to realign the tracks, but the alignments were permanent and users had no way of doing so. The backing track was also too loud at times compared to the recorded track, and there was no way to adjust the volume of the tracks.

Task 3: *Rehearse a piece with a duet partner.*
All nine users that attempted this task was able to record pieces, but only seven out of nine users were able to playback properly due to the incorrect playback bug in the alignment tool. To complete the task, participants had the intuition to mute themselves on Zoom to record together, but this may not be the case of musicians who are not familiar with video conferencing. Figure 6-10 shows a screenshot of one participant's desktop that integrates JamNSync, Zoom, and their sheet music.

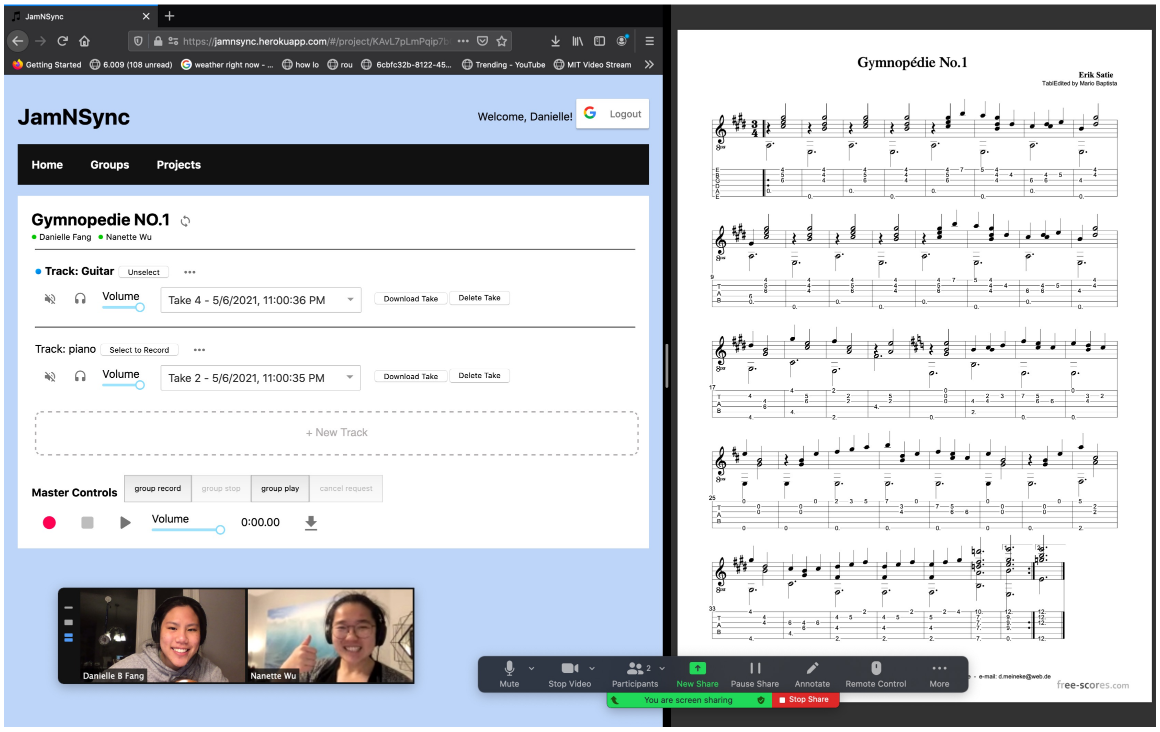


Figure 6-10: Example usage of JamNSync with split screen sheet music and Zoom running in the background.

Participants liked how they were able to quickly iterate on alignments and submit takes, which allowed them to spend less time discussing recording issues and more time discussing the actual music. By having multiple parts, tracks, and projects in one place, users noted that it was easy to navigate between different components of a DAW. Since each group member in a group was able to align their own part to a common backing track, the burden of one person needing to align and mix all tracks was eliminated. When multiple users were on the DAW page, they also liked the active status of other users to see who was present on the website: "the green circle is the universal sign of online". One participant found it helpful to discuss what they were hearing together in real time (i.e. "You're rushing at bar 20, maybe you should take more time at the upbeat"). Another participant stated that their favorite part of using the tool was the experience of playing with others. During a group recording session, they appreciated the feeling of playing live with their partner, even though they were actually playing to a recent recording.

There were a few issues with the user interface. During a group recording session, three out of ten users clicked the "stop" icon that was meant for solo stop rather than group stop. This caused future iterations of group rehearsal to function incorrectly since their duet partner did not know that they had stopped. There was also no feedback for updated requests on group play and group record, which prevented the non-initiating member of the group session from knowing the most recent group status (i.e. knowing 1/3 ready became 2/3 ready). Finally, one user mistakenly created three separate tracks in anticipation of needing one track for each take he wanted to record. In this context, the word choice of "track" was ambiguous and did not clearly suggest that one group member should be assigned one track.

During the rehearsal process, participants who frequently played with each other missed seeing visual cues from their duet partner to transition between sections of a piece. Without the cue of a conductor or an up-bow of a musician, it was difficult for users to know exactly when to begin playing if there was no click track that led up to the beginning of the recording. Some users had to think of clever ways to cue each other over an audio stream, like a distinctly audible breath that each group member takes to cue in the beginning.

When a user listens to other instrument parts while recording, they tend to match the stylistic choices of what they hear. However, there is the potential to overcompensate and over-correct to previously recorded takes, which is problematic if they contain misleading mistakes. For example, if user A and B record a duet, and user A accidentally rushes in take one, then user B is likely to rush in take two as B listens to A while recording. As much as user B may try to ignore the tempo fluctuations, it is difficult to fight against what is heard.

During this task, participants continued to have issues with the audio alignment tool, including incorrect playback speed. Users also wanted to see a longer playback of excerpts: matching the tempo and style of the group is especially difficult at the beginning of a piece, suggesting that track alignment was less reliable when only given the first seven seconds of playback.

The alignment process was more challenging when the recording did not have evenly spaced out distinct peaks, as a click track does. One user noted that while other alignment in other audio editing cases (e.g. subtitling) were more lenient with being "off by tenths of a second", alignment in music recordings required stricter precision. Letting each person do their alignment can become time-consuming if any single group member is slow. Furthermore, the visual alignment guide was still misleading because it resembled a playback cursor in other DAWs.

When asked about potential new features, users agreed that seeking was highly desired, while embedded video conferencing was not. Some even thought that the lack of seeking was a deal-breaker to convince people to use the tool, because "you don't want to re-record an entire track if you messed up once". If implemented during playback, seeking would also actually allow a group to "rehearse" together by reviewing and investigating certain spots in their piece. This issue was more prevalent for experienced DAW users.

On the other hand, video calling within the website did not seem necessary. While an external communication channel is needed to use JamNSync (e.g. to decide when to record, discuss feedback), setting up and meeting a Zoom call was "easy enough to coordinate". Even for duets with two participants, it was easy enough to coordinate a Zoom call among themselves. Some thought that a chat feature would be a nice "watered down version of communication, especially if a computer is slow and may not be able to run intense apps like Zoom at the same time." In any case, one user observed that it was important to not "force people" to be on a video call, or any communication medium, by default, since most users already have a preferred medium of communication. Since watching a video stream is typically out-of-sync and would be distracting to watch during a recording session, an audio call may also be sufficient in place of a video call.

Discussion

Notable observations from testing include:

- JamNSync features seems familiar to other music software interfaces. Participants often compared JamNSync to existing tools like Garageband (e.g. JamNSync's audio alignment waveform dragging: "when you see a track with a waveform in Garageband, you know you can control it by dragging") and Audacity (e.g. audio controls: "the mute and solo buttons reminded me of a DAW").

- JamNSync is compatible with any audio recording device including Bluetooth headphones. The correctness of most recording software depends on wired headphones due to undesirable lag from wireless connections. After each recording, using the audio alignment tool to compensate for non-deterministic browser latency comes with the added benefit of overcoming varying latencies between audio devices. Recording also does not require device-specific configurations, which can be a hassle in other platforms.

- Aside from seeking, JamNSync appears to cover the minimum set of audio control features expected from a DAW. In fact, one user said that "[JamNSync] seemed more like QuickTime rather than a DAW, which was nice because there wasn't an overload of features to keep working on."

- The protocol for group play and group record is intuitive and representative of in-person rehearsals.

We implemented the following modifications based on Round 2 results. Due to time constraints in implementation development, other improvements were saved for future work (discussed in Chapter 7).

- Changing "track" to "part": use "part" instead, which is more user-friendly for musicians with less recording experience. "Part" also has clearer implications of being associated with a single group member.
- Audio alignment tool overhaul: support track realignment, address sample rate mismatch when mixing backing tracks, redesign visual guide UI, elaborate on reasoning for tool in instructions.
- UI and visual feedback improvements: enlarge buttons, unify component sizes (i.e. buttons should have the same thickness), display "1/2 ready" for all users during group record, show incoming updates from a group member (i.e. who selected what track), use a "pointer" mouse cursor when things are clickable, and add clear visual indication of active recording. With any new feature, simplicity must be maintained while improving usability.
- Approach on instructions: reduce number of text blocks but increase the descriptiveness of each to add clarity for those who want to read them.

6.3 Round 3: Duets, Trio, Quartet (2-4 Musicians)

We recruited 18 participants to test usability of the final design and implementation (Chapter 4 and 5) with larger groups. As in Round 2, they represented a wide range of instrument types, years of study, style of music, and previous experience with DAWs (see Table 6.3 for details). Figure 6-11 shows the breakdown of audio hardware used.

10 duet tests were conducted between a participant and the user test facilitator (Users 1-10). An a cappella trio test involved two returning users and one new user

Table 6.4: Summary of Round 3 Users

User	Instrument	Private Lessons (yrs)	Small Group (yrs)	Record. Exp. (1-7)	DAW Exp. (1-7)	Piece Name
1	Voice	4-8	2-4	4	4	Someone Like You
2	Voice	8+	4-8	6	6	Love is an Open Door
3	Voice	8+	8+	5	4	A Spoonful of Sugar
4	Flute	8+	8+	7	4	Take Five
5	Clarinet	2-4	0	4	5	Carmen
6	Vibes	8+	1-2	5	5	Jamshied Sharifi Piece
7	Flute	8+	4-8	6	6	Four Seasons
8†	Clarinet	8+	8+	2	2	Mozart K.487 No.8
9†	Violin	8+	8+	6	6	Mozart K.487 No.1
10†	Piano	8+	8+	7	7	Brahms Sonata
11	Voice	4-8	0	2	1	Star Spangled Banner
12*	Voice	-	-	-	-	Star Spangled Banner
13*	Voice	-	-	-	-	Star Spangled Banner
14	Bassoon	8+	2-4	5	5	Peruvian Melody
15	Piano	8+	4-8	5	4	Peruvian Melody
16	Violin	8+	8+	4	4	Peruvian Melody
16*	Violin	-	-	-	-	Messiaen Quartet
17*	Cello	-	-	-	-	Messiaen Quartet
18*	Piano	-	-	-	-	Messiaen Quartet

* indicates user who participated in Round 2 of testing. † indicates professional musician. Any cell with - indicates data already collected in previous test. The mean rating for recording experience was 4.87 (stdev=1.56), and the mean rating for DAW familiarity was 4.5 (stdev=1.61).

(Users 11-13). Another trio test involved new users who had never rehearsed in person together (Users 14-16). One quartet test involved returning users and the facilitator, who were part of an active chamber group (Users 16-18). User 16 participated in both a trio and quartet test.

Each test was a one-hour session hosted on Zoom, involving recording environment set-up, a set of tasks, and a Google Form questionnaire to provide feedback.

6.3.1 Tasks

The tasks remained the same from Round 2 in both duet and trio tests. For the trios in particular, tasks were distributed evenly so that each musician screen-shared for

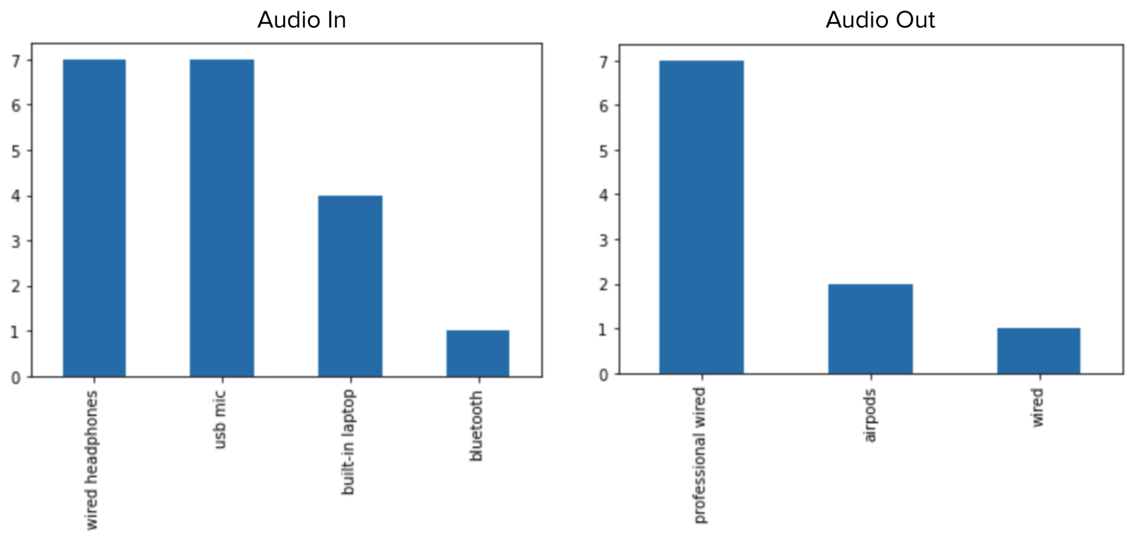


Figure 6-11: Left: number of users who used each type of audio input device. Right: number of users who used each type of audio output device.

one task. Because the quartet test only involved returning users, we were curious about their rehearsal approach and offered no directions. The quartet picked the most difficult piece among all tests (Olivier Messiaen's *Quartet for the End of Time*).

Task 1: *Record to and playback from an ensemble part*

Task 2: *Record, align, and playback an audio file using a backing part*

Task 3: *Record predetermined piece with small ensemble*

6.3.2 Results

All users completed their tasks. Table 6.5 summarizes questionnaire responses.

Task 1: *Record to and playback from an ensemble part*

Like previous rounds of this task, users enjoyed the simple interface and straightforward navigation. User 1 particularly liked the "Ahh I'm not ready!" button - she found the language both friendly and fun. User 11 appreciated how a group could share the same set of projects that was automatically propagated to the group. Since this user had been editing a cappella videos, they found this helpful to easily organize multiple projects in a concert set.

The main issue was the "select-to-record" feature. Because users only had one

Table 6.5: Summary of Round 3 Google Form Ratings

Question	1	2	3	4	5	6	7	Avg	Std
Prior experience making recordings	0	2	0	3	4	3	2	4.86	1.56
DAW familiarity	1	1	0	5	3	3	1	4.5	1.6
Rating: application navigation	0	0	0	0	0	6	13	6.68	0.48
Rating: group creation and project set-up	0	0	0	0	0	4	15	6.79	0.41
Rating: group features	0	0	0	1	3	7	8	6.16	0.90
Rating: alignment tool ease-of-use	0	0	0	0	3	9	7	6.21	0.71
Rating: audio quality	0	1	1	2	3	5	7	5.63	1.50
Overall experience	0	0	0	0	3	5	11	6.42	0.77

Each row represents one question, and each table entry indicates the number of participants who responded with the column label for that row's question (1 to 7).

part for this task, many were confused about why their part had to be selected before recording. In fact, many immediately clicked the record button without selecting and were taken aback by the pop-up asking them to select. All users ultimately figured out the functionality without help and did not encounter this issue again. One suggestion was to select the part by default if only one exists.

User 8, who was less familiar with music software, found the buttons very gray and confusing. Icons like "mute" and "solo" were visually unintuitive, and "download full mix" was semantically ambiguous. The user wanted to see more text labels with more colors for important buttons to "pop out on the page".

Most users thought that audio quality exceeded their expectations of a virtual rehearsal tool, but this was dependent on their microphone choice (Fig. 6-12). Bluetooth and inexpensive headphone microphones created the same crackling issues in Round 2. Bigger instruments with timbres rich in harmonics (i.e. bassoon) also caused aggressive audio clipping. User 10, a professional pianist, noticed that sustained tones were often clipped at the end. However, musicians that used professional-grade hardware experienced no peaking or distortion in their audio feed. Users from the Round 2 test also found that the audio quality improved overall using the same computer and input devices.

An interesting observation involved User 1, who had exceptional audio quality with standard Apple wired headphones. Because they were in a room with heavy

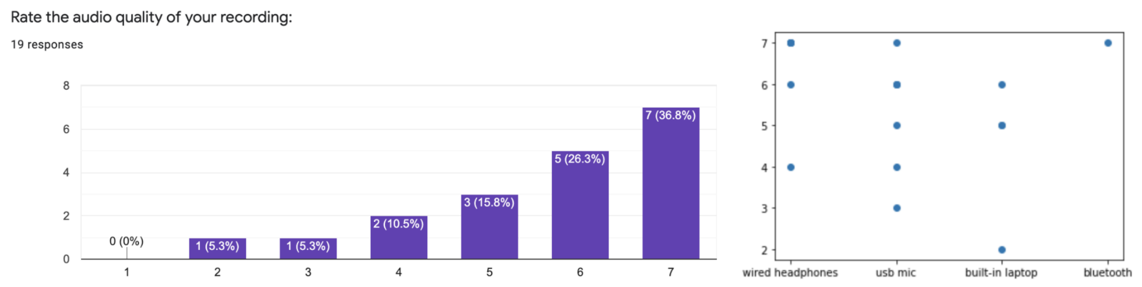


Figure 6-12: Left: ratings of perceived audio quality (mean=5.63, stdev=1.50). Right: audio input type vs. perceived audio quality ratings.

reverberation, the unintentional sound effect improved their audio quality. This suggested that the location of recording is a factor that may greatly impact perceived audio quality, but was not properly considered during testing.

Task 2: *Record, align, and playback an audio file using a backing part.*
The choice of icons for play, stop, and record made this task mostly straightforward, especially for users experienced with recording interfaces. For the purposes of recording, JamNSync offered just the right number of audio control features. However, there were two sources of confusion. First, some users were unsure if recording implied both recording and playback; one user even tried to click the play button before recording. This issue was corrected with verbal explanation from the user test facilitator. Second, other users tried to initiate playback by pressing the space bar - a common keystroke for playback in other music software. With no response, they wondered if the application was broken when the feature was not supported to begin with.

As in Round 2, the facilitator provided a metronome click to use as the initial backing track. User 6 provided their own backing track, which was a MIDI file with a click track used to record a large ensemble piece. However, because the duet did not play until much later in the piece, the user had to use Audacity to trim the file, then re-upload the backing track to JamNSync. While slightly tedious, the workaround was quick and clever.

Feedback for the alignment tool was positive. Users liked how the tool immediately popped up after recording, unlike BandLab and Audacity, which do not direct attention to track alignment and require users to guess the amount of latency adjustment. Although some new users found the tool initially hard, they understood the usage very quickly, especially since the tool was draggable and visual. User 9, a MIT chamber music professor, mentioned that the option to realign takes was particularly helpful in allowing group members to align each other's takes. When the professor taught virtual chamber groups, her students were not familiar with track editing, but helping them during a live coaching was tedious. The realignment feature would have not only expedited this process but also allow her to provide immediate feedback after each alignment.

During the alignment process, most users wanted a longer alignment period. It was unclear why the track stopped at the seven-second mark initially, which gave some users the impression that something went wrong with the recording. While the looping seven-second excerpt helped users become familiar with aligning that section, it is more likely to play out-of-tempo at the beginning of any group piece. As discovered in previous tests, alignment issues were more obvious much further into a piece during playback. In fact, more opportunities to become out-of-sync arise with more complex, lengthy pieces. To support longer playback, seeking functionality would be necessary in this tool. Users needed to hear complex sections multiple times in order to feel confident about alignment. Users also requested a reset button (i.e. over-dragged first time and lost the starting position).

The visual alignment guide was an interesting failure in the alignment tool. All new users did not touch the guide on their first alignment. One user even moved the yellow bar off to the side, so it would not block the waveforms. One potential source of failure was in its graphical design: the visual alignment guide was introduced as a ruler to users, but the yellow rectangle did not reminder users of one. As such, the function was not represented by its appearance. Users familiar with DAWs like Garageband and Logic were also frustrated that playback did not begin where the guide was located. Nonetheless, users did not find it distracting. Once users were directed about its proper usage, they found the guide helpful to align peaks.

User 15 suggested an alternate way to numerically align tracks. He noticed that once the alignment was close, it became annoying to adjust the latency by moving the track just a few pixels. Instead of dragging, he would have preferred to change a numerical value (e.g. 35 to 34) to adjust the final few milliseconds of latency.

When recordings were scrapped, users appreciated the confirmation pop-up. Interfaces like Audacity have instant deletion; once a track is gone, it is impossible to recover any parts or takes of the track.

Task 3: *Record predetermined piece with full ensemble.*
Each ensemble was able to create a set of recordings which they found satisfying. Users who had never played together appreciated hearing each other for the first time. Users who had played together enjoyed playing more expressively when listening to familiar parts from their ensemble, especially when a click track was not used.

However, musicians noticed that they were always following past takes when recording, which often contain mistakes and tempo fluctuations. After discussing issues and potential fixes with their group, it became bothersome to listen to and expect those variations when trying to record a new take. Even when users tried to ignore mistakes, their in-person habits to adapt to what they are hearing in real-time unintentionally influenced the new recording. As such, the potential of over-correcting (described in Round 2 testing) continues to prevail.

In terms of coordinating logistics (i.e. "let's decide to hear two bars for nothing"), those with extensive small group rehearsal background had a straightforward experience. Those lacking the background found the setup experience very frustrating. In the voice trio (users 11-13), two of three people had never sung in a small group in-person or virtually. When the experienced third member delegated tasks, it became very confusing where and how to start without careful elaboration. For example, singers frequently needed to hear "the first note" - but may have already muted on Zoom by the time they needed to recall it. The same two people also had limited experience with audio editing, requiring significant effort to learn for the first time. As they made more recordings, the rehearsal cycle seemed more intuitive.

Group record and play worked well: updates, requests, and alerts happened very quickly. Audio files transferred almost instantly and preserved quality after distribution. Users appreciated the built-in synchronicity of group recording, which was superior to having group members independently press the record button and hope that everyone is in sync. In fact, this experience reminded some users of Netflix Party.

Two problems were identified with the group rehearsal synchronization. First, users were misled by the wording of "group play" - some thought the button would group tracks together for playback, rather than synchronously playback all tracks for members of a group. Improved UI for group controls can address this issue. Second, there was ambiguity in which features propagated among users. Because group members saw each other's parts and takes immediately after creation, they expected muting, soloing, and volume adjust to do so as well. There was no feedback or instructions on the page detailing this scenario, causing confusion during group playback discussions.

The quality of group playback was strongly dependent on the audio devices used by the group. The more users there were, the more likely bad microphones would add extraneous noise to their recordings. These issues compounded when different parts were superimposed on each other for playback. Recording along to the backing mix became more difficult, as the number of pops and glitches increased with the number of parts in the mix. User 16, who rehearsed with both a trio and quartet, commented that she had two very different experiences between tests. The quartet rehearsal went significantly better because of a better choice of microphones among the group (even though logistics to plan the Messiaen Quartet were more challenging).

The lack of seeking functionality was also brought up in this round of testing. If any user misplaced a single beat, it was difficult to fix later on. For perfectionists, this stalled progress for rehearsals because they stopped the full group recording to address a small mistake. On the other hand, some users felt bad interrupting the recording and admitted to their mistake after everyone finished, which could not be used in the next iteration of playback anyway. Users also asked about rehearsing specific sections by using splices of the backing mix where a subset of instruments plays (i.e. clarinet and violin play from rehearsal letter F to G).

The most difficult part of this task was the alignment, especially for users who participated in larger group tests. Six out of nine users expressed frustration during the trio and quartet rehearsals. Similar issues from Round 2 were found: the first set of beats were typically less in time than later beats. Misalignment was also easier to identify later in the piece, which was not possible with the current alignment tool. As such, users mentioned that trying to achieve perfect alignment was "a slippery slope" - a potentially very tedious and time-consuming process. In these larger groups, we observed that the same people always took longer than others to get the perfect track alignment. As such, the slowest person to align tracks (a non-musical task) becomes the bottleneck of the rehearsal progress, which is not ideal.

Two user tests had notable observations. First, the duet of Take Five with User 7 showed the jazzy pieces had more room for error in alignment than classical and pop pieces. Not only was precision of alignment not necessary, but aligning tracks "slightly behind the beat" actually gave the piece the right character, rather than strictly aligning peak-to-peak.

The second interesting test was the Messiaen quartet rehearsal. The group initially rehearsed the sixth movement, which involved all instruments playing the same sequence of notes. Using a backing track was not suitable because each measure did not have the same number of beats, and having a subdivided click would be too fast and annoying to listen to. Instead, the pianist recorded alone first to create a backing track. After everyone recorded, the alignment process was easier because waveforms looked similar between parts. However, the issue of precision was extremely frustrating. A slight misalignment in one instrument made the playback experience near intolerable, especially because the group was accustomed to rehearsing in-person. The issue of one instrument always being slightly ahead or behind would never occur in-person, because musicians continually adjust to each other's tempo.

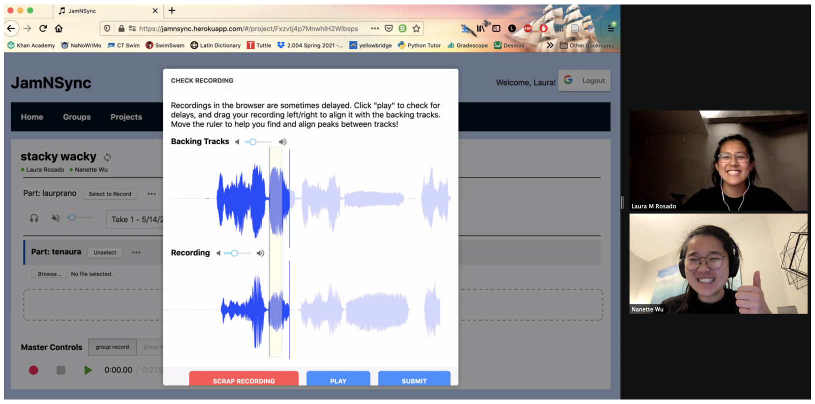
After the sixth movement proved to be difficult, the group rehearsed the first movement, which had a consistent time signature but was musically difficult to put together. Because all four instruments did not begin at the same time, the group had to find ways to produce in-tempo sounds in order to align their tracks. As such, other members counted "1, 2, 3; 1, 2, 3" at the beginning, which did help but felt quite unnatural for a rehearsal.

Discussion

Compared to Round 2, people noted many UI improvements in the questionnaire, but did not consciously notice them during user testing. The most helpful improvement was the highlighted part when selected and "1/4 ready for play" during the group record/playback request. Another user favorite was adjusting volumes in the alignment tool as well as the "invalid tooltip for dragging the backing track". To further improve the alignment tool, each part's exact take number and volume level should be passed into the alignment tool (currently just taking the latest take and full volume) to make sure the alignment mix represents what the user heard during recording. While users also perceived better audio quality overall, it is likely due to the use of better equipment during this round of testing.

The key feature that made the experience worse was the misleading visual alignment guide. Even when the guide helped align peaks, it still led to aural misalignment due to the different attack and decay times per voice and instrument. Aligning tracks was also harder with larger groups due to audio quality issues and parts not beginning within the first seven seconds.

User 1 found that recording harmonies of the same part (in which waveforms looked similar) made the visual alignment process a lot easier (Fig. 6-13). Conversely, the more distinct parts are, the more difficult they may be to align. The same logic applies to trickier pieces with difficult, syncopated rhythms.


CHECK RECORDING

Recordings in the browser are sometimes delayed. Click "play" to check for delays, and drag your recording left/right to align it with the backing tracks. Move the ruler to help you find and align peaks between tracks!

Figure 6-13: User demonstrates aligning harmonies of the same part, where the recording is visually similar to the backing mix.

A positive side effect of synchronous, recording-based virtual rehearsals is that users are fully present during rehearsal. Because all group members must agree before group recording or playing back, any user that zones out or slacks off must re-focus in order to accept the request and move forward. The moment of full focus at the beginning of JamNSync's virtual rehearsal is less guaranteed during in-person rehearsals, where people can easily "check-out" during long rehearsals.

The following table and figures quantifies improvement from Round 2 to Round 3. Table 6.3.2 compares means and standard deviations of Google Form questions with numerical ratings. Figure 6-14 and 6-15 show positively shifted distributions for the likelihood to use JamNSync if in-person rehearsals were not possible and possible.

Table 6.6: Comparing Rating Means from Round 2 and 3

Rating	Round 2 Mean (stdev)	Round 3 Mean (stdev)	Δ Mean
Application navigation	6.2 (1.32)	6.68 (0.48)	0.48
Group creation and project set-up	6.2 (0.79)	6.79 (0.41)	0.59
Alignment tool ease-of-use	5.5 (1.43)	6.21 (0.71)	0.71
Group features	5.3 (0.95)	6.16 (0.90)	0.86
Audio quality	4.6 (1.90)	5.63 (1.50)	1.03

Between the two rounds, the mean for each rating increased while the standard deviation decreased. The positive delta is attributed to improved usability, with the exception of audio quality (people used better audio devices in Round 3).

6.4 Analysis and Evaluation

Based on questionnaire answers, all 16 users who had experience with virtual music rehearsals began doing so after the pandemic started in March 2020. As such, virtual rehearsals were popularized by the pandemic; otherwise, there was no reason not to rehearse in person. Participants had used tools like Zoom, SoundJack, BandLab, and Audacity prior to testing JamNSync, and found them frustrating to use for the reasons outlined in Chapter 3. Results from user testing show that there is a low barrier to using the JamNSync system, which holds significant advantages over the four categories of NMP platforms. All quoted phrases are attributed to user testers.

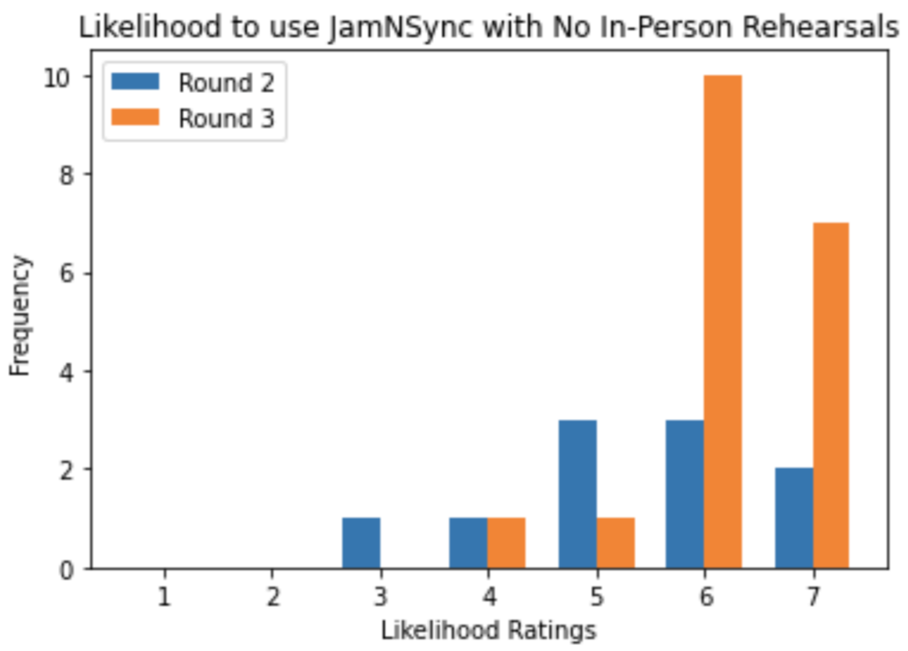


Figure 6-14: Using a scale from 1 to 7, this figure shows the distribution of the likelihood one would use JamNSync if in-person rehearsals were not possible. The mean was 5.4 (std=1.26, sample size 10) for Round 2 and 6.21 (std=0.79, sample size 19) for Round 3.

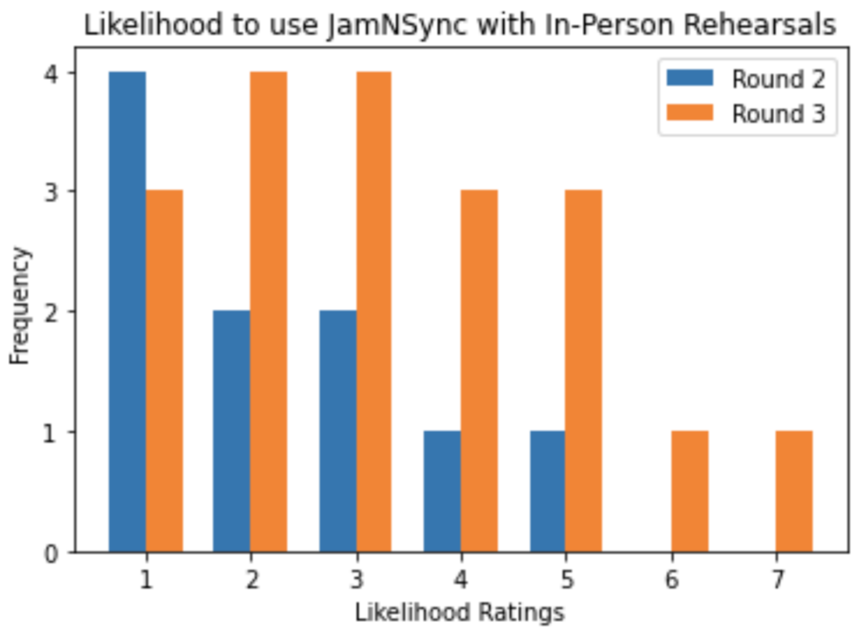


Figure 6-15: Using a scale from 1 to 7, this figure shows the distribution of the likelihood one would use JamNSync if in-person rehearsals were possible. The mean was 2.3 (std=1.41, sample size 10) for Round 2 and 3.31 (std=1.73, sample size 19) for Round 3.

- Generic Video Conferencing System (Zoom): JamNSync allows multiple musicians to be heard at once. JamNSync also provides a clean audio feed that does not distort quality during playback.

- Real-Time Music Rehearsal Systems (SoundJack): JamNSync is a more user-friendly, "no-hardware alternative to SoundJack" that requires little setup and no administrative approval. JamNSync's UI and controls are tailored to musicians and do not require extensive prior training. Overall sound quality is improved, since SoundJack "sacrifices quality to a great degree for latency". While users were "very hesitant to try SoundJack with people who were not tech-savvy", they were more willing to try JamNSync with other musicians.

- Online DAWs (BandLab): JamNSync offers per-group project organization with a simple sign-up process. JamNSync also layers tracks faster, even with multiple users online. While other platforms may have more features, JamNSync offers just enough controls for a user to record and playback audio.

- Home-Spun Solutions (Audacity and Google Drive): JamNSync saves a lot of hassle for a delegated person to stitch tracks together. Users align their own parts which parallelizes the alignment work. JamNSync also acts as a "middle man" that centralizes file upload and rehearsing in a single virtual space, reducing the time to export and upload large audio files.

In general, people were likely to use JamNSync if in-person rehearsals were not possible. JamNSync was a "nice middle ground" that "fills a hole for real-time remote rehearsals" as other options were inconvenient or limiting. "Even though groups still aren't playing synchronously with one another, [this tool had aspects of low latency by giving] people the ability to iterate quickly together." The collaborative aspects of JamNSync created a unified alternative to send and receive tracks, and the interactive components made rehearsal "a more social experience". Users with no prior virtual rehearsal experience appreciated the intuitive application flow. One user who had not played with others since the pandemic started found it fun and rewarding to rehearse with someone for the first time in months. The trio that never met in-person was able to make light-hearted side conversation while recording, which is hard to achieve on other virtual platforms. One user even compared JamNSync to the experience of playing multiplayer games like Fortnite, where users host an audio call on Discord while playing together on a separate game platform. However, several participants found this tool better suited for recording final takes than for in-depth rehearsing. Unless the group was well-prepared, users found this tool geared towards practicing shorter excerpts of pieces. The lack of seeking functionality prevented users from practicing specific sections of music, which would be inefficient with limited practice time.

However, people were unlikely to use JamNSync if in-personal rehearsals were possible. While some saw potential for JamNSync to bring people together in far geographical locations or when practice rooms were unavailable, "nothing really compares" to hearing live music; as one user states, "you don't need to record or set up Zoom and mic settings to rehearse [in-person]". Experienced chamber musicians noted the difficulty in "replicat[ing] immediate feedback you get in [a] live setting: people automatically adjust their pitch, tone, and intensity to each other, which makes a big difference [in the final product]". The lack of body language also made it tricky to align parts. Even with the benefit of having tangible recordings, each recording did not accurately represent the sound of a group from the perspective of a player in a physical room, since volume levels were determined by microphone settings. In fact, people who were skeptical of virtual rehearsals all together "would rather play solo music as opposed to any kind of recording" and "would rather just practice on [their] own and not have to deal with the overhead of working through a website, particularly [with] things like aligning audio and communicating about when to start and stop recording". There were also limitations in the types of pieces groups could play. Although the tool may be well suited for pop singers, other types of music like jazz benefit from real-time feedback and improvisation, which is not suited for recording-based rehearsals.

Notable takeaways include:

- Partitioning the work to independently align individual tracks can greatly speed up audio alignment of a small group piece. However, the slowest person to align their track due to perfectionist tendencies or lack of experience can greatly impede efficiency and progress of a virtual rehearsal.
- A secondary real-time communication tool is necessary to use JamNSync. The experience “was made easier with video calling during the rehearsal", but without video call (and premium Zoom educational accounts which remove time limit constraints), coordination among group members may be difficult.
- One user stated that “whenever [they] think about rehearsing music with others virtually, it always feels like a big ordeal to undertake”. This was not the case with JamNSync.
- Habits from in-person rehearsals carry well into JamNSync virtual rehearsals. For example, breathing sharply (i.e. "the sniff") before starting a piece helped users begin recording at the same time.
- Using JamNSync requires good organization and participation from group members. Coordinating where to start recording, initiating and stopping playback sessions, and leading discussions on areas of improvement will not work with passive members. This is easier to do in-person with body language, and harder to gauge solely through a video call.
- Rehearsing with JamNSync allows groups to review a rehearsal in real-time, which is hard to do in-person (one person must go out of their way to record). Using the video call and synchronous playback allow users to immediately quickly identify mistakes and differences in stylistic choices.
- User feedback revealed expectations tailored more to a recording platform than a group rehearsal. Because users could immediately playback what they rehearsed, they had higher standards for their virtual playing, with aspirations to make a single flawless take with perfect tone quality (uncommon in-person).

To perform holistic analysis on the user tests, we merged user ratings from Round 2 (Table 6.3) and Round 3 (Table 6.5) into one dataset. We present four figures, each representing a pair of ratings that presented an interesting finding. Each figure is a scatter plot with a linear regression line. The blue shading represents a 95% confidence interval for the regression estimate.

Using a significance threshold of 0.05, we found that the p-value is statistically significant in each comparison. Strong correlations were found between recording experience and prior DAW experience (Fig. 6-16), as well as navigation rating vs setup rating (Fig. 6-17). Weaker correlations were found between overall experience and alignment tool ease-of-use (Fig. 6-19), as well as audio quality rating versus group features rating (Fig. 6-18).

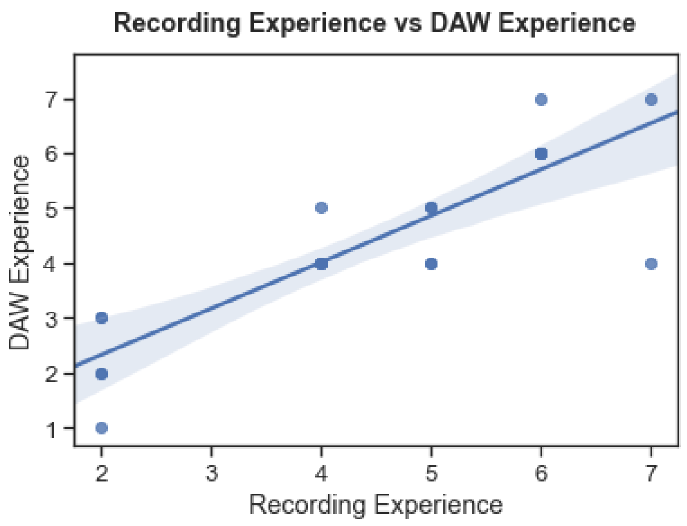


Figure 6-16: Rating of prior experience making audio recordings versus familiarity with a DAW ($R^2 = 0.754$, p-value $= 3.82\text{e}{-8}$). The more experience a user had with recording, the more likely they were familiar with editing and mixing recordings.

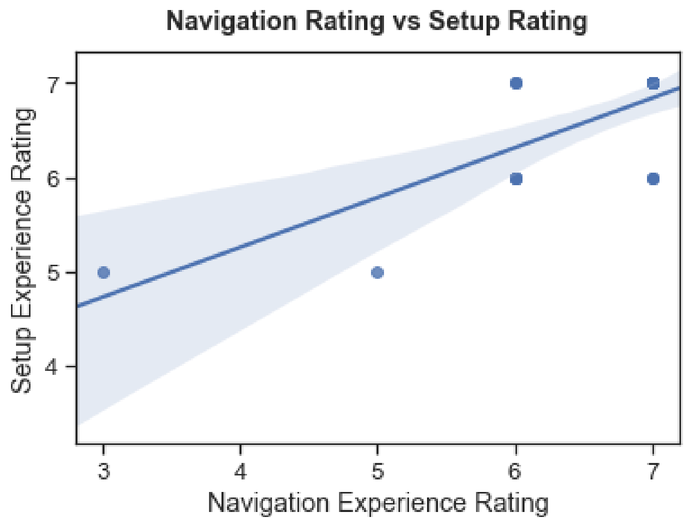


Figure 6-17: Rating of experience with application navigation versus experience setting up a project ($R^2 = 0.539$, p-value $= 6.39\text{e}{-6}$). The easier a user found it to navigate the application, the easier it was for them create a group and set up a project. This shows that users had similar experiences during non-music portions of JamNSync's virtual rehearsal - perhaps with higher ratings for more tech-savvy people and vice versa.

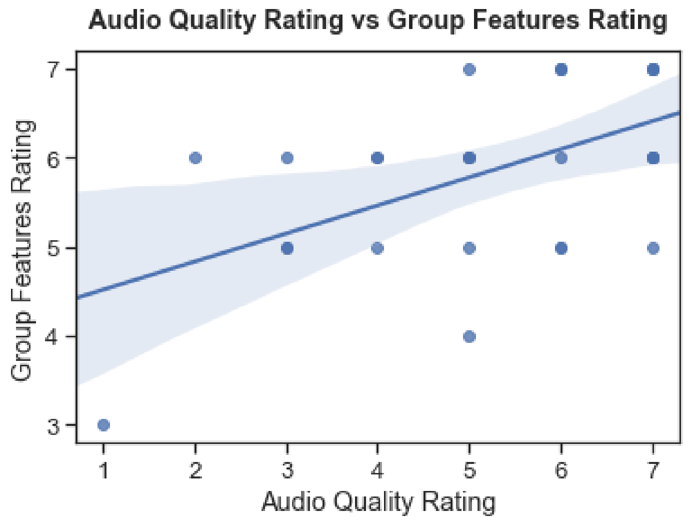


Figure 6-18: Rating of perceived audio quality versus rating of group feature intuitiveness ($R^2 = 0.288$, p-value $= 0.0027$). As mentioned previously, poor audio quality in recordings greatly impacted playback of multiple files simultaneously. Groups who produced high quality recordings tended to have a better experience during synchronized playback and rehearsal, resulting in a more positive overall experience using the group features.

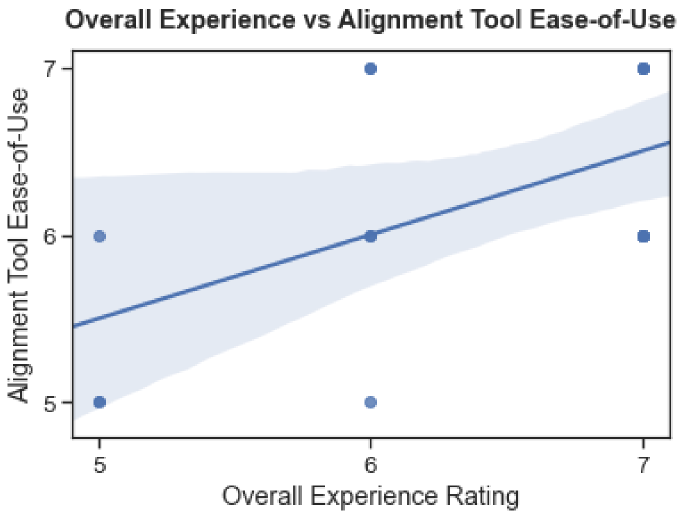


Figure 6-19: Rating of overall experience using JamNSync versus rating of the alignment tool's ease-of-use ($R^2 = 0.290$, p-value = 0.0173). People who had a better experience with JamNSync seemed to find the alignment tool easier to use.

Chapter 7 Conclusion

As a musician-tailored, browser-based application, JamNSync addresses the skepticism and difficulty [13] in continuing music rehearsals during the COVID-19 pandemic. An accessible and user-friendly alternative to existing rehearsal software, JamNSync justifies using technology to enable—rather than overcomplicate—remote music-making. Eliminating tedious set-up prerequisites and network jargon allows JamNSync to reach a wider population of musicians. This democratizes opportunities to make music during a time of social distancing and isolation.

However, as the pandemic ends, people will prefer to make music in-person once again over any virtual rehearsal space. Nonetheless, JamNSync remains relevant not only within the NMP community but also among small music groups. There is potential for JamNSync to connect musicians separated by distance and provide a virtual rehearsal space for small music groups that otherwise have no means of making music together. As such, JamNSync offers just the right amount of convenience and functionality for people to consider it a viable solution for remote music-making.

7.1 Contributions

This thesis makes the following contributions:

First, a user-friendly, latency-agnostic virtual rehearsal tool as a complete solution for small music groups. With little setup, a rehearsal group can efficiently record and playback their desired piece in a web browser. This can be accomplished without sending real-time audio over the network.

Second, the real-time rehearsal protocol that provides a synchronous recording and playback experience for a group. Using a voting consensus algorithm supported by event handlers over WebSockets, the RTR protocol simulates the experience of an in-person rehearsal. The protocol requires all users to agree before recording and playing back together, but allows any user to stop a group session.

Third, a visual audio alignment tool that adjusts latency differences between a recording track and a backing mix. Instead of placing the burden on a single person to master the tracks, each musician is responsible for adjusting their own recording. This distributes the work across the group, thus reducing rehearsal bottlenecks.

Fourth, a built-in per-group project organization that lends itself to managing pieces within a group. Rehearsal groups working on multiple pieces can track their projects in a centralized location, rather than split their work in different tools.

Fifth, a DAW-like, versioned recording system that allows a user to easily download takes. Instead of tracking different projects in multiple Google Drive folders, JamNSync centralizes file upload and download where the actual recording happens.

Finally, JamNSync creates an experience that imitates in-person rehearsals. At the end of a synchronous recording cycle, the feeling of watching other group members finish playing and discussing trouble spots is familiar to rehearsing in-person.

7.2 Future Work

Though the core of the JamNSync system has been implemented, there are many extensions that should be considered:

7.2.1 Feature Improvement

Higher Audio Quality

Crackling and popping in recordings require further investigation. While users can tolerate occasional distortions, it is evident from other web tools like Soundtrap and Upbeat that a solution exists to produce clean audio recordings.

Overhaul Visual Alignment Guide UI

Because the appearance of the visual alignment guide is not indicative of its function, the usage of the guide remains unclear. An alternative design can consider iMovie's track seeking and editing cursor, which shows the mouse cursor as a vertical line that extends across all tracks. Any click along the timeline moves the playback cursor to the location of the click, which links the cursor to the playback time.

Web Audio Scheduler

Playing multiple files simultaneously relies on preloading audio in the browser and sufficient computation power to initiate playback events at the same time. A more reliable method of playback uses a high level scheduling system provided by the Web Audio API. To start files at a precise time, an AudioContext object's currentTime property (accurate to 15 decimal places) can be exploited. In this way, all files can be specified to play at a precise time relative to the AudioContext [3]. Using a scheduler can also transfer audio file latency adjustment logic to the frontend.

Group Controls Interface

The current placement of the group controls implies functionality independent of solo audio controls. Instead, they should be integrated into a single set of audio controls, which switches from solo to group mode when multiple group members are online. If a user wants to individually record or playback, they can hold the shift button to reveal the solo options — similar to Audacity's audio controls design.

7.2.2 Feature Development

Seeking

It is imperative to implement pausing and advancing during playback. This offers a more complete rehearsing experience by replaying trouble spots. One idea considers Spotify's playback system designed to seek with a draggable progress bar.

Rehearsal Playback Annotations

Playback annotations can help groups revisit key timestamps during playback. For example, any user can press the space bar during playback to bookmark a time to listen later. The annotation can contain the creator's name, an editable label, and a timestamp - much like annotations used in Sonic Visualizer [28].

Speed Adjustment

Time-stretching without pitch manipulation allows users to playback audio at any tempo. This feature eliminates the need to generate new backing tracks or re-record all tracks when a group wants to change tempo. A built-in speed adjusting playback extension should resemble what already exist in DAWs like Audacity.

Embedded Video Call

Embedding an open-source video conferencing tool like Jitsi integrates the full rehearsal experience into a single page. This makes JamNSync more lightweight — favorable for computers with difficulty running Zoom with other applications. The Jitsi API also provides programmatic control to mute the talk-back channel (i.e. before group record), taking a step out of the existing rehearsal process.

7.2.3 Scalability

Multiple Browser Support

Bugs internal to the system caused audio and synchronization issues in browsers other than Firefox. This needs to be fixed so JamNSync works in all browsers.

Record Parts Starting at Different Times

The alignment tool should allow users to play back any desired length or portion of a piece. This removes the constraint that pieces require all parts to play within the first seven seconds.

Appendix A Software Libraries

Javascript Packages

Name	**Usage**	**Documentation Link**
@material-ui/core	Pre-built React components for basic web app functionalities	`https://material-ui.com/`
@material-ui/icons	Material icons used for DAW audio controls	`https://material-ui.com/components/material-icons/`
crunker	Helper library to process audio files with Web Audio API	`https://github.com/axios/axios`
ffmpeg	Javascript version of ffmpeg to manipulate audio data	`https://github.com/Kagami/ffmpeg.js/`
rc-slider	React component used for volume slider in DAW interface	`https://www.npmjs.com/package/rc-slider`
react	Main user interface library to design web application	`https://reactjs.org/`
react-countdown-circle-timer	Display countdown prior to begin recording	`https://github.com/vydimitrov/react-countdown-circle-timer`
react-dom	DOM-specific methods to render React elements	`https://reactjs.org/docs/react-dom.html`
react-draggable	Used in audio alignment tool to drag recorded file	`https://www.npmjs.com/package/react-draggable`

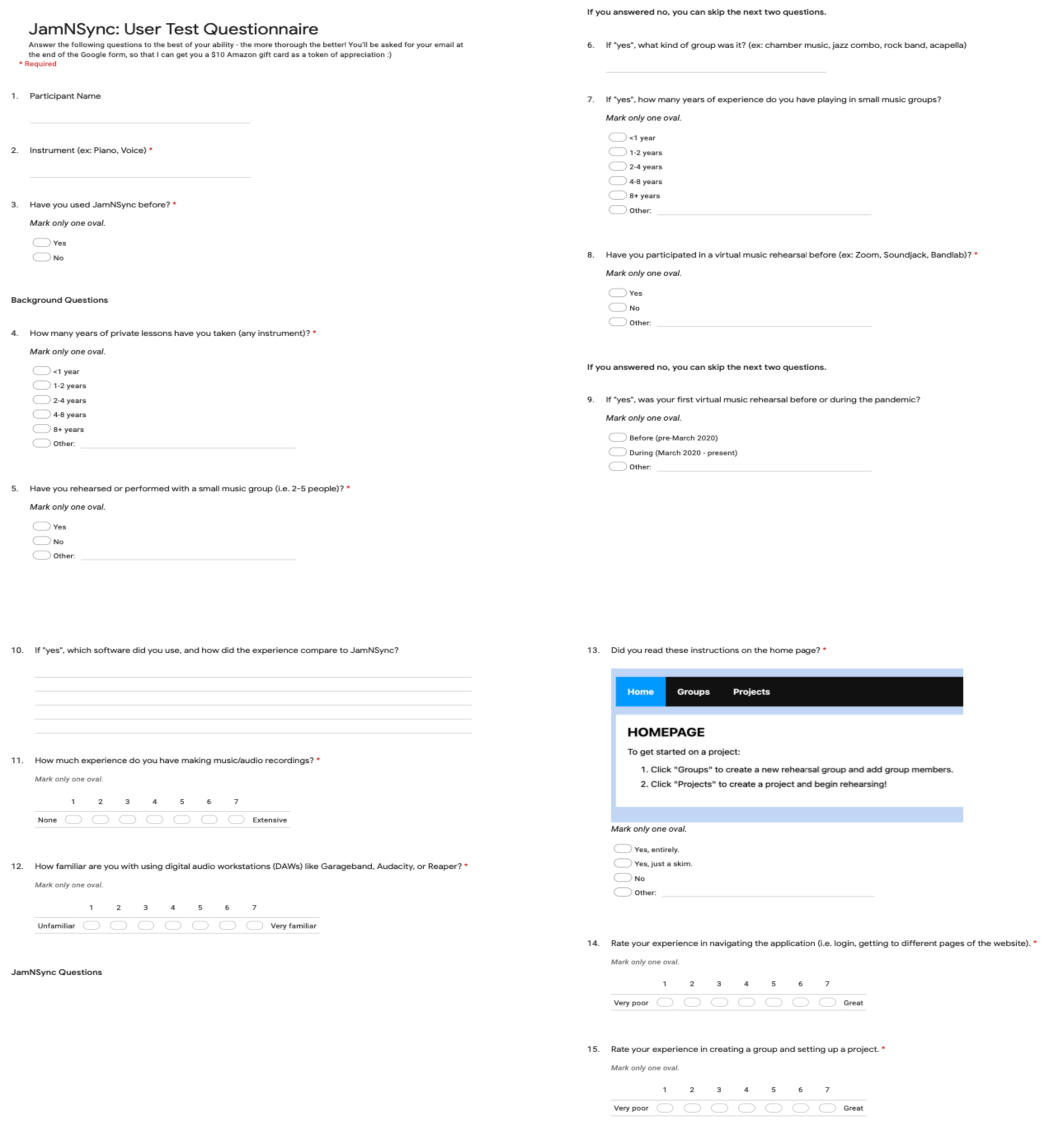
react-dropdown	Dropdown component to select takes for each project track	`https://github.com/fraserxu/react-dropdown`
react-google-login	Google OAuth sign in and log in component for React	`https://github.com/anthonyjgrove/react-google-login`
react-router	Navigational components to easily switch between web pages	`https://reactrouter.com/`
react-router-dom	DOM bindings for React Router	`https://www.npmjs.com/package/react-router-dom`
react-scripts	Scripts and configuration used by Create React App	`https://www.npmjs.com/package/react-scripts`
react-st-modal	Customizable pop up dialog components	`https://nodlik.github.io/react-st-modal/`
socket.io-client.js	Supports real-time, bi-directional, event-based communication between browser and server	`https://socket.io/docs/v4/client-initialization/`
wavesurfer.js	Customizable audio waveform visualization	`https://wavesurfer-js.org/`
Web Audio Recorder	Record audio input (Web Audio API AudioNode object) and encode as MP3 audio file	`https://github.com/higuma/web-audio-recorder-js/`

Python Packages

Package	**Usage**	**Documentation Link**
alembic	Database migration tool used with SQLAlchemy	`https://alembic.sqlalchemy.org/en/latest/`
awscli	Amazon Web Services (AWS) command line interface to configure credentials and region	`https://aws.amazon.com/cli/`
boto3	AWS SDK for Python to manage audio file storage in S3	`https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`
eventlet	Concurrent networking library to synchronize group members	`https://eventlet.net/`
ffmpeg	Audio converter used by pydub to manipulate audio files	`http://ffmpeg.org/ffmpeg.html`
Flask	Micro web framework used for the system's web server	`https://flask.palletsprojects.com/en/1.1.x/`
Flask-Login	Flask extension that provides user session management (i.e. log in, log out)	`https://flask-login.readthedocs.io/en/latest/`
Flask-Migrate	Flask extension that handles SQLAlchemy database migrations using Alembic	`https://flask-migrate.readthedocs.io/en/latest/`
Flask-SocketIO	Flask extension that enables low latency bi-directional communication between client and server	`https://flask-socketio.readthedocs.io/en/latest`
Flask-SQLAlchemy	Flask extension that adds support for SQLAlchemy	`https://flask-sqlalchemy.palletsprojects.com/`
gunicorn	Python Web Server Gateway Interface (WSGI) HTTP server	`https://gunicorn.org/`

jmespath	Query language to extract JSON elements in API responses	`https://jmespath.org/`
MarkupSafe	Helper library to escape characters in text for HTML and XML	`https://markupsafe.palletsprojects.com/en/1.1.x/`
numpy	Multi-dimensional array library used to manipulate audio files	`https://numpy.org/doc/stable/`
psycopg2-binary	PostgreSQL database adapter to connect to interface with database	`https://www.psycopg.org/docs/`
pydub	Audio file processor to edit and trim recorded audio files	`https://github.com/jiaaro/pydub`
python-dateutil	Format audio file timestamps stored in database	`https://dateutil.readthedocs.io/en/stable/`
python-dotenv	Reads database and Flask environment variables as key-value pairs from .env files and Heroku	`https://pypi.org/project/python-dotenv/`
pytz	Provides world timezone definitions to calculate accurate audio file timestamps stored in database	`https://github.com/newvem/pytz`
rsa	Helper RSA library to support cryptographic signing in Flask.	`https://stuvel.eu/software/rsa/`
s3transfer	Helper library to manage AWS S3 file transfers	`https://github.com/boto/s3transfer`
shortuuid	Generates URL-safe hashes to concisely identify each project	`https://pypi.org/project/shortuuid/`
sox	Sound processing library to manipulate recorded audio files	`https://pysox.readthedocs.io/en/latest/`
SQLAlchemy	Object relational mapper that provides a simple API to interact with PostgreSQL database	`https://www.sqlalchemy.org/`

Appendix B User Test Questionnaires



JamNSync: User Test Questionnaire

Answer the following questions to the best of your ability - the more thorough the better! You'll be asked for your email at the end of the Google form, so that I can get you a $10 Amazon gift card as a token of appreciation :)

* Required

1. Participant Name

2. Instrument (ex: Piano, Voice) *

3. Have you used JamNSync before? *

Mark only one oval.

- Yes
- No

Background Questions

4. How many years of private lessons have you taken (any instrument)? *

Mark only one oval.

- <1 year
- 1-2 years
- 2-4 years
- 4-8 years
- 8+ years
- Other:

5. Have you rehearsed or performed with a small music group (i.e. 2-5 people)? *

Mark only one oval.

- Yes
- No
- Other:

If you answered no, you can skip the next two questions.

6. If "yes", what kind of group was it? (ex: chamber music, jazz combo, rock band, acapella)

7. If "yes", how many years of experience do you have playing in small music groups?

Mark only one oval.

- <1 year
- 1-2 years
- 2-4 years
- 4-8 years
- 8+ years
- Other:

8. Have you participated in a virtual music rehearsal before (ex: Zoom, Soundjack, Bandlab)? *

Mark only one oval.

- Yes
- No
- Other:

If you answered no, you can skip the next two questions.

9. If "yes", was your first virtual music rehearsal before or during the pandemic?

Mark only one oval.

- Before (pre-March 2020)
- During (March 2020 - present)
- Other:

10. If "yes", which software did you use, and how did the experience compare to JamNSync?

11. How much experience do you have making music/audio recordings? *

Mark only one oval.

	1	2	3	4	5	6	7	
None	◯	◯	◯	◯	◯	◯	◯	Extensive

12. How familiar are you with using digital audio workstations (DAWs) like Garageband, Audacity, or Reaper? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unfamiliar	◯	◯	◯	◯	◯	◯	◯	Very familiar

JamNSync Questions

13. Did you read these instructions on the home page? *

Home Groups Projects

HOMEPAGE

To get started on a project:

1. Click "Groups" to create a new rehearsal group and add group members.
2. Click "Projects" to create a project and begin rehearsing!

Mark only one oval.

- Yes, entirely.
- Yes, just a skim.
- No
- Other:

14. Rate your experience in navigating the application (i.e. login, getting to different pages of the website). *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	◯	◯	◯	◯	◯	◯	◯	Great

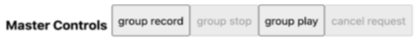
15. Rate your experience in creating a group and setting up a project. *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	◯	◯	◯	◯	◯	◯	◯	Great

Figure B-1: Google Form used to conduct Round 2 of User Testing (Part 1 of 2)

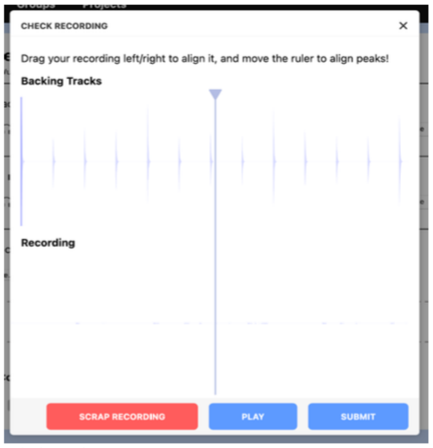
16. The usage of the "group" features was intuitive. *



Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Strongly Agree

17. After recording, the audio alignment tool (i.e. dragging the waveform) was easy-to-understand. *


CHECK RECORDING
Drag your recording left/right to align it, and move the ruler to align peaks!
Backing Tracks
Recording
SCRAP RECORDING
PLAY
SUBMIT

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Strongly Agree

18. Upload a take of your recording here: *

Files submitted:

19. Rate the audio quality of your recording: *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very high

20. What kind of audio input device (microphone) did you use (ex: USB Blue Snowflake)? *

21. What kind of audio output device (speaker) did you use (ex: Bose wired headphones)? *

22. Comments on audio quality and overall recording experience:

General Questions

23. If you *could not* play or rehearse music in-person, how likely would you use JamNSync to rehearse music? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unlikely	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very likely

24. Explain your reasoning.

25. If you *could* play or rehearse music in-person, how likely would you use JamNSync to rehearse music? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unlikely	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very likely

26. Explain your reasoning.

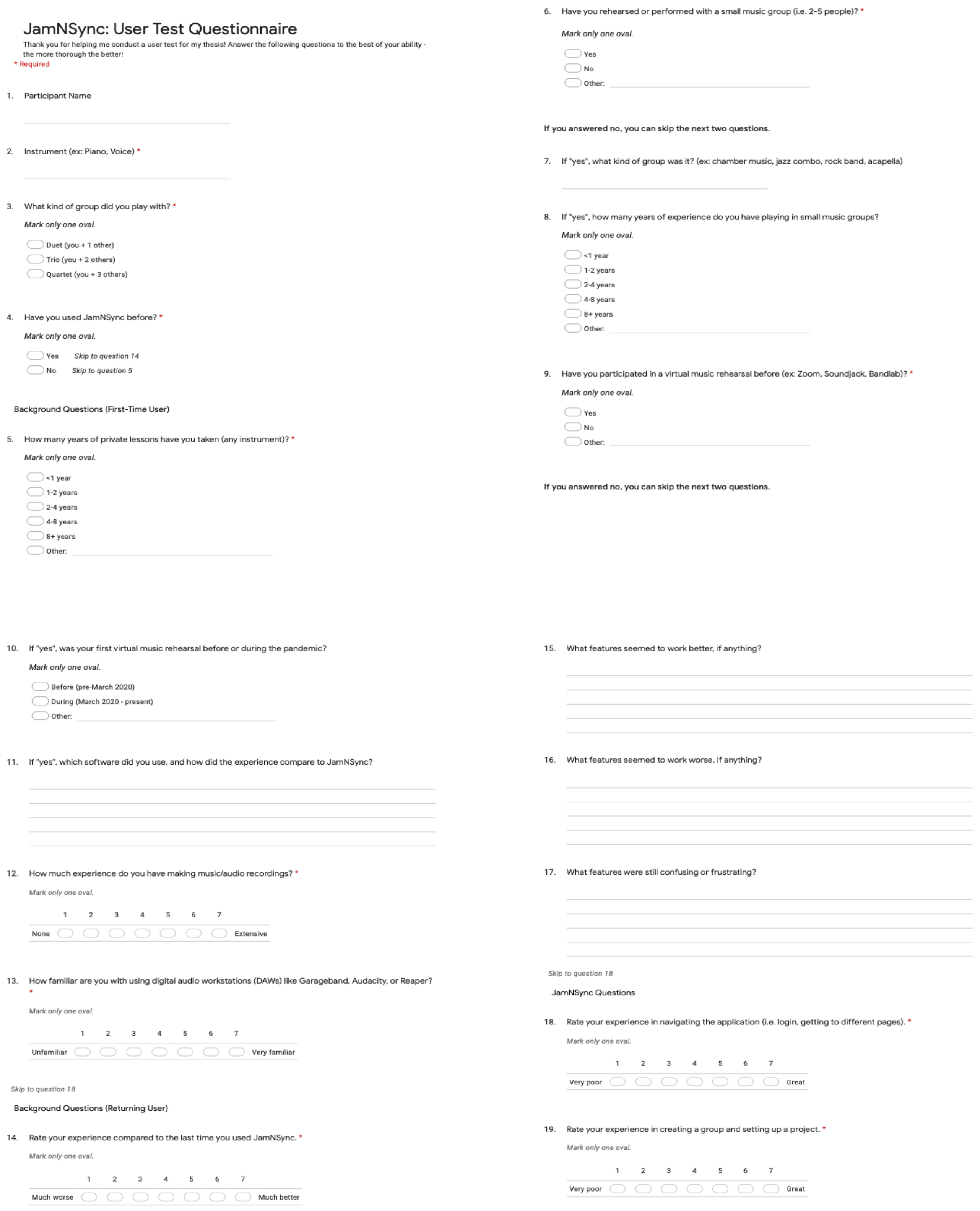
27. What did you think worked particularly well? *

28. What did you find frustrating about the experience? *

29. Final thoughts or recommendations (i.e. for the app, user test experience, etc.)?

30. Thanks for your help! Please enter your email below to receive a $10 Amazon gift card:

Figure B-2: Google Form used to conduct Round 2 of User Testing (Part 2 of 2)



JamNSync: User Test Questionnaire

Thank you for helping me conduct a user test for my thesis! Answer the following questions to the best of your ability - the more thorough the better!

* Required

1. Participant Name

2. Instrument (ex: Piano, Voice) *

3. What kind of group did you play with? *

Mark only one oval.

- Duet (you + 1 other)
- Trio (you + 2 others)
- Quartet (you + 3 others)

4. Have you used JamNSync before? *

Mark only one oval.

- Yes *Skip to question 14*
- No *Skip to question 5*

Background Questions (First-Time User)

5. How many years of private lessons have you taken (any instrument)? *

Mark only one oval.

- <1 year
- 1-2 years
- 2-4 years
- 4-8 years
- 8+ years
- Other:

6. Have you rehearsed or performed with a small music group (i.e. 2-5 people)? *

Mark only one oval.

- Yes
- No
- Other:

If you answered no, you can skip the next two questions.

7. If "yes", what kind of group was it? (ex: chamber music, jazz combo, rock band, acapella)

8. If "yes", how many years of experience do you have playing in small music groups?

Mark only one oval.

- <1 year
- 1-2 years
- 2-4 years
- 4-8 years
- 8+ years
- Other:

9. Have you participated in a virtual music rehearsal before (ex: Zoom, Soundjack, Bandlab)? *

Mark only one oval.

- Yes
- No
- Other:

If you answered no, you can skip the next two questions.

10. If "yes", was your first virtual music rehearsal before or during the pandemic?

Mark only one oval.

- Before (pre-March 2020)
- During (March 2020 - present)
- Other:

11. If "yes", which software did you use, and how did the experience compare to JamNSync?

12. How much experience do you have making music/audio recordings? *

Mark only one oval.

	1	2	3	4	5	6	7	
None	◯	◯	◯	◯	◯	◯	◯	Extensive

13. How familiar are you with using digital audio workstations (DAWs) like Garageband, Audacity, or Reaper? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unfamiliar	◯	◯	◯	◯	◯	◯	◯	Very familiar

Skip to question 18

Background Questions (Returning User)

14. Rate your experience compared to the last time you used JamNSync. *

Mark only one oval.

	1	2	3	4	5	6	7	
Much worse	◯	◯	◯	◯	◯	◯	◯	Much better

15. What features seemed to work better, if anything?

16. What features seemed to work worse, if anything?

17. What features were still confusing or frustrating?

Skip to question 18

JamNSync Questions

18. Rate your experience in navigating the application (i.e. login, getting to different pages). *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	◯	◯	◯	◯	◯	◯	◯	Great

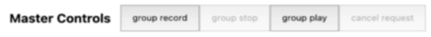
19. Rate your experience in creating a group and setting up a project. *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	◯	◯	◯	◯	◯	◯	◯	Great

Figure B-3: Google Form used to conduct Round 3 of User Testing (Part 1 of 2)

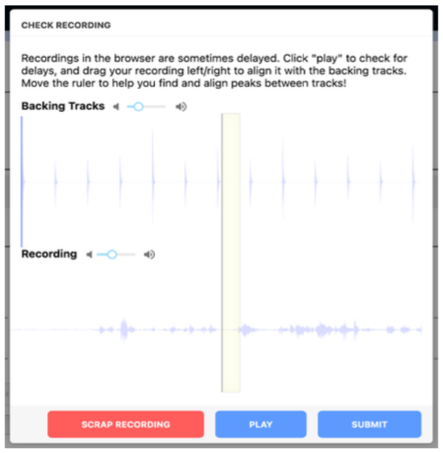
20. The usage of the "group" features was intuitive. *



Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Strongly Agree

21. After recording, the audio alignment tool (i.e. dragging the waveform) was easy-to-understand. *



Recordings in the browser are sometimes delayed. Click "play" to check for delays, and drag your recording left/right to align it with the backing tracks. Move the ruler to help you find and align peaks between tracks!

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Strongly Agree

22. Upload a take of your recording here: *

Files submitted:

23. Rate the audio quality of your recording: *

Mark only one oval.

	1	2	3	4	5	6	7	
Very poor	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very high

24. What kind of audio input device (microphone) did you use (ex: USB Blue Snowflake)? *

25. What kind of audio output device (speaker) did you use (ex: Bose wired heapdphones)? *

26. Comments on audio quality and overall recording experience:

General Questions

27. How was your overall experience with the interface? *

Mark only one oval.

	1	2	3	4	5	6	7	
Not user-friendly at all	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Super user-friendly

28. If you *could not* play or rehearse music in-person, how likely would you use JamNSync to rehearse music? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unlikely	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very likely

29. Explain your reasoning.

30. If you *could* play or rehearse music in-person, how likely would you use JamNSync to rehearse music? *

Mark only one oval.

	1	2	3	4	5	6	7	
Unlikely	⬭	⬭	⬭	⬭	⬭	⬭	⬭	Very likely

31. Explain your reasoning.

32. What did you think worked particularly well? *

33. What did you find frustrating about the experience? *

34. Final thoughts or recommendations (i.e. for the app, user test experience, etc.)?

That's it! Thank you so much once again :)

Figure B-4: Google Form used to conduct Round 3 of User Testing (Part 2 of 2)

Bibliography

[1] Acapella [Mobile application]. (2015). Retrieved from `https://www.mixcord.co/pages/acapella`

[2] Add Google Sign-In to Your Web App [Computer software]. (2020). Retrieved from `https://developers.google.com/identity/sign-in/web`

[3] Advanced Techniques: Creating and Sequencing Audio. (2021, March 7). Web APIs - MDN. Retrieved May 23, 2021, from `https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Advanced_techniques`.

[4] Agora.io [Computer software]. (2019). Retrieved from `https://www.agora.io/en/`

[5] Awesome Networked Media [Computer software]. (2020). Retrieved from `https://github.com/omarcostahamido/awesome-networked-media`

[6] AWS Cloud Products [Computer Software]. (2020). Retrieved from `https://aws.amazon.com/products/?nc2=h_ql_prod_fs_f`

[7] Bandlab [Computer software]. (2014). Retrieved from `https://www.bandlab.com/`

[8] Bernstein, Philip A., et al. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987. Retrieved from `https://research.microsoft.com/en-us/people/philbe/ccontrol.aspx`.

[9] BSQ. (2020, March 13). Virtual Rehearsing. Retrieved October 28, 2020, from `https://bayberrystringquartet.com/2020/03/13/virtual-rehearsing/`

[10] Carôt, A., & Werner, C. (2007, September). Network music performance-problems, approaches and perspectives. In *Proceedings of the "Music in the Global Village"-Conference, Budapest, Hungary* (Vol. 162, pp. 23-10). Retrieved October 27, 2020, from `http://www.carot.de/Docs/MITGV_AC_CW.pdf`

[11] Chafe, C. & Caceres, J. (2009). Jacktrip: Under the Hood of an Engine for Network Audio. Retrieved November 3, 2020, from `https://ccrma.stanford.edu/groups/soundwire/publications/papers/2009-caceres_chafe-ICMC-jacktrip.pdf`

[12] Cleanfeed [Computer software]. (2016). Retrieved from `https://cleanfeed.net/`

[13] Crimmins, P. (2020, April 02). Some choirs rehearse remotely while others wait out coronavirus pandemic. Retrieved October 27, 2020, from `https://whyy.org/articles/choirs-make-the-most-of-virtual-rehearsals-during-coronavirus-pandemic`

[14] Clark, D. (August 1988). The Design Philosophy of the DARPA Internet Protocols. *ACM SIGCOMM Conference*, 18(4), 106-114. Retrieved April 24, 2021, from `https://doi.org/10.1145/52325.52336`

[15] Egozy, E. (2020). Low Latency Audio at MIT: Overview and Core Concepts. Retrieved from `https://canvas.mit.edu/courses/9153`.

[16] Fischer, V. (2015). Case Study: Performing Band Rehearsals on the Internet with Jamulus. Retrieved November 3, 2020, from `https://jamulus.io/PerformingBandRehearsalsontheInternetWithJamulus.pdf`

[17] Gu, X., Dick, M., Kurtisi, Z., Noyer, U., & Wolf, L. (2005). Network-centric music performance: practice and experiments. *IEEE Communications Magazine*, 43(6), 86-93. Retrieved October 28, 2020, from `https://ieeexplore.ieee.org/abstract/document/1452835`

[18] Handley, M. (2006). Why the Internet only just works. *BT Technol J*, 24, 119–129. Retrieved April 24, 2021, from `https://doi.org/10.1007/s10550-006-0084-z`

[19] Hardman, R. (2020, April 23). Social Distancing, Lagging Technology Make Choir Rehearsals A Challenge. Retrieved September 30, 2020, from `https://www.wnpr.org/post/social-distancing-lagging-technology-make-choir-rehearsals-challenge`

[20] Howell, I., et al. (2020, March 25). Audio Quality of Four Video Conferencing Platforms. Retrieved October 28, 2020, from `https://www.ianhowellcountertenor.com/preliminary-report-testing-video-conferencing-platforms`

[21] Jamkazam [Computer software]. (2020). Retrieved from `https://www.jamkazam.com/`

[22] Jitsi [Computer software]. (2003). Retrieved from `https://github.com/jitsi/jitsi-meet`

[23] Lazzaro, J., & Wawrzynek, J. (2001, January). A case for network musical performance. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video* (pp. 157-166). Retrieved

October 28, 2020, from
https://john-lazzaro.github.io/sa/pubs/pdf/nossdav01.pdf

[24] NINJAM [Computer software]. (2005). Retrieved from
https://www.cockos.com/ninjam/

[25] OpenShot [Computer software]. (2008). Retrieved from
https://www.openshot.org/

[26] Pogue, D. (2020, June 4). How to Make Your Virtual Jam Session Sound-and Look-Good. Retrieved September 30, 2020, from
https://www.wired.com/story/zoom-music-video-coronavirus-tips/

[27] Saroiu, S., Gummadi, K., and Gribble, S. (2002, January). A measurement study of peer-to-peer file sharing systems. In Proceedings of Multimedia Computing and Networking. Retrieved from https://people.mpi-sws.org/ gummadi/papers/p2ptechreport.pdf.

[28] Sonic Visualizer [Computer software]. (2021). Retrieved from
https://www.sonicvisualiser.org/

[29] SoundJack [Computer software]. (2020). Retrieved from
https://www.soundjack.eu/howto/

[30] Soundtrap [Computer software]. (2012). Retrieved from
https://www.soundtrap.com/

[31] Symonics' Fastmusic Box [Computer hardware]. (2012). Retrieved from
https://symonics.com/fastmusic/.

[32] Upbeat [Computer software]. (2020). Retrieved from
https://upbeatmusicapp.com/

[33] WebEx [Computer software]. (1995). Retrieved from https://www.webex.com/

[34] Xambó, A. (2020, June 02). Network Music Performance During COVID-19 and Beyond: A Quick Review of Available Software. Retrieved October 28, 2020, from http://annaxambo.me/blog/research/2020/06/02/network-music-performance/

[35] Zoom [Computer software]. (2011). Retrieved from http://zoom.us/