

SANTA CLARA UNIVERSITY

Department of Computer Science and Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY


Zachary Graber, William Henrion

ENTITLED

Music For Me

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE AND ENGINEERING

DocuSigned by:

D8129749E98B404...

6/10/2021

Thesis Advisor(s) (use separate line for each advisor)

date

Nam Ling
Nam Ling (Jun 15, 2021 09:51 PDT)

Jun 15, 2021

Department Chair(s) (use separate line for each chair)

date

Music For Me

By

Zachary Graber, William Henrion

SENIOR DESIGN PROJECT REPORT

Submitted to the Department of Computer Science and Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements

for the degree of

Bachelor of Science in Computer Engineering

Santa Clara, California

2021

Music For Me

Zak Graber

William Henrion

Department of Computer Engineering

Santa Clara University

ABSTRACT

Music For Me is an online web application that enables users to compose improvised music alongside an AI. Users have an array of understandable, easy to use yet powerful options available to them. While previous AI-collaborative music improvisation tools exist, they are either limited in scope, with minimal structure and user customization or completely overwhelming with a large amount of provided options so as to be inaccessible except to those dedicated to learning the program. This web application provides an intersection between these two extremes, providing a pick-up and play model that allows for a rich improvisational experience while still presenting an easy-to-use and understandable interface.

0.1 Acknowledgements

Thanks so much to our advisor Dr. Ackerman who set us on this journey of building this project that we've enjoyed so much. Also thank you to the friends and family who supported us and the project every step of the way.

Table of Contents

1	Introduction	1
	1.1 Background.....	1
	1.2 Problem Statement.....	1
	1.3 Stakeholders.....	2
	1.4 Existing Solutions.....	2
	1.5 Our Solution.....	4
2	Requirements	5
	2.1 Functional Requirements.....	5
	2.2 Nonfunctional Requirements.....	5
	2.3 Design Constraints.....	5
3	Use Cases	7
	3.1 Use Case Diagram.....	7
	3.2 Use Case Description Table.....	7
	3.3 Conceptual Model.....	9
4	Technologies Used	10
	4.1 Functional Technologies.....	10
	4.2 UI Technologies.....	11
5	Functional Diagrams	12
	5.1 System Sequence Diagram.....	12
	5.2 Architecture Diagram.....	13
	5.3 Architecture Diagram Description.....	14
6	Design Rationale	15
7	App Implementation	16
	7.1 Overview.....	16
	7.2 UI/UX.....	16
	7.3 Application Logic.....	19
8	Development Timeline	20
9	Ethical Analysis	21
10	Conclusion	22
	10.1 Summary.....	22
	10.2 Obstacles Encountered.....	22
	10.3 Lessons Learned.....	22
	10.4 Future Work.....	23

List of Figures

1.1	Impro-Visor UI	3
1.2	AI Duet UI	3
2.1	Use Case Diagram	7
5.1	System Sequence Diagram	12
5.2	Architectural Diagram	13
7.1	UI before Session starts	17
7.2	UI Post Session	17
7.3	Chord Settings Box	18

Chapter 1: Introduction

1.1 Background

Before going into the problem we solved with our senior design project, we must go into some of the background information that is necessary to understand. Musical improvisation is the act of coming up with music and playing that music on the spot. This is usually done within a musical context such as a backing chord or continuing another musical melody. It can also be done with a specific style of music in mind which affects the rhythm and notes the musical improviser plays. These styles can be jazz, blues, rock, etc. Musical improvisation can also be a collaborative activity by trading playing. Trading is where two or more musicians switch off improvising/playing music after a set amount of time usually in units of musical bars. It's analogous to writing a story with another person by trading writing one sentence at a time. Trading playing is a fun activity for either practicing your improvisation skills in the context of another musician or even writing a song by taking elements from your improvisations.

Also, a necessary thing to understand is the progression of musical artificial intelligence. This is AI that is capable of generating musical notes when trained with musical data. This has been accomplished with many different algorithms in AI and Machine Learning but what is a new development are these AI models and algorithms being available in open source. The library used to power our project is Magenta, which provides a Recurrent Neural Network (RNN) model already trained with musical MIDI data. What this model can do is it can be given a musical melody in notes and then it can continue that sequence of notes with what it learned from its prior training. It is in this context we consider the possibility of AI to provide the experience of playing and improvising music with another person in a meaningful way. This can be used for practicing your playing, coming up with an idea for a song, or just having fun creating music.

1.2 Problem Statement

In our present situation, many musicians are restricted to either practicing alone or over an audio or video-based communication system with other musicians. However, these are both unideal

solutions, as by practicing alone you lose the enjoyment of playing in the context of other musicians, and playing with people over the internet presents technical challenges such as latency, which poses a large threat to something like music where exact timings are essential. For these reasons, we considered the possibility of AI to provide that experience of playing with another musician. There are currently web and desktop applications, such as *AI Duet* and *Impro-Visor* that do provide features where you can play along with an AI, but these apps are either too barebones and limited to provide a meaningful playing and improvising experience, or they offer too many features and complicated UIs for casual music experiences. With this in mind, we decided there was a good opportunity to create an app that finds a good mix between these two applications. Something that provides a low learning curve but sufficient features and options to add more structure and purpose into the improvising experience.

1.3 Stakeholders

Our team identified the following stakeholders in our project:

- Musicians
- Musical Artificial Intelligence Developers
- Casual Internet users
- Santa Clara Engineering Community

1.4 Existing Solutions

As previously mentioned, there are currently solutions that provide the functionality of playing music with artificial intelligence. Let's consider *Impro-Visor* first. *Impro-Visor* is a fantastic musical notation desktop application intended for jazz musicians wanting to improve their solo playing by utilizing an AI music generation model to compare their playing to. The application itself has a plethora of features but the one we thought was most relevant was the play-along feature where you can trade with the AI model with backing chords. This app and feature work great, but it has a very high learning curve with a large number of features it offers and a complex UI (Fig. 1.1) to navigate through. It is not intended for casual musicians.

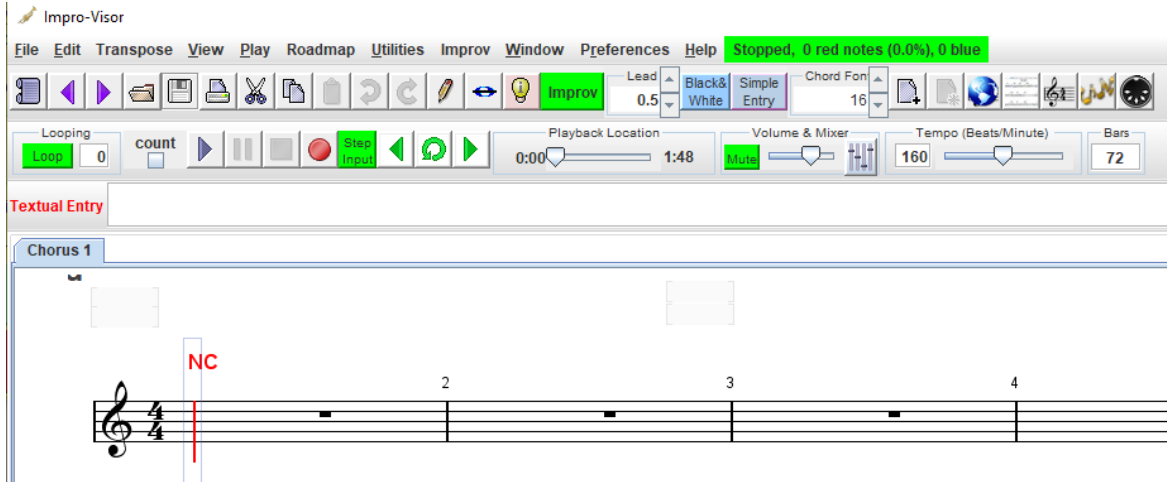


Figure 1.1: Impro-Visor UI

Another solution we will consider is *AI Duet* which is a web application with a simple UI (Fig. 1.2) that allows the simple functionality of just plugging in a MIDI keyboard and can start playing immediately. The AI then responds to your playing at an unspecified time. This application does not provide any backing chords or timing so we found that it was a confusing experience and hard to have a meaningful practice experience.

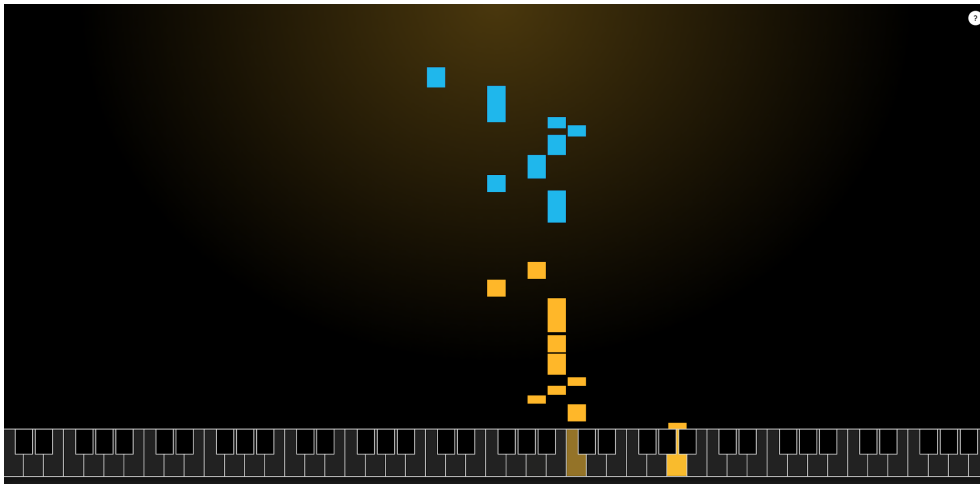


Figure 1.2: AI Duet UI

1.5 Our Solution

We present a web application that provides an AI that plays along with you in the browser with features such as backing tracks and programmable chord progressions as well as a timing element to simulate a musical trading experience. We combine this with a pick-up-and-play ease of use to provide a playing experience that doesn't overwhelm with setup and features but still provides an opportunity for meaningful and enjoyable practice. To provide this, there are options to select backing chords to guide both the machine learning model and the musician in playing. There is also a timeline divided up into sections where either the user player will play or the AI will play. This will simulate the back-and-forth nature of a musical trading session. Once these are selected, it's as simple as pressing a button to start. To implement the AI model, we use Magenta.js, a TensorFlow machine learning library that provides musical note-based models on the client-side of the browser. With Music For Me, we have tried to find the right level of flexibility for new and experienced musicians to have a worthwhile improvising experience, as well as use AI in an imaginative way.

Chapter 2: Requirements

Below are the functional requirements, non-functional requirements, and design constraints of our platform. These are actions that our platform must complete or specific ways that our platform must be built.

3.1 Functional Requirements

- Allow a user to connect a MIDI device to the browser
- MIDI input should playback through web audio as a chosen SoundFont
- Allow the user to trade playing bars with the AI
- Record user MIDI input such that it can be played back
- Playback created music made with the AI after the session is finished
- Play chords over user playing and AI sequence playing
- Change chords that play in the background
- User interface displays notes played and generated in real-time
- Schedule recording and playing events with accuracy
- Play a metronome when session is active
- Allow user to change tempo
- Count off sequence before session starts
- Session continues until user presses stop

3.2 Non-Functional Requirements

- Experience should be in near real-time
- UI is intuitive and easy to use
- Amateur musicians should be able to understand the settings
- App should be able to run with decent performance on most computers

3.3 Design Constraints

- Web application

- Compatible with most browsers
- Compatible with MIDI devices
- Non-mobile devices only

Chapter 3: Use Cases

2.1 Use Case Diagram

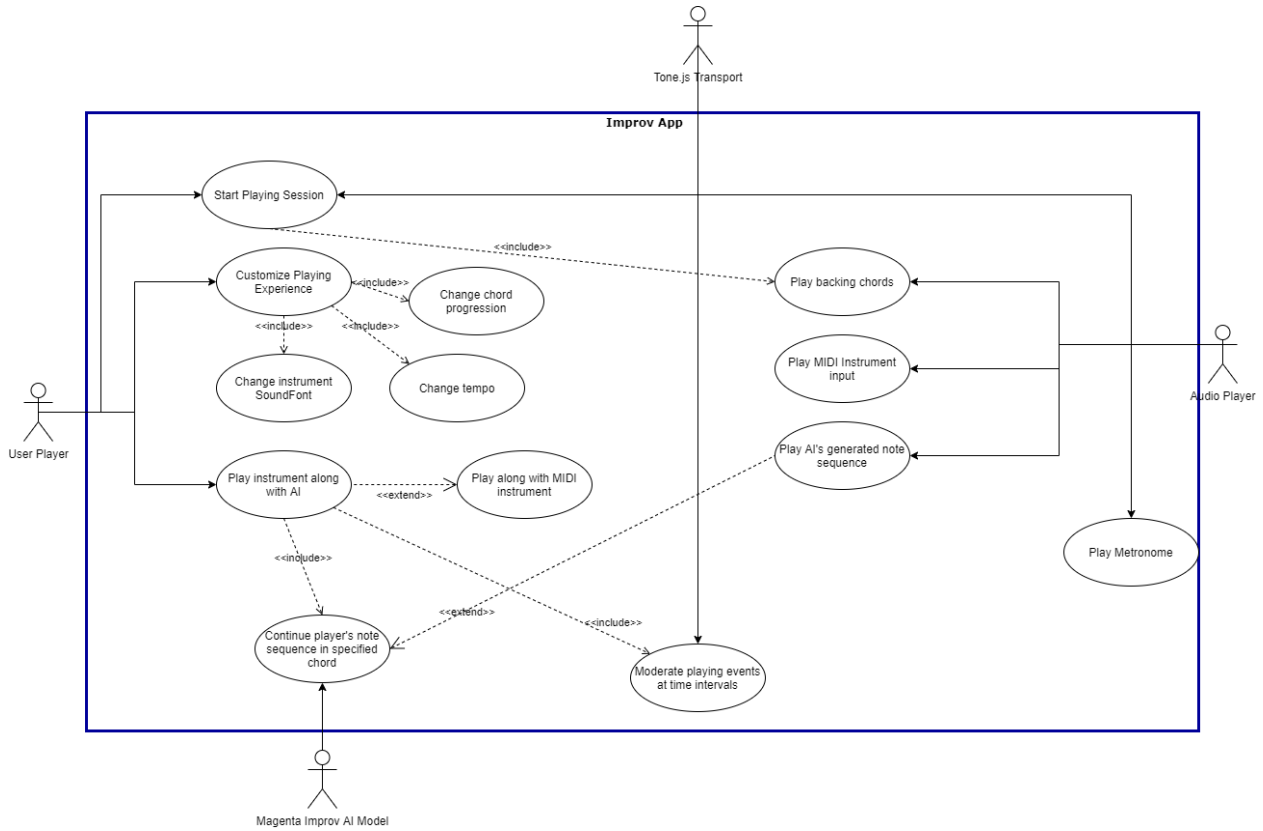


Figure 1: Use Case Diagram

2.2 Use Case Description Table

Use Case #	Feature	Description	Priority
1	Start Playing Session	Initiates the start of the session where the user and AI are ready to start playing together and trading	Very High
2	Customize Playing Experience	Options are available that will change the nature of the playing session in chords, speed, etc.	High
3	Change chord	The backing chords will be changed to affect the	High

	progression	AI model's parameters and the backing chords	
4	Change instrument SoundFont	Audio output when user and AI play can be many different selected instruments (piano, guitar, etc.)	Moderate
5	Play instrument along with AI	The user can trade bars with the Magenta AI so it will be like playing with the AI as you would with a human	Very High
6	Play along with MIDI instrument	The user can use a MIDI keyboard as their instrument to play with the AI	Very High
8	Continue player's note sequence in chord	Given a user-made note sequence, the AI can take the sequence and generate a continuation of that sequence with a chord accounted for in the algorithm. This will be done by Magenta's ImprovRNN	Very High
9	Moderate playing events at time intervals	Will enforce when things will be played and recorded in a way that is seamless to the user	High
10	Play MIDI instrument input	When the user plays a note on their MIDI instrument, the audio will playback that note	Very High
11	Play backing chords	When the user and the AI are playing, there will be a backing chord sequence playing to accompany	High
12	Play AI's generated note sequence	Given the generated sequence, the audio will playback that sequence with the correct tempo and timing of notes	Very High
13	Play Metronome	Throughout the session, there will be a metronome to keep time for the user	Moderate
14	Change Tempo	User can change playing tempo before session starts and is reflected in timing of events	High

2.3 Conceptual Model

Given our importance on ease-of-use, we want to keep our GUI clean and simple. As such, we provide the most important settings, such as instrument type, MIDI device, and tempo. Featured most prominently should be the musical output visualizer, as well as the recording and playback buttons which are vital for interacting with it. We aim to provide an interface that is largely understandable at first glance. The interface will have subdivisions within it that will indicate whether it's where the AI or user will be playing and the chord assigned to that section.

Settings that would clutter the main page like setting chords will be set aside in context menus and popups so as to keep the GUI simple while incorporating the depth of options we wish to include. We will tune these hypothetical settings so as to be perfect for casual musicians while giving more control to those who wish for more customization.

Chapter 4: Technologies Used

Listed below are the technologies we plan to use to build *Music For Me* at this time.

4.1 Functional Technologies

Magenta.js

Magenta is an open-source research project created by Google Brain that explores the role of machine learning in the creative process. They offer many different features but the most important thing provided for us was the out-of-box music Machine Learning models created using recurrent neural networks. These models were trained on many different MIDI files by the Magenta team. We are specifically using the Improv RNN which allows us to provide a chord progression to generate the next musical sequence over. For our purposes, we use its `continueSequence()` function to respond to the user's playing and continue in a similar style. It also provides a `NoteSequence` class that stores music note sequences in a MIDI-like structure. This API is vital to our application as it provides the main functionality of playing music with an AI.

Tone.js

Tone.js is an audio library that abstracts away WebAudio and allows us to time events and creates samplers to play music notes. Without Tone, working with Web Audio would've been much more complex as the Web Audio API is more intended for those familiar with audio and music engineering. The main features that we used were the features that let us play samples of instruments with a simple interface, and the global transport object. The transport object allows you to schedule and synchronize events within a timeline. This element is very important to us because our application requires many events to play or occur at the true time we specify.

WebMidi.js

WebMidi.js is a library that allows us to easily interact with MIDI devices and the WebMidi API. It can listen for MIDI events and respond to them with relative speed and ease. We arranged it so we play the instrument sampler provided by Tone.js on the note down event, and then record the note played and the time it was played.

JavaScript

JavaScript is the main language we will use for this project as it is used natively on the web. We will use it for managing UI and also the logic of our application.

4.2 UI Technologies

HTML5

HTML is a standard language we will use in order to develop the front-end of our application. It's very useful for playing audio and structuring our application.

CSS

CSS was used to make our application look presentable and provides it with a good user interface and experience. We have continuously changed this for the color palette and arrangement of visual elements.

React

React allows us to easily manage the UI of our application in components so we can separate our logic into these components. It also makes it easy to host and test our application in the terminal instead of the browser console. This also makes it easy to host on websites

Chapter 5: Functional Diagrams

5.1 System Sequence Diagram

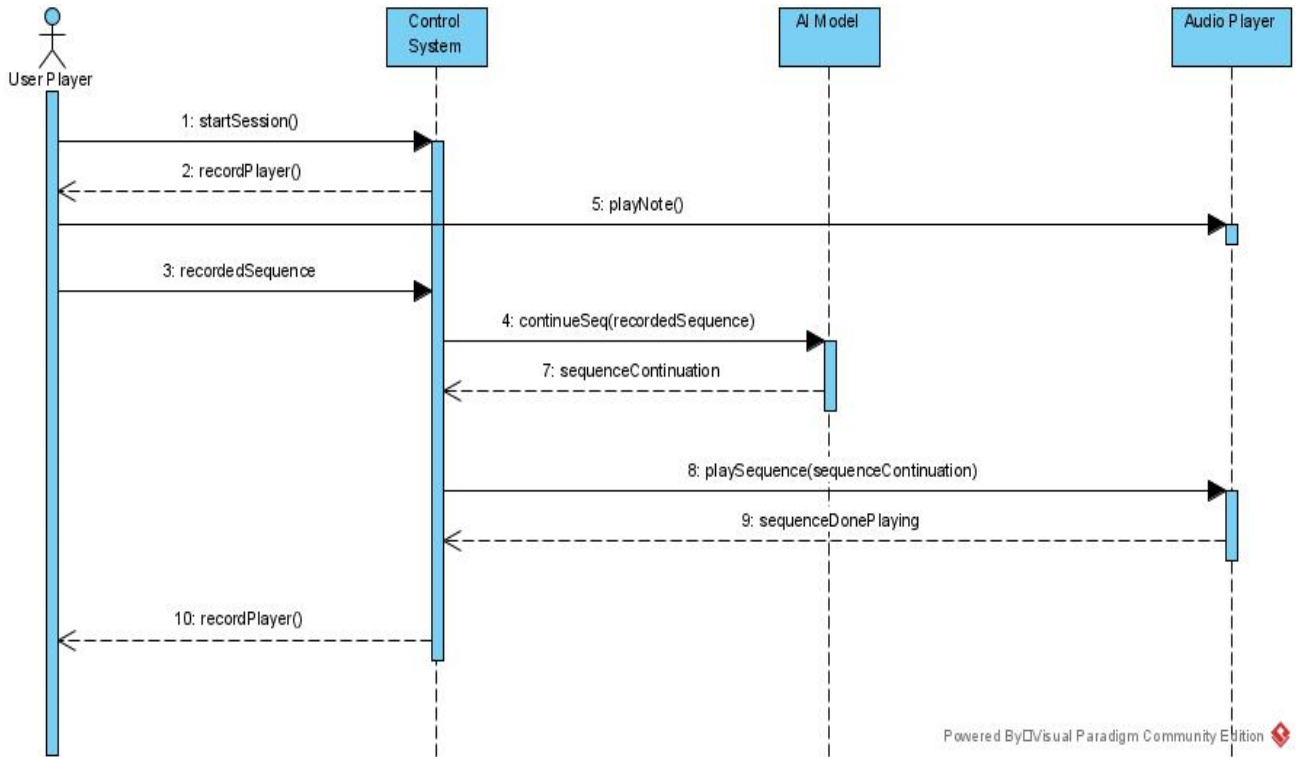


Figure 5.1: Sequence Diagram

5.2 Architectural Diagram

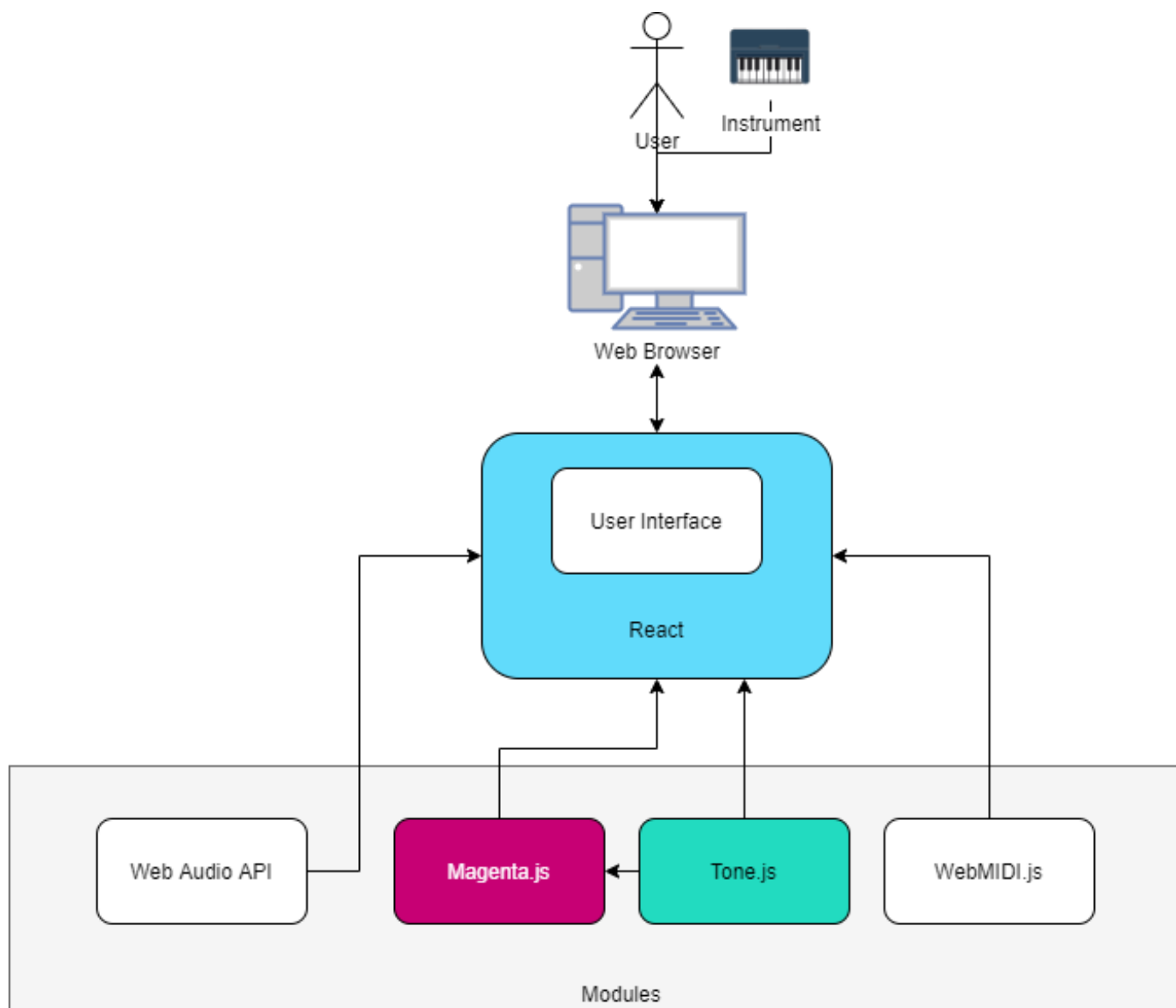


Figure 4: Architectural Diagram

5.3 Architecture Diagram Description

Our architecture is primarily front-end oriented. Since we are trying to make latency as low as possible, we are using front-end libraries to support the React app, where our application and UI logic resides. Magenta.js sits as a front-end library that provides the Machine Learning models we'll be using. The fact that Magenta provided machine learning models that can be run in browser was a large motivation to make our app front-end oriented. Magenta is built on top of Tone.js which schedules all the events in the application as well as provides audio playback functionality. WebMIDI.js is the intermediate between the user's MIDI instrument and the application.

Chapter 6: Design Rationale

The rationale behind making the application a web application as opposed to a desktop app, which some similar programs are, was for the ease of access, namely the ability to immediately begin using the app without needing to download and set up a desktop application. A web app only requires navigating to the website to begin, which suits the desired simplicity of this application. Additionally, given the use case of being a pick-up-and-play type of application, there is very little need for persistent file storage (i.e. accessing a previous day's session) which a desktop app would be superior at. The only persistent storage required by the application would be in settings, which are simple for a web application to store using cookies or similar tools. A web app provides a friendly and inviting interface that provides high ease of access at the cost of complexity which would not fit this application's desired use case anyway.

The web application is built on top of React. React utilizes a Virtual DOM to dynamically reload only parts of the web page that have changed, without needing to recalculate and reload the CSS and layout of the entire page, which is what normally happens when directly updating the DOM. This normal method of the DOM updating is efficient when navigating to different web pages on a single website that would require reloading everything regardless, but becomes very inefficient when trying to dynamically change components on the same page. As such, React's ability to distinguish only the components that have been updated, and selectively reload those, makes it well suited to single-page web applications, which this program is.

Chapter 7: App Implementation

7.1 Overview

Our application is built on Node and npm to manage the execution and packages of the program, respectively. React served as the framework and frontend library, and HTML and CSS were used to create the frontend. JavaScript provided the logic for the program, alongside the essential JavaScript packages Magenta, Tone, and WebMidi.

7.2 UI/UX

Most of our UI components were built using React-Bootstrap, a third-party package that rebuilds the UI components of Bootstrap as React components. Our UI can be divided into two main components; the settings/control buttons, and the piano roll that visualizes notes (Fig. 7.1, 7.2). We've also applied a modern color palette to the entire UI to make the interactive elements pop out to the user

For controlling the application, there were three simple types of React-Bootstrap components used. The first was buttons, used for the single-click components of the UI. Secondly, dropdowns were used for instrument and MIDI device selection. Finally, a modal was used to place the chord progression settings inside (Fig. 7.3). Additionally, button groups and form control components were used to group the UI elements within the code.

The most critical component of the UI, the note visualizer, was built not with React-Bootstrap but with the standard `<svg>` HTML tag. Pre-packaged note visualizers provided too little control over the visualizer, but a custom SVG allows that control to come back to us. `<rect>` objects were used to represent the notes, and a custom React component built around a `<rect>` was used for the sliding bar on the visualizer. We also color-coded each section for the user player and AI sections to make it clear to the user what to expect for each section.

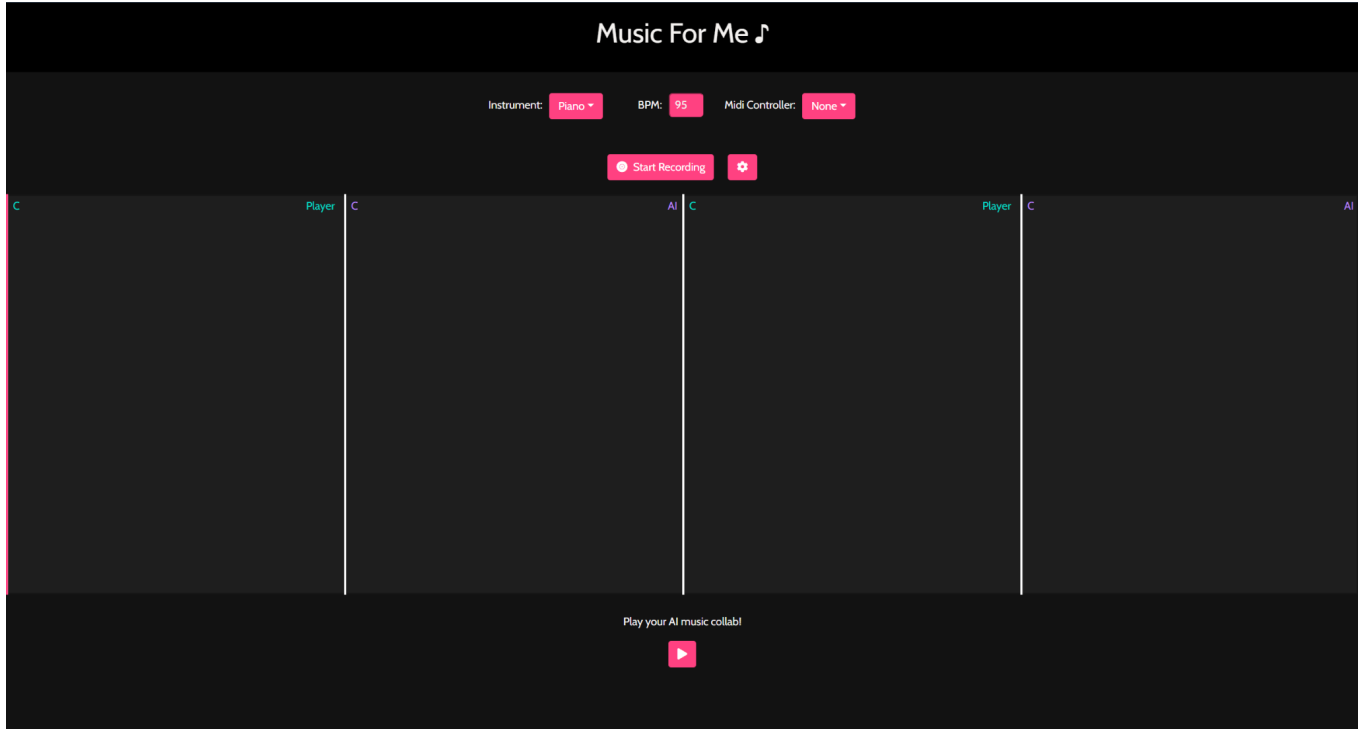


Figure 7.1: UI before Session starts

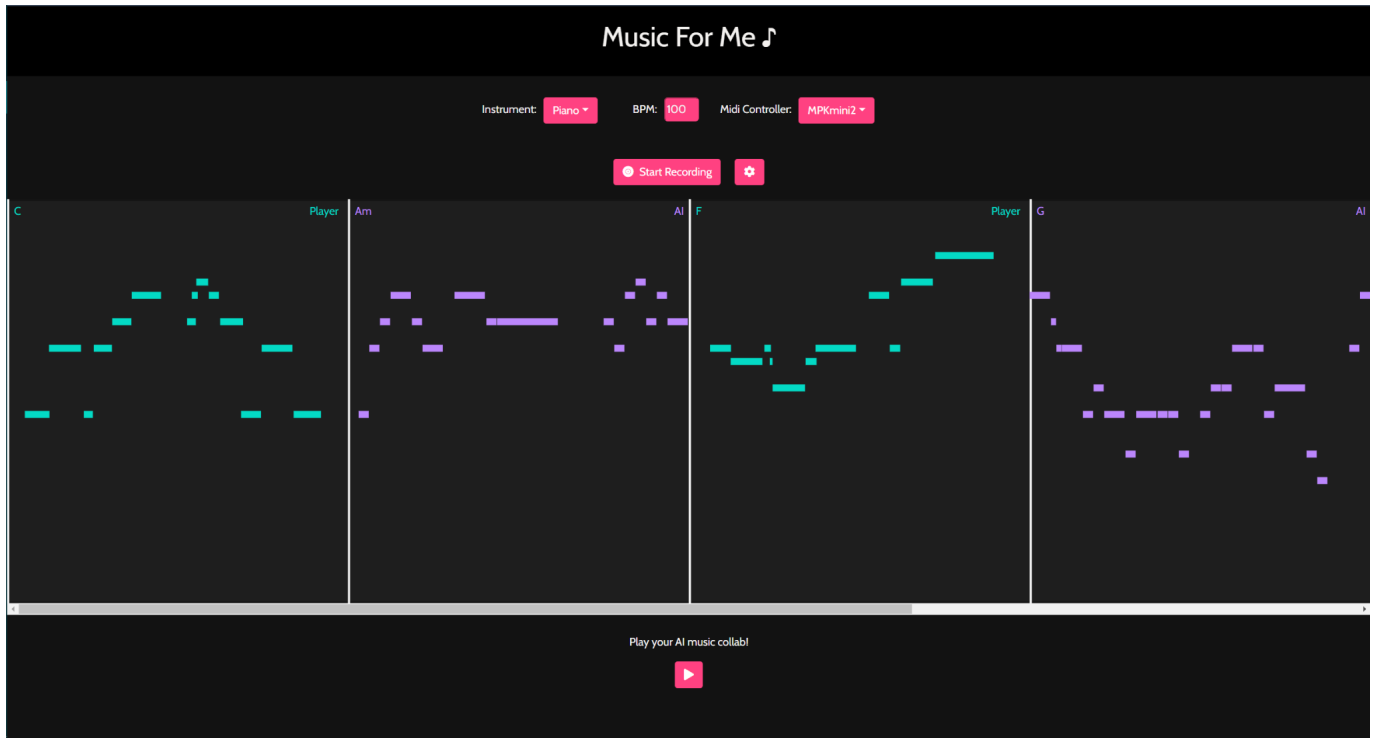


Figure 7.2: UI Post Session

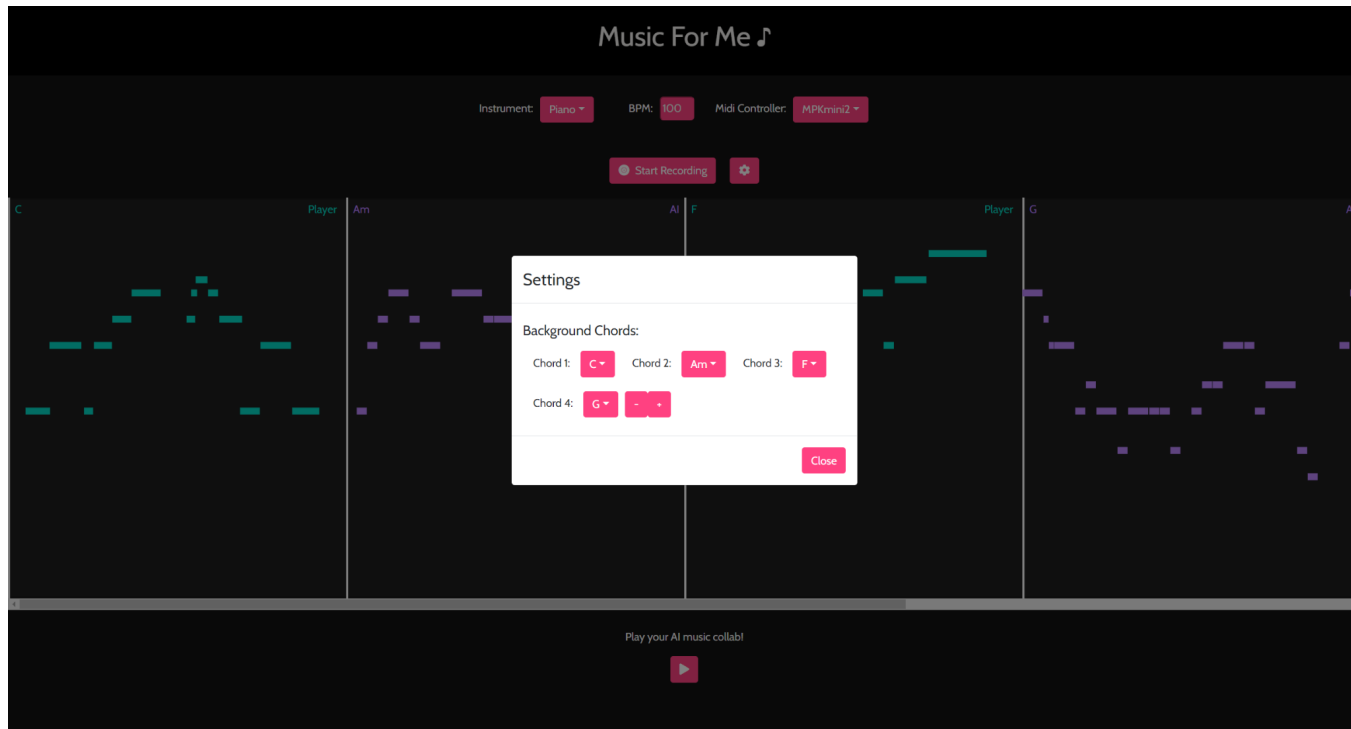


Figure 7.3: Chord Settings Box

7.3 Application Logic

The critical logic components in Music For Me are the recurrent neural network that creates improvised music and the timing control that enables the audio to stay in sync with the visuals.

The recurrent neural network was created by the Magenta team and repurposed for our uses in our web application. This network, called Improv RNN, in addition to providing us with the core functionality of music generation, also supports the usage of chord progressions to condition the generated music, which separates it from other models created by the Magenta team. By feeding the Improv RNN the notes from the session, including all previous player and AI-generated note sequences, our application creates and then plays its own generated note sequence.

The timing control was provided by Tone.js, which was also used to play or pause the generated notes. Tone provides a Transport object that keeps time and allows for the scheduling of events. This Transport object was used not just to control the scheduling of notes and backing chords, but also within the piano roll visualizer to ensure the graphical representation of the notes, stop and start points, and the bar that shows the current position in the sequence are all properly in sync.

Chapter 8: Development Timeline

Here is a timeline of the development milestones we made whilst working on this project.

Fall Quarter 2020

- Refined conception and planned implementation, requirements, user experience
- Created a rough prototype as a proof of concept and for future development

Winter Quarter 2021

- Refined, optimized, and added more features through iterations of development, working off the prototype
 - Visualizer
 - Infinite playing session
 - Drawing of notes on Visualizer
- Researched alternative implementations of existing features to optimize processes
 - Ended up using Tone.js more for scheduling events
- Ongoing tests with team members, advisors, and peers

Spring Quarter 2021

- Completion of feature implementation
 - New instruments
- Bug Fixes
- Complete front end development for aesthetic and UX purposes
 - Modern color palette
 - Effective arrangement of visual elements
- Completion of Final Senior Design Thesis and Design Conference

Chapter 9: Ethical Analysis

This application offers a new way of interacting with AI and playing music. This can be hard to grasp because playing music is what makes us human and if we let automated systems create it, are we dehumanizing music? How can AI provide any benefits in artistic fields such as music? Like with most applications of AI, we believe that AI is simply a tool for humans to use in conjunction and not something that should completely dominate a task. In our case, it's a tool to practice playing music or help start writing songs perhaps. Our application is largely human-driven. The user player initially starts playing and then the AI has to respond giving most agency to the player. We also don't view AI as a substitute for another human to play music with. There are some things that AI can't replace about the joy and creativity that comes with playing with another human. We simply believe that it's worth looking into what novel ideas musical AI can give humans in creative endeavors. Also due to the COVID-19 crisis, playing with other people has been hard so we hope this could be a useful tool when you can't play with other musicians.

With regard to how we handled this project as engineers, we believed that we did well following the habits of ethical engineering. As covered before, we are aware of the impact we mean to make within Techno-social society and the proper use cases of the application. We also believe that our project is doing the public good of empowering musicians with technology. Music is important to society on many levels. On a teamwork level, we understood our strengths and weaknesses when working on the project as well as the different work styles we brought to this project. We accommodated this by setting deadlines for ourselves and having a consistent work schedule.

Analyzing the risks of the project, we don't believe our application will put any user in physical danger which is the nature of webpages as they are. If we mean to host this for the public, we would have the obligation to our users that it will be a secure webpage that won't be used for malicious purposes. We'd also keep the obligation to make sure that the website is as user-friendly as possible.

Chapter 10: Conclusion

10.1 Summary

In conclusion, our team has designed a web application capable of improvising music alongside a human using machine learning AI techniques. Using a minimalistic GUI and intuitive controls, a pick-up-and-play style of experience is created, where even beginners can easily jump into an improvisation session. Music For Me offers backing chords, a metronome, and clear stop and start indicators to create a more controlled experience. Additionally, Music For Me contains various options for the user to customize their experience, namely custom chord progressions, tempo control, and instrument sound.

10.2 Obstacles Encountered

For the majority of the project, the Covid-19 pandemic prevented us from working together in person, which, given the physical component present in playing music, was occasionally an issue. Fortunately, our project being a web application helped to minimize this problem most of the time.

One obstacle we encountered very early on in the project was when we attempted to allow our application to interpret notes from the live audio of an instrument playing rather than receiving MIDI input directly. While this feature is possible, it quickly showed that it would become far too time-consuming, and so the idea was abandoned for the sake of the greater project.

Other features also provided difficulty in implementation, such as the various iterations of the piano roll Music For Me went through. However, these were tackled one at a time and solved, and none of the other features we worked on had to be abandoned.

10.3 Lessons Learned

Planning, designing, and developing this web application taught us valuable lessons in project development. Planning out a project to be created over the course of nearly a year allowed us to

develop valuable skills in planning, coordinating, and distributing work. Separating out the features we wanted into small tasks allowed us to incrementally and steadily improve the working prototype we came up with very early on in the project.

Another important lesson was in the value of version control. Using Git made keeping track of our work and combining it together very easy, even though we had to work together remotely for the majority of the project.

10.4 Future Work

Our team has plans to continue with Music For Me in some capacity. Current plans for the future of Music For Me involve publishing it as a standalone, open-source project as opposed to continuing as solo developers of the project.

There are also features and improvements that our team discussed but did not implement, be it for reasons of time, feasibility, or something else. These could be revisited in the future and implemented, either by us or others. Early on, we theorized a feature, also discussed in section 8.2, that would allow a user to play a normal instrument, and the application would convert live playing into a MIDI sequence that the AI could respond to. This was scrapped due to the high difficulty but would provide a serious improvement to the application. Other smaller ideas include machine learning models for different genres of music or optimizing the machine learning model to reduce the delay between the end of a user section and the beginning of an AI section. These steps would help turn Music For Me into a more fleshed-out application.