LSTMusic: A Web-Based AI Melody Extension Tool

Generating music from user-created melodies.

Thomas Dent

Candidate 246518

School of Engineering and Informatics

Supervisor: Kingsley Sage

2024

Abstract

Within the domains of art and image editing, tools and toys are surfacing which allow non-professional folk to expand and augment digital content to a standard which previously would require a long processes of skill acquisition and refinement through professional software.

This report recounts the research, implementation, successes and failures of LSTMusic, a melody extension web-app designed to bridge this gap within the domain of music, allowing users to experiment with and benefit from generative AI beyond the confines of professional music production software.

Statement of Originality

This report is submitted as part requirement for the degree of Computer Science with Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

I hereby give permission for a copy of this report to be loaned out to students in future years

Acknowledgements

I would like to give a massive thank-you to my project supervisor, Kingsley Sage, for his continual support and guidance throughout the duration of this project, and for being an incredibly engaging professor throughout my years at Sussex.

Contents

1 Introduction

1.1 Project Overview and Inspiration

Within the past few years, there has been a notable surge in the popularity and mainstream usage of AI-driven tools that aim to complement user creativity, increase productivity, or just serve as a form of entertainment by augmenting, improving, and expanding user-provided content with machine-generated elements.

These expansion and augmentation tools are now widely accessible through a diverse range of user-friendly desktop, web, and mobile apps, with many companies now opting to integrate AI directly into their platforms for fast and convenient use.

For instance, Microsoft has incorporated AI features like 'Designer', an AI image generator, and 'Copilot', a text-based AI Assistant into their new Windows 11 Operating System[37]. Samsung, too, have implemented several new AI-Driven features on their new Galaxy S24 smartphone like 'Note Assist', a note generation and summarising tool built into their Notes app, 'Generative Edit', an AI-powered image editor which can fill parts of images in with AI generated elements, and various other AI-powered features that come standard with the phone[51], requiring minimal setup from users.

The widespread development and adoption of these tools indicates a demand for user-friendly AI applications, particularly in creative realms where users traditionally must undergo a gradual process of learning and skill refinement.

With AI now capable of assisting and augmenting individuals' creative content, it raises the question of whether this type of easy-to-use AI will make its way to the realm of music.
Within this space, research initiatives have been coming from prominent organisations such as OpenAI[45], Google[57], Microsoft[36], and Meta[11], all with the shared goals of providing AI support in the music creation process.

One part of Google's AI Music generation research project, 'Magenta Studio', stands out as an noteworthy inspiration for this project. Providing a set of AI-powered tools to Ableton's 'Live' music production software in the form of a downloadable plugin that works alongside the production software, allowing users to expand their musical tracks using machine learning technologies with its 'Continue' tool.[50]

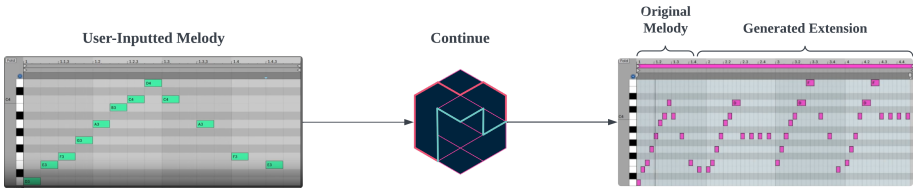


Figure 1: A simple diagram showcasing the use case of Magenta Studio's 'Continue' tool.

However, the requirement of learning to use Ableton Live (or learning how to run any provided code when investigating the other research initiatives) can be perceived as a barrier for those who simply wish to experiment with the technology without requiring additional software installation or skill acquisition.

Therefore there currently exists a gap in the domain of music between the skills required to use these available machine learning tools and the existing skills of novices. Applications like Designer and Generative Edit address this gap within their creative domains, with this 'bridge' seemingly not yet existing within music creation.

LSTMusic has been created as a platform to bridge that gap, with this report detailing the planning, design and implementation of a machine learning powered melody extender, with the goal of providing a platform for users to experiment, toy with, and benefit from generative AI in music *beyond* the confines of professional production software to eliminate the difficulties linked to installing, mastering, and operating music production software, and instead being accessible as a

straightforward, easy-to-use web app.

1.2 Project Relevance & Motivation

Several aspects of this project draw and build upon skills which I have acquired during my time at Sussex, including algorithmic thinking, iterative development, understanding Human-Computer-Interaction principles, proficiency with programming languages, and use of version control systems.

I recognised towards the project's outset that the provided project specification was incredibly flexible, giving me the freedom to shape this project in any direction I desired. This motivated me to view the project as a valuable opportunity for personal growth, prompting me to chase new skills and learn programming languages and technologies that were previously unfamiliar to me.

Stepping beyond the confines of my usual comfort zone of simple Python applications, I set out to make the project something which was aspirational, yet attainable, leading me to specify my project in the way I did in my Project Proposal.

This allowed me to establish a set project aims that would push me to gain an understanding of concepts and areas of computing which I never previously looked, including:

- The use of HTML, CSS and JavaScript.
- The usage of web frameworks.
- Website design.
- Website hosting.
- Fundamental concepts of music theory.

Additionally, I've long had a personal interest in picking up web development and hosting for use in personal projects, making this an ideal opportunity to acquire this vastly applicable skill.

In terms of the project's objective, there is currently a gap in the availability of online melody extenders. While existing platforms[16] allow for user submissions through traditional audio files, the incorporation of a piano roll for user input does not currently exist on any web tool, with the closest implementations being limited to music production software as plugins, such as Magenta Studio[50].

2 Project Aims

As stated in section 1.1, the general aim of this project is to develop a web-based system which allows users to extend their melodies through the use of AI.
This goal was split into a set of fallback, primary and stretch objectives in my Project's Proposal, being updated to better scope the project around the time of the project's interim.
The details of these objectives can be seen in the below sections, with some slight rephrasing from my interim report.

2.1 Core Objectives

- Design and implement a website frontend that allows users to input short, note-by-note melodies into a piano roll.
- Explore music generation techniques which allow for the extension of melodies.
- Develop a backend which implements a music generation technique.
- Connect the front and backend using a suitable web framework.
- Host the tool online.
- Evaluate the tool's effectiveness through qualitative assessment to determine areas of improvement.
- Reflect on the successes and challenges encountered during the project.

2.2 Extension Objectives

- Add complexity to the notes which users can input into the piano roll. (i.e variable note durations, velocity adjustment)
- Allow the tool to generate multiple extensions from a supplied melody.
- Cleanly stitch the extensions to the originally submitted melodies.
- Investigate the plausibility of implementing the system into a Digital Audio Workstation (DAW).

2.3 Fallback Objectives

- Create a tool which allows users to input short melodies into a piano roll and output MIDI files.
- Experiment with, and create a music generation model which outputs something resembling music.

3 Professional Considerations

The bulk of this project involved the research and development of a web application, with minimal ethical concerns. Later stages of the project involved basic user testing and user-evaluation.

All evaluation performed within this project was designed to follow the 12 points detailed in the "User Testing Compliance Form for UG and PGT Projects". A signed copy of this form can be found in the appendix of this report.

3.1 Public Interest

The final product of this project does not contain any content which would negatively impact the public health, privacy, security or well-being of others or the environment. Any third-party libraries, software or ideas used are rightfully credited and licensed for their work and contribution to this project.

My activities regarding this project have been conducted without any discrimination on the ground of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age, disability, or of any other condition or requirement.

The product of this project is a publicly accessible website that anyone can use, providing equal access to any benefits it can provide.

3.2 Professional Competence and Integrity

The development of this project used skills which I already learned throughout this course. Any elements of the project that required learning new skills were determined to be within my capabilities, as suitable university and external resources existed to acquire these skills.

Continual evaluation on the suitability of the project's scope during development was carried out, with appropriate alterations being performed to the project's requirements on an as-needed basis when issues were encountered.

I ensured throughout the project that I was knowledgeable of any relevant legislation related to the project, and avoided injuring others, their property, reputation, or employment by any false, malicious, or negligent action or inaction. Any forms of bribery or unethical inducement I encountered during this project were to be rejected and reported to my supervisor.

3.3 Duty to Relevant Authority

Throughout the project, I aimed to avoid any situation that could have created a conflict of interest between me and my project supervisor. I followed any guidelines that were imposed by my supervisor or the University.

I did not, and will not, disclose confidential information without my project supervisor's permission, except as required by legislation.

Throughout the project I aimed to be honest about the project's performance, avoiding misrepresentation or withholding of information, and aimed to not exploit others' lack of knowledge or experience, unless bound by a duty of confidentiality.

3.4 Duty to the Profession

Throughout this project I aimed to uphold the reputation and good standing of the BCS and the profession, seeking to improve professional standards through participation in their development and use of enforcement.
I aimed to support and encourage my fellow members in their professional development, providing opportunities for professional development of new members and students when possible through mutual assistance with other IS professionals to further the reputation of the profession and assist individual members.

Part I
Pre-Development

4 Research

The majority of this project was dedicated to the design and development of a web application, with a period of background research being carried out towards the beginning of the project.

Other research was performed on an as-needed basis when the necessary knowledge was required for development to continue, typically at the beginning of new prototype iterations when familiarising myself with tools, libraries, technologies or languages through documentation and online resources. These resources were typically in the form of user guides and tutorials aimed at teaching and guiding beginners through setup processes, proving to be very helpful given the learning-focus of this project.

4.1 Music Theory Concepts

A basic understanding of some core music theory concepts was required throughout the duration of this project. Working with music and Machine Learning (ML) systems naturally required the need for pre-processing steps, which, in the context of music, required knowledge of music 'normalisation' strategies.

This basic understanding would also allow me to accurately articulate my ideas using the correct terminology, allowing me to find the correct resources and functions within the libraries and packages used, and to accurately explain to others the steps that the project takes when working with music.

Below is a list of the core music theory concepts which were explored and used throughout this project. Brief explanations of all these concepts can be found in this report's Appendix.

- Monophony and Polyphony (Musical Layering).
- The properties of Notes (Duration and Pitch).
- The Key, Tonic and Mode of music.
- Musical Intervals.
- Transposition over Intervals.

4.2 MIDI

Unlike conventional audio files that store music as a waveform, MIDI (.mid) files function descriptively, storing instructions for musical performance which can be executed by specially built programmes like MIDI Sequencers and MIDI Players[2][58].

MIDI files comprise one or more tracks containing numerous timestamped musical 'events' and a metadata-rich 'header' containing details like the file's time signature, key, tempo, and copyright information.

For this project, I was mainly concerned with 'NoteOn' and 'NoteOff' events, which correspond to the activation and release of a note played by a virtual instrument[3][58]. The pitch of the played note is contained within this event as an integer, creating units of information which can be thought of as a tokenised representation of a song's instructions.

A MIDI file's events can be manipulated through several methods, like through MIDI sequencers, Digital Audio Workstations (DAWs), and importantly for this project, code.

Numerous packages and libraries, available in both Python and JavaScript, allow for the modification and creation of MIDI files, such as Music21[13], and Midi-Writer-JS[24]. These tools,

alongside some others, will be discussed further in Section 4.3.

MIDI's capacity for tokenisation (which could simplify the process of encoding music into an Machine-Learning-friendly format), its capability of being edited through code, and the availability of varied online datasets ([26], [31], [35] to name a few) made it an appealing choice for the project. And despite eventually not using these advantages, MIDI's widespread adoption by music production software and sequencers made exporting extended melodies as MIDI files a clear choice.

4.2.1 MIDI Messages

The above section describes what is known as the "Standard Midi File", which is a storage medium for the MIDI protocol, which predates MIDI's `.mid` file format. The MIDI protocol was originally created as a communication protocol between keyboards, containing the same information stored in the events of a Standard Midi File, like note on/off messages, velocity information and pitch bend, as 'Messages' which are transferred in real time between devices.[58]
This protocol is still used today to connect keyboards and other instruments to computers, allowing music to be inputted directly from the instrument to be parsed by a DAW.

4.3 Music Packages and Libraries

Several external Packages and Libraries were used throughout the duration of this project to read, write, analyse and modify musical representations.

4.3.1 Music21

Music21[13][12] is a Python Library used throughout the backend of this project. It provides an extensive set of tools to interact with several popular formats of deterministic music representation, like MIDI, KERN[30] and MusicXML[22].

Music21 has the capability to parse these files, converting them into an internal `Stream` representation [40] which contains the musical data of the parsed song.
These tracks are built up of a series of core 'building block' objects.

Notes
Represented by the `music21.note.Note` class.
Music21's Note Objects contain the same fundamental information which actual notes in music hold. They hold characteristics like the pitch and duration, defining the note.

Rests
Represented by the `music21.note.Rest` class.
A Rest is an object with a duration but no pitch. It can be thought of as a note of purposeful silence.

Notes and Rests can be added to an already existing, or brand new, music21 Stream, allowing musical tracks to be created from scratch, or appended to using a simple `.append()` function[40], proving advantageous for a project which is concerned with the extension of pre-existing melodies.

Beyond these core building blocks, Music21 includes other classes that can capture note relationships, melody properties, and analyse melodic and rhythmic structures.

Key
Represented by the `music21.key.Key` class.
The Key class in music21 represents musical key. It holds information such as the key's tonic pitch, and the mode. It is an essential class for any tonal analysis of a melody.

Music21 conveniently provides functionality to analyse a melody's key through it's `analyze()` method found in the `Stream` class, if a track doesn't already provide information about its key in its metadata, which Music21 can also read.

Intervals & Transpositions
Represented by the `music21.interval.Interval` class.
The Interval in music21 is a complex class which has incredible flexibility. For the purposes of this project, it is used as a descriptive class to represent an interval between two different keys[39].
It can be used as an argument within a transposition function to transpose a melody to a different key, proving handy for key normalisation of music.

All of the above classes and functions had extensive use throughout the backend and proved invaluable for the manipulation and creation of music files.

4.3.2 Midi-Writer-JS

Midi-Writer-JS[24] is a JavaScript library found in the initial research phase of the project. It was used as part of my first frontend prototypes to collect user input into a MIDI track. It operates in a similar, but massively simplified, manner to Music21, allowing programmatic creation and alteration of MIDI files through Note and Rest objects.

```
const track = new MidiWriter.Track();

const note = new MidiWriter.NoteEvent({pitch: ['C4', 'D4', 'E4'], duration: '4'});

track.addEvent(note);
```

Listing 1: Javascript code showing the creation of a simple MIDI track using Midi-Writer-JS.

This allowed for saving of inputted melodies in the project's first frontend prototype as part of the testing process.

4.3.3 HTML-Midi-Player

HTML-Midi-Player [29] is an open source project by Ondřej Cífka which provides a set of HTML elements which allow for the creation and customisation of web-based MIDI players and visualisers.

It was a plugin discovered late into the project's frontend development and performs the job of visualising and playing MIDI files in a flexible and clean implementation.

It has seen usage on several demo websites for other music generation models[33][8], and comes with a large potential for customisation which makes it a very attractive library as it, with the correct theming, could nicely compliment my webapp's design.

A screenshot of a HTML-Midi-Player player and visualiser in their default styling can be seen below.

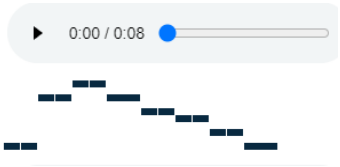


Figure 2: Twinkle Twinkle little star, shown on HTML-Midi-Player's visualiser, accompanied by a play button and seekbar. Source: [9]

4.3.4 WebAudio-PianoRoll

WebAudio-PianoRoll[20] is an open source project created by GitHub user g200kg which was found in the later stages of the project.

It provides an interactive, web-based piano roll in the form of a custom HTML element that allows users to input and play melodies, showcasing an extensive set of customisation options which allow the usage and theming of the piano roll to be customised in a way to suit the project's needs, similar to HTML-MIDI-Player. The open source nature of this project meant I was able to easily explore its codebase and even add desired features as-needed.

Some examples of piano rolls created using WebAudio-PianoRoll can be found below.

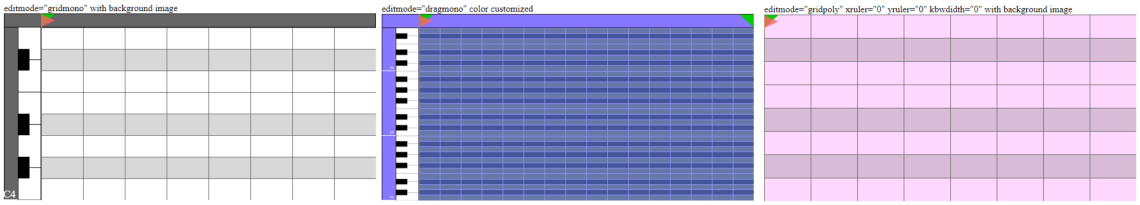


Figure 3: Examples of customised WebAudio-PianoRoll Elements. Source: [21]

4.4 Backend Frameworks and Tools

4.4.1 TensorFlow and Keras

TensorFlow is a flexible library for numerical computations. It enables users to efficiently program and train neural network and other machine learning models and deploy them to production[43]. It has a number of APIs available in different languages, notably Python for this project.

Keras is a higher-level API which sits on top of Tensorflow, aiming to simplifying the process of creating Tensorflow networks.
It offers a range of pre-defined layer modules, which can be stacked together to form a complete model, only then requiring the use of compile and fit methods to build and train it. This simplifies the process of network creation, allowing for quick and easy prototyping[43].

```
# Create Model.
model = tf.keras.Sequential([
layers.Dense(64, activation='relu', input_shape=(32,)),
layers.Dense(64, activation='relu'),
layers.Dense(10, activation='softmax')])

# Compile the model.
model.compile(
        optimizer=tf.train.AdamOptimizer(0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy'])
# Train the model.
model.fit(data, labels, epochs=10, batch_size=32)
```

Listing 2: An example of the Python code required to create and train a simple Neural Network using Keras. Source: [43]

4.4.2 Flask

As this project aimed to be accessible via the web, usage of a web framework was naturally required to allow my python backend to handle web requests. At the beginning of the project, I considered building this backend around Django, but experimentation with it found that it's 'batteries included' nature was rather overkill for this project, with it including tools and functionality for larger scale web apps which I had no use for in this small scale project.

A recommendation from my project supervisor then lead me to Flask[17], which I found to be a much more suitable framework due to its ease of use, fast setup process and lightweight nature. Flask uses a simple decorator system to create URLs which bind to regular python functions within a surrounding Flask app, allowing for rapid testing through an included local development server.

```
from flask import Flask
app = Flask(__name__)

@app.route('/helloworld')
def hello():
    return 'Hello, World'
```

Listing 3: The code required for a hello world Flask application.

4.5 Magenta studio's 'Continue'

As discussed previously in Section 1.1, several research initiatives and existing projects exist to provide music extension in a similar manner to what this project aims to achieve, with Google's Magenta standing out for its unique functionality as both a practical tool and a research endeavour.

Magenta is an open-source research project formed by Google's AI research group to explore the role of machine learning as a tool in creative processes[50]. Magenta Studio is one of Magenta's larger projects; aimed at building AI tools for Ableton's 'live' Digital Audio Workstation, in the form of a set of downloadable plugins.
Magenta Studio consists of 5 plug-ins which all provide unique functionality in slightly different ways, generating melodies and drum tracks from user input. 'Continue' is the most relevant plug-in to this project, acting as a primary inspiration.

Continue is a compositional tool that can be used to generate notes that are likely to follow an input drum beat or melody. The user provides an input clip, and the plug-in will extend it by up to 32 measures using a Recurrent Neural Network (RNN) to assist users in adding variation or new material to a melodic track, typically picking up on musical features like note-duration and key[50].

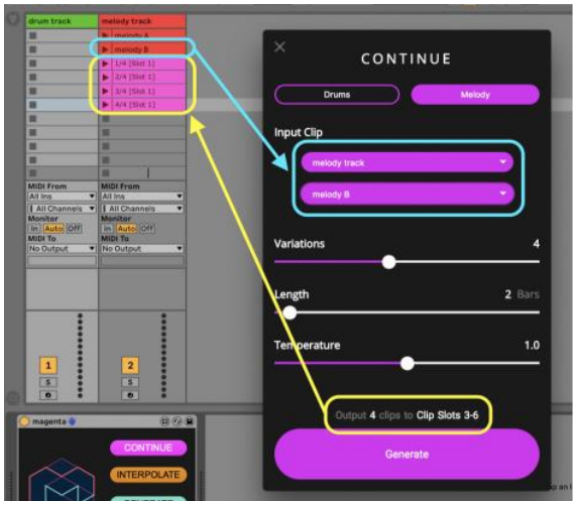


Figure 4: Example usage of Continue. The user selects an input clip (annotated in blue) and the plug-in outputs multiple continuations (annotated in yellow). Source: [50]

4.6 Representing and Encoding MIDI

The main technical task of this project requires the implementation of a model tailored to the generation of music. This naturally requires the use of a model-friendly musical representation, as machine learning models are unable to work with untouched MIDI files.
In the human world, music can be represented in a variety of ways, each with its own advantages and disadvantages. Sheet music is difficult for beginners to pick up, but can hold extremely complex melodies in a dense yet readable format. Piano rolls can be helpful for beginners to remember which keys they should be pressing, but can't be translated well to paper.

This problem holds in the realm of computing too, as a MIDI track can be encoded into a variety of representations. How fit for purpose a representation is for a machine learning task can be heavily influenced by factors like the amount of memory available, the computing power available, and the difficulty of working with the format. With all of these factors potentially affecting the speed of music generation, how quickly prototypes can be made, and the speed that experiments can be carried out, it was important that a suitable musical representation was chosen.

4.6.1 Time Series Representation

A time series representation is essentially a list showing what occurs at each time step of a song. With each index of the list corresponding to a fixed duration 'step' of time within the song, where

an event occurs.

The set of possible tokens used encapsulated all of the functionality which would be required for simple, monophonic melodies to be represented as a list. This set consisted of the following:

- The activation of a new note. (Denoted by its MIDI number)
- The start of a rest. (Denoted by an '`r`' character)
- The prolonging of a note or rest. (Denoted by an underscore)
- The end of a file. (Denoted by a forward slash)

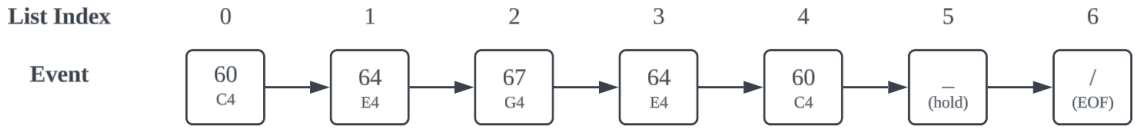


Figure 5: The time series of a C Major chord being arpeggiated.

All together, this set of events could represent monophonic melodies over any period of time, with enough flexibility to make this simple representation sufficient for a first model prototype.

This representation was taken from an online guide created by Valerio Velardo [59] due to its simple nature, allowing for initial prototyping to stay simple and understandable, before tackling any more difficult representation methods.

4.6.2 The Issue of Polyphony

As suitable as a time series representation is for prototyping, its inability to represent polyphonic music leads to a lack of depth when it comes to the music that a model implementing this representation could produce.

Workarounds do however exist for a list-based time series representation. One possible solution could be the usage of combined tokens, which could represent multiple pressed notes at once to represent any key combination.

While this would technically allow a 1-dimensional list to represent *any* key-combination, the shear size of the one-hot encoded representations used by a model's input and output would be ridiculously sparse, likely causing memory issues with approximately 2^{128} output classes being possible across the 128 possible pitches which midi supports.
So without the use of advanced tokenisation methods[1], models using any sort of list-based time series representation seem to be stuck generating monophonic music.

4.6.3 Image-Based Representation

Transitioning from a 1-dimensional representation to a 2-dimensional representation allows for the limitations imposed by the previously mentioned monophonic time-series representation to be bypassed.
By representing each time step as a vector, rather than a single value, a matrix can be produced where each column delineates the time axis and each row signifies the pitch played at that time step. This approach would allow for a representation of data which can support any combination of the 128 pitches MIDI supports, with active notes being denoted by '1' and inactive notes by '0'.[5]

[1] See MidiTok: [19]

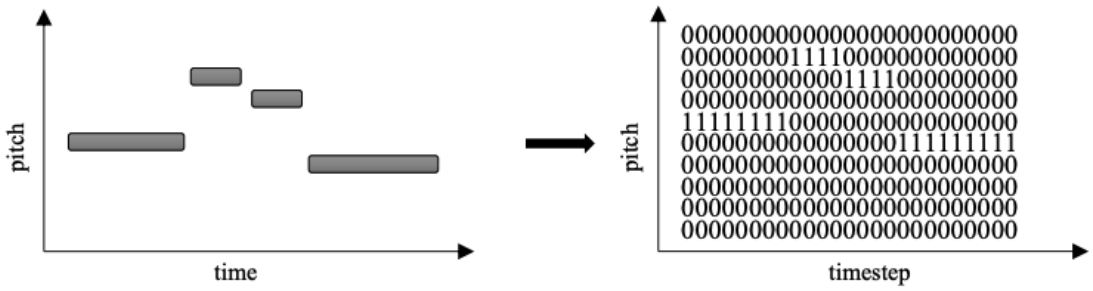


Figure 6: MIDI Notes represented as a piano roll. Source: [5]

Visualising one of these matrices reveals a monochrome image resembling a piano roll, which can be fed into an image-based generative model.

Figure 7: MIDI Notes represented as a monochrome image.

4.7 RNNs and LSTMs

Given the project's primary functionality relied on the implementation of a generative model, research on generative techniques was required.

During the initial research phase, various generation techniques were researched, with a focus on Recurrent Neural Networks (RNNs) due to their usage in Magenta Studio, the main inspiration of this project.

4.7.1 RNNs

Music, with its rhythmic progression, is fundamentally stateful. Each note in a song is a part of a greater sequence to create melody and rhythm.

The anticipation of upcoming notes in a musical sequence requires a prior context that traditional Neural Networks do not have, preventing the recognition and continuation of melodic and rhythmic structures in a musical piece.

Recurrent Neural Networks address this issue, allowing information to persist within a network by feeding a state's output back into the network for future use[41], essentially allowing the network to store an inner representation of the past context that it's operated in.

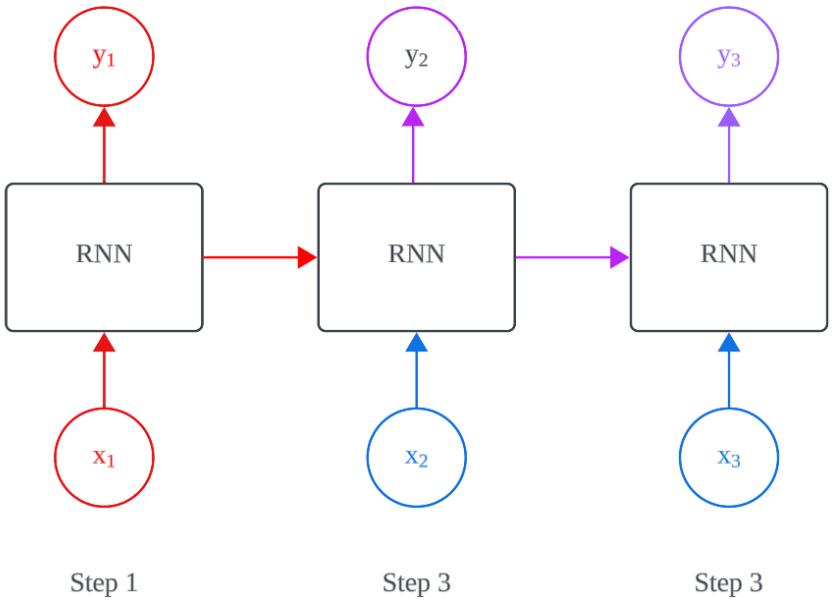


Figure 8: A diagram of an unravelled RNN, illustrating information from an initial step (red) being retained to influence the network's output (purple)

Unfortunately, RNNs encounter an issue of 'forgetfulness' when working with longer sequences, unable to connect the information between large state gaps[41].

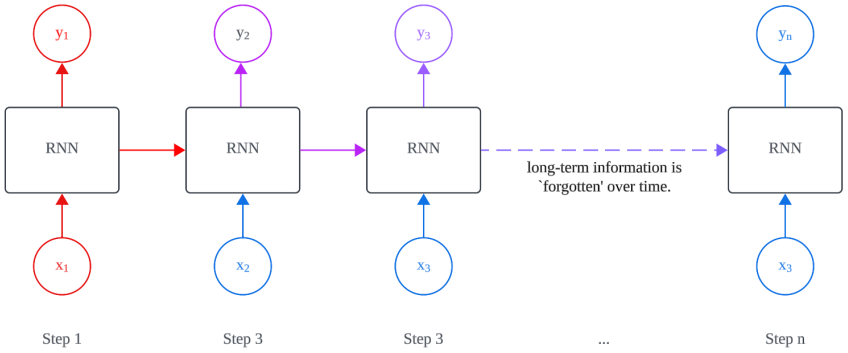


Figure 9: A diagram of an unravelled RNN, with the initial step's red information being 'forgotten' over time.

This phenomenon was explored by Sepp Hochreiter and Jürgen Schmidhuber, who proposed a solution: Long short-term memory (LSTM) cells[28].

4.7.2 LSTMs

LSTM cells are capable of remembering long-term dependencies which allows them to mitigate the problem of 'forgetfulness' in RNNs.
They achieve this by incorporating a memory cell with a set of gates which regulate the flow of information into, out of and within the cell, enabling the LSTM cell to selectively retain or discard information over time[28].
As a result, LSTMs are able to capture long-range dependencies in sequential data by design, making them well-suited for tasks involving using both long and short-term data sequences to influence their output.

This naturally makes them quite an attractive choice for use with music, as a model using LSTMs could potentially 'learn' the long and short-term sequences that are found in music. Many others[10][44], including Magenta's team[54], have explored exactly this, with varying levels of success. All of these factors led me to choose an LSTM-based approach as a starting point for my project's generative model.

4.8 GANs

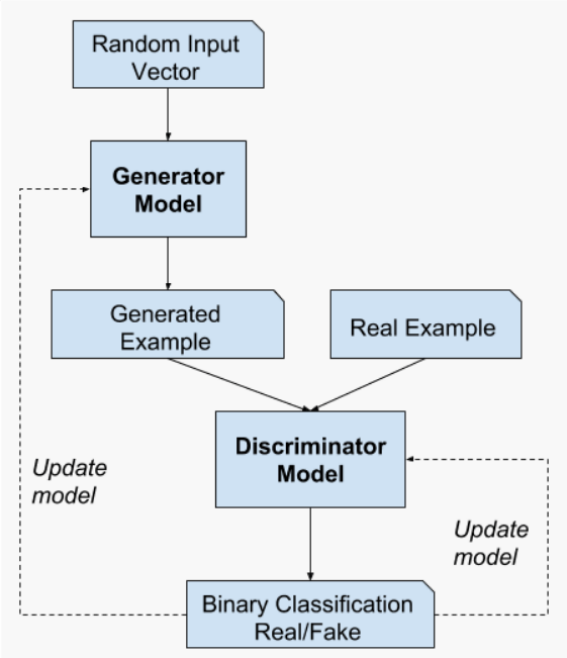


Figure 10: Example of the Generative Adversarial Network Model Architecture. Source: [6]

The usage of GANs within image inpainting [27] and image synthesis[49] papers gave me the desire to experiment with them with the aim of creating a system which synthesises an extension of an image-based musical representation, like seen in Figure 7, rather than general image infilling/synthesis.

4.9 N-Gram Models

The use of an N-Gram model was also considered towards the beginning of the project as a way of generating chord progressions for a monophonic melody generator.

N-gram models operate by constructing a large probability distribution from a training dataset to capture the likelihood of specific chords occurring after a given chord in a sequence. This concept is illustrated below in Figure 11.

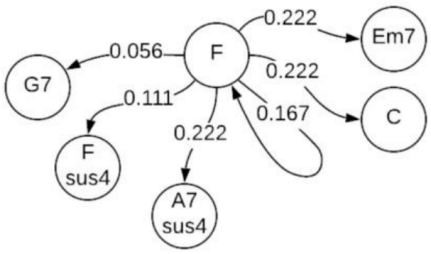


Figure 11: An illustration of the probabilities of different chords being played, given an F chord is played. Source: [42]

These models can make their predictions from either a single preceding chord or multiple chords, allowing for more nuanced patterns to be captured and reducing the likelihood of repetitive loops in generated progressions.

However, there exists a trade-off in terms of sparsity, as models that consider an extensive history of preceding chords may produce chord progressions that include patterns so rare that they are improbable in practice.

Despite this drawback, coupling an N-Gram model as a chord sequencer alongside a monophonic melody generator could increase the musical depth of generated music without too much implementation effort.

4.10 Investigating the Suitability of DAW Integration

One of this project's extension objectives was to investigate the suitability of integrating its music generator into a Digital Audio Workstation (DAW), a piece of software used by musicians to compose, record, mix and master audio tracks.
While this objective doesn't exactly align with the project's original goals, it was an interesting extension task to expand the project's scope and gain some insight into the challenges that Magenta's team may have faced when developing Magenta Studio for Ableton Live.

If possible, integration would offer would-be users direct access to the music generation model within their preferred music production software, allowing usage of AI techniques into their existing workflow.

Some rough designs in the form of a UI sketch and high-level sequence diagram were produced towards the beginning of the project to show potential styling and usage of the generator within a DAW.

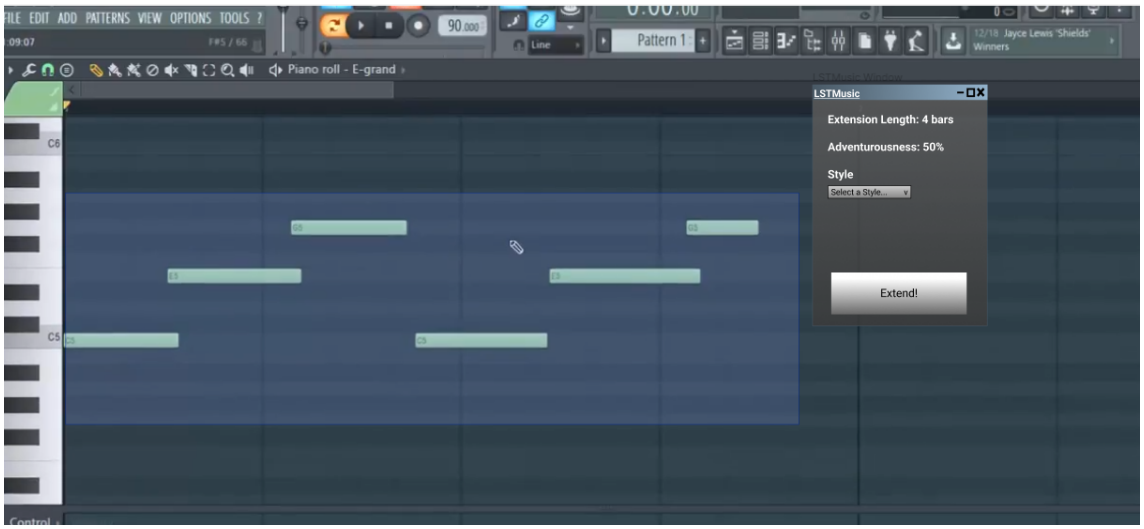


Figure 12: A UI Design concept of what DAW integration could look like.

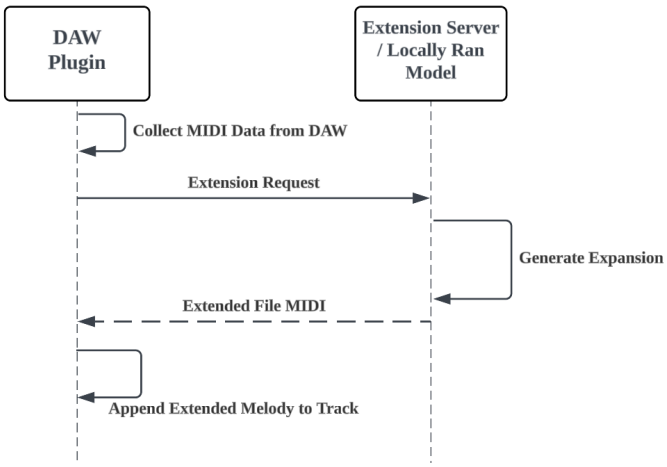


Figure 13: A High-Level Sequence Diagram of the plugin's operation.

With these basic designs made, I began investigating the possibility of DAW integration.

4.10.1 VSTs Using C++

Under the recommendation of my project supervisor, I started by investigating the functionality and properties of Steinberg's VST plugins.

Surface-level reading suggested that VSTs were mainly concerned with the manipulation or synthesis of audio signals, rather than directly generating MIDI events or interfacing with existing tracks within the hosting DAW, allowing them to emulate the sound of physical instruments, create new sounds, or apply effects to existing sounds. They appear to function more as Input/Output systems for audio signals, rather than systems capable of directly controlling the DAW's MIDI tracks for expansion.

However, a subset of VSTs called MIDI Effects exist which can be used to augment and generate MIDI tracks, bringing hope to the possibility of extending existing MIDI tracks in the DAW.

These seemed like a starting point to look into integrating my model, with several MIDI Effects existing that performed tasks similar to what I aimed to investigate within this project, like text-prompted melody generation[34] and generative midi sequencing[56].

Unfortunately, due to the fact that VSTs are developed in C++, or similarly complex frameworks, I decided to not look any further into the possibility of developing a VST as part of the project. Learning another language and building a plugin in addition to the project's core objectives was deemed to be too large of a task, with the creation of a plugin like this having the potential to be its own final year project in and of itself.

4.10.2 Using Python

To get over the hurdle of needing to develop in C++, I also briefly investigated the possibility of integrating the project's model into a python-based plugin.
Although VSTs cannot be developed using Python, I discovered that FL Studio, a renowned DAW developed by Image-Line, offered support for MIDI interaction through Python Scripts[32].

While at first this sounded promising, I found that the python scripts used in FL were unable to interact with the individual midi events within a pattern, which is the container that user-provided melodies would be stored in, with the system appearing to be more of a platform for technical users to create highly-specific macros and custom controllers to work with the messages transmitted by MIDI devices, rather than being one for developers to create GUI-based Python plugins.

With this, I gave up on the possibility of DAW integration and started to focus on achieving the project's core objectives.

5 Planning and Development Strategy

Like the process undertaken in my background research, an initial high-level planning phase was completed towards the beginning of the project, giving me a general idea of the project's shape and purpose.

Surface level knowledge on the data flow process of web apps, as well as what I learned from my initial background research, led me to create a high-level flowchart outlining the project's overall structure, noting the languages and technologies I aimed to use.

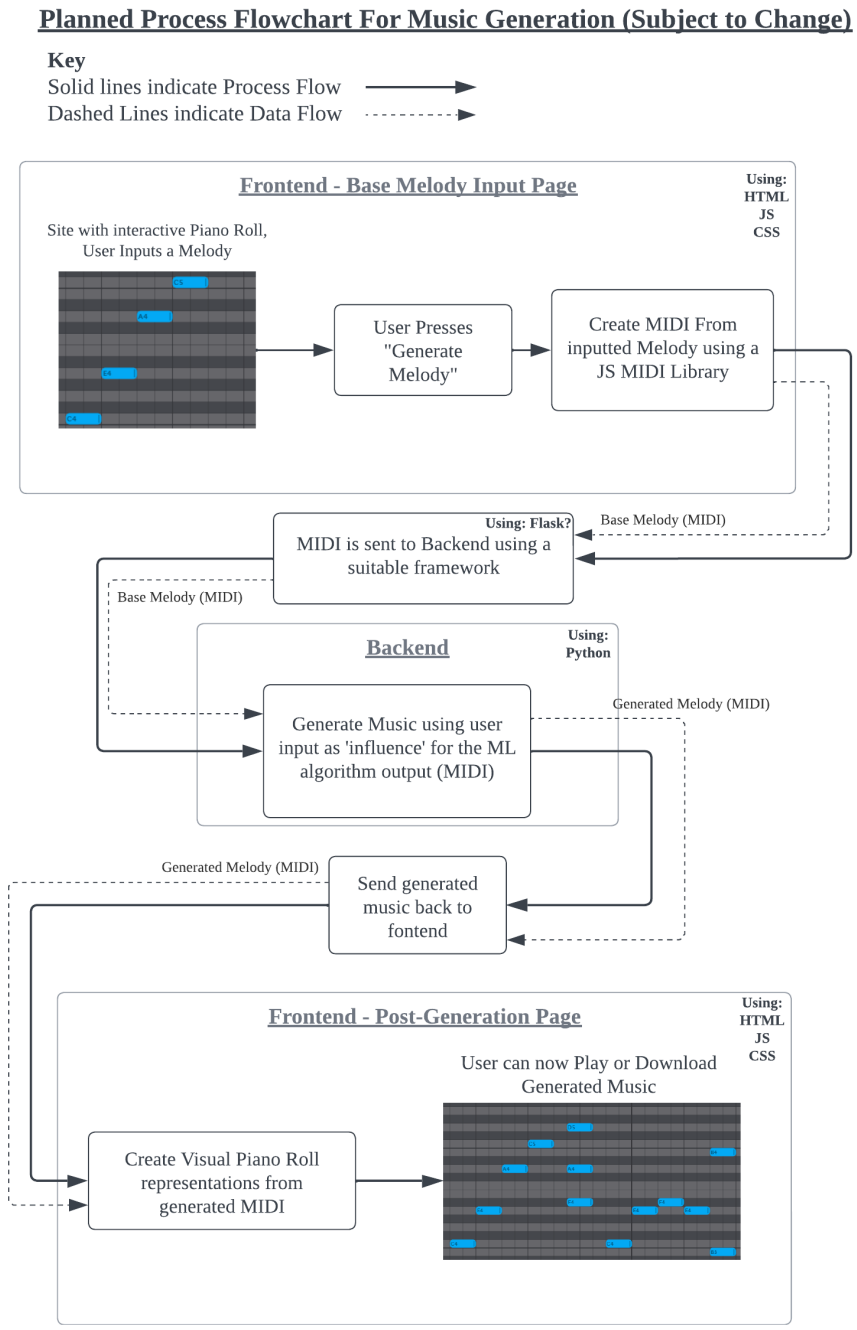


Figure 14: The initial high-level flowchart of the tool's operation process, created towards the start of the project.

The lower-level details of specific parts of the project would then be planned, implemented and tested in an as-needed basis similar to background research, giving an Agile-like structure to design and development.

This led to a separation in the planning and development between different areas, creating four distinct 'phases' of the project, which can be found below. Of course, some overlap occurred when necessary to allow for integration and bug-fixing.

1. The planning and development of the project's first front-end prototype.
2. The planning and development of the project's back-end.
3. The planning and development of the project's improved front-end.
4. The process of hosting the project online.

While this Agile-like form wasn't the initial strategy I aimed to use for the project, the continual improvement of the newly-learnt skills used to create my project meant that I was frequently recognising mistakes and parts of the project that could be improved, leading multiple iterations of the project to exist.

An informal testing system was employed during the development of the project's four phases. Discovered issues were typically fixed on-the-spot as part of the development process, with any persisting or challenging issues being raised on the project's GitHub Repository.

Part II
Frontend Prototype 1

October 2023 to November 2023

During the initial phase of the project, my primary focus was on creating a functional prototype for the main parts of the tool's frontend, with the goal of creating a frontend prototype that would hold enough functionality to allow for unimpeded backend testing.

By prioritising this frontend prototype early on, I aimed to achieve three things:

1. Gain momentum on frontend development.
2. Gain early hands-on experience to build my confidence and knowledge of the languages and technologies associated with frontend development.
3. Gain valuable insights for refining the structure and organisation of the project.

As development at this point was only for prototyping purposes, I wasn't overly concerned with quality or optimisation of my code, embracing any errors or issues as learning opportunities to improve my code, keeping in mind that I would likely have to rework and enhance the frontend in subsequent prototypes.

6 Design

The focus of the frontend's design process mainly revolved around user interface design, rather than code architecture. My limited experience in web development hindered my ability to effectively plan the lower-level aspects of the frontend's code, leading to low-level design choices to be hastily made, with oversights and unintended consequences that would later pose challenging.

High level plans were created however, with the project's high-level requirements being formalised into a use case diagram, encapsulating all of the functionality that the project would require in a first frontend prototype.

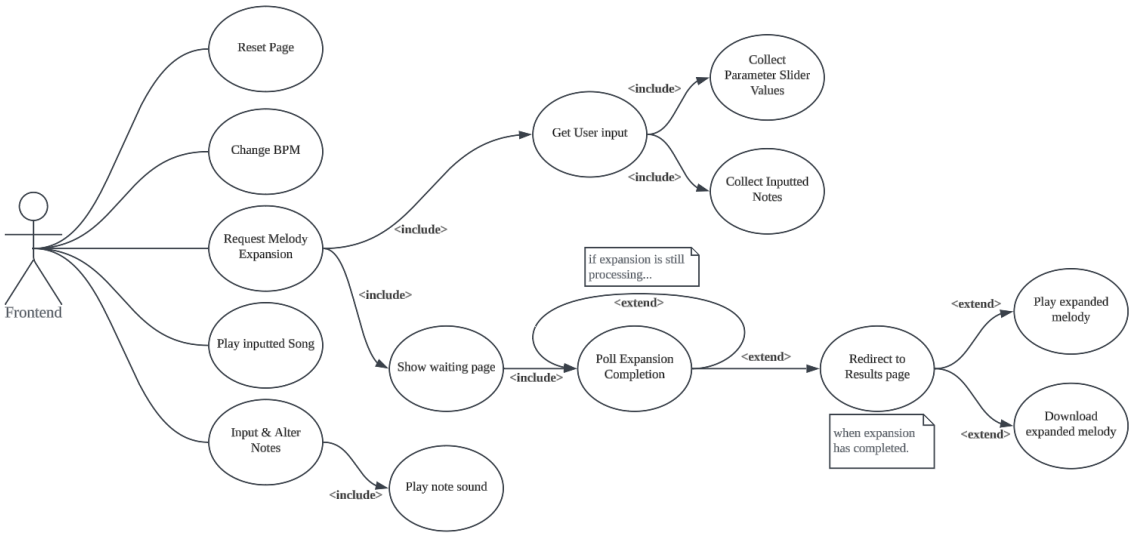


Figure 15: The proposed use cases of frontend prototype 1.

For this prototype, I set out to design three pages which would satisfy all of these identified use cases; a melody input page, a waiting page, and a results page.

Interface mock-ups were created on Figma to define the page structure, design language and interaction methods that would be used throughout the frontend. A dark theme with colourful

interactive elements was chosen to make clear what parts of the page were to be interacted with, giving the frontend a clean and simplistic yet functional design.

These mock-ups, as well as justifications for some design choices, can be found below:

Melody Input Page

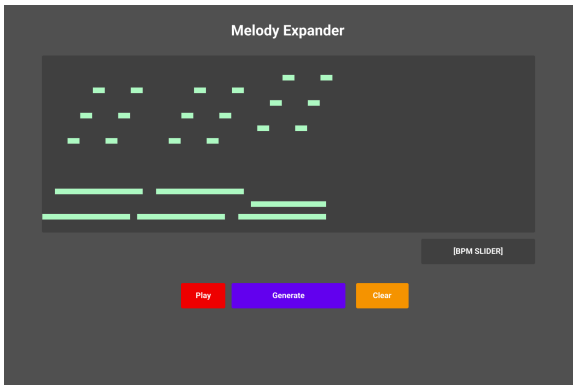


Figure 16: The design mock-up of the melody input page.

The music representation and input method across the entire frontend was designed to follow that of a piano roll, drawing inspiration from the piano rolls commonly used in Digital Audio Workstations, and the uncomplicated design of online music sequencers such as Chrome Music Lab's[23] music sequencer and piano roll. Their user-friendly nature and straightforward use were very advantageous given the project's target audience of music novices.

Waiting Page

A waiting page was designed for use in the case that one was required, in the event that generating melody extensions required a long wait. Sending a request to a backend and having the input page pause for any longer than a few seconds could lead to confusion, so using a page like this would be good way to keep users 'in the loop' while waiting.

The general design of the waiting page was made to be simplistic, only conveying the essential information needed while waiting; acting more as a transitional page which assured users that their requests were actually getting processed.

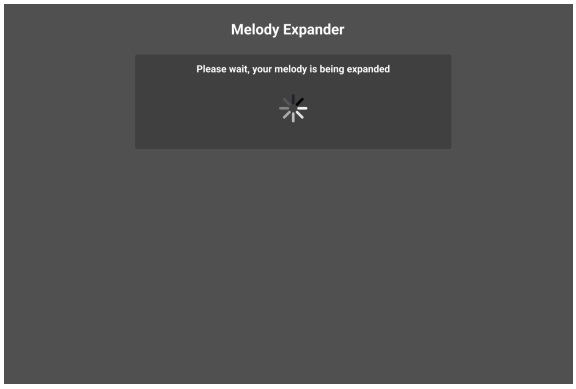


Figure 17: The design mock-up of the waiting page.

Results page

The results page was designed with multiple possible generations in mind, but were also compatible with single melodies. Two designs were originally created, with the vertical style found in Figure 19 being preferred due to its better usage of page space, minimising blank space.

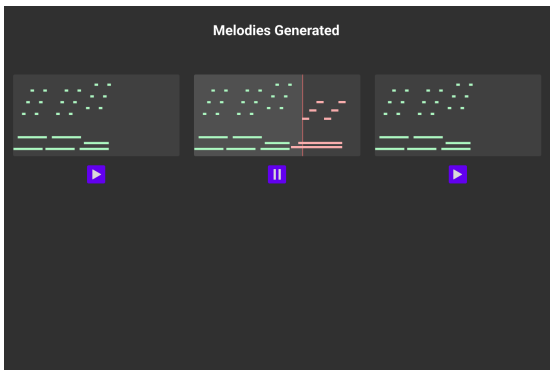


Figure 18: The first design mock-up of the results page.

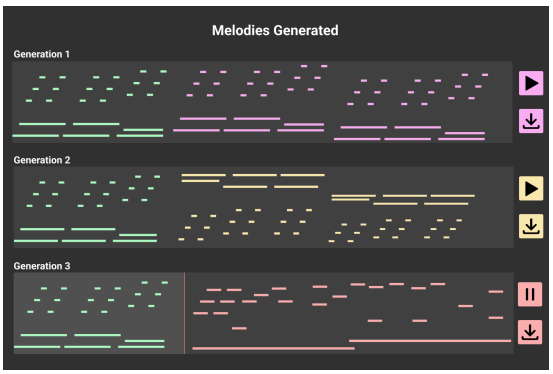


Figure 19: The second design mock-up of the results page.

Like the other pages, the results page was designed to contain all of the the essential functionality required to satisfy the project's use cases, allowing users to play and download their generated melodies.
The clear colour separation between the user-inputted melody and the generated melody was designed to communicate to users what parts of their melodies are AI generated, versus what was inputted. Other UI elements like the play and download buttons were styled around this colouring, clearly separating the different generated melodies.

7 Implementation

With the above mock-ups in mind, development began on implementing the three pages.

7.1 The Melody Input Page

The underlying implementation of the piano roll on the melody input page was based off of a loosely followed online tutorial for a basic web-based Minesweeper board [60]. This created a starting point which could then be altered and built upon with the goal of creating a piano roll.

Continued work eventually resulted in a clickable, interactive grid of HTML elements, each with an event listener corresponding to a coordinate in a boolean matrix. This was the first piano roll prototype, with the rest of frontend prototype 1 being built around it.

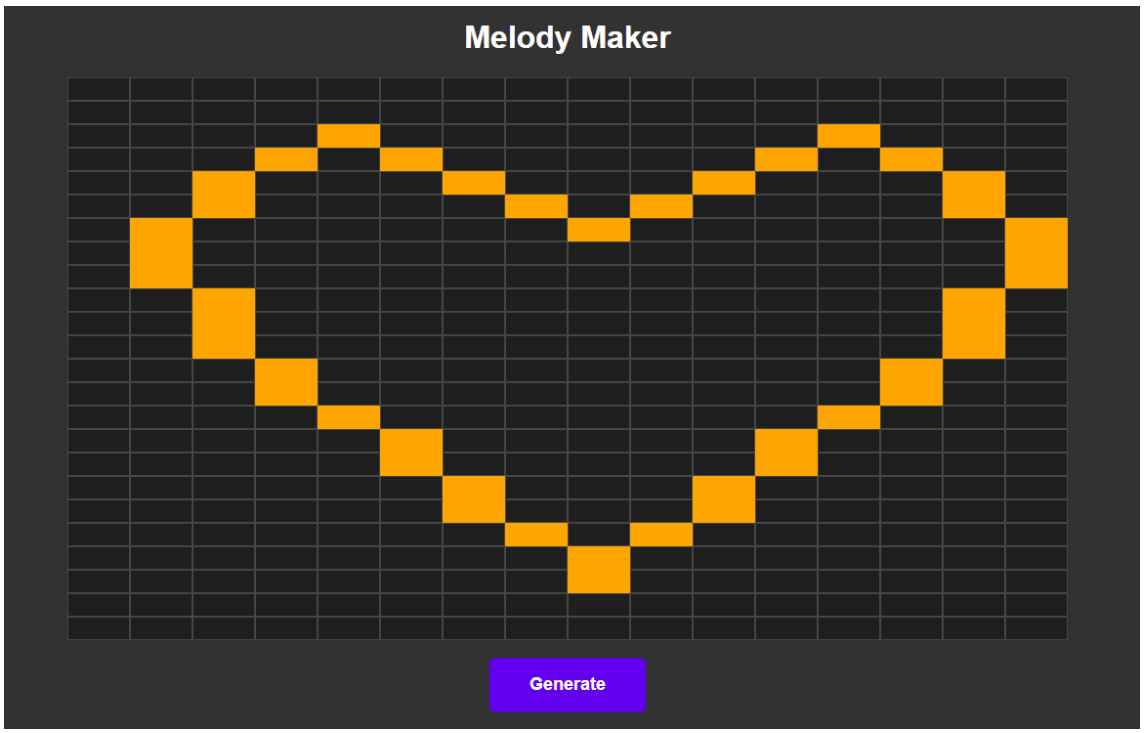


Figure 20: The first version of the frontend's input page, with a heart drawn on it.

Development continued further, with new features and functionality being added frequently. By the time that the project's interim report was due, the front page's feature set was as follows:

- Users could input individual notes onto the piano roll using the mouse.
- Inputting notes would sound an appropriate noise from a digital piano.
- A visual representation of a piano to the left of the piano roll assisted users in positioning their notes.
- A basic player allowed the user to play their melody.
- Users could change the BPM of the played melody.
- Users could clear the board.
- Users could press a placeholder "Generate Melody" button, which would convert the inputted notes into a base-64 encoding of a MIDI file using Midi-Writer-JS.

This page is pictured below.

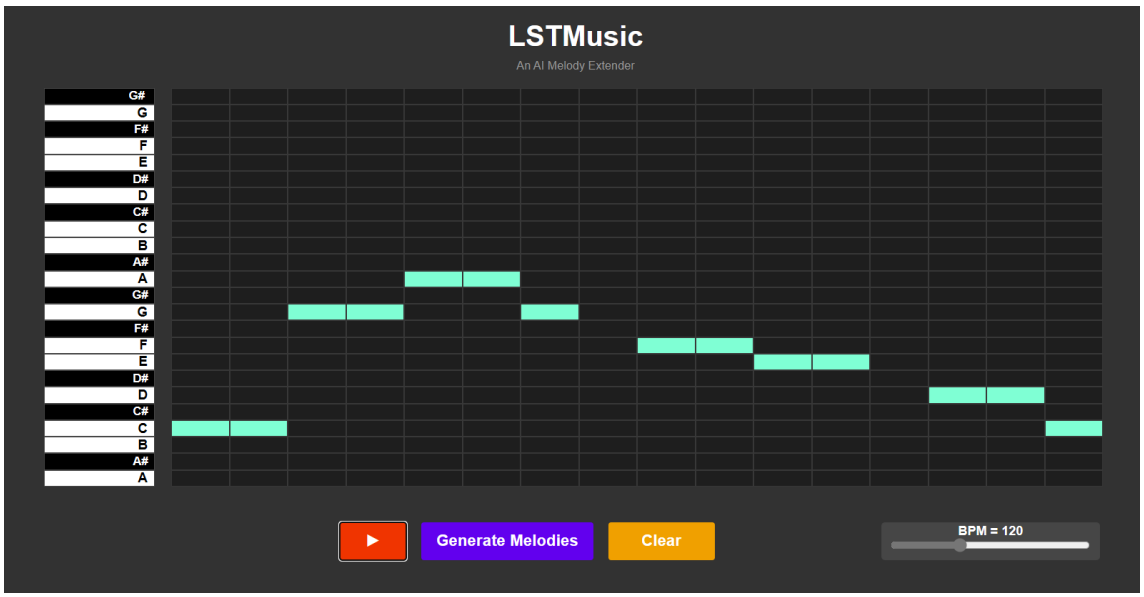


Figure 21: The final version of the first prototype's input page, with Twinkle Twinkle Little Star inputted.

While this feature set was rich enough to technically allow it to be hooked up with a future backend, the bodged together nature of the piano roll meant that it had some issues, which were uncovered during a short testing period.

These issues included, but were not limited to:

1. Identical notes not giving audio feedback when placed in rapid succession.
2. Identical notes not playing during playback if placed back to back.
3. Scaling issues when resizing the browser.
4. Issues with playback when the "play" button was pressed during playback.

Although most of these issues were not huge concerns given the prototype's early stage, the decision to manually construct the grid using a boolean matrix significantly limited the duration of notes that could be entered to just whole notes, therefore restricting the possible melodies that users could input into the page.

7.2 The Waiting Page

The waiting page's implementation was simple, requiring little more than a recreation of the Figma design to show a loading icon. No other functionality was included in this page due to the lack of a backend, with it acting as a skeleton to be completed if required.
Below is a screenshot of the waiting page's implementation.

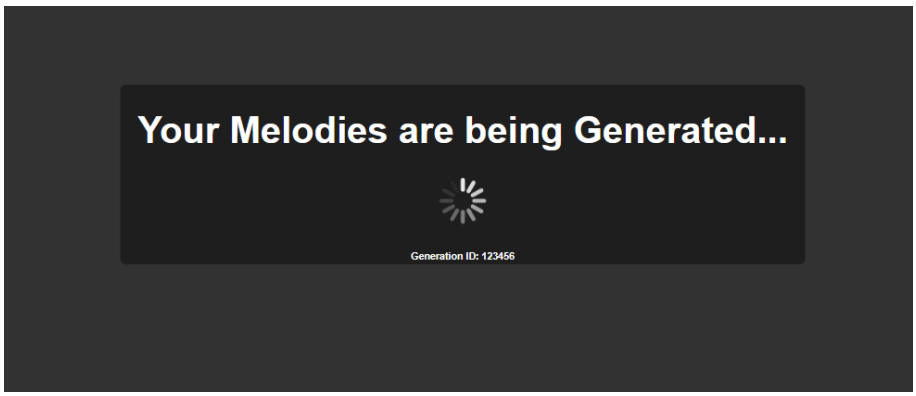


Figure 22: The waiting page from the first prototype.

An extension ID element was added with possible future debugging purposes in mind, lacking any functionality or justification at this stage.

7.3 The Results Page

The prototype 1 iteration of the results page stuck closely to the designs made, but remained lacking in functionality, serving as a placeholder until backend development progressed enough to allow MIDI files to be sent back to the frontend. This led to the page lacking any interactive features, only consisting of static HTML elements and styling.

Figure 23: The skeleton of the results page from prototype 1.

After these pages were developed, I decided to shift focus from developing the frontend to starting the project's backend as, at this point, I believed that the main functional components of the project's frontend were suitable to allow for future backend testing, once everything was hooked up.

Part III
Backend

January 2024 to March 2024

The primary design and development processes of the backend revolved around the need to create a set of API endpoints for the frontend to use to complete its use cases. It would use a simple prototype generative model, with model improvement and parameter adjustment being planned for a later development phase.

I chose to tackle the design and creation of the generative model first, as it'd allow me to gain an insight into the extent of functionality and modularity that the model's surrounding Flask app would require, allowing for a better planning process when tackling it.

8 Generative Model

8.1 Design

The choice of dataset, musical representation, model structure, and general design of the backend's music generation model was inspired heavily from the work of Valerio Velardo. His set of online resources [59] proved to be extremely helpful in guiding the creation of the LSTM-based music generation model used. Slight alterations to his designs were required to change the generative model from one built to generate melodies from scratch, to a model which could extend pre-existing melodies, but his contribution here cannot be understated.

Dataset
The Essen Folksong Collection (EsAC) is a database of some 6,255 folk song transcriptions, consisting primarily of folk songs from Germany and other Germanic regions[52]. While it is not exactly the most mainstream of music that one would expect from a melody extender, its composition of simple, copyright-free, single-track monophonic melodies meant it was compatible with the chosen time-series representation used, and avoided any concerns regarding copyright. It was therefore chosen as the dataset the prototype model.

EsAC's music isn't stored in MIDI files, but instead KERN files. KERN is a similar symbolic music representation[30] which can also be parsed by Music21, therefore causing no issues when pre-processing.

Other monophonic datasets were considered (primarily [46]), but the plan to eventually replace or improve the model later in development meant that I was content enough to keep the folk song dataset as a proof of concept, with larger, polyphonic datasets (such as [26], [31], or [35]) being noted for any future development.

Pre-processing
The pre-processing steps to turn the ESaC dataset into usable training data would follow Velardo's work, consisting of the following steps:

1. Use Music21 to parse the dataset into Music21 streams.
2. Discard any streams with 'irregular' note lengths, with accepted note durations being whole, half, dotted half, quarter, dotted quarter, eighth, dotted eighth and sixteenth notes.
3. Transpose the streams to C major or A minor (dependant on mode) to normalise the song key.
4. Encode all streams into the time series representation, with each list index representing a sixteenth note.
5. Flatten the entire dataset's streams into a single file.

6. Construct a map to map the dataset's 'vocabulary' (possible tokens) into Neural Network compatible classes, represented by integers.
7. Map all of the tokens within the flattened dataset into their integer representations.

These steps would result in a single file representation of the entire encoded dataset, which could be later used to create training pairs for the model to use.

Model Configuration and Training
The model's configuration and training process were to follow Velardo's work, with design choices here being deemed acceptable within his model. This 'default' configuration was deemed suitable enough for a prototype generation model, with aims to perform parameter optimisation later in the project.

The model itself would consists of 4 layers:

1. An input layer with a size matching the dataset's vocabulary, allowing for all possible tokens to feed into the network.
2. An LSTM layer with an output size of 256.
3. A dropout layer to avoid overfitting.
4. A dense output layer using a sigmoid activation function, also with the size of the dataset's vocabulary.

Each output node of the network would correspond to a possible note within the dataset's vocabulary, with the model's task essentially being a supervised classification problem. This means it would be necessary to create a set of expected outputs, given various inputs.
For the time series representation, this would again be achieved by following Velardo's method, creating a set of 'training pairs' by labelling pairs of sequential notes from the training dataset with the next expected note.

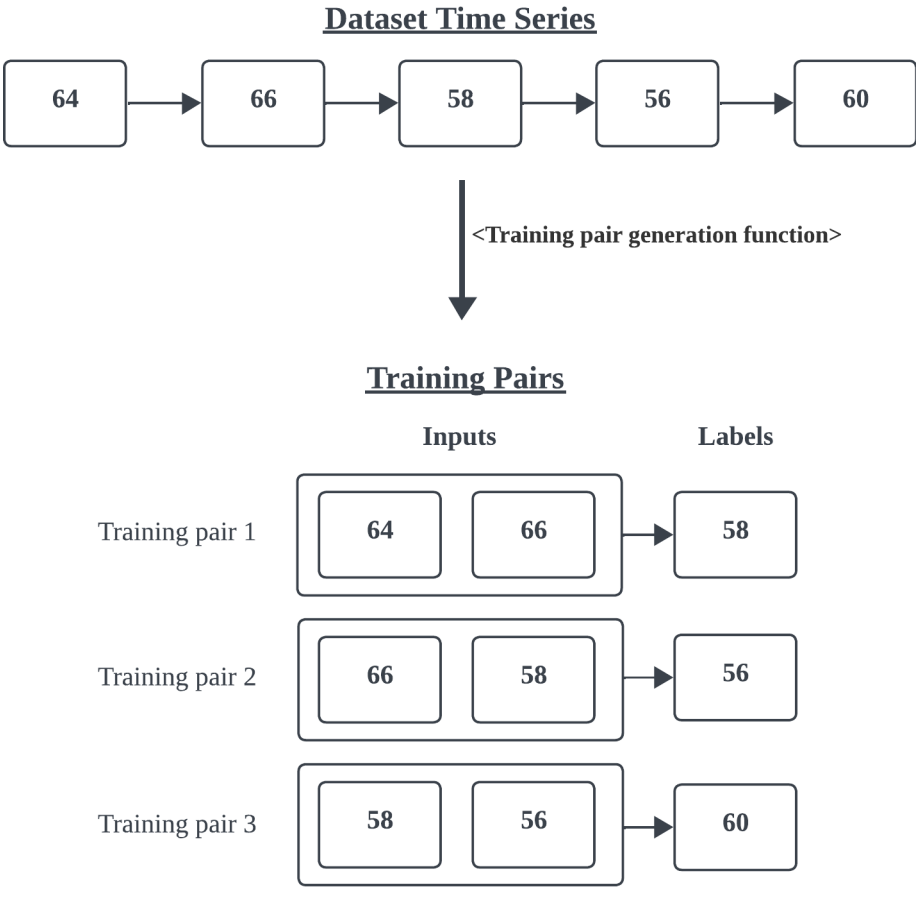


Figure 24: A diagram illustrating the training pair structure.

Training would then be carried out using Velardo's suggested batch size of 64 pairs to fit the model to the dataset.

Generation process
The generation process of the model would follow Velardo's designs with slight changes. This would allow the model to use a user's input melody as a seed, rather than a randomly generated string of notes.
A flowchart illustrating plans for the process can be found below.

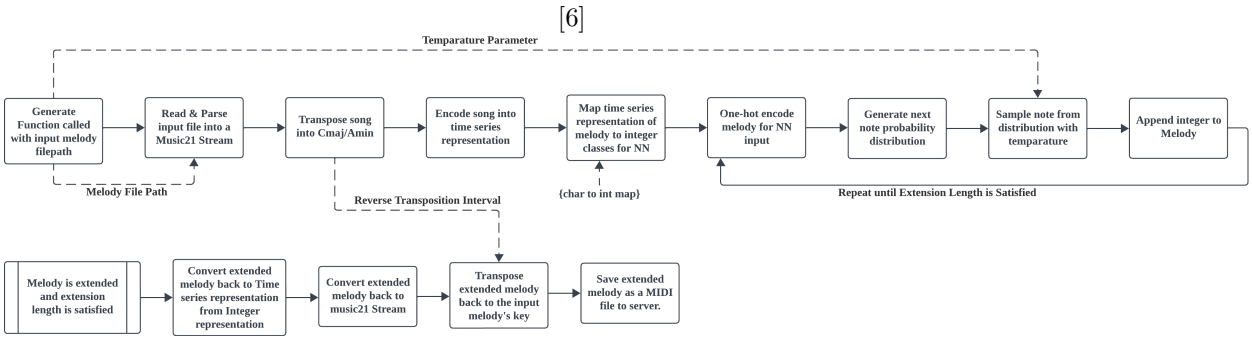


Figure 25: A flowchart illustrating the planned generation process.

The model would generate notes by creating a classification based on the input it is given, creating a prediction for the next note. This would be appended to the sequence to 'extend' the melody, with the process repeating until the desired extension length is satisfied.
The extended melody would then be converted back into a music21 stream and transposed back into the key of the user's input melody, and saved to the server, completing the generation process.

8.2 Implementation

Implementation of the model was straightforward and happened without issue, as most of its implementation followed Velardo's guides.

Training occurred over 16 epochs, a figure chosen somewhat arbitrarily to ensure the model's ability to pick up on patterns within the training data.

The model's performance wasn't too important at this point, as this model was simply a proof of concept to show that the LSTM could work in the entire extension sequence. However, a short testing phase was still carried out to ensure that the model could successfully complete its generative steps.

As there was no way for the frontend and model to communicate at this point, testing consisted of feeding externally created MIDI files into the model with manually set temparature and extension length parameters.

I was pleasantly surprised when these tests revealed that the model was able to create short, simple melodies that had clear melodic and rhythmic influence from the input melody, with melody extensions often following the key and basic rhythmic structure of inputted melodies.
These generated melodies leaned clearly towards the rhythmic structure of folk-songs, with long notes and a general 'jolly' feeling. This wasn't much of a surprise given the training data used, but these results gave a good indication that the model used was able to pick up the rhythmic structure of its training data.

Wait times for generations ranged from around 2 seconds for shorter extensions, to 10 seconds for longer extensions, which felt largely acceptable for an ML task.

9 Flask API

9.1 Design

After the generation model's functionality was designed and implemented, work started to design the Flask API that would surround the model. Modularity was an important focus when creating this API, as I had plans to improve my model later, which would likely involve completely changing the model's pre-processing and generation functions.

The design process for this API revolved around the need to satisfy the required use cases from figure 15. This created a set of use cases for the backend itself, from which three API endpoints were identified; illustrated in the diagram below.

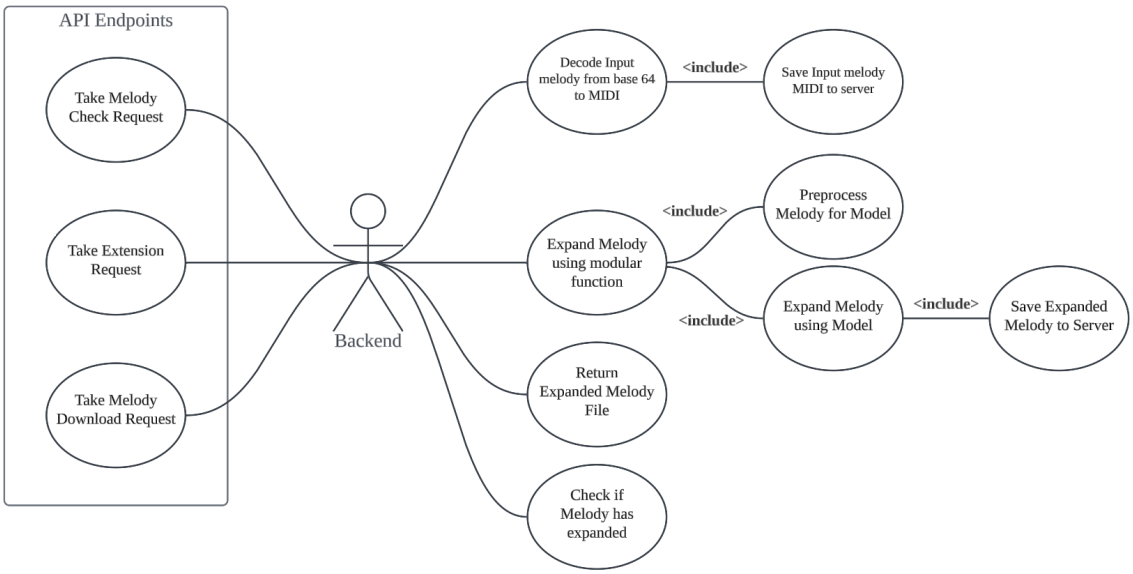


Figure 26: A use case diagram of the backend's desired functionality.

These endpoints were designed to be used alongside the frontend to create the following sequence of events for a melody extension, illustrated below.

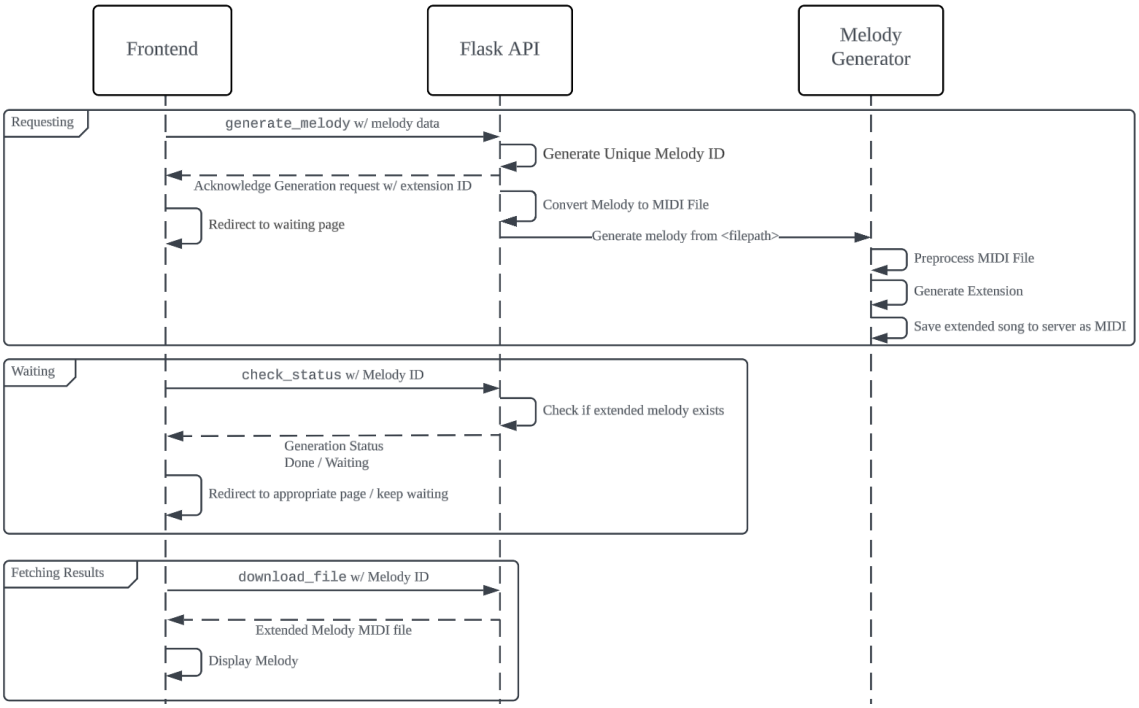


Figure 27: A sequence diagram illustrating the entire melody extension process using the API endpoints.

The below sections elaborate on the functionality and operation of each API endpoint's design.

9.1.1 Extension Request Endpoint

Using the `/generate_melody` endpoint.[2]

Extension requests would be handled through an ID system, with each extension request having a unique, randomly generated ID. All subsequent check and download requests to the API would contain a request's respective ID, allowing multiple users to use the API at once.

Designed to be called by the frontend's input page, this endpoint would handle all of the request handling and melody generation functionality that the backend would need, including the distribution of unique melody IDs, saving of input melodies to the server, and calling the generation process.

The generation process itself would be handled by a separate generation function called from the endpoint, taking the file path of the saved melody. This would separate the request processing and actual melody generation processes, making the extension request endpoint modular and therefore easing the process of swapping the pre-processing and generation function, if required in the future.

A flowchart detailing the planned operation of this endpoint can be found below.

[2]Later changed to the `/generate_melody_new` endpoint, for the second frontend prototype

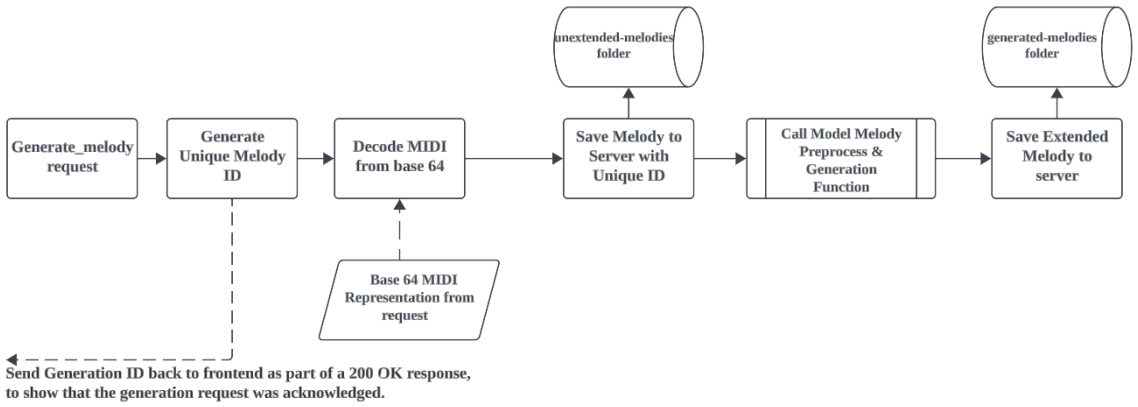


Figure 28: A flowchart showing the planned operation of the generate_melody endpoint.

9.1.2 Extension Status Endpoint

Using the `check_status` endpoint.

The need for an endpoint to monitor the status of a melody's extension arose due to the wait times experienced for melody extensions found during model testing, justifying the implementation of the previously skeletal waiting page, along with an endpoint to give it its functionality.

The designs for the extension status endpoint were straightforward. Calling the endpoint would verify the existence of a file with a specified extension ID in the model's output folder, providing a response indicating the expansion's status as either "waiting" or "complete".

To connect the waiting page to this new endpoint, a simple script to poll the endpoint with an extension ID would also be necessary.

A flowchart detailing the planned operation of this endpoint can be found below.

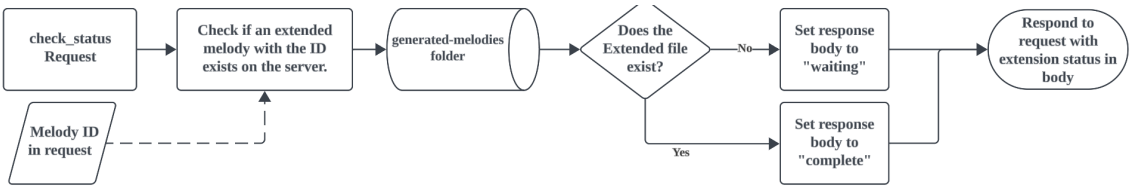


Figure 29: A flowchart showing the planned operation of the check_status endpoint.

9.1.3 Melody Retrieval Endpoint

Using the `/download_file` endpoint.

The implementation of a melody retrieval endpoint was necessary to allow users to retrieve their generated melodies. The design of this endpoint was straightforward. Upon receiving a request with a specific extension ID, the endpoint would locate and retrieve the corresponding MIDI file from the server's storage.

9.2 Implementation

During some initial tests with Flask, it was discovered that endpoints must provide a response by acknowledging a request, or by sending data back. This meant that all of a function's inner processes had to be completed to return.

This became problematic when attempting to implement the event sequence outlined in Figure 27 for extension requests, as the request acknowledgement (alongside the extension ID) were designed

to be returned *before* the function's completion.

Flask wouldn't allow this, forcing the response to be sent at the end of the function call. This led to the frontend hanging while waiting for a request acknowledgement from the backend, due to the time taken for an extension to generate, often causing the original request to time out.

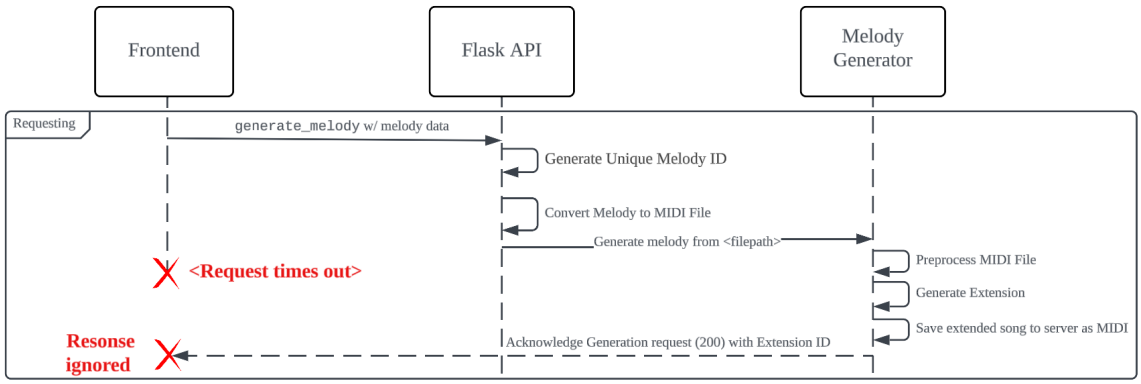


Figure 30: A sequence diagram illustrating the timeout issue.

However, by dispatching the generation function's call to a second thread, the main thread was able to immediately acknowledge the request and return an extension ID, permitting the implementation I originally designed with some slight alterations to enable threading. Resolving the timeout issue with no other problems.

The remaining endpoints were implemented without issue, though slight alterations were made to the `download_file` and `check_status` endpoints, as I had discovered Flask's "Variable Rule" system[18] during development. This simplified the process of passing extension IDs between the frontend and backend by using a parameter contained within the API's URL, rather than within the body of the HTTP request. The simplicity of this approach is illustrated in the below code snippet.

```
@app.route('/download_file/<song_id>')
def download_file(song_id):
    filename = f"generated-melodies/extended_melody_{song_id}.mid"
    return send_file(filename, as_attachment=True
```

Listing 4: Flask's variable rule system in use.

In addition to the backend modifications, some frontend adjustments were made to ensure full communication between the frontend and backend. These included:

- Adding a script on the results page to fetch a download of the extended MIDI file upon completion of the melody extension process.
- Modifying the "Generate" button on the input page to send the base-64 representation of the inputted melody to the backend to start the extension process.
- Adding the previously mentioned polling script to the waiting page.

With these adjustments, the frontend interface gained full functionality, enabling users to navigate through the generation process without direct interaction with the backend, marking the project's first fully operational locally hosted prototype.
After this phase of implementation concluded, a brief system testing phase was conducted to assess the API's robustness and functionality. This testing not only revealed limitations within the API, but also exposed issues with the frontend that needed to be addressed, outlined below:

- The backend begins to struggle when more than 6 extensions requests are being processed.
- Temperature and Extension Length adjustment was absent from the frontend and backend.
- Any rests between notes from the frontend's piano roll would be ignored, impacting the melodic structure of inputted melodies.

- Any form of UI beyond the waiting page was absent.
- The frontend's piano roll lacked any note duration variety, severely limiting the input melodies, thus limiting the model's generative capacity.

These issues highlighted the shortcomings of the project's first frontend prototype, with a critical need for improvement towards the input page's piano roll and the results page's non-existent functionality, kick-starting the design of the project's second frontend.

Part IV
Frontend Prototype 2

March 2024 to April 2024

The second iteration of frontend started with an urgent need to rework the input page to include a piano roll capable of capturing variable-length notes. This was the starting point of frontend prototype 2's design.

10 Design

10.1 The Melody Input Page, Version 2

For the piano roll's second implementation, I initially entertained the idea of modifying the first prototype's piano roll to accommodate variable note lengths. This was quickly dismissed due to concerns that such an implementation would likely create more challenges than benefits from its messy and unconventional implementation.

Instead, I turned to online resources in search of existing web-based piano roll implementations. I discovered WebAudio-PianoRoll, and immediately recognised the advantages its implementation could bring to my project, with the caveat that its implementation would be tied to my ability to decode the custom format in which it exported entered melodies. This will be touched upon further.

In addition to introducing the improved piano roll, new extension length and temperature sliders were planned, drawing inspiration from similar features in Magenta Studio.

Incorporating these sliders required a revision of the method used for sending extension requests to the backend. As previously, only a base-64 encoding of the inputted melody MIDI was sent to the backend with the request.

To accommodate these changes, each component of the request would be delimited by semicolons, forming an informal "melody extension request" object for API communication. These adjustments also required modifications to the existing backend function responsible for processing the expansion requests.

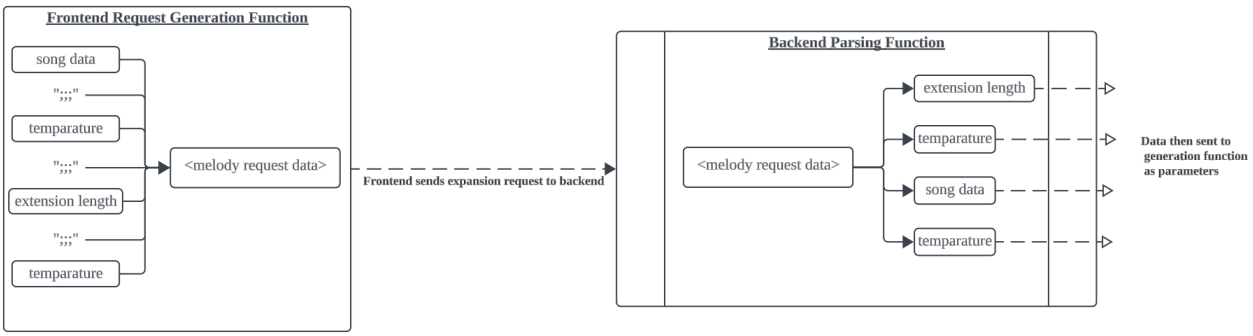


Figure 31: A diagram illustrating the information contained within an expansion request.

These feature additions were accompanied by visual alterations to the input page's design. These changes involved repositioning interactive elements to reduce clutter, grouping the page's parameters, playback controls, and generate/clear buttons into specific areas.

At this point the project's name was also revised to LSTMusic, chosen as a nod towards the backend's generative model, drawing inspiration from ChatGPT, and similarly incorporating its model into its name.

The final input page design with all of these changes can be seen in the below mock-up.

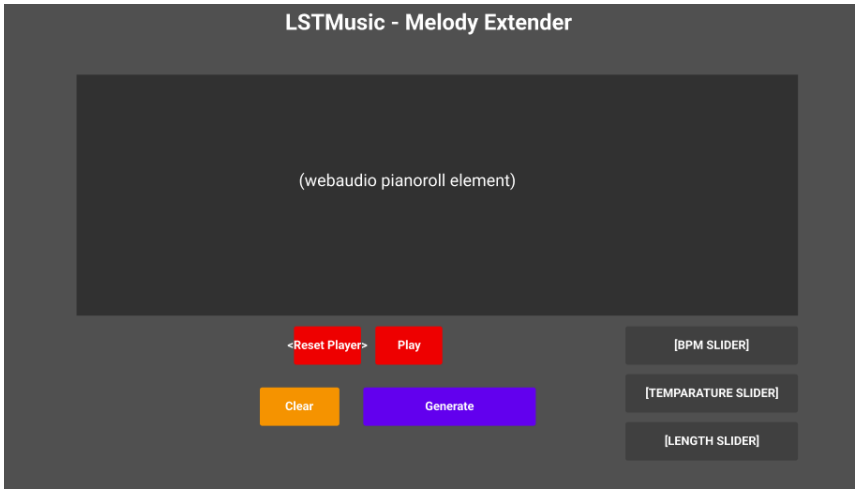


Figure 32: The second visual mock-up of the melody input page.

10.2 The Error Page

As prior backend testing uncovered unforeseen issues leading to occasional failed melody extensions, a decision was made to introduce an error page. This error page would serve as an extra means of keeping users informed throughout the melody extension process, like that of the waiting page.

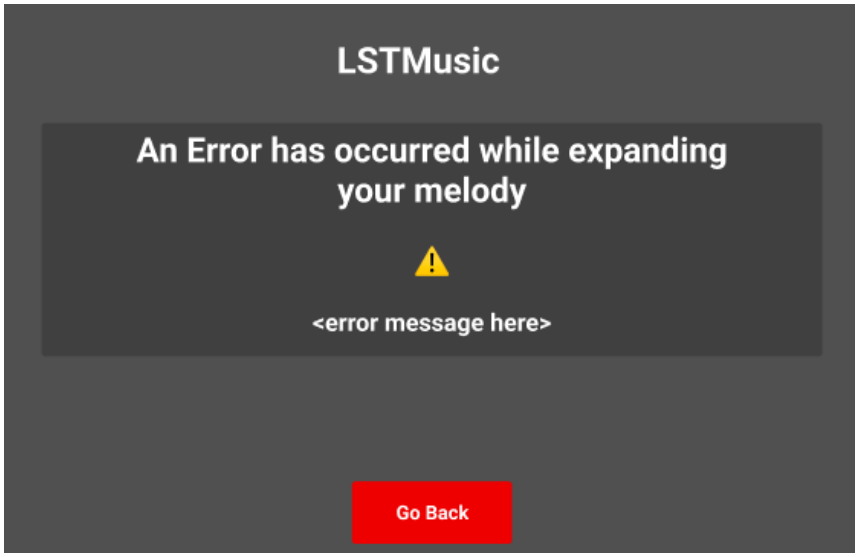


Figure 33: The design mock-up of the error page.

The visual design of this error page closely resembled that of the waiting page, with the addition of a simple back button, and an error message box, designed to dynamically update to offer information in the event that a melody extension fails.

This was achieved through the creation of an 'error type' system, aimed at providing codes to correspond to types of errors the system commonly encountered.

Error type	Error Message to Display	Actual Issue
1	'The generation server is not available right now. Please try again later.'	Frontend can't access backend.
2	'An error has occurred while trying to generate your melody. Please try again.'	Exception raised during extension by backend.
3	'Your melody could not be generated. Please try again.'	Request timed out.

Table 1: A Table illustrating the planned errors and their messages.

While error types 1 and 3 would be only concerned with the frontend, error type 2 required some slight changes to the project's backend, shown as blue elements in the below flowcharts.

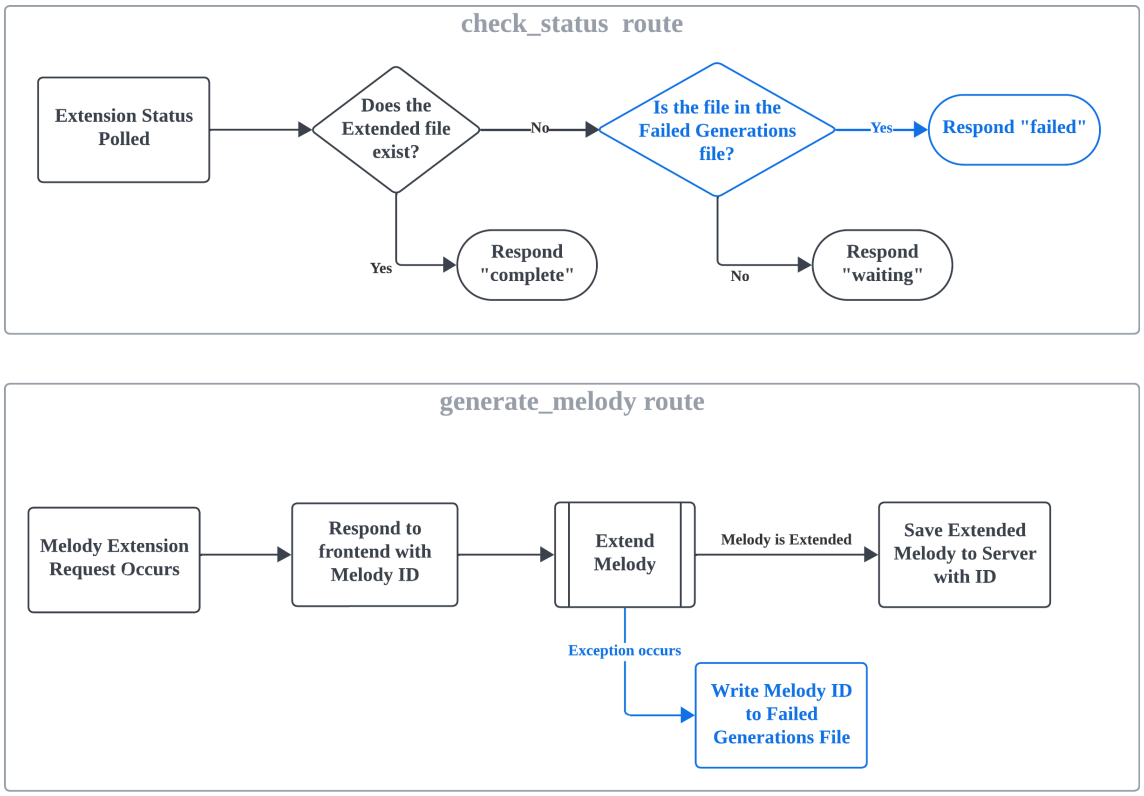


Figure 34: The proposed changes to the backend to implement error handling.

A new response message "`failed`" would be added to the backend's generate_melody route as part of this new error functionality through the use of try/catch blocks and a new custom `GenerationError` exception type.

10.3 The Waiting Page, Version 2

The design of the frontend's waiting page required minimal enhancement from the first iteration as it already functioned well enough for what it aimed to achieve.
Apart from a simple alteration to the waiting page's polling script to redirect users the new error page when receiving a 'failed' message from the backend, no changes were planned.

10.4 The Results page, Version 2

The implementation of the results page was planned with the usage of HTML-MIDI-Player in mind.
I planned to adapt the skeleton implementation of the previous iteration's results page, altering it to contain custom-styled instances of the player and visualiser elements which HTML-MIDI-Player provided. This would create a visually appealing results page with the supporting functionality to

satisfy the use cases which frontend prototype 1's result page was unable to provide in its skeletal state.

The large buttons at the side were scrapped in favour of the sleeker player that HTML-MIDI-Player provided, with custom "download .mid" buttons being created to fit the new layout, allowing users to still download their melodies.

Unfortunately, the colour separation of the input and extended sections of the returned melodies would not be implementable through HTML-midi-player without changes to its source code, which weren't planned for this iteration of the frontend. The melodies would still be cleanly stitched together though, with this only being a visual issue.

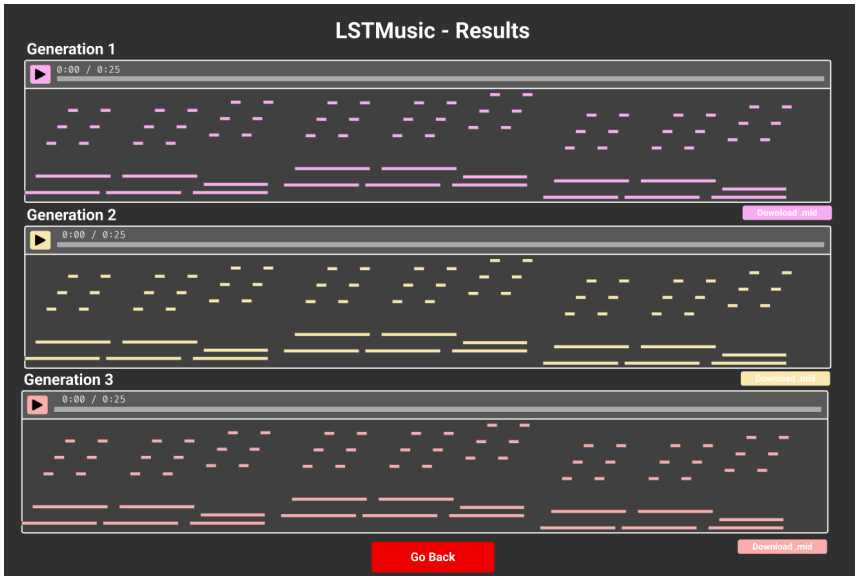


Figure 35: The design mock-up of the results page using HTML-MIDI-Player.

11 Implementation

11.1 Melody Input Page Version 2

The second iteration of the melody input page was started through the quick integration of a WebAudio-Pianoroll element, which immediately proved to be more functional than the first prototype's odd piano roll implementation.

Figure 36: The first implementation of WebAudio-PianoRoll on the input page.

Work followed over the next month, with various improvements and changes to the front page. However, two challenges stood out during this time period: WebAudio-PianoRoll's playback functionality, and its data representation.

Playback Functionality Issues
WebAudio-PianoRoll comes with a rather modular setup when playing an inputted melody. Users are able to hook up a custom callback function to customise the sound which the piano roll makes using a WebAudio element. This callback function would take a custom event object which g200kg created themselves, lacking any well-translated documentation.

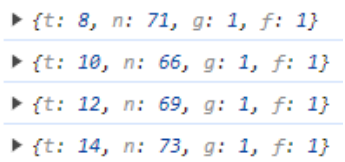
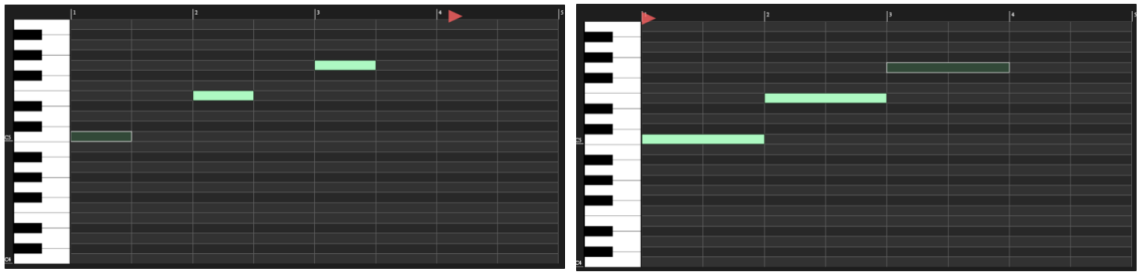


Figure 37: A screenshot of some WebAudio-PianoRoll event objects with vague variable names.

Unfortunately, this flexible play/pause functionality lacked a default callback function out-the-box, requiring me to create my own.
Looking through g200kg's source code[20] and Mozilla's AudioContext documentation[4] gave enough guidance to figure out how to create a simple synth, which could play notes using a sawtooth wave. This resolved the issue, but left the sounds played by the piano roll in a worse-off state than what was found in the frontend's first prototype, which used a sampled piano note.

Encoding user input into MIDI.
Another challenge which arose as I integrated WebAudio-PianoRoll was caused by its data-exportation format.
WebAudio-PianoRoll's documentation suggested that it was able to export its piano roll data through the use of a function called `getMMLString()`, returning the piano roll's contents in a String format, refered to as 'MML' by g200kg.



t120o4l8o5c2r2e2r2g2

t120o4l8o5c1e1g1

Figure 38: Two piano rolls, with their exported MML Strings below.

Various implementations of MML exist[38], each with their own slight variations in representing musical elements. With no standard for MML, finding a specification or more information about the format used by g200kg was likely impossible.
Attempts to match WebAudio-PianoRoll's MML format with the formats listed in an electronic music wiki[38] yielded no helpful results, forcing me to accept that I would need to decode and parse the language into MIDI myself.

An idea of the structure of MML's was finally pieced together after several weeks of on-off work to figure out the format. Through inputting simple melodies into the piano roll, and comparing the outputted MML strings, I was able to conclude the following:

- `'o'{INT}` Denotes a change in octave.
- `'t'{INT}` Denotes a change in tempo.
- `'l'{INT}` Sets the default note duration.
- `['a'-'g']{DURATION}` Denotes a note, with a specified duration.

The duration of a note is specified using a combination of numbers, dots, and ampersands:

- A number denotes the denominator of the note length.
- A dot signifies an extension of half of the current note's length.
- An ampersand indicates the addition of two notes, used when a duration cannot be expressed through a combination of numbers and dots.

With this understanding, I began work on converting MML to MIDI, drafting the beginnings of a Finite State Machine (chosen due to the sequential nature of MML Strings) for a parsing function.

At this time I was also undertaking a personal project to improve the English in WebAudio-PianoRoll's documentation, due to its poor translation. This exploration led to the discovery of a hidden internal data structure storing user input sequences in a MIDI-like structure.

Using this data structure, I was able to implement a set of changes to allow my app to export this data structure as JSON to my backend, circumventing the need to parse MML directly. With the addition of a simple parsing function to my backend, the export issue was resolved, saving time that would've otherwise been wasted to create an MML to MIDI function.

Fortunately, not all parts of the second input page were this convoluted to implement, with most of the other features being rather straightforward additions or fixes, following or adding upon the previously created designs.
These included:

- The addition of the planned parameter sliders, with "temperature" being changed to "adventureousness" as a better term for those not familiar with the concept of temperature in machine learning.

- The usage of the new melody extension request data format, with backend changes also being implemented to parse the new format.
- Implementing the new button and slider arrangement.
- Updating the page's title.
- Adding functionality to clear the piano roll with the existing "Clear" button, by setting the piano roll's MML string to be an empty String.
- Limiting the piano roll's input to be monophonic, by setting the piano roll's 'grid type' to be monophonic.
- Adding auto-scaling to the site's elements to prevent visual bugs, by reworking parts of my CSS.
- Adding an "About" section to provide more context on the app's purpose and operation.

With these additions, the page was just about in its final form, with a slight change being made to change the extension length slider to generalise the wording it used, as testing found that the backend would commonly extend the melody by an amount which wouldn't respect this slider exactly.
Now that more generation customisation was implemented and variable note events were possible, several backend issues were uncovered when testing:

- Melodies would sometimes not expand, returning the melody which was inputted.
- Shorter melodies would sometimes generate an empty 'buffer' between the end of the inputted melody and the generated music.
- The extension length parameter would not be respected, typically being off by one or two bars.
- Inputted note lengths longer than a whole note would be separated into whole notes.

With the exception of the extension length parameter issue (which was resolved by generalising the extension length slider from a numerical value to an option of "short", "medium" and "long"), the above issues are still apparent and can be encountered when using the site.

11.2 Waiting Page Version 2

The implementation of the waiting page was extremely straightforward, with the alteration to the waiting page's polling functionality (in both the front and backend) going smoothly.
Other, unplanned alterations were made to the waiting page as slight 'spur of the moment' additions which helped improve the general feeling and usability of the waiting page. These included:

- The addition of a back button for easier navigation, redirecting users to the homepage.
- The addition of a stylish new loading GIF[15], as there was a feeling of emptiness and blandness to the page with the old one.

11.3 Error Page

The implementation of the project's error system in both the frontend and backend was a made easy due to the thorough planning documents created. The error page was easily adapted from the existing loading page, and the addition of a script to handle the error messages led to the page showing all error messages as expected.

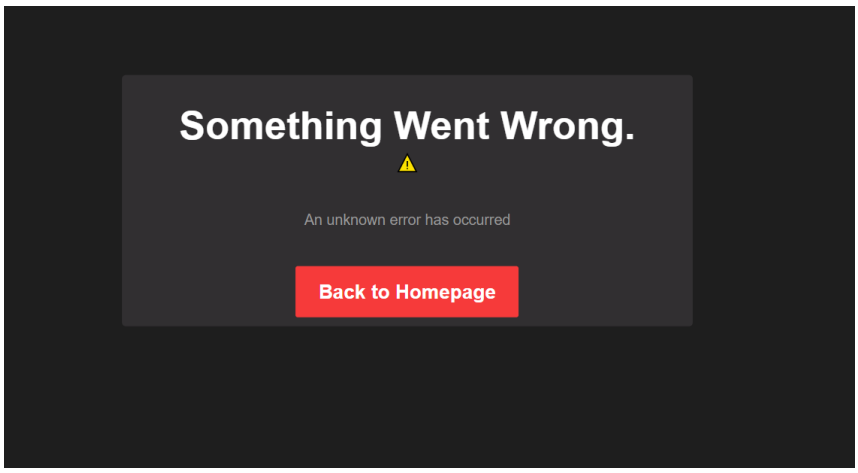


Figure 39: The final implementation of the error page.

11.4 Results Page Version 2

The implementation of the results page encountered challenges due to timeout issues when attempting multiple generation requests simultaneously. As a result, it was decided to limit the page to generate and display only a single melody extension.

While this approach alleviated the risk of overloading the server and potentially causing a denial-of-service attack, it led to the page feeling somewhat sparse.

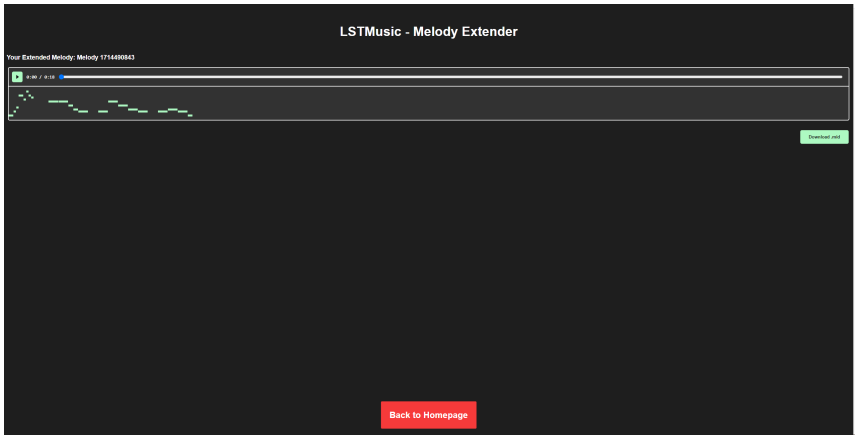


Figure 40: A screenshot of the final implementaion of the results page.

Despite this limitation, the implementation of the results page largely adhered to the designs I created, with the first prototype's skeleton being adapted easily. The clear documentation provided on HTML-MIDI-Player's README[29] made fitting the player and visualiser into my skeleton very straightforward.

Part V
Hosting

April 2024

As I aimed for the project to be accessible through the web, I had to consider hosting options for both the project's frontend and backend; both of which I lacked experience with.

12 Frontend Hosting

Deploying the project's frontend proved to be a quick and straightforward task. Choosing to use GitHub's 'pages' system, I was able to directly host my project's static frontend pages from my repository's `main` branch. This simple process made getting my frontend online very convenient, with updating the deployed frontend being as easy as pushing changes to the project's `main` branch.

Some minor bugs related to the redirects on my site surfaced during initial deployment, but were quickly fixed with some slight alterations to the project's file structure.

13 Backend Hosting

13.1 PythonAnywhere

Due to my lack of experience with backend deployment, I began by deciding to tackle deployment using a beginner-friendly web hosting platform. After some reading on beginner Python forums, Anaconda's PythonAnywhere (https://www.pythonanywhere.com) seemed to be an attractive option, due to its easy-to-use graphical user interface, extensive documentation and community forum.

My attempt began with purchasing a plan for disk space and CPU time to run my generation model for expanding requests, giving me a URL which would be eventually be used to access my project.
Uploading files and installing dependencies proved difficult due to PythonAnywhere's dated file manager and limited memory, causing server stoppages on occasions when installing larger packages like TensorFlow. However, by manually recreating and populating the project's file structure, and installing my requirements with specific memory-saving flags, I was able to get the project's dependencies and source code ready to run.

Running my Flask app launched it on the server without any obvious issues, and after tweaking the frontend to connect it with my provided backend URL, I was ready to see the project live.
However, attempts to expand a simple melody resulted in several timeout errors, despite my debugging tools indicating the acknowledgement, pre-processing and execution of generation attempts.
Further investigation revealed PythonAnywhere's lack of multi-threading support, crucial for my implementation.

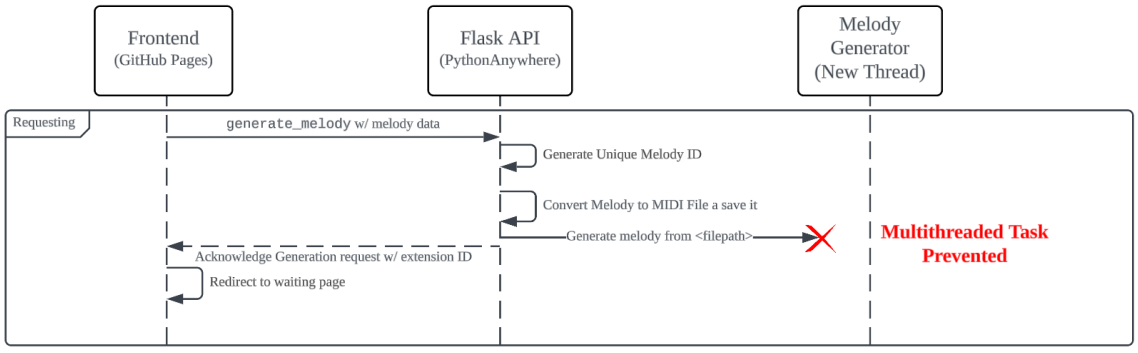


Figure 41: A sequence diagram showing the issue encountered using PythonAnywhere.

Responses in an online forum post from someone experiencing a similar issue[14] suggested a complete restructure of the app to use PythonAnywhere's task scheduler. While this approach had the potential to solve the problem at hand, the time-expense of needing to restructure the project and my understanding that tasks would be executed at predetermined intervals rather than immediately led me to the decision that that this approach wasn't viable, as long wait times would leave users unsatisfied.

With the realisation that there was no good solution to my problem, I opted to instead create my own web server, which wouldn't have any threading restrictions.

13.2 AWS Lightsail

I was made aware of Amazon Lightsail[1] as an alternative to PythonAnywhere on a recommendation from another student while working through the aforementioned issue.

To get started with Lightsail, I chose to follow a resource[47] that provided steps to host a Flask application on it with two web-server tools:

1. Gunicorn [25], used to locally host a Flask application outside of a development environment.
2. Caddy [53], acting as a web server and reverse proxy to direct incoming HTTP and HTTPS requests to the locally hosted Gunicorn Instance.

These used together created a simple "Hello World' app, accessible from the web.

Despite using Linux, requiring the use of a bash shell rather than a graphical interface, the process of hosting this app on Lightsail was much easier than that of PythonAnywhere, largely due to the beginner-friendly nature of the resource I followed.

Transitioning from this HelloWorld setup to the generation backend was made easy thanks to my use of GitHub and Python virtual environments, which made the task of getting my code onto the web server and the management of python packages very easy.

Testing at this point would reveal a reoccurring issue where my backend would stop running after disconnecting from my SSH session, as a result of Gunicorn running as a foreground process. This was quickly resolved by instead running it as a daemon, ensuring Gunicorn continued running in the background even when not directly connected through SSH.

Upon resolving the issue, I proceeded with some final hosting steps, which involved registering the server's IP to a domain and updating the frontend's code to direct requests to this domain.
With these tasks completed, the setup process for my backend server was complete and, more importantly, the application was accessible through the web!

A diagram of this final setup can be seen below, showing how all of the server's different backend components interact.

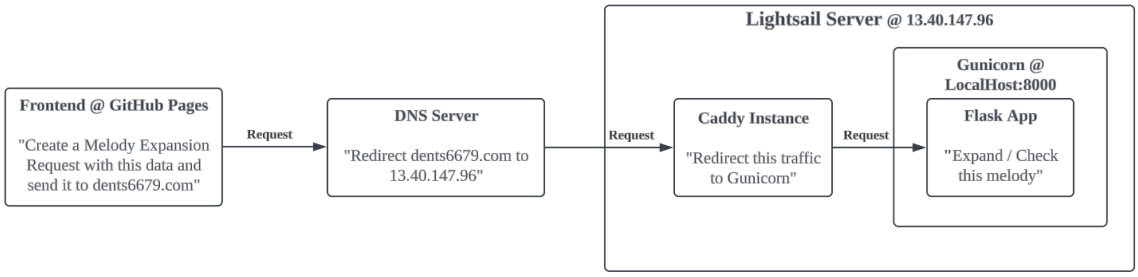


Figure 42: A diagram illustrating the interaction between various components of the project's backend, enabling its exposure to the web.

Part VI
Project Discussion and Future Work

14 User Evaluation

Upon completion of the development phase, my final objective was to conduct a qualitative user evaluation, aimed to assess the suitability of my project's final form and to collect insights regarding potential project enhancements from user feedback.

14.1 User Evaluation Methodology

The project's evaluation was to be performed through a HCI-Style survey, designed to gather feedback from users with diverse levels of expertise. Participants were to engage with LSTMusic for a brief period before responding to questions concerning its generated music, interface designs, and overall suitability.

Initially, I planned to include a second evaluation section featuring a Turing-style test.
In this section, participants would have been presented with three melodies, two extended by my model and one extended by a human.
However, considering the project's focus on developing a web-based toy rather than creating the *best* generation model, I opted to proceed solely with the regular user evaluation survey.

All of the filled forms can be found within the report's appendix.

14.2 User Evaluation Findings

14.2.1 Participant Demographics

User demographics encompassed a spectrum of experience levels regarding music, including several beginners and a professional music producer. This led to several insights from the perspectives of users who would typically use LSTMusic as a toy, and a professional who would use LSTMusic as a part of their workflow.

14.2.2 Generation Model Musicality

Regarding the quality of the generated music, respondents expressed varying opinions. While some appreciated the smooth transition between input and model-generated notes, others found the actual length of the extensions unsatisfactory and the generator unstable, reportedly sending users to the error page on more than one occasion.

The model's tendency towards creating folk-song like melodies was observed and pointed out, with respondents expressing a desire for the model to pick up on other genres of input melodies.

There was a general agreement that the model could correctly pick up on the key of input melodies, but struggled to respect the duration of inputted melodies, with one user critiquing the generated extensions as "dull, uncreative and seemingly random". The majority of users also expressed the desire to input and generate polyphonic music, further highlighting the limitations of the model and music representation used.

This feedback comes with little surprise given the number of outstanding bugs and lack of optimisation that the model has, with user-feedback largely reflecting my own thoughts; that the model requires improvement, and is absolutely not suitable for any sort of professional usage within a DAW plugin.

14.2.3 User Interface

Melody Input Page
Feedback for the melody input page expressed an overall appreciation for the page's design and functionality, with some suggestions for improvement.

In terms of positive feedback, users found the piano and note design engaging and enjoyable to interact with, contributing to a fun, toy-like user experience. The page's aesthetic appeal was praised, although one respondent expressed a preference for centrally aligned buttons.

However, concerns were raised regarding the usability and functionality of the input page. The music producer noted that while the overall layout was expected for a piano roll, they found it clunky to use at times and requested a more modern and streamlined appearance, similar to what can be found in modern DAWs.

Respondents also offered constructive suggestions for features and tools that could enhance the usability and efficiency of the input page, including:

- Clearer instructions for button functionality.
- The inclusion of scissors tool for note editing/splitting.
- The introduction of hotkeys for common actions like note deletion and play/pause functionality.
- An expansion to the range of possible note pitches.

Despite the positive feedback and suggestions, several respondents encountered issues and bugs while using the input page. These included glitches during playback causing notes to sound disjointed, and inconsistencies in note playback when inputting notes into the piano roll.

Overall, feedback highlighted that there are several improvements required for the input page to become entirely fit for purpose, most notably usability refinements, stability and reliability enhancements through bug fixes, and the incorporation of the requested features to align the page closer to user needs and expectations. Perhaps a consultation on an expected feature set during development would've helped as an addition to the project's agile-like development strategy.

Waiting Page
Respondents generally agreed that the purpose of the waiting page was clear and illustrated a feeling of progression through its stylish loading GIF, which frequently received praise for being 'fun' and 'smooth'.

No comments were made on the loading times for melodies, leading to the assumption that users were satisfied with the wait times for their melodies.

Results Page
Feedback for the results page indicated a mix of positive sentiments alongside criticisms and feature requests.

Among the positives, users appreciated the visual representation of their melodies through the MIDI visualisers and the option to download their results.

For improvements, there were suggestions for a more DAW-focused design from the music producer. They indicated their desire for the addition of bar lines, a more precise decimal time representation, more precise play head control, and the inclusion of informational counters to communicate musical features which the model picked up on, like key-signature.

These insights from the music producer clearly highlight a gap in my understanding of the needs of my users, especially for those who I'd consider to use an LSTMusic-like plugin in a DAW.

Again, more user research would've helped me better understand my user requirements, allowing them to help and shape the tool into a form which is more fit for purpose, rather than shaping it myself.

14.2.4 Overall Thoughts

In summary, user feedback has indicated that LSTMusic serves its purpose as a platform for playful experimentation with AI-generated music, but its potential is limited by the shortcomings in its generative capabilities. While users generally praised the site's design and aesthetic, they also

highlighted areas for improvement, particularly in the implementation of more advanced tools and features. So without substantial enhancements to both the model and interface, the likelihood of usage outside of a 'messing about with it'-use-case from professional musicians appears slim.

This feedback generally reflects the aims I had for the project's use case, acting as a platform for those users without DAW skills to pick up and essentially 'mess about' with AI music generation.

The critiques provided by respondents have been extremely helpful in identifying numerous opportunities for refinement in both the project's model and user interface, as well as in the detection of bugs which were overlooked during testing phases, marking the project's evaluation a great success for assessing LSTMusic's suitability and generating improvement ideas.

15 Outstanding Issues and Project Improvements

15.1 Outstanding Bugs / Issues

System testing throughout the project's different phases led to the discovery of bugs and issues with the project, many of which made it into the project's final iteration. The user evaluation portion of the project highlighted additional bugs which weren't discovered during my testing phases.
A final table of the project's known bugs and issues, as well as known causes and potential solutions can be found in the report's appendix.

15.2 Possible Scalability Improvements

After backend development initially concluded, robustness testing revealed long wait times leading to 'timeout' errors for users when multiple extension requests occurred, requiring the scrapping of the plan allowing users to generate multiple extensions. This oversight was entirely mine, as I neglected to anticipate such an issue.
A potential solution could be reached by implementing a task queuing system, where a task broker would delegate extension requests to a network of "consumer" computers, enabling them to process several extensions independently. RabbitMQ[48] and Celery[55] serve as examples of task broker and consumer software, respectively.

Despite recognising the potential benefits of implementing these technologies, the project's timeline constraints prevented any further research or work with them. A high level sequence diagram was created to illustrate a possible configuration of a more scalable system, but no further work was planned.

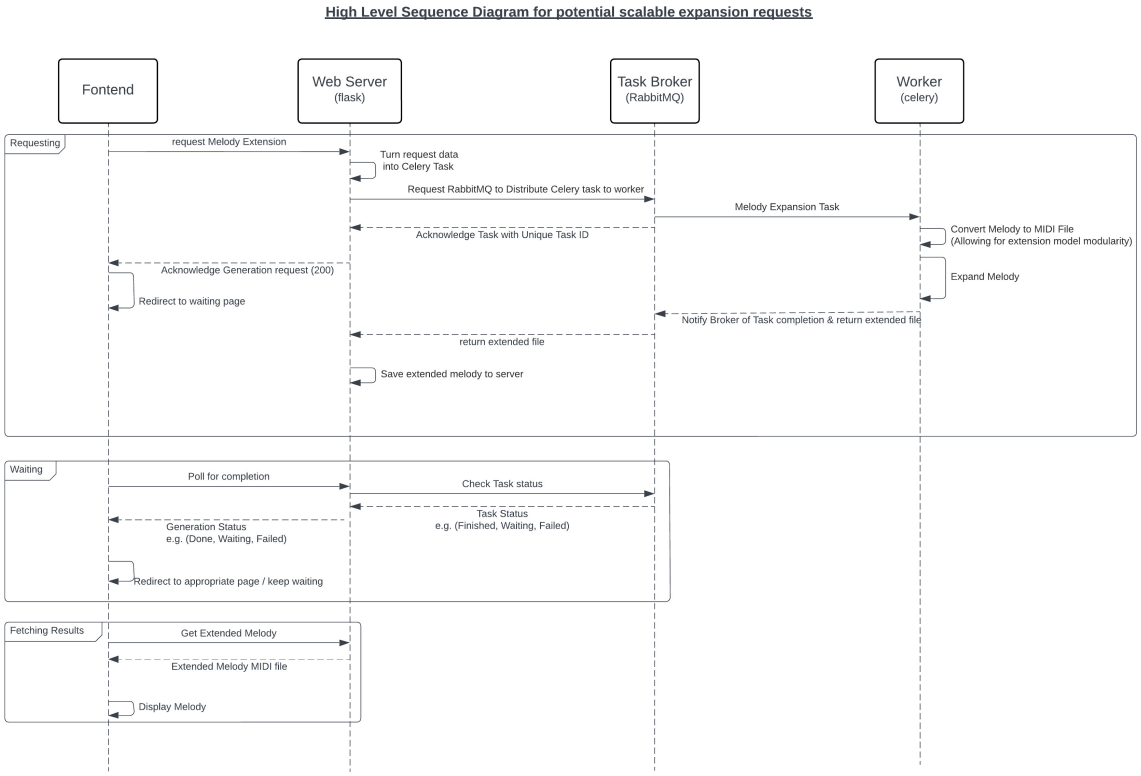


Figure 43: A high level sequence diagram of a more scalable system employing RabbitMQ and Celery.

15.3 Model improvements

Unfortunately, due to time constraints and the looming deadline of the project, I was unable to perform the model improvements which I originally set out to do while designing the backend.

When backend development initially concluded, time for development was quickly running out, and I had to make the choice to either prioritise the refinement of the project's frontend and hosting everything, or to prioritise the refinement of my backend's model through technical experiments and optimisation.

Given the project's aim towards accessibility and playful experimentation, I made the choice to focus on refinement and hosting, as a failure to host the project with a suitable frontend would, for me, be a larger failure than having an unrefined generative model.

I believe this time crunch was caused primarily due to overly-optimistic planning, the presence of the unexpected issues throughout development, and the need to essentially remake the frontend.

This failure in planning left the project's generative model in an unrefined, prototype state, leaving a lot of potential for future improvement, with many different avenues of improvement to explore.

Improvement and optimisation of the project's current monophonic LSTM model would be a good short term effort that could involve a quantitative comparison of hyper-parameters, training processes, dataset usage, preprocessing steps and more.

Thinking longer term, potential improvements could involve experimentation with the N-Gram and GAN models, which were researched but had no further testing. Similar quantitative comparison methods could be used with these and other generative models to evaluate the efficacy of these models, with eventual implementation into my backend.[3]

[3]This would unfortunately require the re-naming of my project from LSTMusic to a different name, GANmusic doesn't exactly roll off the tongue.

Fortunately, the system I have developed is modular enough to accommodate any changes or experiments with little difficulty, making the undertaking of such future work undoubtedly possible.

16 Completion of Project Aims

Regarding the project's core objectives, I believe that my developmental journey for LSTMusic has been largely successful. All core objectives have been satisfactorily achieved, resulting in the availability of LSTMusic for public toying and experimentation at `https://Dents6679.github.io/LSTMusic`.

While the amount of model improvements I ended up performing may have fallen short of expectations, I made an effort to engage with and explore alternative generative models, with both GANs and N-Gram models appearing to be promising for further investigation.

My project's extension objectives were partially fulfilled, with addition of variable notes in the second frontend iteration, somewhat-consistent melody stitching, and a surface level investigation of DAW integration being performed. And although the generation of multiple melodies remains unachieved, the approach I've proposed using RabbitMQ and Celery could be promising for future work. Laying a solid high level plan, like those found during the design parts of my project's different phases, for any implementation to come.

17 Conclusion.

So, while the generative model still has multiple avenues for improvement, with user feedback agreeing, this project has undeniably succeeded in providing me with the hands-on introduction to the frontend, backend, and web hosting fundamentals that I sought at the project's beginning.

LSTMusic is available for use, with it now providing anyone the opportunity to experiment, find entertainment, or benefit from the use of Generative AI in music, beyond the confines of code or DAWs, fulfilling its purpose as the bridge platform it intends to be.

I am happy and confident in saying that the knowledge and skills I've acquired in the duration of this project will directly benefit my future personal development projects, marking the development of LSTMusic, if not a practical success, a significant personal success.

References

[1] *amazon lightsail*. URL: https://aws.amazon.com/lightsail/.

[2] The MIDI Association. *About MIDI-Part 1: Overview*. URL: https://midi.org/about-midi-part-1overview.

[3] The MIDI Association. *About MIDI-Part 3: MIDI Messages*. URL: https://midi.org/about-midi-part-3midi-messages.

[4] *AudioContext - Web APIs — MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/API/AudioContext.

[5] Trym Bø. *MIDI music generation, the hassle of representing MIDI*. Apr. 2023. URL: https://mct-master.github.io/machine-learning/2023/04/19/tryb-midi-hassle-copy.html.

[6] Jason Brownlee. *A Gentle Introduction to Generative Adversarial Networks (GANs)*. 2019.

[7] Matteo Carcassi. *New and improved method for the guitar*. Boston : Oliver Ditson and Co., 1792-1853, p. 8. URL: archive.org/details/newimprovedmeth00carc/.

[8] Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang. *Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding*. 2021. arXiv: 2108.05064 [cs.SD].

[9] Ondřej Cífka. *HTML MIDI Player Basic Example*. URL: https://codepen.io/cifkao/pen/WNwpLzL.

[10] Michael Conner et al. *Music Generation Using an LSTM*. 2022. arXiv: 2203.12105 [cs.SD].

[11] Jade Copet et al. *Simple and Controllable Music Generation*. 2024. arXiv: 2306.05284 [cs.SD].

[12] Michael Scott Asato Cuthber. *Music21 Documentation*. 2006 - 2023. URL: https://web.mit.edu/music21/doc/index.html.

[13] Michael Scott Cuthbert and Christopher Ariza. "music21: A toolkit for computer-aided musicology and symbolic music data". In: (2010).

[14] deleted-user-1224105. *PythonAnywhere Forums: How to enable threads?* Forum Post. 2016. URL: https://www.pythonanywhere.com/forums/topic/3627/.

[15] Thomas Dent. *LSTMusic Better Loading Gif*. URL: https://dents6679.github.io/LSTMusic/frontend-assets/better-loading.gif.

[16] Mark Doppler. *ExtendMusic*. URL: https://www.extendmusic.ai/.

[17] Flask. *Welcome to Flask — Flask Documentation (3.0.x)*. URL: https://flask.palletsprojects.com/en/3.0.x/.

[18] *Flask Quickstart*. Pallets. URL: https://flask.palletsprojects.com/en/2.3.x/quickstart/#variable-rules.

[19] Nathan Fradet et al. "MidiTok: A Python package for MIDI file tokenization". In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[20] g200kg. *webaudio-pianoroll*. GitHub Repository. URL: https://github.com/g200kg/webaudio-pianoroll.

[21] g200kg. *WebAudio-PianoRoll Samples*. URL: https://g200kg.github.io/webaudio-pianoroll/index2.html.

[22] Michael Good. "MusicXML for notation and analysis". In: *The virtual score: representation, retrieval, restoration* 12.113–124 (2001), p. 160.

[23] Google. *Chrome Music Lab*. URL: https://musiclab.chromeexperiments.com/.

[24] Garrett Grimm. *MidiWriterJS*. Github Repository. 2023. URL: https://github.com/grimmdude/MidiWriterJS.

[25] *gunicorn - python wsgi http server for unix*. URL: https://gunicorn.org/.

[26] Curtis Hawthorne et al. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=r1lYRjC9F7.

[27] Mohamed Abbas Hedjazi and Yakup Genc. "Efficient texture-aware multi-GAN for image inpainting". In: *Knowledge-Based Systems* 217 (2021), p. 106789. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106789. URL: https://www.sciencedirect.com/science/article/pii/S0950705121000526.

[28] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[29] *html-midi-player*. GitHub Repository. 2020 - 2023. URL: https://github.com/cifkao/html-midi-player.

[30] David Huron. *KERN Representation*. humdrum, 2016. URL: https://www.humdrum.org/rep/kern/index.html.

[31] Bernd Krueger. *Classical Piano Midi Page*. 1996. URL: http://www.piano-midi.de/.

[32] Image Line. *MIDI Scripting Device API reference*. URL: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/midi_scripting.htm.

[33] Antoine Liutkus et al. "Relative Positional Encoding for Transformers with Linear Complexity". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 7067–7079. URL: https://proceedings.mlr.press/v139/liutkus21a.html.

[34] AudioCipher Technologies LLC. AudioCipher's Homepage. URL: https://www.audiocipher.com/.

[35] Doug McKenzie. *Jazz Piano MIDI Library*. URL: https://bushgrafts.com/midi/.

[36] Microsoft. Aug. 2019. URL: https://www.microsoft.com/enus/research/project/ai-music/.

[37] Microsoft. *Copilot in Windows Other AI-Powered Features — Microsoft*. URL: https://www.microsoft.com/en-gb/windows/copilot-ai-features?r=1.

[38] *Music Macro Language — Electronic Music Wiki*. URL: https://electronicmusic.fandom.com/wiki/Music_Macro_Language.

[39] *music21 interval — music21 documentation*. URL: https://web.mit.edu/music21/doc/moduleReference/moduleInterval.html#music21.interval.GenericInterval.transposePitchKeyAware.

[40] *music21.stream.base — music21 documentation*. URL: https://web.mit.edu/music21/doc/moduleReference/moduleStreamBase.html.

[41] C. Olah. *Understanding LSTM networks*. Aug. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[42] Alexander Osipenko. *Markov Chain for music generation*. 2019. URL: https://towardsdatascience.com/markov-chain-for-music-generation-932ea8a88305.

[43] Bo Pang, Erik Nijkamp, and Ying Nian Wu. "Deep Learning With TensorFlow: A Review". In: *Journal of Educational and Behavioral Statistics* 45.2 (2020), pp. 227–248. DOI: 10.3102/1076998619872761. eprint: https://doi.org/10.3102/1076998619872761. URL: https://doi.org/10.3102/1076998619872761.

[44] Sheetal S. Patil et al. "Music Generation Using RNN-LSTM with GRU". In: *2023 International Conference on Integration of Computational Intelligent System (ICICIS)*. 2023, pp. 1–5. DOI: 10.1109/ICICIS56802.2023.10430293.

[45] Christine Payne. *MuseNet*. Apr. 2019. URL: openai.com/blog/musenet.

[46] Martin Pfleiderer et al., eds. *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[47] Anthony (Pretty Printed). *How to Deploy a Flask App to a Linux Server with a Domain Name*. July 2023. URL: https://youtu.be/vfZgHX5ttsY?si=AFdrNmZZpWmxYK2j.

[48] RabbitMQ. *Messaging that just works — RabbitMQ*. 2019. URL: https://www.rabbitmq.com/.

[49] Scott Reed et al. *Generative Adversarial Text to Image Synthesis*. 2016. arXiv: 1605.05396 [cs.NE].

[50] Adam Roberts et al. “Magenta Studio: Augmenting Creativity with Deep Learning in Ableton Live”. In: *Proceedings of the International Workshop on Musical Metacreation (MUME)*. 2019. URL: http://musicalmetacreation.org/buddydrive/file/mume_2019_paper_2/.

[51] Samsung. *Galaxy AI — Mobile AI on Galaxy S24 Ultra*. URL: https://www.samsung.com/us/smartphones/galaxy-s24-ultra/galaxy-ai/.

[52] Helmut Schaffrath. *The Essen Folksong Collection in Kern Format*. Ed. by D. Huron. [computer database]. Menlo Park, CA, 1995.

[53] Caddy Web Server. *Caddy 2 - The Ultimate Server with Automatic HTTPS*. URL: https://caddyserver.com/.

[54] Ian Simon and Sageev Oore. *Performance RNN: Generating Music with Expressive Timing and Dynamics*. https://magenta.tensorflow.org/performance-rnn. Blog. 2017.

[55] Ask Solem. *Celery - Distributed Task Queue — Celery 5.4.0 documentation*. URL: https://docs.celeryq.dev/.

[56] ADSR Sounds. *Hexcel by ADSR - Hex-Based Midi Generator for Mac/Windows*. URL: https://www.adsrsounds.com/product/software/hexcel/.

[57] Google Brain Team. *Welcome to Mangenta!* June 2016. URL: https://magenta.tensorflow.org/blog/2016/06/01/welcome-to-magenta/.

[58] Dominique Vandenneucker. *MIDI tutorial for programmers*. URL: https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html.

[59] Valerio Velardo. *Generating Melodies with LSTM Nets*. May 2020. URL: https://www.youtube.com/watch?v=FLr0r-QhqH0&list=PL-wATfeyAMNr0KMutwtbeDCmpwvtul-Xz.

[60] Kenny Yip. *Code minesweeper game with Javascript*. Mar. 2022. URL: https://www.youtube.com/watch?v=AfhfAxKFP-s.

Part VII
Appendix

A Github Repository

The GitHub repository used within this project can be found on the LSTMusic GitHub Repository at `Https://GitHub.com/Dents6679/LSTMusic`.

B Music Theory Concepts

Summarised below are some of the core music theory concepts which were used throughout this project.[4]

B.0.1 Monophonic and Polyphonic Music

Monophony and Polyphony are terms that can be used to describe the texture or 'layering' of a musical composition.

Monophonic music, the simplest form of musical composition, comprises a single melody without any accompanying harmonies or chords.

Polyphonic music, conversely, can involve multiple independent melodies simultaneously, combined to create harmonies and intricate musical pieces. This complexity enriches the music, offering a multi-layer listening experience.

B.0.2 Notes

Notes in music represent the fundamental building blocks of melody. Each note has two essential components: its pitch and duration.
Together, the manipulation of pitch and duration allows musicians to create melodies and rhythmic patterns which can feel 'musical'.

Pitch
Pitch refers to the frequency of the sound produced by an instrument. It is essentially how high or low of a sound is produced.
In musical notation, pitch is denoted using letters from A to G, with variations known as 'accidentals' that can slightly modify the pitch of the note. These accidentals include sharps (♯) and flats (♭), which raise or lower the pitch by a semitone (the smallest pitch step that I am concerned with in this project) respectively.
Pitches can span across multiple octaves, with each octave representing a doubling or halving of a note's frequency.

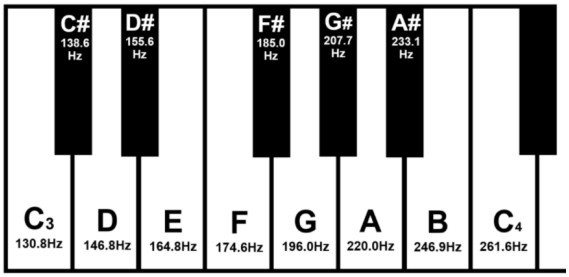


Figure 44: The range of frequencies and corresponding notes in a single octave of a piano.

Duration
Duration indicates how long a note is played or held, contributing to a feeling of rhythm in a piece.

Duration is represented by different types of note symbols, each indicating a specific length of time that the note is sustained after being pressed. Note durations are represented as a fraction of a semibreve (or 'Whole Note'), which lasts of 4 beats, or 1 bar, in $\frac{4}{4}$.[5]

Shorter duration notes, such as eighth notes or sixteenth notes, create a faster rhythmic pattern, while longer duration notes, such as whole notes or half notes, result in a slower pace. The combination of various durations creates rhythmic diversity within a piece.

[4] Music theory is an astonishingly huge field of work which some dedicate their entire lives to. I will only be touching upon the most basic concepts required to understand the decisions made in this project.

[5] $\frac{4}{4}$ was chosen for this project because of its widespread familiarity among English users, particularly those without a background in music theory, which the project primarily targets.

B.0.3 Key, Tonic and Mode

The Key of a melody can be thought of a central note which acts as the melody's home state. All of the notes in a melody can revolve around this note (known as a Tonic). Modes are the 'character' of a Key[7], coming in both Major and Minor. The mode used in a peice can lead to slight differences in the notes accompanying base note, making the melody sound 'bright' when in major or 'sad' when in minor[6].

B.0.4 Intervals and Transpositions

An interval is simply a label for the difference in two notes. Many intervals have their own unique names, but can also be represented as an integer denoting the number of semitones (or 'steps' on a piano, shown below) between one note and another.

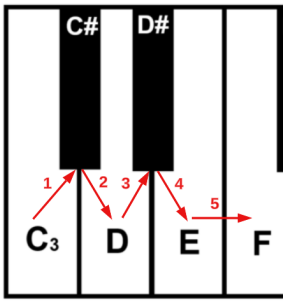


Figure 45: A 5 semitone interval on a piano roll, from a C note to an F note.

This concept can also apply to the key of a melody, with a C Major to F Major key also having an interval of 5 semitones.
The movement of a melody over an Interval is called a Transposition, which allows the melody's key to be changed while preserving the tonal differences between the Melody's individual notes.

[6]These are large generalisations, but are good enough explanations for the purpose of this section.

C Existing Bugs

Unresolved Bugs / Issues

Bugs & Issues

Bug / Issue	**Project component**	**Cause**	**Solution?**
Melodies sometimes do not expand	Extension Model	Unsure: Potentially due to model representation.	Short term: Unknown Long term: Replace Model / Improve Model Representation
Shorter melodies generate empty ‘buffer’ space between the end of user input and the beginning of generative input.	Extension model / Input melody pre-processing function.	Unsure: Requires further investigation.	Short term: Unknown Long term: Replace Model
Inputted Note lengths longer than a whole note are separated into whole notes.	Extension Model / Input melody pre-processing function	Function parsing piano roll events may be incorrect. Exact cause unknown	Unknown
Generations Frequently fail.	Extension Model	KeyError when mapping model output from integer representation to the time-series representation.	Short term: Add a try/catch block which retries generating the note. Long term : Re-train model with an additional sample melody that includes all 128 MIDI

Bug / Issue	**Project component**	**Cause**	**Solution?**
The Model doesn’t respect users’ input genre	Extension Model	Lack of training data variety	Allow users to choose the genre of their melody, and train multiple models on genre, hot-swapping the model’s .keras parameter files when necessary.
The model doesn’t respect users’ input melody durations	Extension Model	Lack of training, or unsuitable training dataset.	Use a more suitable dataset towards Melodies
Melodies are monophonic	Extension Model & Music Representation	music representation is inflexible.	Easy: Add an N-Gram model to the existing model to generate chords. Hard: Use a better music representation
Piano roll playback bugs out.	Piano roll	Requires further investigation.	Fix issue in source code and submit PR to repo.
The first inputted note doesn’t play music.	Piano roll AudioContext	Browsers seem to require user input before audio contexts can play	Slight bodge: Show a welcome splash, with an ‘ok’ button which initialises this audio context. Actual Fix: Requires further research
The piano roll’s sound doesn’t sound like a piano	Piano roll callback synth function.	Only a simple sawtooth wave is used for playing notes on the piano roll	Create a more intricate AudioContext callback function which sounds more like a piano
Mobile devices are not Supported	PianoRoll	Scaling issues with the piano roll make the site	Unknown

Bug / Issue	**Project component**	**Cause**	**Solution?**
		near impossible to use with a touchscreen.	
Results page Timer updates inconsistently.	Results page - HTML-MIDI-Player	Unsure of actual cause, assuming timer only updates when a note is played.	Unknown, investigate source code to find a potential fix.

D Project Proposal

Proposal: Generating Original music from user-created melodies.

Thomas Dent
Student 246518
Computer Science and Artificial Intelligence
Supervisor: Kingsley Sage

Project Background

In recent years, there has been a notable surge in the application of AI-Driven productivity tools to enhance user creativity and productivity. Many of these tools enable users to augment their digital content with AI-Generated elements, Adobe's generative expand tool in photoshop being a prominent example. [1]

Figure 1: Examples of Content Expansion

The recent trend has sparked my curiosity and led to a question: Can this concept be applied to music production? It seems that researchers have the same question and have created research initiatives and projects with this aim. Google's generative music research project 'Magenta Studio' stands as an interesting and noteworthy example, with specific focus on providing AI tools to the widely used Ableton music production software in the form of a plugin. [2]

Magenta's 'continue' tool directly addresses the question I've posed, allowing users to generate music that seamlessly follows user-provided melodies within a chosen track.

Objectives

The general aim of this project is to develop a similar online musical tool akin to magenta's 'continue' tool, allowing the generation of 'original' music based on a user-provided melody.

Specifically, the project will investigate the following primary, extension and fallback objectives:

Primary objectives

1. Research and evaluate existing work in the field of AI Music generation to gather an idea of currently existing technologies.
2. Identify Successes and Failures of existing AI Music tools in professional software, research studies and online toys.
3. Design and plan the tool's frontend using the insights gained from Objectives 1 & 2.
4. Explore the effectiveness of various backend implementations through prototyping.

5. Conduct qualitative and quantitative assessment to determine how successful the tool is. (As it stands, I would like the tool to create melodies which are indistinguishable from human made melodies. This may be subject to change).
6. Reflect on the successes and challenges encountered during the project.

Extension Objectives

1. Modify the tool to generate music around the user's melody, rather than creating a separate track.
2. Enhance the tool to provide multiple melodies, giving users the option to select from a variety of generated tracks.
3. Enable users to export their generated melodies as MIDI files for importation into music production software.

Fallback Objectives

1. Create a web-based tool for users to create short MIDI melodies.

Relevance

This project's significance lies in its potential to aid music creation, empowering users to generate original compositions from their own melodies with the assistance of AI technology.

The objectives I've outlined above will serve a dual purpose to drive the development of my tool while also providing an opportunity for me to assess and enhance my existing technical skills and learn new languages and frameworks and parts of computing which are currently a mystery to me.

I currently have technical experience with machine learning libraries using Python from my modules in prior years and slight UI design experience from personal projects, with no experience creating web content.

As it stands, these skills alone will not allow me to complete this project. I anticipate that I will need to learn the following skills and technologies to be able to fulfil my primary and extension objectives:

- Proficiency in HTML, CSS and JavaScript for the Tool's frontend.
- Familiarity with a framework for integrating the frontend with A python backend (Django seems like an attractive option).
- Website Design.
- Website Hosting.
- Fundamental knowledge of music theory.

Resources Required

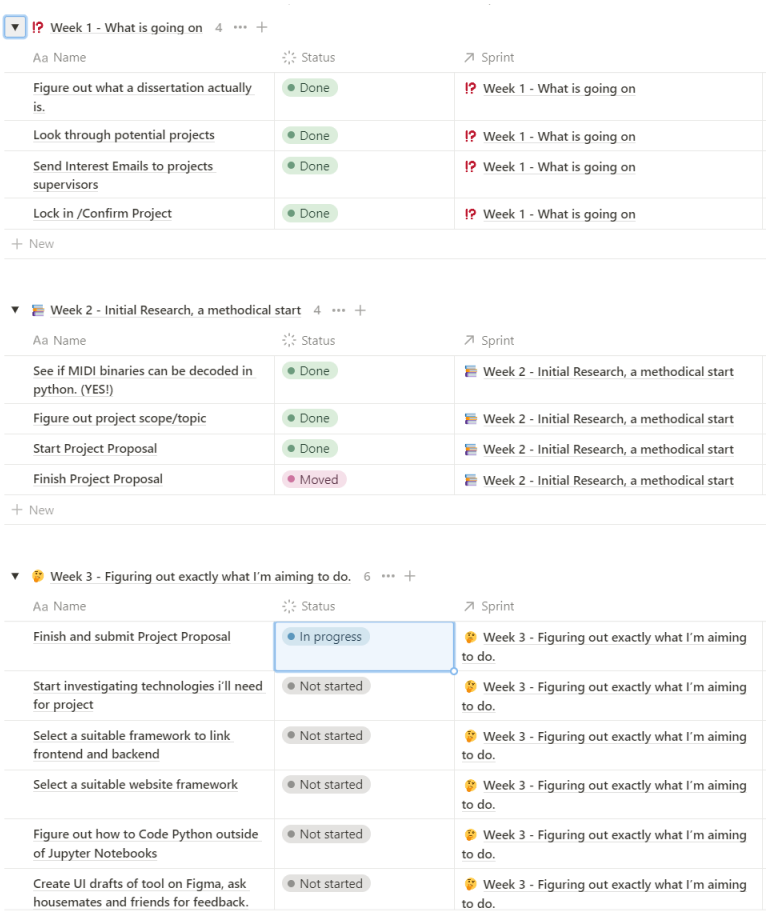
This project may involve reserving study or meeting rooms for conducting user testing on an as-needed basis.

Although I currently lack experience in web hosting, I anticipate the need for some sort of online hosting resource to make my tool available on the web. I plan to specify the necessary resources once I research the hosting strategy for my tool.

Planning

I've created a notion workspace to facilitate my project's planning process, as I've found it to be an extremely helpful tool for organisation in a prior project (Software engineering Group 4's documentation).

My intention is to maintain this notion workspace, using it as a central hub for monitoring my background research, references, project timeline, weekly work updates and keeping track of supervisor meetings. Below is a screenshot of one part of my workspace, the current task list, which shows my week-by-week tasks.



!? Week 1 - What is going on 4

Aa Name	Status	Sprint
Figure out what a dissertation actually is.	Done	!? Week 1 - What is going on
Look through potential projects	Done	!? Week 1 - What is going on
Send Interest Emails to projects supervisors	Done	!? Week 1 - What is going on
Lock in /Confirm Project	Done	!? Week 1 - What is going on

+ New

Week 2 - Initial Research, a methodical start 4

Aa Name	Status	Sprint
See if MIDI binaries can be decoded in python. (YES!)	Done	Week 2 - Initial Research, a methodical start
Figure out project scope/topic	Done	Week 2 - Initial Research, a methodical start
Start Project Proposal	Done	Week 2 - Initial Research, a methodical start
Finish Project Proposal	Moved	Week 2 - Initial Research, a methodical start

+ New

🤔 Week 3 - Figuring out exactly what I'm aiming to do. 6

Aa Name	Status	Sprint
Finish and submit Project Proposal	In progress	🤔 Week 3 - Figuring out exactly what I'm aiming to do.
Start investigating technologies i'll need for project	Not started	🤔 Week 3 - Figuring out exactly what I'm aiming to do.
Select a suitable framework to link frontend and backend	Not started	🤔 Week 3 - Figuring out exactly what I'm aiming to do.
Select a suitable website framework	Not started	🤔 Week 3 - Figuring out exactly what I'm aiming to do.
Figure out how to Code Python outside of Jupyter Notebooks	Not started	🤔 Week 3 - Figuring out exactly what I'm aiming to do.
Create UI drafts of tool on Figma, ask housemates and friends for feedback.	Not started	🤔 Week 3 - Figuring out exactly what I'm aiming to do.

Figure 2: A Screenshot of the Tasks from weeks 1-3 of my notion workspace.

I currently do not have an overarching plan for the whole year. I would like to discuss this in a meeting to get an idea of how long each stage of a project like this would take.

Availability

Below is my calendar's typical weekday plan. Each day is subject to change and may differ slightly from how I have planned it. My timetable will of course change between terms.

My work shifts are consistently on Monday, Tuesday and weekends. I've made it clear to my work about my academic commitments and will not hesitate to quit if my job interferes too much with my studies.

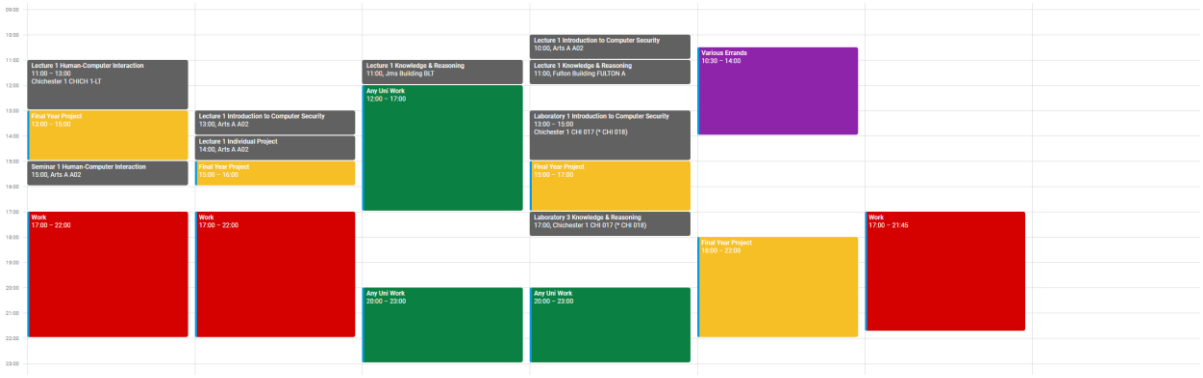


Figure 3: Calendar Availability

I am available for meetings at any time I'm not in a lecture or at work.

Works Cited

[1] Clark, P. (2023, 07 27). *Photoshop releases new Generative Expand workflow and global language support for Firefly-powered features*. Retrieved from Adobe Blog: https://blog.adobe.com/en/publish/2023/07/27/photoshop-releases-new-generative-expand-workflow-global-language-support-for-firefly-powered-features

[2] Roberts, A. E. (2019). *Magenta studio: Augmenting creativity with deep learning in ableton live.* research.google.

E User Testing Compliance Form

User Testing Compliance Form for UG and PGT Projects*
School of Engineering and Informatics
University of Sussex

This form should be used in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects".

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with **all** of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation. Note points are written in past tense, as you are effectively confirming, once your project has been completed, that it was conducted in a way that complied with all of the points.

If this is not the case, you should refer back to the "Research Ethics Guidance for UG and PGT Projects" document for further guidance.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

2. The study materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

---

*This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information are deemed to be high risk and require ethical approval through the relevant C-REC.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

4. No incentives were offered to the participants.
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

6. No participant was under the age of 18.
   Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

7. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.
   Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

8. Neither I nor my supervisor are in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

10. All participants have been informed of my contact details, and the contact details of my supervisor.

All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.

11. The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session.
    Participants must be provided with sufficient information prior to starting the session, and in the debriefing, to enable them to understand the nature of the investigation.

12. All the data collected from the participants is stored securely, and in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymised form.

Project title: LSTMusic: a Web-Based AI Melody Extension Tool

Student's Name: Thomas Dent

Student's Registration Number: 246518

Student's Signature:

Date: 01.05.2024

Supervisor's Name: **Dr Kingsley Sage**

Supervisor's Signature:

Date: **1 May 2024**

F User Evaluation Form + Results

LSTMusic - Evaluation

Hello!
Thank you for choosing to evaluate my final year project. Your assistance is massively appreciated, and will massively help towards the evaluation and reflection of my project.

Study Briefing

The aim of this study is to conduct a user evaluation to collect feedback and data on the web-application developed as part of my Final year Project, called LSTMusic.

LSTMusic is a Web-Based Music Generation Toy which aims to give people the chance to experiment and toy with generative AI, and how it can interact with music.

The main motivation behind this project is to allow people to do this experimentation and toying without the hassle of downloading large music production software packages, plugins, or dealing with unfriendly command-line interfaces.

This evaluation is needed to assess different parts of LSTMusic. like its user interface design, music generation 'musicality', and the general 'feel' of LSTMusic.
It's important that a wide range of people are used when assessing this project, so that any evaluation is not biased towards those with certain interests or experience.

This study consists of 3 main parts, where you will be asked to do the following:

- Answer some 'about you' multiple choice questions to gauge what sort of person you are.
- Use & toy around with LSTMusic for a short amount of time.
- Answer a series of multiple choice & written questions relating to your experience while using the web application.

All of your responses to this form will be collected after it is submitted.

If you have any questions, please feel free to ask me by contacting me through any of my contact information, found below.

You have the right to withdraw from this study at any time by simply exiting the form. The contents of this form will not be used unless you have submitted it via the 'submit' button on the final page.
Please be assured that any data collected in this form will be stored securely and anonymously.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

Please confirm that you have read the above text, and agree to take part in this evaluation study. *

- (●) I Agree to take part in this study (Continue)
- ( ) I do not Agree to take part in this study (Please Exit)

About you.

To start out, please answer the following questions about yourself.

These questions are important to gauge your familiarity & knowledge with some of the concepts relating to this project, such as music theory, web applications and AI.

How familiar are you with the following concepts? *

Please pick the option which describes your level familiarity/experience the most.

	I'm not familiar.	Beginner	Hobbyist	Student	Professional
Basic music theory (notes & scales)	( )	(●)	( )	( )	( )
Rhythm and timing in music	( )	( )	(●)	( )	( )
Creating computer software	( )	( )	( )	(●)	( )
How generative AI works	( )	( )	( )	(●)	( )

How often do you use the following technologies / tools? *

Please pick the option which describes your frequency of use the most.

	Never	Rarely (A few times a year)	Infrequently (Every few months)	Often (Once or more per month)	Frequently (Once or more per week)	Very Frequently (Almost daily / Daily)
Large Language Models (ChatGPT or similar)	○	○	○	○	○	◉
AI image generation software (Dall-E, Photoshop's generative expand tool)	○	○	○	○	◉	○
Music production software (FL Studio, Ableton, Logic)	◉	○	○	○	○	○
Instruments (playing piano, guitar, etc...)	◉	○	○	○	○	○

Testing LSTMusic.

Please now take some time to experiment and toy about with LSTMusic.

- Please spend at least 5 minutes experimenting and toying about with the site.
- You are free to generate as many melodies as you like, and can note down any thoughts or findings while toying.
- If something breaks, don't worry, note it down!
- After experimenting, please return to the survey and press "Next".

LSTMusic can be accessed by clicking HERE, or typing "https://Dents6679.github.io/LSTMusic" into your browser.

Once you've finished experimenting, press 'Next'.

Please note:
If you're accessing this from a university network, there is a high likelihood that access to LSTMusic processing server will be blocked by a firewall, resulting in a "the generation server is not available right now. Please try again later." message. Please use a VPN or complete this survey on a different network if this occurs.

If any other catastrophic issues occur, please message me via WhatsApp on +447860131409.

Thank you

Thank you for spending the time to experiment and toy with LSTMusic. The rest of this survey is intended for you to give feedback about the different parts of LSTMusic.

Questions: Generated Music.

These questions are only concerned about the music LSTMusic generated.

To what extent to do you agree or disagree with the following statements regarding the music LSTMusic generates. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't Know
The PITCH and SCALE of the generated notes felt like they were influenced by the melody you inputted.	○	○	○	◉	○	○
The LENGTH of the generated notes felt like they were influenced by the melody you inputted.	○	○	○	◉	○	○
The transition between your music and the generated music felt smooth.	○	○	○	○	◉	○
The length of the generated music felt long enough.	○	○	○	◉	○	○
The rhythm of the generated music complemented the input melody.	○	○	○	◉	○	○
The generated music felt too arpeggiated.	○	◉	○	○	○	○

Is there anything you like or dislike about the generated music?

Transition between input and AI generated notes was smooth and unnoticeable

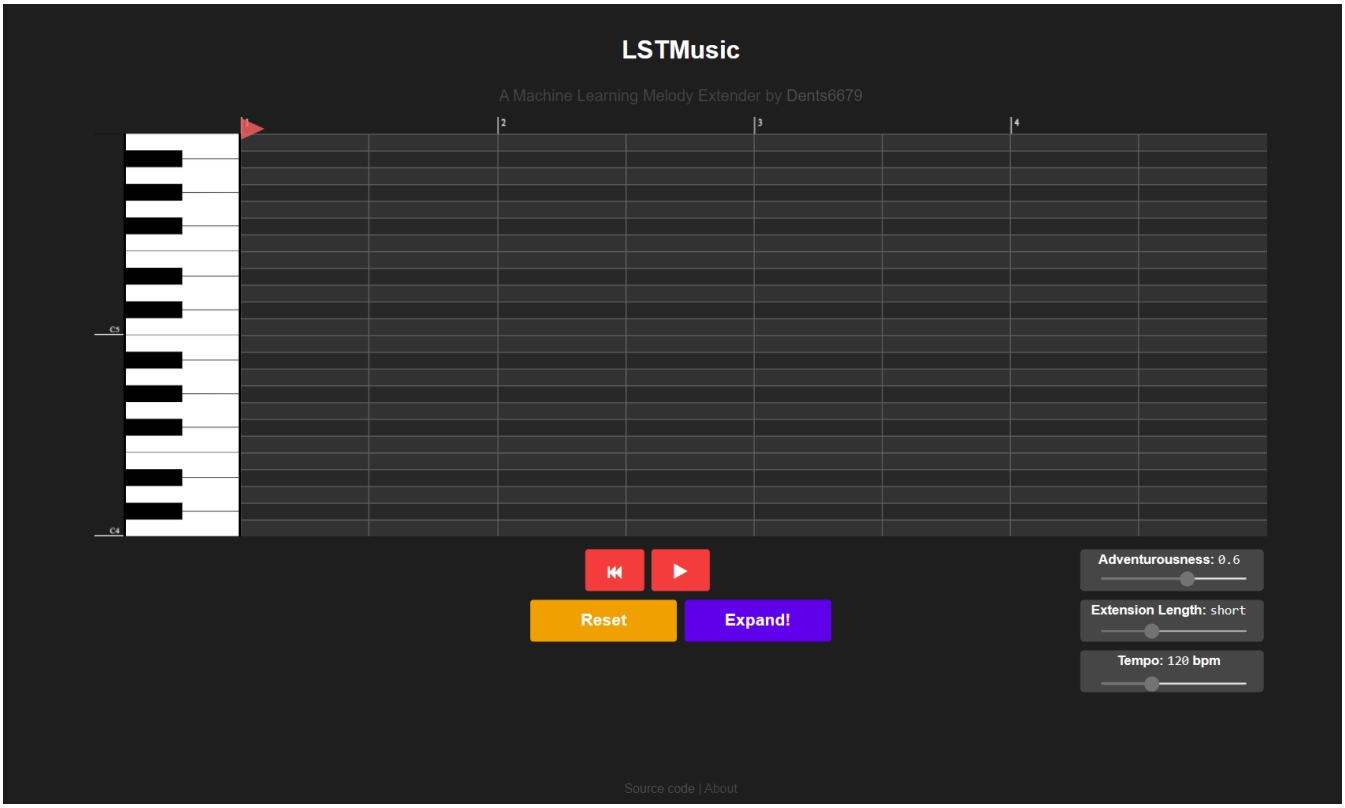
Are there any features, abilities or capabilities which you feel should be added or changed to improve the music LSTMusic produces?

It broke on many occasions when trying to run, but when it worked it worked fairly well

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Melody input page / Homepage

The below questions are concerned about only the input page of LSTMusic, pictured below.


LSTMusic
A Machine Learning Melody Extender by Dents6679
1
2
3
4
C5
C4
Reset
Expand!
Adventurousness: 0.6
Extension Length: short
Tempo: 120 bpm
Source code | About

To what extent to do you agree or disagree with the following statements regarding the melody input page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	◉	○	○
The functionality of the different buttons are clear.	○	○	○	○	◉	○
There was a feeling of cause and effect when interacting with the page.	○	○	○	○	◉	○
The interface is visually appealing.	○	○	○	◉	○	○
The pitch range of the piano roll was large enough.	○	○	○	○	◉	○
The interface of the page matched expectations.	○	○	○	○	◉	○
The sound produced by the piano roll is suitable and appealing.	○	○	○	◉	○	○
Altering the inputted melody was	○	○	○	◉	○	○

intuitive.

intuitive.

Is there anything you like or dislike about the input page?

the piano and chosen note design was fun to play around with

Are there any features, options or tools you feel should be added or changed to improve the input page?

A few instructions on what some of the buttons do, but they were fairly easy to figure out with experimentation

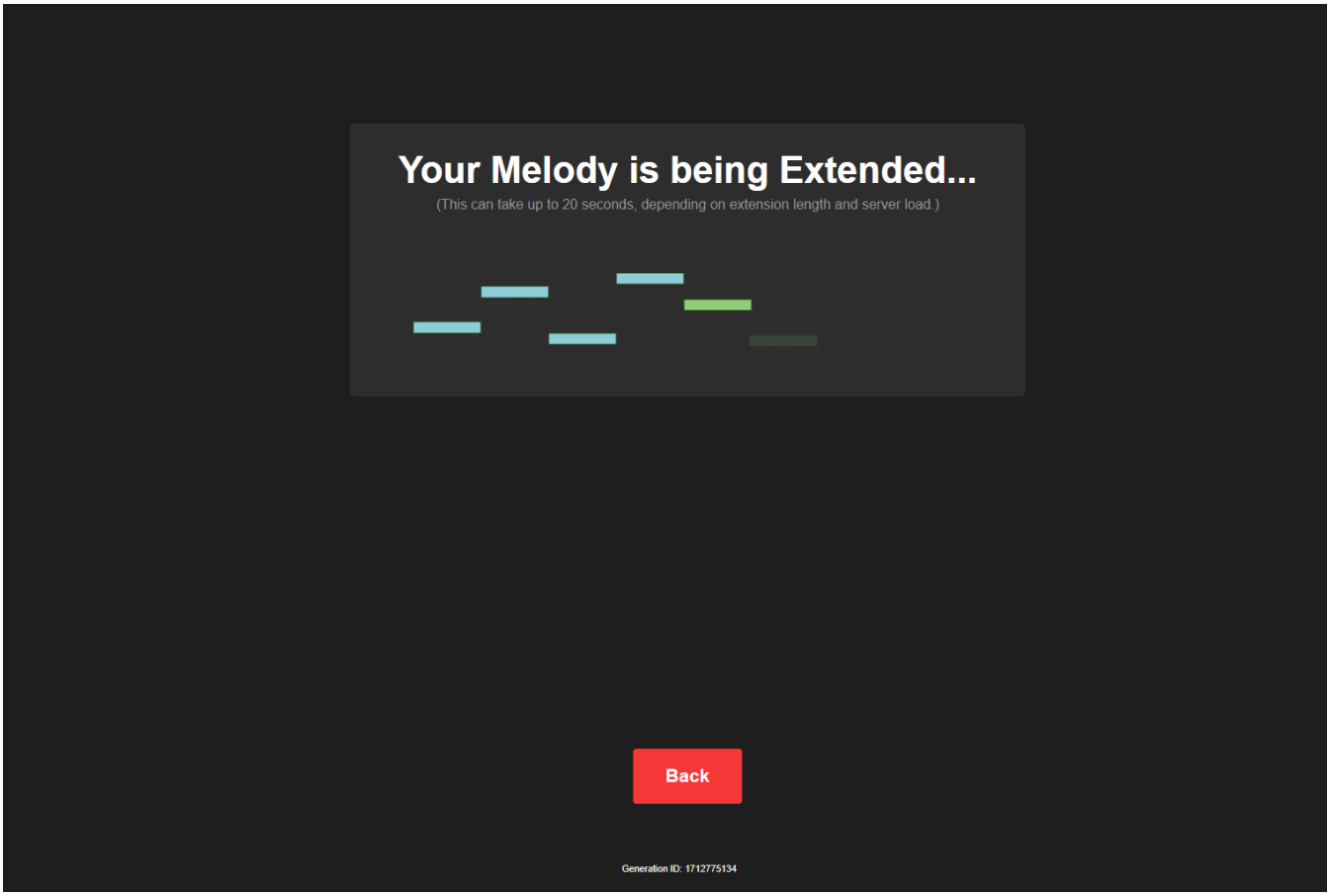
Did you encounter any issues or bugs with the input page? If you did, please specify them below.

It didn't want to run a few times when expanding an inputted melody

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Waiting page

The below questions are concerned about only the waiting page of LSTMusic, pictured below.


Your Melody is being Extended...
(This can take up to 20 seconds, depending on extension length and server load.)
Back
Generation ID: 1712775134

To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
There is a feeling of progression while waiting.	○	○	○	○	◉	○
The waiting page is visually appealing.	○	○	○	○	◉	○
Enough information was given about the expected waiting time.	○	○	○	○	◉	○
Waiting times were acceptable.	○	○	○	○	◉	○

Is there anything you like or dislike about the waiting page?

It was smooth and the animation was fun

Are there any features, options or tools you feel should be added or changed to improve the waiting page?

Sufficient as is

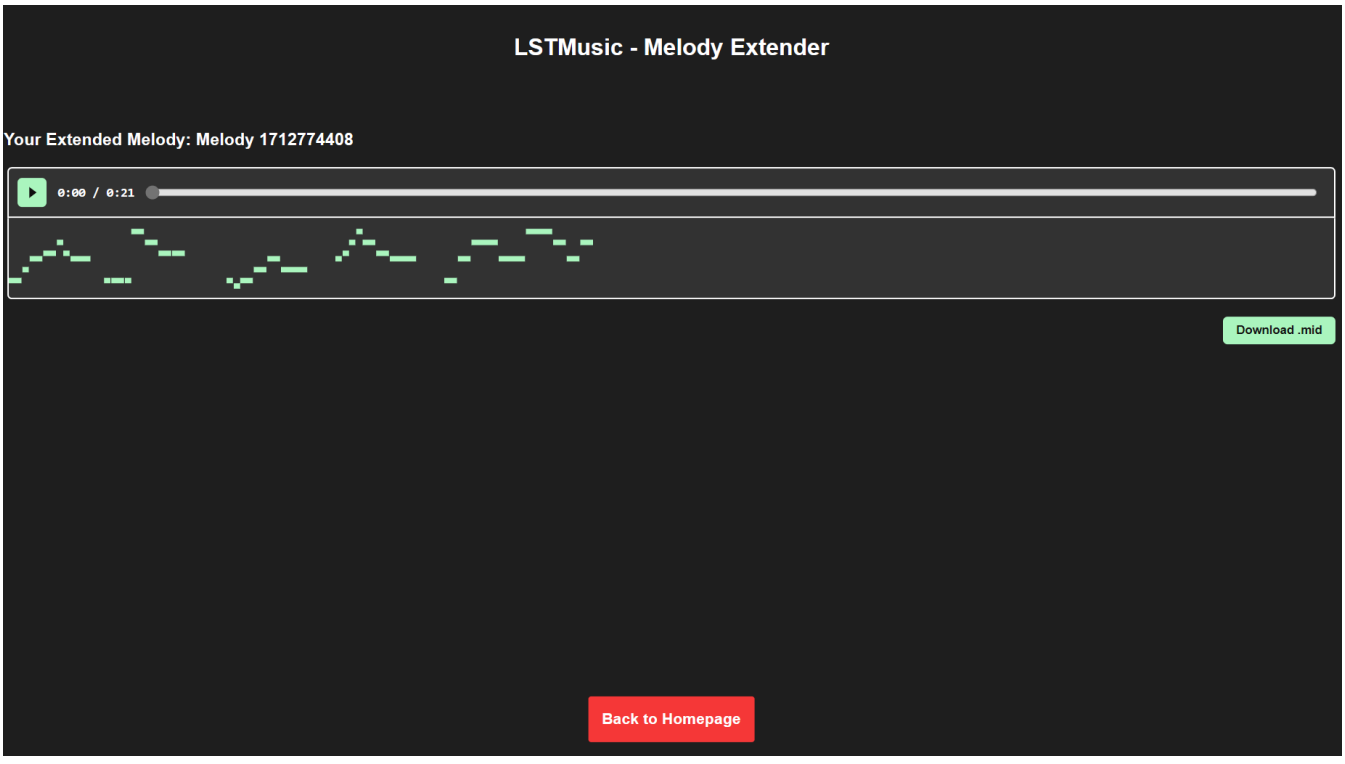
Did you encounter any issues or bugs with the waiting page? If you did, please specify them below.

no bugs encountered

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Results page

The below questions are concerned about only the results page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The sound produced by the player is suitable and appealing.	○	○	○	○	◉	○
The page is visually appealing.	○	○	○	○	◉	○
The controls available on the player are sufficient.	○	○	○	○	◉	○
The song visualiser is fit for purpose.	○	○	○	○	◉	○

Is there anything you like or dislike about the results page?

I liked the visualisation of the melody notes and the fact you could download

Are there any features, options or tools you feel should be added or changed to improve the results page?

No comment

Did you encounter any issues or bugs with the results page? If you did, please specify them below.

No issues encountered

Feel free to make any further comments on the results page which weren't addressed by the above questions.

Questions: LSTMusic as a whole

The following questions are concerned about the entirety of LSTMusic.

To what extent to do you agree or disagree with the following statements regarding the entirety of LSTMusic. *

	Strongly Disagree	Disagree	Neither Disagree or Agree	Agree	Strongly Agree	Don't know
LSTMusic fulfils its purpose as a experimental tool / toy.	○	○	○	◉	○	○
LSTMusic is easy to use.	○	○	○	○	◉	○
LSTMusic is something you could see yourself using for entertainment.	○	○	◉	○	○	○
LSTMusic has given you a better idea of how generative AI could be used in music production	○	○	○	◉	○	○

Were there any parts of LSTMusic that you didn't understand, or which lacked clarity?

just what adventurous means maybe

Do you have any thoughts on LSTMusic's overall *visual theming*? (colour scheme, design language, use of screen space, button shapes, etc..)

The theme is great

Is there anything else you like or dislike about LSTMusic that hasn't been addressed in the previous sections?

If you have any final thoughts, opinions, feedback or suggestions about any part of LSTMusic, please feel free to write them below. This can be as detailed or vague as you like.

Optional Music Production Questions

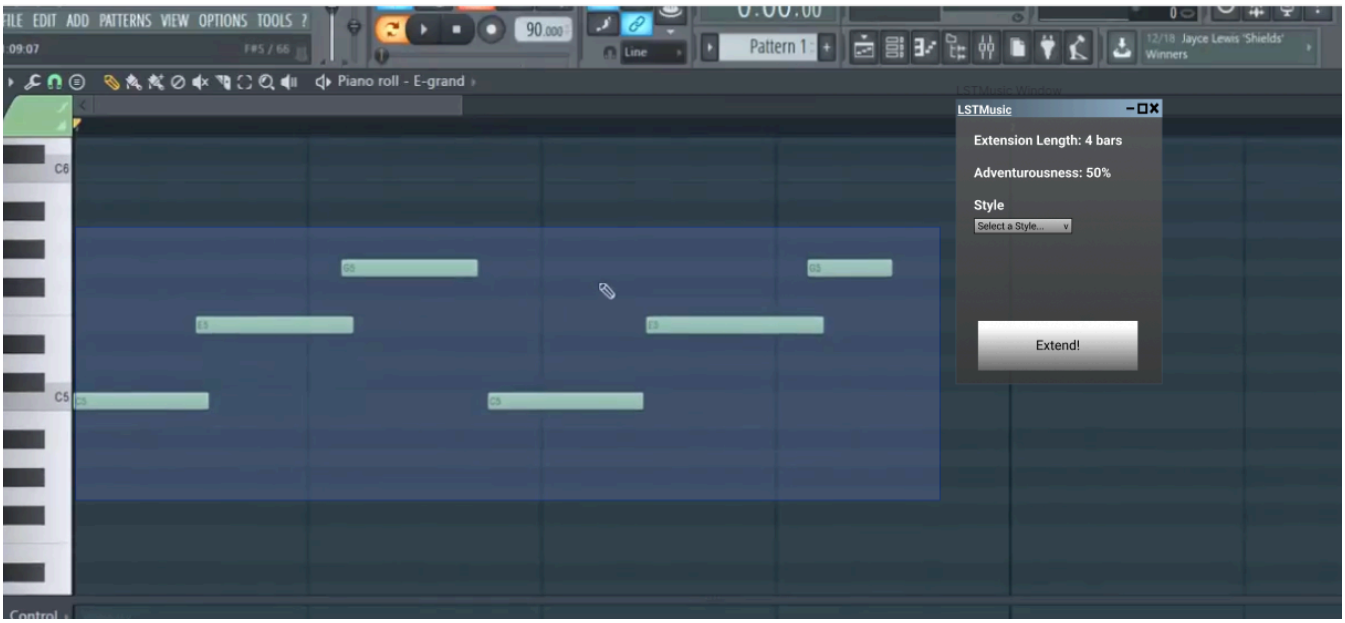
For those interested in music production, this form includes an optional section with additional questions for those familiar with music production software such as FL Studio, Ableton Live, Logic Pro, and more.

Do you use music production software? *

- ○ I use music production software.
- ◉ I do not use music production software.

Additional Questions: Using LSTMusic in music production

Part of this project was dedicated to researching and testing if LSTMusic could be made into a plugin for DAWs, to allow for on-the-fly melody extension during production. Some rough concept art of this can be seen below.



Would you ever consider using AI as part of the music production process? *

- Yes
- No

Have you ever used AI as part of the music production process? *

- Yes
- No

If Yes, in what way?

Would you be interested in using LSTMusic as part of your music production process?

- ○ Yes
- ○ No

Are there any changes or improvements to LSTMusic which would make using it as a plugin attractive?

Study Debriefing

The questionnaire section of the study is now complete.

The main aim of this experiment was to investigate the suitability of LSTMusic by collecting user feedback. In particular, to collect feedback regarding the site's user interface, and the site's AI model, used to extend the melodies inputted into it.

My contact details have been included again at the bottom of this page as well as the contact details of the project's supervisor. Thank you very much for your help. If you have any further questions, please let me know.

Please click "Submit" to finish the study.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

LSTMusic - Evaluation

Hello!
Thank you for choosing to evaluate my final year project. Your assistance is massively appreciated, and will massively help towards the evaluation and reflection of my project.

Study Briefing

The aim of this study is to conduct a user evaluation to collect feedback and data on the web-application developed as part of my Final year Project, called LSTMusic.

LSTMusic is a Web-Based Music Generation Toy which aims to give people the chance to experiment and toy with generative AI, and how it can interact with music.

The main motivation behind this project is to allow people to do this experimentation and toying without the hassle of downloading large music production software packages, plugins, or dealing with unfriendly command-line interfaces.

This evaluation is needed to assess different parts of LSTMusic. like its user interface design, music generation 'musicality', and the general 'feel' of LSTMusic.
It's important that a wide range of people are used when assessing this project, so that any evaluation is not biased towards those with certain interests or experience.

This study consists of 3 main parts, where you will be asked to do the following:

- Answer some 'about you' multiple choice questions to gauge what sort of person you are.
- Use & toy around with LSTMusic for a short amount of time.
- Answer a series of multiple choice & written questions relating to your experience while using the web application.

All of your responses to this form will be collected after it is submitted.

If you have any questions, please feel free to ask me by contacting me through any of my contact information, found below.

You have the right to withdraw from this study at any time by simply exiting the form. The contents of this form will not be used unless you have submitted it via the 'submit' button on the final page.
Please be assured that any data collected in this form will be stored securely and anonymously.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

Please confirm that you have read the above text, and agree to take part in this evaluation study. *

- (●) I Agree to take part in this study (Continue)
- ( ) I do not Agree to take part in this study (Please Exit)

About you.

To start out, please answer the following questions about yourself.

These questions are important to gauge your familiarity & knowledge with some of the concepts relating to this project, such as music theory, web applications and AI.

How familiar are you with the following concepts? *

Please pick the option which describes your level familiarity/experience the most.

	I'm not familiar.	Beginner	Hobbyist	Student	Professional
Basic music theory (notes & scales)	( )	( )	( )	( )	(●)
Rhythm and timing in music	( )	( )	( )	( )	(●)
Creating computer software	( )	( )	( )	(●)	( )
How generative AI works	( )	( )	( )	(●)	( )

How often do you use the following technologies / tools? *

Please pick the option which describes your frequency of use the most.

	Never	Rarely (A few times a year)	Infrequently (Every few months)	Often (Once or more per month)	Frequently (Once or more per week)	Very Frequently (Almost daily / Daily)
Large Language Models (ChatGPT or similar)	○	○	○	○	○	◉
AI image generation software (Dall-E, Photoshop's generative expand tool)	○	○	◉	○	○	○
Music production software (FL Studio, Ableton, Logic)	○	◉	○	○	○	○
Instruments (playing piano, guitar, etc...)	○	○	○	○	◉	○

Testing LSTMusic.

Please now take some time to experiment and toy about with LSTMusic.

- Please spend at least 5 minutes experimenting and toying about with the site.
- You are free to generate as many melodies as you like, and can note down any thoughts or findings while toying.
- If something breaks, don't worry, note it down!
- After experimenting, please return to the survey and press "Next".

LSTMusic can be accessed by clicking HERE, or typing "https://Dents6679.github.io/LSTMusic" into your browser.

Once you've finished experimenting, press 'Next'.

Please note:
If you're accessing this from a university network, there is a high likelihood that access to LSTMusic processing server will be blocked by a firewall, resulting in a "the generation server is not available right now. Please try again later." message. Please use a VPN or complete this survey on a different network if this occurs.

If any other catastrophic issues occur, please message me via WhatsApp on +447860131409.

Thank you

Thank you for spending the time to experiment and toy with LSTMusic. The rest of this survey is intended for you to give feedback about the different parts of LSTMusic.

Questions: Generated Music.

These questions are only concerned about the music LSTMusic generated.

To what extent to do you agree or disagree with the following statements regarding the music LSTMusic generates. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't Know
The PITCH and SCALE of the generated notes felt like they were influenced by the melody you inputted.	○	○	◉	○	○	○
The LENGTH of the generated notes felt like they were influenced by the melody you inputted.	○	◉	○	○	○	○
The transition between your music and the generated music felt smooth.	○	○	◉	○	○	○
The length of the generated music felt long enough.	◉	○	○	○	○	○
The rhythm of the generated music complemented the input melody.	○	◉	○	○	○	○
The generated music felt too arpeggiated.	○	◉	○	○	○	○

Is there anything you like or dislike about the generated music?

Kind of dull, uncreative, seemingly random

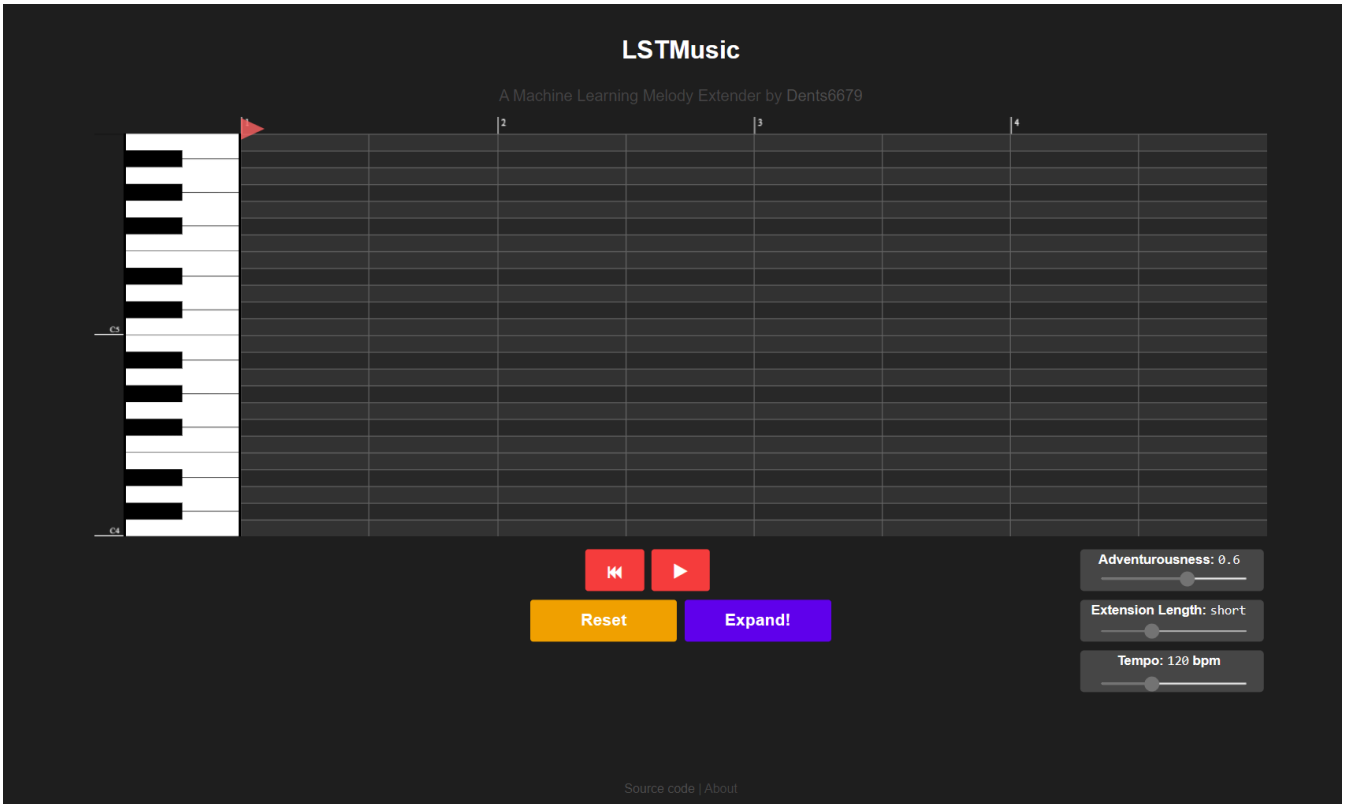
Are there any features, abilities or capabilities which you feel should be added or changed to improve the music LSTMusic produces?

Multiple notes played at once

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Melody input page / Homepage

The below questions are concerned about only the input page of LSTMusic, pictured below.


LSTMusic
A Machine Learning Melody Extender by Dents6679
1
2
3
4
C5
C4
Reset
Expand!
Adventurousness: 0.6
Extension Length: short
Tempo: 120 bpm
Source code | About

To what extent to do you agree or disagree with the following statements regarding the melody input page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The functionality of the different buttons are clear.	○	○	○	○	◉	○
There was a feeling of cause and effect when interacting with the page.	○	○	○	◉	○	○
The interface is visually appealing.	○	○	○	○	◉	○
The pitch range of the piano roll was large enough.	○	◉	○	○	○	○
The interface of the page matched expectations.	○	○	◉	○	○	○
The sound produced by the piano roll is suitable and appealing.	○	◉	○	○	○	○
Altering the inputted melody was	○	○	○	○	◉	○

intuitive.

intuitive.

Is there anything you like or dislike about the input page?

Are there any features, options or tools you feel should be added or changed to improve the input page?

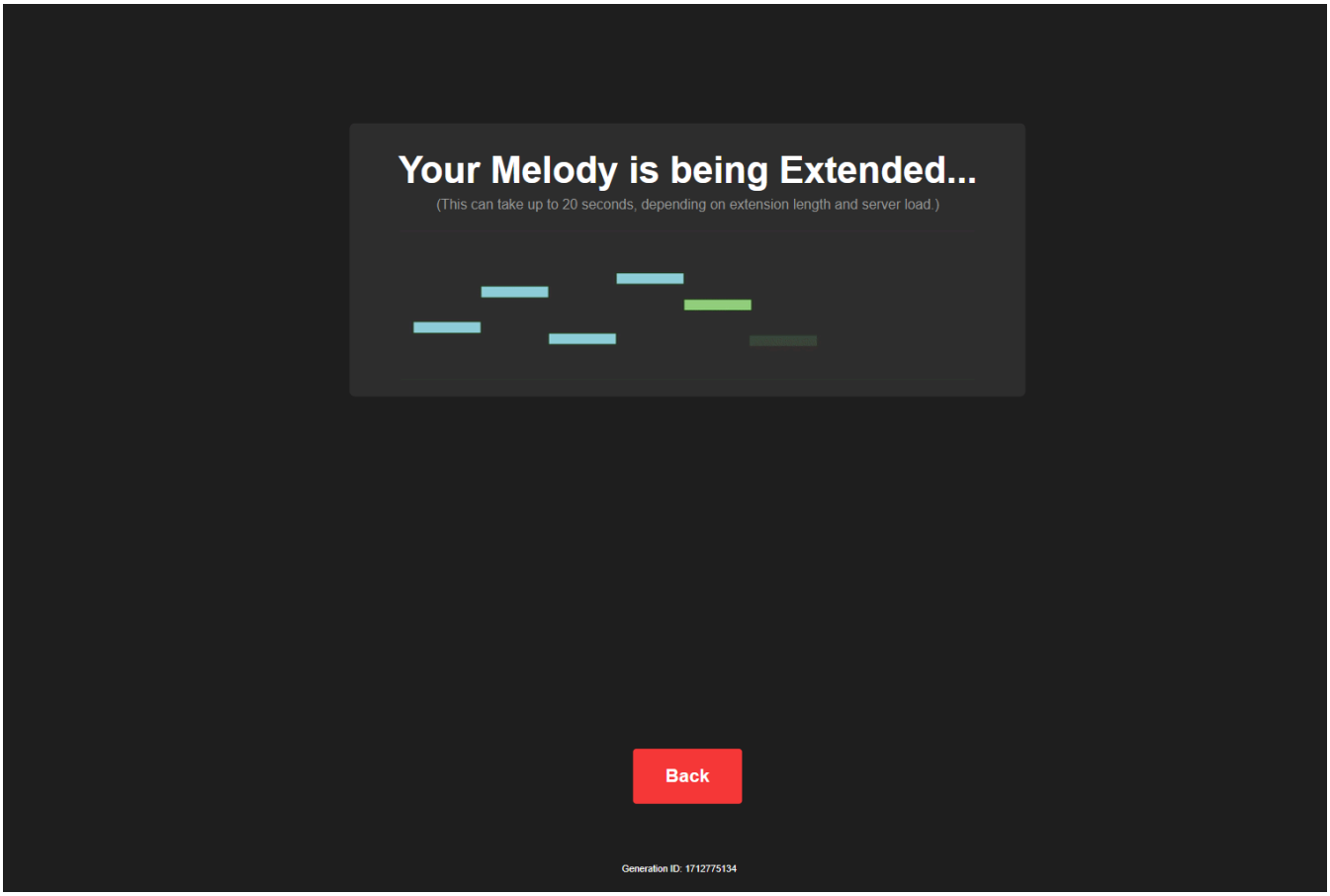
Did you encounter any issues or bugs with the input page? If you did, please specify them below.

Often "something went wrong" while generating

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Waiting page

The below questions are concerned about only the waiting page of LSTMusic, pictured below.


Your Melody is being Extended...
(This can take up to 20 seconds, depending on extension length and server load.)
Back
Generation ID: 1712775134

To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
There is a feeling of progression while waiting.	○	○	○	○	◉	○
The waiting page is visually appealing.	○	○	○	○	◉	○
Enough information was given about the expected waiting time.	○	◉	○	○	○	○
Waiting times were acceptable.	○	○	○	○	◉	○

Is there anything you like or dislike about the waiting page?

Something went wrong often

Are there any features, options or tools you feel should be added or changed to improve the waiting page?

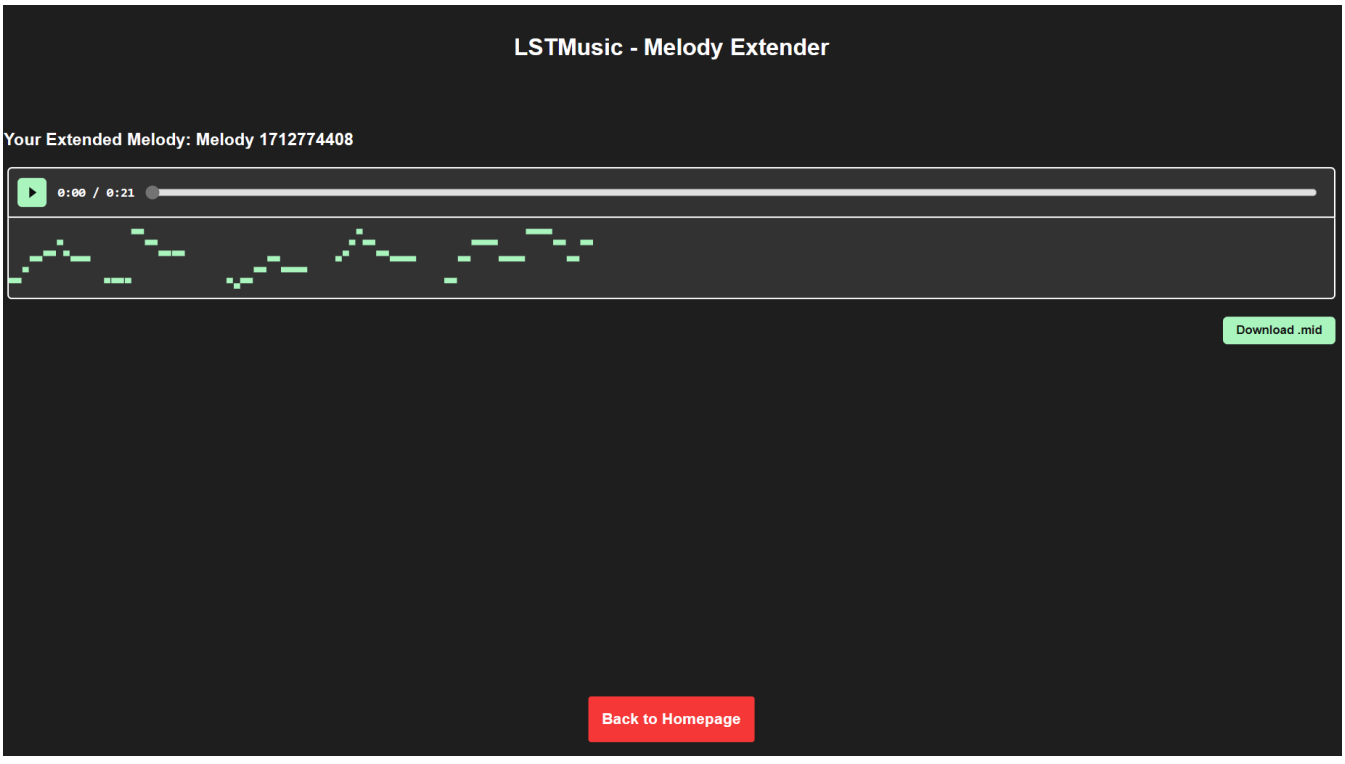
Did you encounter any issues or bugs with the waiting page? If you did, please specify them below.

Yes something went wrong

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Results page

The below questions are concerned about only the results page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The sound produced by the player is suitable and appealing.	○	◉	○	○	○	○
The page is visually appealing.	○	○	○	○	◉	○
The controls available on the player are sufficient.	○	○	○	○	◉	○
The song visualiser is fit for purpose.	○	○	○	○	◉	○

Is there anything you like or dislike about the results page?

Are there any features, options or tools you feel should be added or changed to improve the results page?

Did you encounter any issues or bugs with the results page? If you did, please specify them below.

Feel free to make any further comments on the results page which weren't addressed by the above questions.

Questions: LSTMusic as a whole

The following questions are concerned about the entirety of LSTMusic.

To what extent to do you agree or disagree with the following statements regarding the entirety of LSTMusic. *

	Strongly Disagree	Disagree	Neither Disagree or Agree	Agree	Strongly Agree	Don't know
LSTMusic fulfils its purpose as a experimental tool / toy.	○	○	◉	○	○	○
LSTMusic is easy to use.	○	○	○	○	◉	○
LSTMusic is something you could see yourself using for entertainment.	○	○	○	◉	○	○
LSTMusic has given you a better idea of how generative AI could be used in music production	◉	○	○	○	○	○

Were there any parts of LSTMusic that you didn't understand, or which lacked clarity?

Do you have any thoughts on LSTMusic's overall *visual theming*? (colour scheme, design language, use of screen space, button shapes, etc..)

Great

Is there anything else you like or dislike about LSTMusic that hasn't been addressed in the previous sections?

If you have any final thoughts, opinions, feedback or suggestions about any part of LSTMusic, please feel free to write them below. This can be as detailed or vague as you like.

Overall a nice idea that could be expanded on with better AI models

Optional Music Production Questions

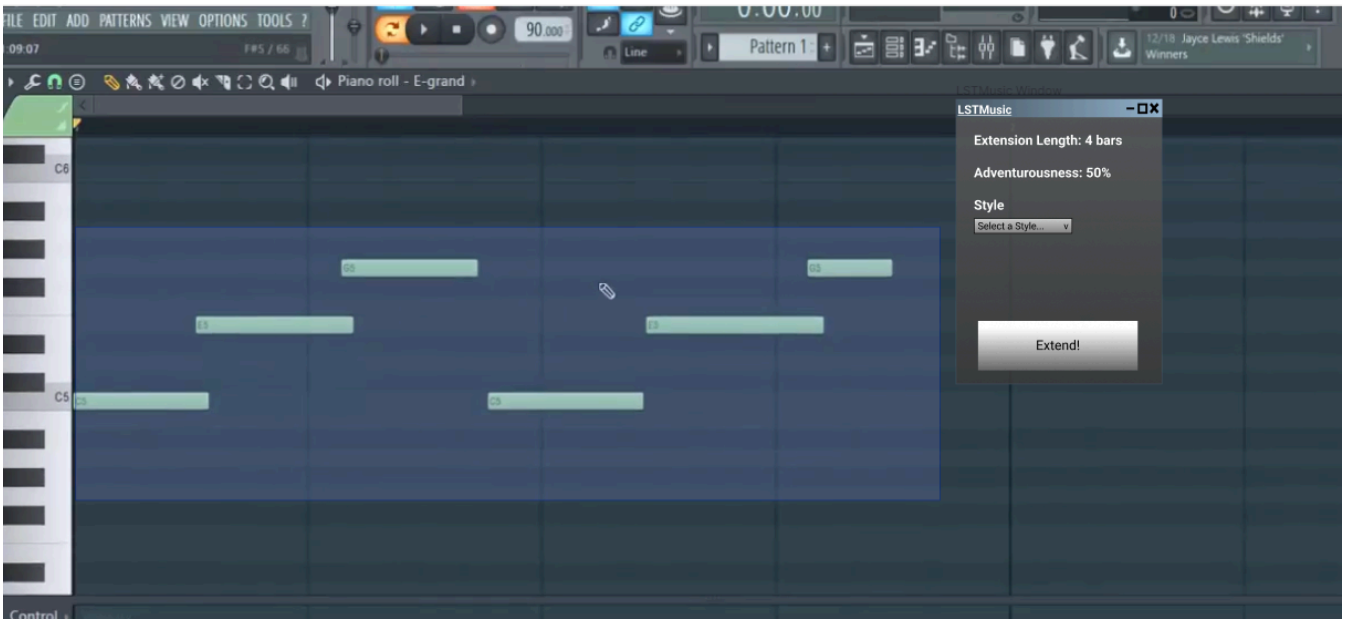
For those interested in music production, this form includes an optional section with additional questions for those familiar with music production software such as FL Studio, Ableton Live, Logic Pro, and more.

Do you use music production software? *

- (●) I use music production software.
- ( ) I do not use music production software.

Additional Questions: Using LSTMusic in music production

Part of this project was dedicated to researching and testing if LSTMusic could be made into a plugin for DAWs, to allow for on-the-fly melody extension during production. Some rough concept art of this can be seen below.



Would you ever consider using AI as part of the music production process? *

- (●) Yes
- ( ) No

Have you ever used AI as part of the music production process? *

- (●) Yes
- ( ) No

If Yes, in what way?

Dissertation, magenta music maker

Would you be interested in using LSTMusic as part of your music production process?

- ○ Yes
- ◉ No

Are there any changes or improvements to LSTMusic which would make using it as a plugin attractive?

Better AI model and multi instrumental

Study Debriefing

The questionnaire section of the study is now complete.

The main aim of this experiment was to investigate the suitability of LSTMusic by collecting user feedback. In particular, to collect feedback regarding the site's user interface, and the site's AI model, used to extend the melodies inputted into it.

My contact details have been included again at the bottom of this page as well as the contact details of the project's supervisor. Thank you very much for your help. If you have any further questions, please let me know.

Please click "Submit" to finish the study.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

LSTMusic - Evaluation

Hello!
Thank you for choosing to evaluate my final year project. Your assistance is massively appreciated, and will massively help towards the evaluation and reflection of my project.

Study Briefing

The aim of this study is to conduct a user evaluation to collect feedback and data on the web-application developed as part of my Final year Project, called LSTMusic.

LSTMusic is a Web-Based Music Generation Toy which aims to give people the chance to experiment and toy with generative AI, and how it can interact with music.

The main motivation behind this project is to allow people to do this experimentation and toying without the hassle of downloading large music production software packages, plugins, or dealing with unfriendly command-line interfaces.

This evaluation is needed to assess different parts of LSTMusic. like its user interface design, music generation 'musicality', and the general 'feel' of LSTMusic.
It's important that a wide range of people are used when assessing this project, so that any evaluation is not biased towards those with certain interests or experience.

This study consists of 3 main parts, where you will be asked to do the following:

- Answer some 'about you' multiple choice questions to gauge what sort of person you are.
- Use & toy around with LSTMusic for a short amount of time.
- Answer a series of multiple choice & written questions relating to your experience while using the web application.

All of your responses to this form will be collected after it is submitted.

If you have any questions, please feel free to ask me by contacting me through any of my contact information, found below.

You have the right to withdraw from this study at any time by simply exiting the form. The contents of this form will not be used unless you have submitted it via the 'submit' button on the final page.
Please be assured that any data collected in this form will be stored securely and anonymously.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

Please confirm that you have read the above text, and agree to take part in this evaluation study. *

- (●) I Agree to take part in this study (Continue)
- ( ) I do not Agree to take part in this study (Please Exit)

About you.

To start out, please answer the following questions about yourself.

These questions are important to gauge your familiarity & knowledge with some of the concepts relating to this project, such as music theory, web applications and AI.

How familiar are you with the following concepts? *

Please pick the option which describes your level familiarity/experience the most.

	I'm not familiar.	Beginner	Hobbyist	Student	Professional
Basic music theory (notes & scales)	( )	( )	( )	( )	(●)
Rhythm and timing in music	( )	( )	( )	( )	(●)
Creating computer software	(●)	( )	( )	( )	( )
How generative AI works	( )	( )	(●)	( )	( )

How often do you use the following technologies / tools? *

Please pick the option which describes your frequency of use the most.

	Never	Rarely (A few times a year)	Infrequently (Every few months)	Often (Once or more per month)	Frequently (Once or more per week)	Very Frequently (Almost daily / Daily)
Large Language Models (ChatGPT or similar)	◉	○	○	○	○	○
AI image generation software (Dall-E, Photoshop's generative expand tool)	◉	○	○	○	○	○
Music production software (FL Studio, Ableton, Logic)	○	○	○	○	○	◉
Instruments (playing piano, guitar, etc...)	○	○	○	○	◉	○

Testing LSTMusic.

Please now take some time to experiment and toy about with LSTMusic.

- Please spend at least 5 minutes experimenting and toying about with the site.
- You are free to generate as many melodies as you like, and can note down any thoughts or findings while toying.
- If something breaks, don't worry, note it down!
- After experimenting, please return to the survey and press "Next".

LSTMusic can be accessed by clicking HERE, or typing "https://Dents6679.github.io/LSTMusic" into your browser.

Once you've finished experimenting, press 'Next'.

Please note:
If you're accessing this from a university network, there is a high likelihood that access to LSTMusic processing server will be blocked by a firewall, resulting in a "the generation server is not available right now. Please try again later." message. Please use a VPN or complete this survey on a different network if this occurs.

If any other catastrophic issues occur, please message me via WhatsApp on +447860131409.

Thank you

Thank you for spending the time to experiment and toy with LSTMusic. The rest of this survey is intended for you to give feedback about the different parts of LSTMusic.

Questions: Generated Music.

These questions are only concerned about the music LSTMusic generated.

To what extent to do you agree or disagree with the following statements regarding the music LSTMusic generates. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't Know
The PITCH and SCALE of the generated notes felt like they were influenced by the melody you inputted.	○	○	○	◉	○	○
The LENGTH of the generated notes felt like they were influenced by the melody you inputted.	○	○	◉	○	○	○
The transition between your music and the generated music felt smooth.	○	○	○	◉	○	○
The length of the generated music felt long enough.	○	◉	○	○	○	○
The rhythm of the generated music complemented the input melody.	○	◉	○	○	○	○
The generated music felt too arpeggiated.	○	◉	○	○	○	○

Is there anything you like or dislike about the generated music?

I tried giving it a few different genres of music, and with all of them the tool mostly failed to stay within that style. It definitely seems to naturally sway to classical music style writing. However, with genres aside, the melodies it generated were complex and cohesive for the most part. More complex than what I tried to give it as I wanted to keep it simple to test.

All in all, the music it generated itself was impressive.

Are there any features, abilities or capabilities which you feel should be added or changed to improve the music LSTMusic produces?

Note velocities - different volumes (or a more muffled tone on "quieter" notes) contribute massively to the feeling of the music, and thus could probably easily be learned by the AI.

Length of phrases, how this is written on UI - the length setting being written as a decimal number isn't clear for how long I want the phrase to be. Each time I generated I couldnt get a solid extra bar or 2 bars. This causes 2 problems, the first being that the AI won't finish the phrase in time with the song that the user is using the AI to improve or complete. Music phrases effectively need a "resolution" at the end of each phrase, and those phrases will be 2 or 4 bars long 99% of the time. The other which doesn't need to be addressed straight away,
is the functionality of using this as an actual tool. If the UI isn't clear on how long the phrase is it just won't be applicable, and if the AI itself can't determine when the phrase should end or how to resolve the melody as such, it is again, not applicable.

Piano roll - Of course this is just a prototype, but being able to write chords would be a big improvement. From what I think about adaptive AI, this would also be a big help to getting the tool to understand what the user is trying to create. Also, being able to write in 8th notes instead of 4th notes, and eventually 16th notes when you've developed it further.

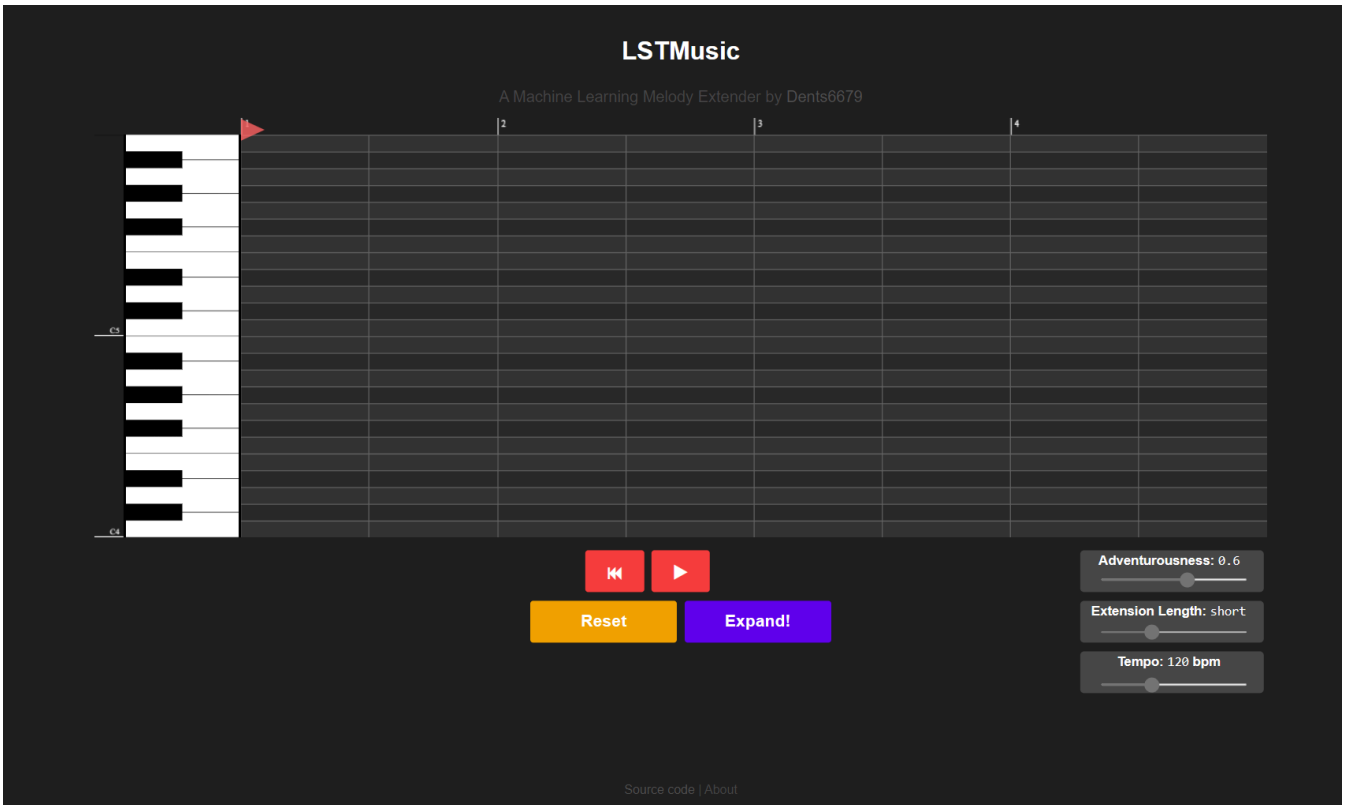
Feel free to make any further comments on the generated music which weren't addressed by the above questions.

My overall conclusion is that the tool needs a lot of work. As much as the melodies it generated were impressive on their own, it wasn't cohesive with what I had written. I would assume this is because it isn't paying attention to "genres" as such, but from my perspective something like adaptive AI shouldn't need to worry about that term as it's effectively a human construct. You do hear similar melody styles and patterns within genres we know, but each of those melodies on their own are cohesive and have recognisable patterns irrespective of the genre they are tied to.

All of this tied with the lack of clear phrasing makes it at best a fun novelty tool at the moment, but clearly shows potential and is probably not too far off from being able to do all of this.

Questions: Melody input page / Homepage

The below questions are concerned about only the input page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the melody input page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The functionality of the different buttons are clear.	○	○	○	◉	○	○
There was a feeling of cause and effect when interacting with the page.	○	○	○	◉	○	○
The interface is visually appealing.	○	○	◉	○	○	○
The pitch range of the piano roll was large enough.	○	◉	○	○	○	○
The interface of the page matched expectations.	○	○	○	◉	○	○
The sound produced by the piano roll is suitable and appealing.	○	○	◉	○	○	○
Altering the inputted melody was	○	○	◉	○	○	○

intuitive.

intuitive.

Is there anything you like or dislike about the input page?

Overall it's what I would expect from a piano roll. However it felt clunky to use at times, and could be made to look more modern and slick.

Are there any features, options or tools you feel should be added or changed to improve the input page?

In all DAWs there are many small things that add up to a very smooth workflow. Each thing will only save a few seconds and most are things I wouldn't expect to see on something like this.

Scissors tool - being able to slice a note in half when altering your melody is very useful because it leaves you with 2 notes instead of having to click to make another after shrinking one.

Placing the playhead - With this prototype you can currently only play from the start or where you paused from. If the user is tweaking the last few notes of the bar they have to wait for it to loop through again to hear it.

Deleting notes - I'm not sure if it was just bugged but i couldn't really find a way to delete notes. A quick hotkey for this is right click, or backspace of course.

Spacebar to pause and start - again, not sure if it's bugged but having to click the pause and play button makes it feel antiquated.

Did you encounter any issues or bugs with the input page? If you did, please specify them below.

When I paused half way through the track, pressing play would cause the notes to completely glitch. It sounded like it was playing at 2 different points in the song. This meant I had to let it play all the way to the end every time if I wanted to go back to the start.

Some notes I already had placed when clicked on didnt play the note back, and some did.

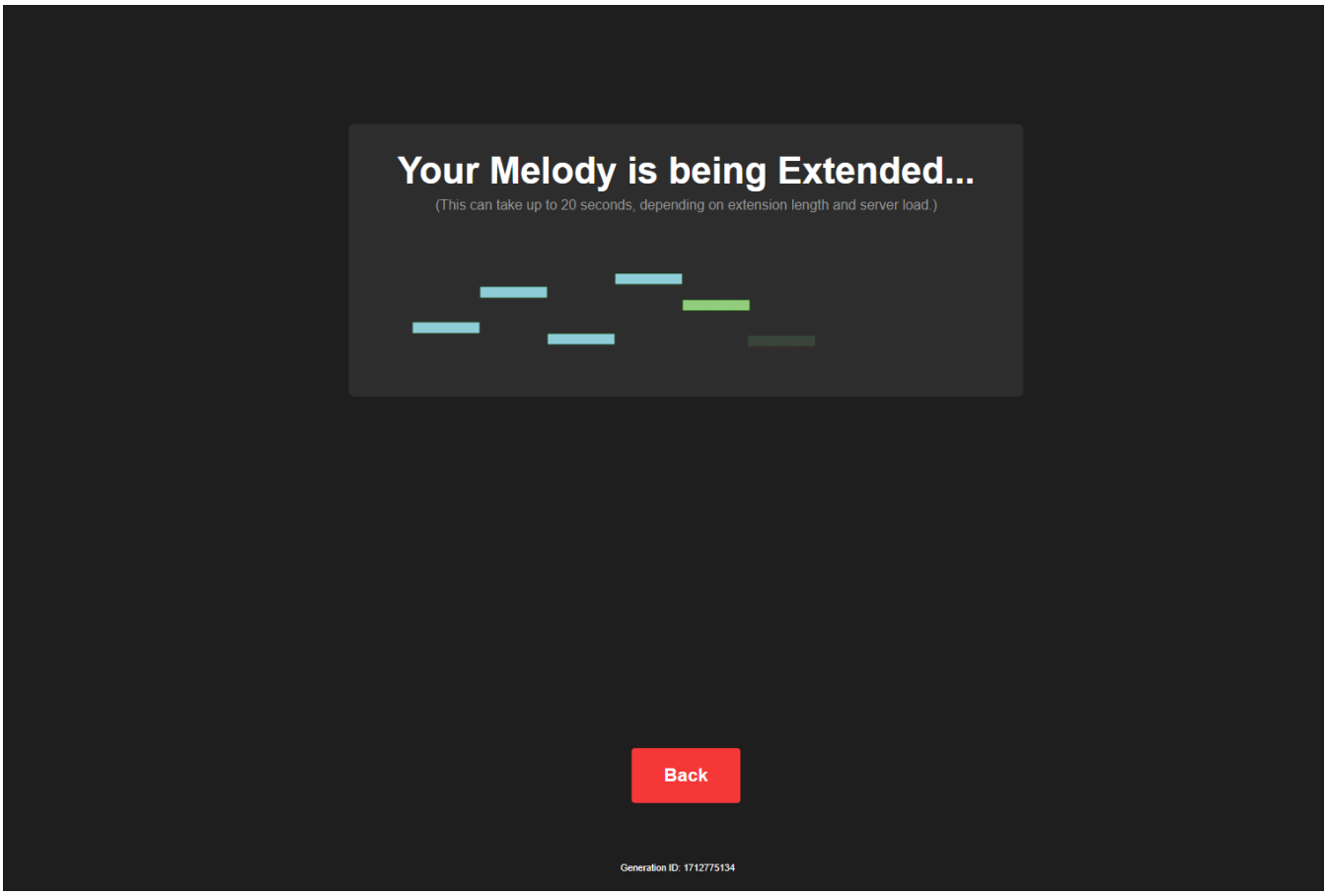
The first note on the piano roll doesn't play when I place it.

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

The sound of the notes on the piano roll being different to the notes played back didn't make sense to me. But this will probably be irrelevant in the future once you have multiple patches available for the AI to use.

Questions: Waiting page

The below questions are concerned about only the waiting page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
There is a feeling of progression while waiting.	○	○	○	○	◉	○
The waiting page is visually appealing.	○	○	○	○	◉	○
Enough information was given about the expected waiting time.	○	○	○	○	◉	○
Waiting times were acceptable.	○	○	○	○	◉	○

Is there anything you like or dislike about the waiting page?

Could look more slick and animated, but this is just a prototype.

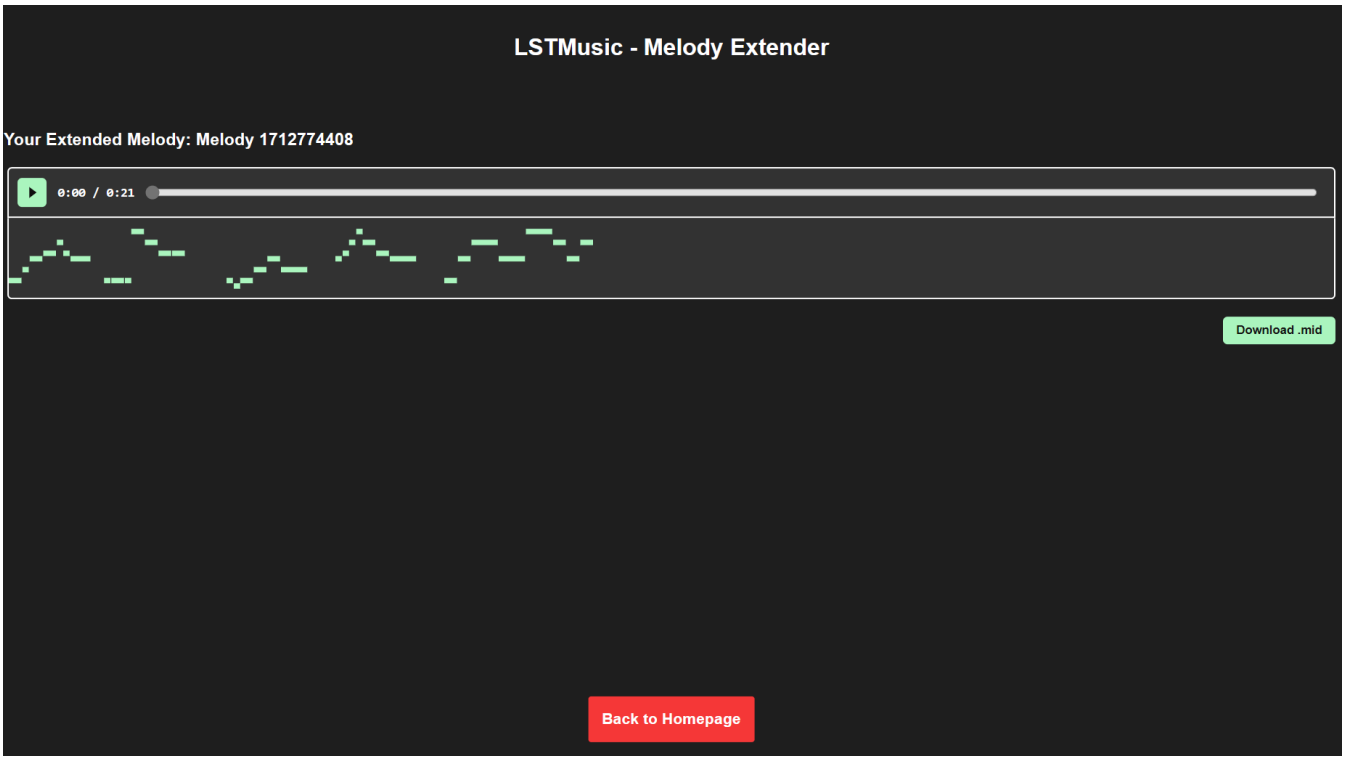
Are there any features, options or tools you feel should be added or changed to improve the waiting page?

Did you encounter any issues or bugs with the waiting page? If you did, please specify them below.

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Results page

The below questions are concerned about only the results page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The sound produced by the player is suitable and appealing.	○	○	○	○	◉	○
The page is visually appealing.	○	○	◉	○	○	○
The controls available on the player are sufficient.	○	○	◉	○	○	○
The song visualiser is fit for purpose.	○	○	○	○	◉	○

Is there anything you like or dislike about the results page?

The design should be much more focused towards a DAW. It needs bar lines, and should display bars/beats at the top. The time should also have decimal places, and would better be displayed in stopwatch format.

Are there any features, options or tools you feel should be added or changed to improve the results page?

Something that I feel makes these tools appear generally more professional when actually being sold as a plugin is basically having a lot of functional eye-candy on the screen. Counters like I mentioned previously, and just things written about the music around.

Key/scale - Have the AI display the scale at the top. This also lets the user know straight away if the tool has failed and he needs to go back.

BPM again - just to make it clear incase they forget the specific number and need to write it elsewhere.

Playhead - having the marker here and being able to place it will also speed up playback.

Being able to tweak the AI generation from here - This is of course somewhat obsolete because you can just download the midi, and further down the line when you could click and drag the midi straight into the piano roll of your actual DAW. However, I still think it is useful to be able to edit the AI melody on here.

Did you encounter any issues or bugs with the results page? If you did, please specify them below.

Feel free to make any further comments on the results page which weren't addressed by the above questions.

Questions: LSTMusic as a whole

The following questions are concerned about the entirety of LSTMusic.

To what extent to do you agree or disagree with the following statements regarding the entirety of LSTMusic. *

	Strongly Disagree	Disagree	Neither Disagree or Agree	Agree	Strongly Agree	Don't know
LSTMusic fulfils its purpose as a experimental tool / toy.	○	○	○	○	◉	○
LSTMusic is easy to use.	○	○	○	◉	○	○
LSTMusic is something you could see yourself using for entertainment.	○	○	○	◉	○	○
LSTMusic has given you a better idea of how generative AI could be used in music production	○	○	◉	○	○	○

Were there any parts of LSTMusic that you didn't understand, or which lacked clarity?

Do you have any thoughts on LSTMusic's overall *visual theming*? (colour scheme, design language, use of screen space, button shapes, etc..)

It does its job from a functional point of view, but could be made more modern.

Is there anything else you like or dislike about LSTMusic that hasn't been addressed in the previous sections?

If you have any final thoughts, opinions, feedback or suggestions about any part of LSTMusic, please feel free to write them below. This can be as detailed or vague as you like.

Optional Music Production Questions

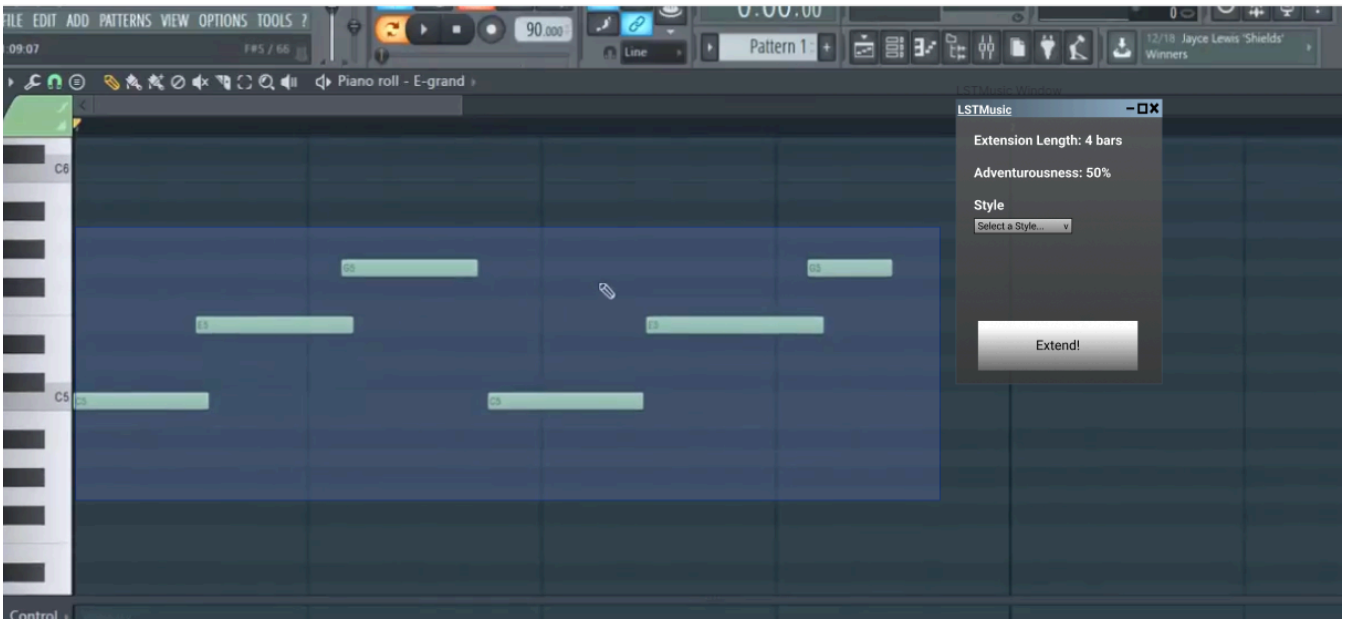
For those interested in music production, this form includes an optional section with additional questions for those familiar with music production software such as FL Studio, Ableton Live, Logic Pro, and more.

Do you use music production software? *

- (●) I use music production software.
- ( ) I do not use music production software.

Additional Questions: Using LSTMusic in music production

Part of this project was dedicated to researching and testing if LSTMusic could be made into a plugin for DAWs, to allow for on-the-fly melody extension during production. Some rough concept art of this can be seen below.



Would you ever consider using AI as part of the music production process? *

- (●) Yes
- ( ) No

Have you ever used AI as part of the music production process? *

- ( ) Yes
- (●) No

If Yes, in what way?

_____

Would you be interested in using LSTMusic as part of your music production process?

- ○ Yes
- ◉ No

Are there any changes or improvements to LSTMusic which would make using it as a plugin attractive?

To be used as an actual plugin it has a long way to go. It would require more than what I've talked about, and the fine tuning and refining of all of that.

Study Debriefing

The questionnaire section of the study is now complete.

The main aim of this experiment was to investigate the suitability of LSTMusic by collecting user feedback. In particular, to collect feedback regarding the site's user interface, and the site's AI model, used to extend the melodies inputted into it.

My contact details have been included again at the bottom of this page as well as the contact details of the project's supervisor. Thank you very much for your help. If you have any further questions, please let me know.

Please click "Submit" to finish the study.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

LSTMusic - Evaluation

Hello!
Thank you for choosing to evaluate my final year project. Your assistance is massively appreciated, and will massively help towards the evaluation and reflection of my project.

Study Briefing

The aim of this study is to conduct a user evaluation to collect feedback and data on the web-application developed as part of my Final year Project, called LSTMusic.

LSTMusic is a Web-Based Music Generation Toy which aims to give people the chance to experiment and toy with generative AI, and how it can interact with music.

The main motivation behind this project is to allow people to do this experimentation and toying without the hassle of downloading large music production software packages, plugins, or dealing with unfriendly command-line interfaces.

This evaluation is needed to assess different parts of LSTMusic. like its user interface design, music generation 'musicality', and the general 'feel' of LSTMusic.
It's important that a wide range of people are used when assessing this project, so that any evaluation is not biased towards those with certain interests or experience.

This study consists of 3 main parts, where you will be asked to do the following:

- Answer some 'about you' multiple choice questions to gauge what sort of person you are.
- Use & toy around with LSTMusic for a short amount of time.
- Answer a series of multiple choice & written questions relating to your experience while using the web application.

All of your responses to this form will be collected after it is submitted.

If you have any questions, please feel free to ask me by contacting me through any of my contact information, found below.

You have the right to withdraw from this study at any time by simply exiting the form. The contents of this form will not be used unless you have submitted it via the 'submit' button on the final page.
Please be assured that any data collected in this form will be stored securely and anonymously.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS

Please confirm that you have read the above text, and agree to take part in this evaluation study. *

- (●) I Agree to take part in this study (Continue)
- ( ) I do not Agree to take part in this study (Please Exit)

About you.

To start out, please answer the following questions about yourself.

These questions are important to gauge your familiarity & knowledge with some of the concepts relating to this project, such as music theory, web applications and AI.

How familiar are you with the following concepts? *

Please pick the option which describes your level familiarity/experience the most.

	I'm not familiar.	Beginner	Hobbyist	Student	Professional
Basic music theory (notes & scales)	( )	(●)	( )	( )	( )
Rhythm and timing in music	( )	( )	(●)	( )	( )
Creating computer software	(●)	( )	( )	( )	( )
How generative AI works	( )	(●)	( )	( )	( )

How often do you use the following technologies / tools? *

Please pick the option which describes your frequency of use the most.

	Never	Rarely (A few times a year)	Infrequently (Every few months)	Often (Once or more per month)	Frequently (Once or more per week)	Very Frequently (Almost daily / Daily)
Large Language Models (ChatGPT or similar)	○	○	◉	○	○	○
AI image generation software (Dall-E, Photoshop's generative expand tool)	○	○	○	○	◉	○
Music production software (FL Studio, Ableton, Logic)	○	◉	○	○	○	○
Instruments (playing piano, guitar, etc...)	○	◉	○	○	○	○

Testing LSTMusic.

Please now take some time to experiment and toy about with LSTMusic.

- Please spend at least 5 minutes experimenting and toying about with the site.
- You are free to generate as many melodies as you like, and can note down any thoughts or findings while toying.
- If something breaks, don't worry, note it down!
- After experimenting, please return to the survey and press "Next".

LSTMusic can be accessed by clicking HERE, or typing "https://Dents6679.github.io/LSTMusic" into your browser.

Once you've finished experimenting, press 'Next'.

Please note:
If you're accessing this from a university network, there is a high likelihood that access to LSTMusic processing server will be blocked by a firewall, resulting in a "the generation server is not available right now. Please try again later." message. Please use a VPN or complete this survey on a different network if this occurs.

If any other catastrophic issues occur, please message me via WhatsApp on +447860131409.

Thank you

Thank you for spending the time to experiment and toy with LSTMusic. The rest of this survey is intended for you to give feedback about the different parts of LSTMusic.

Questions: Generated Music.

These questions are only concerned about the music LSTMusic generated.

To what extent to do you agree or disagree with the following statements regarding the music LSTMusic generates. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't Know
The PITCH and SCALE of the generated notes felt like they were influenced by the melody you inputted.	○	○	○	◉	○	○
The LENGTH of the generated notes felt like they were influenced by the melody you inputted.	○	○	○	◉	○	○
The transition between your music and the generated music felt smooth.	○	○	○	◉	○	○
The length of the generated music felt long enough.	○	○	◉	○	○	○
The rhythm of the generated music complemented the input melody.	○	○	○	◉	○	○
The generated music felt too arpeggiated.	○	○	○	○	○	◉

Is there anything you like or dislike about the generated music?

only complaint is sometimes the music outputted is very short

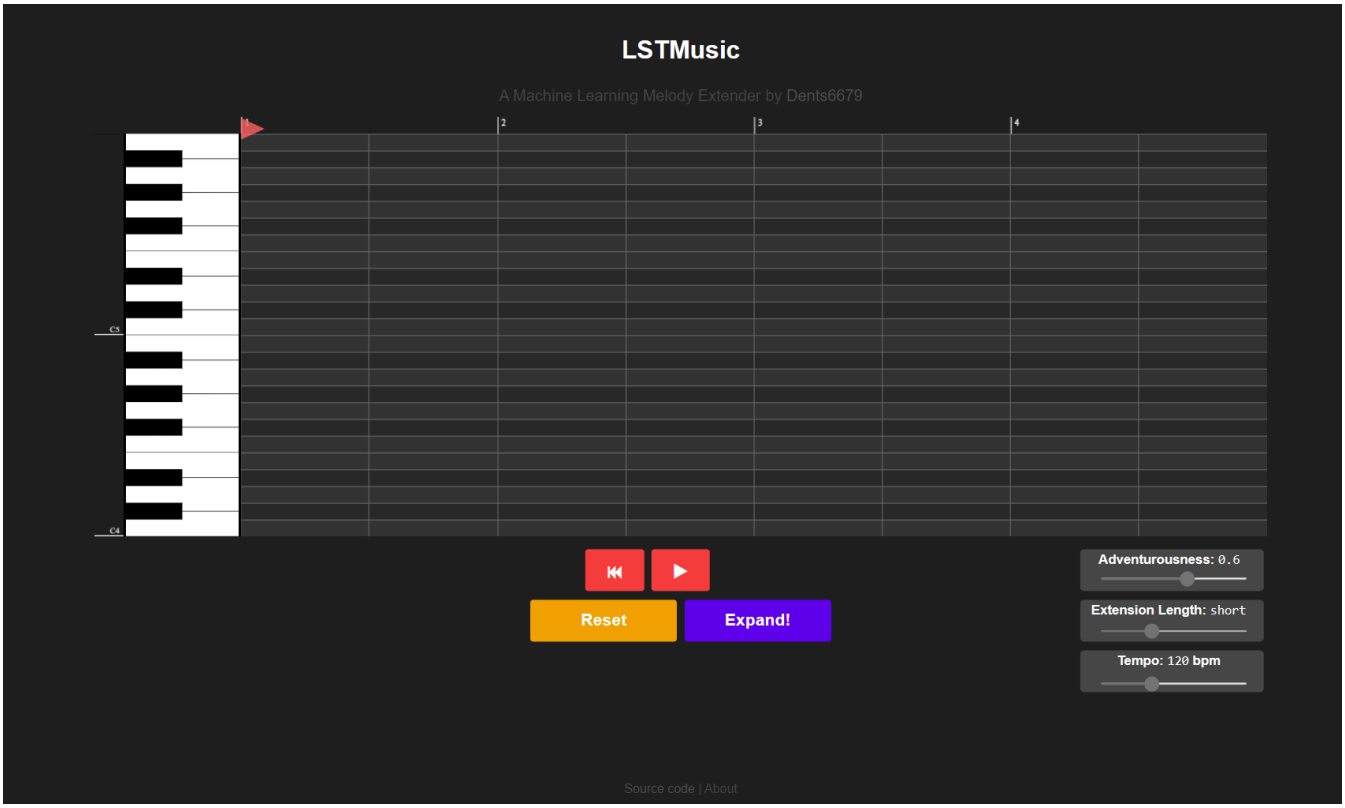
Are there any features, abilities or capabilities which you feel should be added or changed to improve the music LSTMusic produces?

maybe the ability to use different note types (like a guitar or vocaloid)

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Melody input page / Homepage

The below questions are concerned about only the input page of LSTMusic, pictured below.


LSTMusic
A Machine Learning Melody Extender by Dents6679
1
2
3
4
C5
C4
Reset
Expand!
Adventurousness: 0.6
Extension Length: short
Tempo: 120 bpm
Source code | About

To what extent to do you agree or disagree with the following statements regarding the melody input page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The functionality of the different buttons are clear.	○	○	○	○	◉	○
There was a feeling of cause and effect when interacting with the page.	○	○	○	◉	○	○
The interface is visually appealing.	○	○	○	◉	○	○
The pitch range of the piano roll was large enough.	○	○	◉	○	○	○
The interface of the page matched expectations.	○	○	○	◉	○	○
The sound produced by the piano roll is suitable and appealing.	○	○	○	◉	○	○
Altering the inputted melody was	○	○	○	◉	○	○

intuitive.

intuitive.

Is there anything you like or dislike about the input page?

I love the aesthetic of the page, only complaint is I would prefer it if the buttons were centre on the page but that is very picky of me!

Are there any features, options or tools you feel should be added or changed to improve the input page?

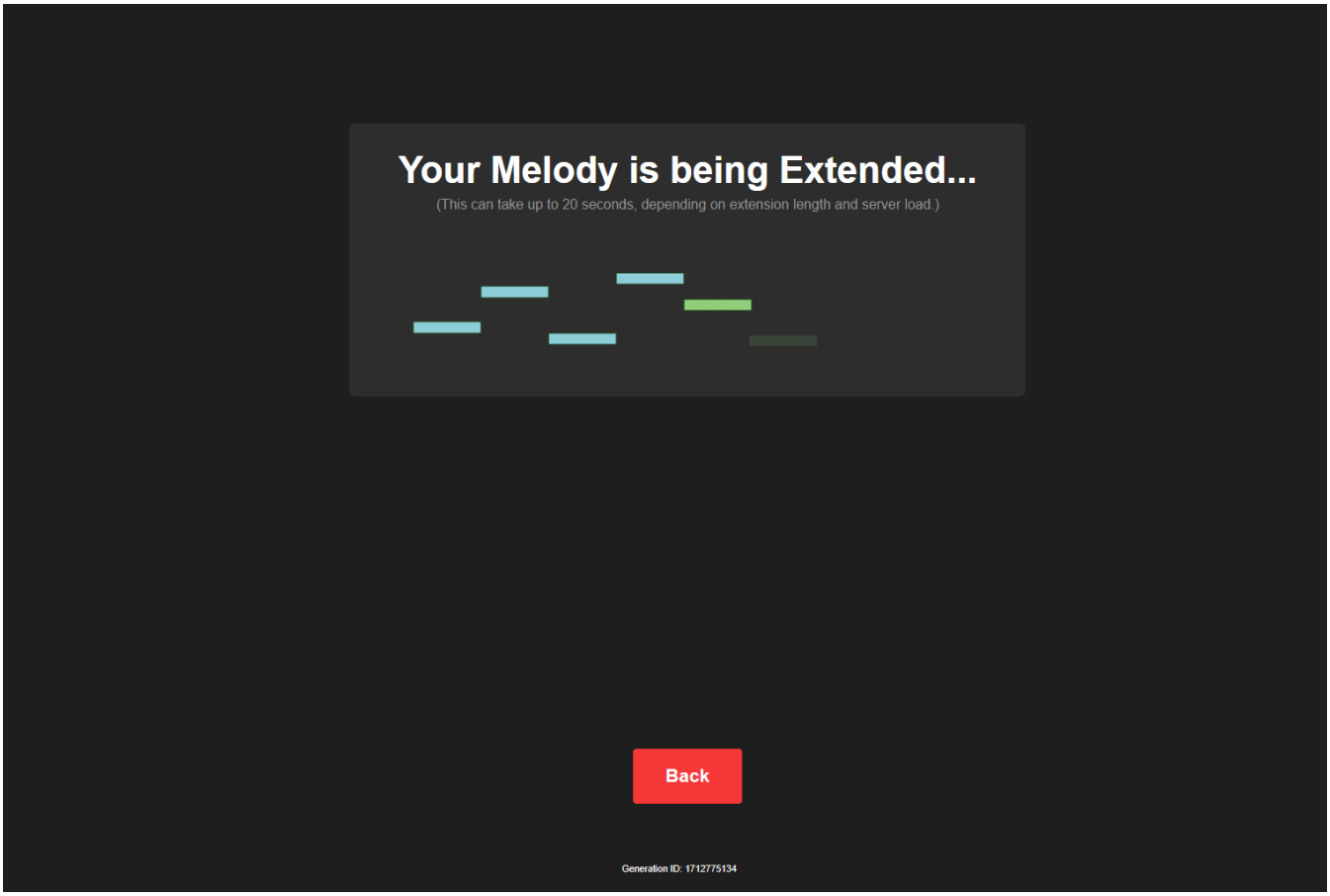
for what it is, I don't think so

Did you encounter any issues or bugs with the input page? If you did, please specify them below.

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Waiting page

The below questions are concerned about only the waiting page of LSTMusic, pictured below.


Your Melody is being Extended...
(This can take up to 20 seconds, depending on extension length and server load.)
Back
Generation ID: 1712775134

To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
There is a feeling of progression while waiting.	○	○	○	○	◉	○
The waiting page is visually appealing.	○	○	○	○	◉	○
Enough information was given about the expected waiting time.	○	○	○	◉	○	○
Waiting times were acceptable.	○	○	○	◉	○	○

Is there anything you like or dislike about the waiting page?

I love the animation!

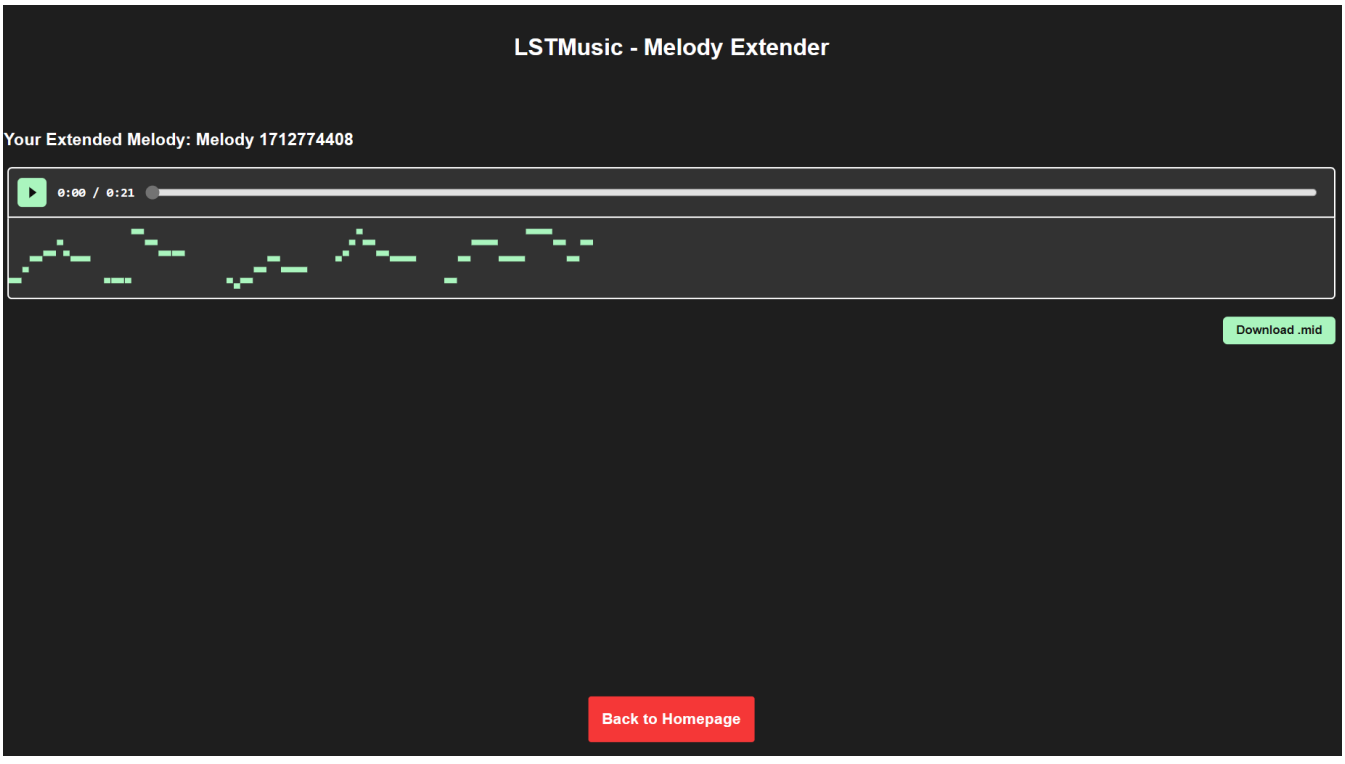
Are there any features, options or tools you feel should be added or changed to improve the waiting page?

Did you encounter any issues or bugs with the waiting page? If you did, please specify them below.

Feel free to make any further comments on the generated music which weren't addressed by the above questions.

Questions: Results page

The below questions are concerned about only the results page of LSTMusic, pictured below.



To what extent to do you agree or disagree with the following statements regarding the waiting page. *

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Don't know
The purpose of the page is clear.	○	○	○	○	◉	○
The sound produced by the player is suitable and appealing.	○	○	○	◉	○	○
The page is visually appealing.	○	○	○	◉	○	○
The controls available on the player are sufficient.	○	○	○	◉	○	○
The song visualiser is fit for purpose.	○	○	○	◉	○	○

Is there anything you like or dislike about the results page?

I love the aesthetic of the results page and I like how you can see the piano roll, only complaint is the timer only updates when a note is played and it would be great if it acted separately from the note roll (instead of for example jumping from 2 to 5 seconds)

Are there any features, options or tools you feel should be added or changed to improve the results page?

Did you encounter any issues or bugs with the results page? If you did, please specify them below.

no issues!

Feel free to make any further comments on the results page which weren't addressed by the above questions.

Questions: LSTMusic as a whole

The following questions are concerned about the entirety of LSTMusic.

To what extent to do you agree or disagree with the following statements regarding the entirety of LSTMusic. *

	Strongly Disagree	Disagree	Neither Disagree or Agree	Agree	Strongly Agree	Don't know
LSTMusic fulfils its purpose as a experimental tool / toy.	○	○	○	◉	○	○
LSTMusic is easy to use.	○	○	○	◉	○	○
LSTMusic is something you could see yourself using for entertainment.	○	○	○	◉	○	○
LSTMusic has given you a better idea of how generative AI could be used in music production	○	○	○	◉	○	○

Were there any parts of LSTMusic that you didn't understand, or which lacked clarity?

it wasn't very clear how to make notes longer but it made sense when i worked it out

Do you have any thoughts on LSTMusic's overall *visual theming*? (colour scheme, design language, use of screen space, button shapes, etc..)

it's visually pleasing!

Is there anything else you like or dislike about LSTMusic that hasn't been addressed in the previous sections?

If you have any final thoughts, opinions, feedback or suggestions about any part of LSTMusic, please feel free to write them below. This can be as detailed or vague as you like.

i'm so proud of you <3

Optional Music Production Questions

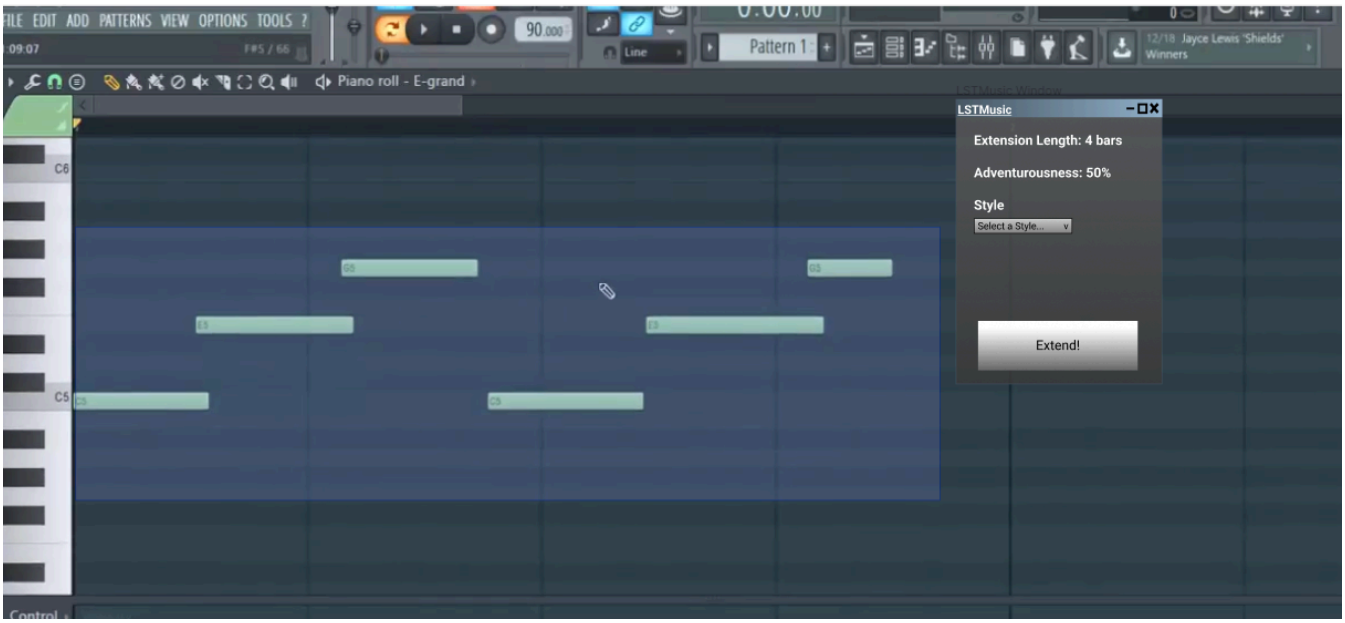
For those interested in music production, this form includes an optional section with additional questions for those familiar with music production software such as FL Studio, Ableton Live, Logic Pro, and more.

Do you use music production software? *

- ○ I use music production software.
- ◉ I do not use music production software.

Additional Questions: Using LSTMusic in music production

Part of this project was dedicated to researching and testing if LSTMusic could be made into a plugin for DAWs, to allow for on-the-fly melody extension during production. Some rough concept art of this can be seen below.



Would you ever consider using AI as part of the music production process? *

- Yes
- No

Have you ever used AI as part of the music production process? *

- Yes
- No

If Yes, in what way?

Would you be interested in using LSTMusic as part of your music production process?

- ○ Yes
- ○ No

Are there any changes or improvements to LSTMusic which would make using it as a plugin attractive?

Study Debriefing

The questionnaire section of the study is now complete.

The main aim of this experiment was to investigate the suitability of LSTMusic by collecting user feedback. In particular, to collect feedback regarding the site's user interface, and the site's AI model, used to extend the melodies inputted into it.

My contact details have been included again at the bottom of this page as well as the contact details of the project's supervisor. Thank you very much for your help. If you have any further questions, please let me know.

Please click "Submit" to finish the study.

Contact Details

Thomas Dent - Project Developer and Study Creator
Email: td336@sussex.ac.uk
Mobile / WhatsApp: +447860131409

Dr. Kingsley Sage - Project Supervisor
Email: k.h.sage@sussex.ac.uk
CHICHESTER 1 CI302
ENGINF SCHOOL - INFORMATICS