Audio to Transcribed MIDI
Interim Report

TU856
BSc in Computer Science

Jason Garcia
C22516126

Sean O'Leary

School of Computer Science
Technological University, Dublin

23/11/2025

Abstract

This project focuses on creating a website that allows users to input audio files to get a piano roll with audio playback of a MIDI recreation. The purpose of this system is to allow hobbyist or untrained musicians to create their own transcriptions by using this website to automate the process. This system uses technologies and concepts in the area of Automatic Music Transcription to allow for instrument-agnostic, polyphonic input detection. Plans for creating audio detection and MIDI creation by using Spotify's Basic Pitch model for pitch inference, and processing that output into music notes to be converted to MIDI are detailed.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Jason Garcia_____

Jason Garcia

23/11/2025

Acknowledgements

I would like to thank my supervisor, Seán O’ Leary for his guidance throughout this project.

1. Introduction

1.1 Project Background

Music transcription is the act of listening to a piece of music and writing it down into a readable format. It helps people learn songs and gives them the ability to play it themselves. Music transcription is not a skill that is open to beginner musicians. It takes a trained ear to be able to accurately deduce what notes make up a certain melody. In this aspect, music can be considered inaccessible. The problem comes when there are no transcriptions available to the public. Musicians will often run into scenarios where they are not able to find any transcripts for a song they want to play.

There are several reasons why it is so difficult to recognize correct music notes, such as chords. The challenge with identifying chords is that when the notes are being played simultaneously, they blend into a single combined sound, which makes it troublesome to recognize each individual note.

Because music notation is so hard, there is a need to develop technologies that can help us with transcribing songs. This demand has pushed forward research into automated approaches. There have recently been technological advances in machine learning, automatic chord recognition, music information retrieval, and automatic music transcription.

With the rise of these technologies, the tools are now in place to create a system that gives beginner musicians a tool to mimic the musical understanding of an experienced musician with trained ears. This project will solve the issue of musical inaccessibility.

1.2 Project Description

This project will be a website that allows users to upload audio files, and returns a transcription with audio playback of MIDI. It would be in the style of a piano roll, where keys display their note names. The system will be able to handle any type of melodic instrument. The logic will all happen offline. This would be done to maximize accuracy and performance while also getting rid of the complexity added from live audio transcription. To assist users in understanding the uploaded song, chord names will be displayed whenever one is played. Users will also have the ability to adjust playback speed.

The use of a pretrained model to detect audio will be required for this project. My reasoning for this is to enable polyphonic input detection. Polyphonic input detection is the ability to recognize overlapping pitches. In the vast majority of songs, there will be times where notes are playing simultaneously, such as chords. A model that can only detect one pitch at a time will be no good.

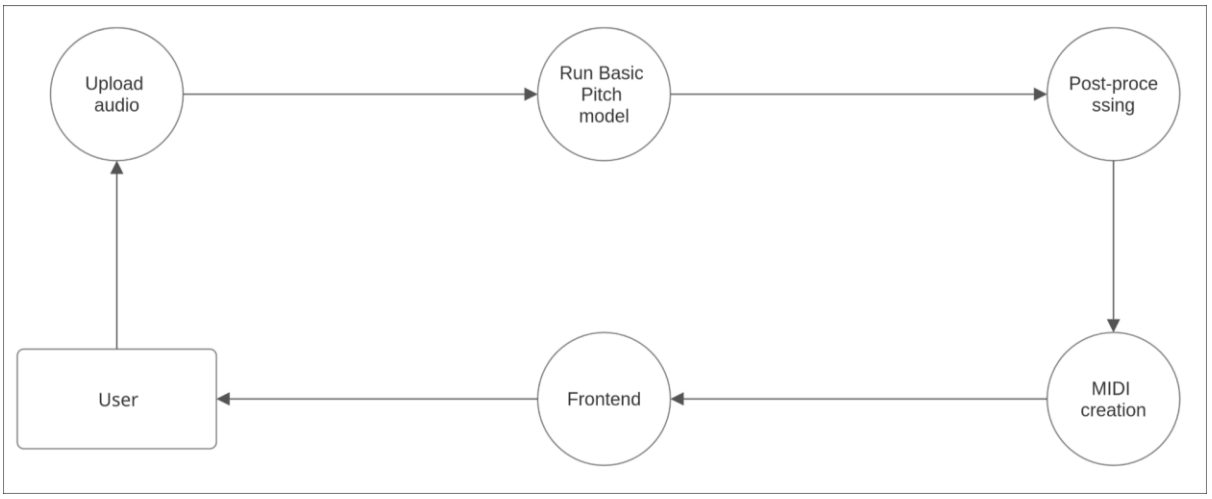
The pretrained model that I will be using to detect audio is Spotify's Basic Pitch [1]. Basic Pitch is a lightweight model that allows for polyphonic input detection. Another advantage of using Basic Pitch is that it works with any melodic instrument. These characteristics align with the philosophy of the program. It addresses problems related to the barriers of music transcription as well as being resource efficient.

The frontend of the website would be responsible for taking audio inputs, displaying the notation, and audio playback. It will be programmed using React. My reasoning for this is that the website will be dynamic. There will be a screen for audio uploads as well as a screen for audio playback. React develops websites in components, which simplifies the process of creating this dynamic frontend.

The most important part of the system would be the backend. All of the automatic music transcription logic would happen here such as audio detection, post processing and MIDI

creation. The backend would be coded in Python. I am choosing python because it has many pre-existing tools for audio processing, MIDI creation and audio rendering. It is also the most popular language for Automatic Music Transcription. This backend will be hosted on Render.

Data Flow Diagram:



1.3 Project Aims and Objectives

This project aims to make music more accessible by empowering beginner musicians to have the ability to generate their own music notations.

The biggest objective in this project is polyphonic input detection. This can be considered the foundation of the project. Without the ability to detect simultaneous notes, this website would be completely useless. This website has to be a viable option that people can use to help them with the process of learning a song.

The next objective is to create a transcription system that is suitable to be created as an output from the polyphonic input detection model. A transcription system that fits these needs is a piano roll. You can visually see where each note lies, and is intended to be used alongside audio playback generated from MIDI. The reason why transcription is necessary is that it lets the user see the structure of the song, rather than just hearing it. This aids the user in understanding how a song is composed.

The piano roll will have to be able to have audio playback. Users should have the ability to change playback speed. Audio playback will allow users to verify transcriptions are correct by comparing it to the original audio.

The chord name display will be a very nice complementary piece to the piano roll. This is important as it will let the user learn what the chord is, as well as each individual note within that chord.

All of these things must be displayed in a web based UI. This UI must be responsive and accessible. This is so that it can be used on different devices, and to include as many users as possible in its design. I decided to use a website as the medium because it is convenient for users.

1.4 Project Scope

The purpose of this project is to develop a website that lets users understand how a piece is composed by seeing its transcription. Users can verify the correctness of this transcription through the MIDI playback.

I will be developing the handling of audio into the pre-trained Basic Pitch [1] model, post-processing of the model output, MIDI creation, and the website. I will not be developing or training a model to detect audio. My reasoning for this is because it is not feasible to train a machine learning model to detect polyphonic inputs. I will not be doing live audio detection and transcription. However, I will allow users to record audio to be sent to the backend and be processed offline.

1.5 Thesis Roadmap

The literature review will be about all of the research on technologies that make this project possible and how they work. It will be heavily about Music Information Retrieval(MRI) and Automatic Music Transcription(AMT).

The system analysis will cover the needs of my project, based on the research done in the literature review.

The system design will cover how to implement the needs of my project specified by the system analysis.

The testing and evaluation will go over how I will be testing the design of my system, ensuring that the project meets the requirements and that it works as intended.

The prototype will cover what I will be creating for the prototype, and how I will be implementing it.

The issues and future work section will describe the limitations of this project idea as well as what the next iterations of this project should include.

2. Literature Review

2.1 Introduction

The purpose of this literature review is to prove that the technology required to automate the process of music transcription exists, as well as demonstrating the complexity involved for this project. I will be exploring areas that are relevant to my project such as Music Information Retrieval (MIR), Automatic Music Transcription (AMT), pitch detection, and MIDI creation.

This research is relevant to my project because I am tackling a problem that is outside what we have done in classes. Most technologies and concepts used for this project will be from independent learning, outside of what was taught in my course.

My approach to research was to look for existing open-source solutions and to analyse what those solutions do at a high level. Only after finding a solution that I can use to implement into my own project will I dive deeper into the inner workings of it. This approach is effective because it is practical. Existing open-source code will give me insights on how to build my project, as well as allowing me to analyse the system at a high level which can be used to influence later decisions about my own system.

2.2 Alternative Existing Solutions

To ensure that my idea was feasible, I had to look for existing solutions. One solution I found was Basic Pitch [1], developed by Spotify. This is an open-source system that takes audio inputs, detects polyphonic audio, and creates transcriptions. It is also instrument agnostic and lightweight. This matches perfectly with my project idea and will be something that I will be studying for my project. Weaknesses do exist with Basic Pitch. It is not as accurate as some of the non open source alternatives and the web demo is slow and often stops working.

Another solution was the open source piano transcription by Bytedance [3]. This program takes audio files and creates a transcription with MIDI playback. This shares many similarities with my own project idea and was a viable option for me to use as a base for my own project. It created accurate notations for piano inputs. However, this program was designed with piano inputs in mind. When using non piano songs, there was a big drop in accuracy.

I wanted to compare these open source solutions with a non-open source alternative. For that I used songscription. Songscription is another Automatic Music Transcription website. It works similarly to the other 2, with the difference being that when inputting audio, you also have to specify which instrument is used in the song. I found that Songscription has the most accurate piano transcription. However, it was not as accurate as Basic Pitch for non piano songs.

I decided to use Basic Pitch's pretrained model for audio detection in my own project. I find the instrument agnostic aspect to be invaluable, as this project aims to create musical notation for any song, so long as it has a melody. Another reason for using Basic Pitch is that documentation for it and how it works is extensive. A research paper that details how the model was made [5] is linked on the GitHub page.

2.3 Technologies Researched

I have researched different technologies to help build a system to automate music transcription. For the backend, I have chosen to use Python. My reasoning for this is because it comes equipped with an extensive list of tools that will help me with processing audio and MIDI creation. The business logic such as post-processing or notation creation will all be

implemented using python. To host this backend, I have 2 options which are Render and Pythonanywhere. These are free hosting services for python servers.

For the frontend, I will be using React. My website will be dynamic as there will be a page for audio input and another page for MIDI and transcription. React is good at creating dynamic webpages, as it builds websites in components. The alternative to this is using HTML, CSS and JavaScript. However, I am picking React over HTML and JavaScript because it is better suited to handle dynamic webpages, as well as good integration with hosting services like Vercel, which is what will be used to host the frontend. Vercel has great synergy with React and works seamlessly with it.

For note transcription, I will be using Spotify's Basic Pitch model[1]. This will handle note detection. I chose Basic Pitch because of its instrument agnostic, lightweight nature. It is impressively accurate across different instruments and is a great option for general music transcription.

To create MIDI, I will be using pretty_midi. This tool will allow me to create MIDI notes using the output from Basic Pitch as data to construct these MIDI notes.

Altogether, these particular technologies synergise well with each other and will be used to create a cohesive system.

2.4 Other Research

This project required me to learn about pitch and how humans perceive pitch. Sounds are composed of different frequencies, called harmonics. Harmonics are frequencies that are higher in pitch than the sound we perceive. [6] The lowest peak in frequency is the fundamental frequency which is the note that the pitch corresponds to. It is accompanied with other frequencies, which give the sound timbre [4]. Accompanied harmonics do not necessarily match the note name of the pitch. Another important thing to note is that the spacing between musical notes is logarithmic and that octaves double in frequency.

I also had to research fields of study that I have not yet learned as part of my course. The biggest thing that helped me was learning about Automatic Music Transcription (AMT) and the theory behind it. AMT is simply the process of converting an audio recording into a readable format, through the use of algorithms to extract musical information [7], which is an integral part of my project.

AMT is a difficult problem to solve. Music has different, overlapping sources of sound like different instruments and voices. To detect each musical note from this mix of sound is what gives AMT its difficulty. When trying to infer pitch, it is possible that the harmonics that come with a pitch will cancel each other out which leads to either disappearances or amplification in harmonics. Another issue with AMT is the lack of datasets [7].

I was able to find information on this in a research paper that dealt with AMT, titled "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation" [5]. This research paper documents the process of creating basic pitch [1], which is the pretrained model that I will be using in my project. In this paper, they go through the process of how they created their Notes and Multipitch model [5].

The model takes input and feeds it to a Constant-Q Transform (CQT). A constant-Q transform is a music-specialized spectrogram. Like a spectrogram, it uses time as the X axis and frequencies as the Y axis. Where it differs is that it segments the frequencies so that it is logarithmically sized and spaced. This segmentation allows frequencies to be mapped to notes on the chromatic scale. These segments are called frequency bins. The reasoning for spacing and sizing them logarithmically is because musical pitch is logarithmic.

This CQT is then copied and shifted vertically to the position of harmonically related frequencies. The shifted frequency bins will be used to capture harmonically related information. This is a technique known as Harmonic Stacking. The model described in the paper uses 9 layers of CQTs. The output is then sent to a Convolutional Neural Network (CNN).

A CNN is a neural network that uses a convolutional filter to learn patterns [7]. CNNs were originally created for image processing but they work well here because of how time-frequency representations of audio are similar to images. The CNN is broken up into layers, where each layer uses filters that learn patterns of the song and where each layer learns more detailed features than the one before.

A posteriorgram is a matrix that contains data about the pitch, time frames, and the probability of the presence of a note. The CNN produces 3 posteriorgrams. These posteriorgrams represent different aspects of a note, which are pitch likelihood, pitch onset and pitch activation. Pitch onset refers to the probability of a note starting at a specific time and pitch activation refers to the probability that a note is currently sounding during a specific time. To actually use this data and convert it into a usable transcription, it will require post processing, which I will discuss in the prototype section.

This research will be massively beneficial to me as it gives me context on how the Basic Pitch model works as well as how to handle post-processing. It has made the project feasible by giving me an understanding of AMT, the science behind pitch, CNNs, etc. I am now equipped to solve the problems in AMT by employing techniques that I have learned in my research.

2.5 Existing Final Year Projects

A similar final year project is “Seinn: Learn Irish Tunes by Ear” by Jennifer Kearns. It is a web application tailored for traditional Irish musicians with the goal of preserving culture. This application allows musicians to learn songs by breaking the songs down into smaller sections and have the user practice each bar of the song until they completely learn it. This project aims to provide live feedback and progress tracking.

This project is similar to mine as they both deal with pitch detection, with the difference being that Jennifer’s project detects monophonic input (single note), while mine aims to have polyphonic input detection. She also uses React like my project, but uses CREPE and a database, whereas I will be using Basic Pitch and my application will have no database layer.

An important thing to note for my project are the issues that she dealt with. A big one being inadequate support for MIDI playback in browsers. This serves as a useful early warning for me. I plan to solve this issue by using audio files rather than MIDI files. She also talked about the problem of finding note start and stop times. This is a well known issue in Automatic Music Transcription and was discussed in both of the research papers[5], [7] that I dissected in section 2.4. This issue of finding note starting and stop times is caused by several factors such as the lingering harmonics of a sound and reverberation.

Another final year project is “Listen2Me” by Kaiqiang Huang. This project is related to mine in that both projects deal with analysing audio through machine learning and using that audio to output something onto a website. This project uses voice recognition to manage large amounts of videos. The goal of this project is to simplify the process of video classification based on voice recognition technology. This project dealt with the issue of unwanted noise that interfered with processing. This will serve as a good learning point for me when dealing with pre-processing of audio.

2.6 Conclusions

For this section, I have concluded that my project idea is feasible as there are existing technologies that exist that directly deal with the issues that my project faces. Automatic Music Transcription is a difficult issue because of problems like lingering harmonics, overlapping pitches, and reverberation. This is made viable with modern methods such as using harmonically stacked Constant-Q Transforms and Convolutional Neural Networks. By using the pitch information that is outputted from the Basic Pitch model, I can process that data into something usable to create a transcription with playback of MIDI.

I will be using Basic Pitch as it aligns with my project's philosophy of musical accessibility, allowing not only pianists but other types of musicians to be able generate their own transcriptions.

3. System Analysis

3.1 System Overview

Initial general description of system from a user perspective`

Non-technical description

This system allows users to upload audio files and creates a transcription of that audio. This transcription will be in the form of a piano roll. There will be audio playback of the transcription in MIDI form. The system will also display the names of notes and chords that are currently being played.

3.2 Requirements Gathering

Key stakeholders of my project will generally be musicians, notably beginner musicians and hobbyist musicians. This system is aimed towards users who are unable to figure out how to play a song purely from listening to it. Trained musicians may have some use for this site for purposes such as verifying their own transcriptions are correct, but they won't have as much use for it as untrained musicians

As an untrained hobbyist musician myself, I decided that the correct approach to gathering requirements would be to analyse existing solutions and see what they did and did not do well. I chose this approach over questionnaires because data from non musicians will not be meaningful to my system.

Functional Requirements

Priority	Requirement
Essential	Allow audio input from users.
Essential	Create transcriptions in the form of a piano roll from the audio input.
Important	Audio playback from generated MIDI.
Important	Allow users to pause and resume audio playback.
Important	Allow users to skip to different sections of the audio playback.
Optional	Display names of chords that are currently being played.
Optional	Adjust playback speed.
Optional	Allow a metronome on top of the audio playback, where users can choose the BPM.

Non-Functional Requirements

Priority	Requirement
Essential	Accurately infer polyphonic pitch.

Priority	Requirement
Essential	Work with different instruments.
Important	Follow WCAG accessibility guidelines.
Important	Transcription and audio playback should be interactive.
Important	The backend should be lightweight to allow for hosting onto a server.
Important	Piano roll transcription should be synchronized to the audio playback.

3.3 Requirements Analysis

Using the requirements listed in the previous section, the system would follow a client-server architecture. The frontend would be responsible for UI and UX related tasks such as audio playback, displaying the piano roll, and handling audio input. To do this, the frontend requires an audio input interface, a visualisation of the piano roll, and an audio player for playback with interactive controls such as pause and resume. Optional features include components for a metronome, playback speed adjustment, and chord display. All of these together make up a robust frontend that allows users to interact with the piano roll and audio playback.

The backend would be responsible for the business logic where audio is processed and pitch is inferred. This will require audio input handling, the Basic Pitch model for pitch detection, and components responsible for post-processing, MIDI creation, and audio rendering. An API to allow for communication between the frontend and backend will also be necessary. These components are required because they make up the pitch detection pipeline and allow for MIDI creation for audio playback.

3.4 Initial System Specification

Functional Requirements

Priority	Requirement
Essential	Allow audio input from users.
Essential	Create transcriptions in the form of a piano roll from the audio input.
Important	Audio playback from generated MIDI.
Important	Allow users to pause and resume audio playback.
Important	Allow users to skip to different sections of the audio playback.
Optional	Display names of chords that are currently being played.
Optional	Adjust playback speed.

Priority	Requirement
Optional	Allow a metronome on top of the audio playback, where users can choose the BPM.

Non-Functional Requirements

Priority	Requirement
Essential	Accurately infer polyphonic pitch.
Essential	Work with different instruments.
Important	Follow WCAG accessibility guidelines.
Important	Transcription and audio playback should be interactive.
Important	The backend should be lightweight to allow for hosting onto a server.
Important	Piano roll transcription should be synchronized to the audio playback.

This system will follow a client-server architecture. The frontend will be responsible for audio uploads, and everything UI UX related. The backend will be responsible for processing the audio, pitch detection, MIDI creation, and rendering the audio file to be sent to the frontend.

3.5 Conclusions

Through a combination of reviewing existing technologies and alternate solutions as well as identifying my own project needs and philosophy, I was able to analyse those systems to gather requirements for my own project. I created the initial model of the system by using the requirements as a base of what the system should do. Client-Server architecture was a suitable architecture option because of how the presentation layer is separate from the logic layer and that no database is needed for my system. I turned requirements into responsibilities handled by the frontend and backend, with the frontend handling user interactions and the backend handling audio processing, pitch detection, and MIDI creation. This analysis for what the system should do will enable me to create designs of how my system should run.

4. System Design

4.1 Introduction

In this section, I will go through how I will be implementing the requirements set out in section 3, through the use of Figma designs that were created with knowledge from the literature review in mind. I will also be talking about my process in how I will be implementing these designs as well as an overview of what will be implemented.

4.2 Software Methodology

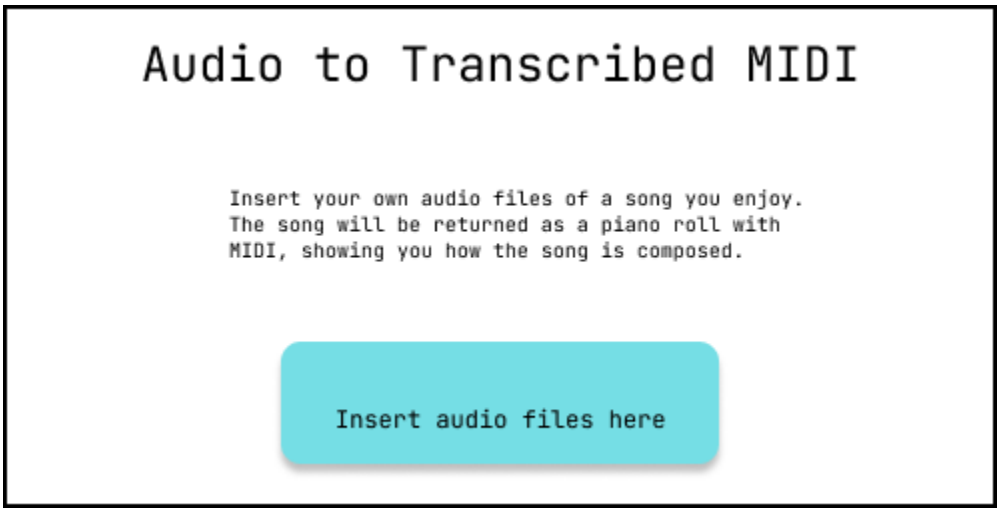
The methodology I will be following is Feature Driven Development. This approach will work best for implementing my system because it will allow for flexibility in how I allocate my time on each feature, which will allow for greater experimentation. I picked FDD over a linear approach like waterfall because a linear development won't allow for as much change in solutions according to the challenges I face along the way.

4.3 Overview of System

This system will be using client-server architecture. The frontend will be made using React. It will be responsible for UI and UX. This is where the user will input their audio file. The backend will be made using Python and will contain the music transcription pipeline. The uploaded audio file will first be sent to the Basic Pitch model. This will output 3 posteriorgrams that hold data for pitch likelihood, the probability that a note starts at a given time, and the probability that a note is currently sounding at a given time [5]. This is where post-processing will occur. Music notes will be created using this data. From here MIDI notes will be created using pretty_midi. The resulting audio file and transcription data will be sent back to the frontend where the user will be able to listen to it and see it in the form of a piano roll. From here, users will be able to interact with the piano roll and also have the options to adjust playback speed or add a metronome on top of it.

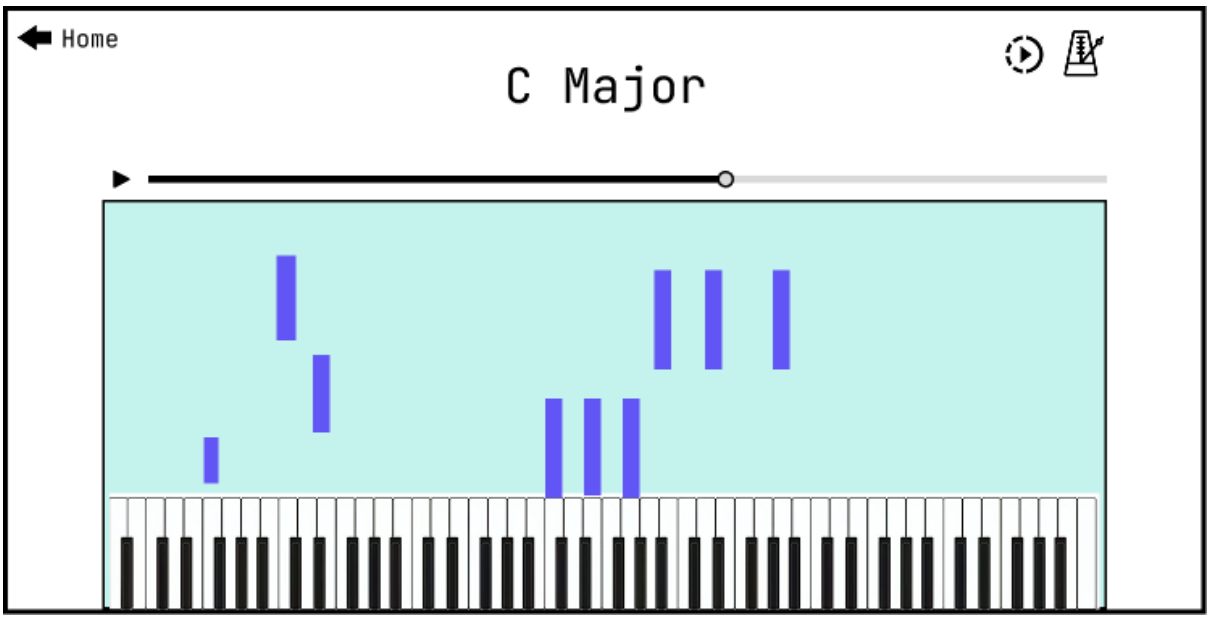
4.4 Design System

Audio input screen



This screen is the starting page of the website. Users will be asked to insert an audio file. Instructions on how to use the site are included. After uploading an audio file, they will be taken to the piano roll screen.

Piano roll screen



After receiving an audio file, it will be processed in the backend and a transcription with MIDI audio will be returned. In this screen, users can interact with the piano roll. They can skip around to different sections of the song as well as pausing and resuming. The chord and note names that are currently being played are displayed on the top of the screen. On the top right side, there are 2 icons for metronome and playback speed. Clicking on these icons will display a dropdown menu where you can adjust the metronome or playback settings. For the metronome, you can adjust the BPM, as well as time signature. For the playback speed, users will be able to adjust the playback anywhere from 20% speed to 200% speed. This design is suitable because it satisfies all of the requirements from section 3.

4.5 Conclusions

By reviewing the system requirements and seeing what should be implemented, I created a high-level explanation of how the system should be implemented by explaining the methodology and giving an overview of what the system should do. I also included Figma designs for each screen that the website will use. These designs demonstrate what the website will look like as well as how users can interact with my system. I will be implementing these features once at a time, ordered by priority, using Feature Driven Development.

5. Testing and Evaluation

5.1 Introduction

The purpose of this section is to verify and validate the designs created from the previous section to ensure that the system is robust and satisfies user needs. I will be identifying which parts of my project require testing, and make high level test cases for each one. I will be following a priority system to minimize the amount of bugs in the program while not spending too much time fixing inconsequential flaws. I will not allow any system breaking, or partially system breaking bugs. I will allow at most 3 failed test cases that cause user inconvenience and 5 minor bugs.

5.2 Plan for Testing

My approach to testing is to ensure that the essential features (pitch detection pipeline) are tested extensively and are robust. Most of the focus will go into post-processing and note creation, as there are many known issues such as determining note end times [7]. Other less essential features such as frontend components will not require as extensive testing, as the core functionality will still remain even with a flawed frontend. This approach can be reflected in my release criteria. There should be no bugs that break the entire system or parts of the system. However, I will allow up to 3 failed test cases at a user inconvenience level and 5 test cases at a minor flaw level.

For the front end, the parts that require testing are the UI, piano roll, audio playback, metronome, and playback speed adjustments. For the backend, I will need to do testing for audio handling, the Basic Pitch model, note creation, MIDI creation, and audio rendering.

Release Criteria:

P1 - System breaking: 0

P2 - Partial break in system: 0

P3 - User inconvenience: 3

P4 - Minor flaw: 5

Test Cases

Number	Priority	Module	Description
1	P1	Note creation	Process basic pitch output for a simple piano song and see if there is a noticeable amount of missing or inaccurate notes.
2	P1	MIDI creation	Play simultaneous notes and see if each one plays correctly
3	P1	Basic Pitch model	Clone the GitHub repo and check if it can create a MIDI version of audio that I select.
4	P1	Audio Rendering	Turn MIDI data into an audio file in Python.
5	P2	Audio	Play audio sent by the backend on the website and

Number	Priority	Module	Description
		Playback	see if it maintains its state.
6	P2	Piano Roll	Let the audio play on the website, see if the piano roll follows along with it.
7	P2	Metronome	Turn metronome on, adjust speed to 80BPM and check to see if 80 beats play in a minute.
8	P2	Playback speed adjustment	Set playback speed to 200%. Check if the piano roll is still synchronized and that the song finishes in half the time.
9	P2	Note Offsets	Input a simple piano song. Check if notes end prematurely or if they go on for too long.
10	P3	Audio input	Upload a non audio file and see if the system allows it.
11	P3	Keyboard Navigation	Upload an audio file, set the metronome to 60BPM, adjust playback speed to 80%, and skip forward in the song only using keyboard inputs.
12	P3	UI	Run the website on different screen sizes e.g. mobile, tablet, desktop. Check if it resizes correctly.
13	P3	Accessibility	Run the website through an accessibility scanner and check if every level A and AA guideline is met
14	P3	Note Creation	Create test audio where I note down which notes and chords are played and when. Use this audio as test input and check if it matches.

5.3 Plan for Evaluation

Evaluation Plan

Module	Description	Evaluation
Note creation	Compare notes created with the correct notes from a test song.	Accurate start and end times of notes. Not a noticeable amount of incorrect or missing notes.
MIDI creation	Generate MIDI in python and save it as an audio file. Play back generated audio.	MIDI notes matched what was specified, Audio was in good quality/not corrupt.
Basic Pitch	Input a test audio file with known notes	Pitch likelihood, note start

Module	Description	Evaluation
	into the model.	and activation times are accurate.
Piano roll	Play audio and see the piano roll.	Piano roll is synchronized with audio.
Audio playback	Play the audio, pause and resume, skip to different times in the song.	Audio playback works with no issues.
Audio Input	Upload non audio files.	Reject non audio files, only accept audio files.
Metronome	Toggle the metronome and set the BPM.	BPM is correct and metronome plays without issues.
UI	Test the website on different layouts.	UI elements are correctly scaled and match the container.
Accessibility	Run the website through an accessibility scanner.	Meet all level A and AA guidelines, allow some level AAA issues.

5.4 Conclusions

In this section I talked about how the system will be tested and evaluated by creating test cases with details on how they should be conducted. My testing plan prioritises that the backend is as robust as possible as it is the most important part of my system. My test and evaluation plan are all structured, ensuring that testing and evaluation are consistent processes for the future.

6. System Prototype

6.1 Introduction

In this section I will be going over what my code submission will be like. For my prototype, I will be focusing on post-processing of Basic Pitch output, particularly note and MIDI creation. This is not as simple as getting notes from Basic Pitch and converting it to MIDI. Basic Pitch gives probability on the likelihood of a note, the probability that a note starts at a given time, and the probability that the note is currently active. It does this in the form of posteriorgrams. Posteriorgrams are matrices that represent time and frequency.

Basic pitch does not give direct data on when a note ends or confirmations of notes. This is something that I will have to do myself in post-processing. There will be no UI for this prototype, I will be using a command line interface.

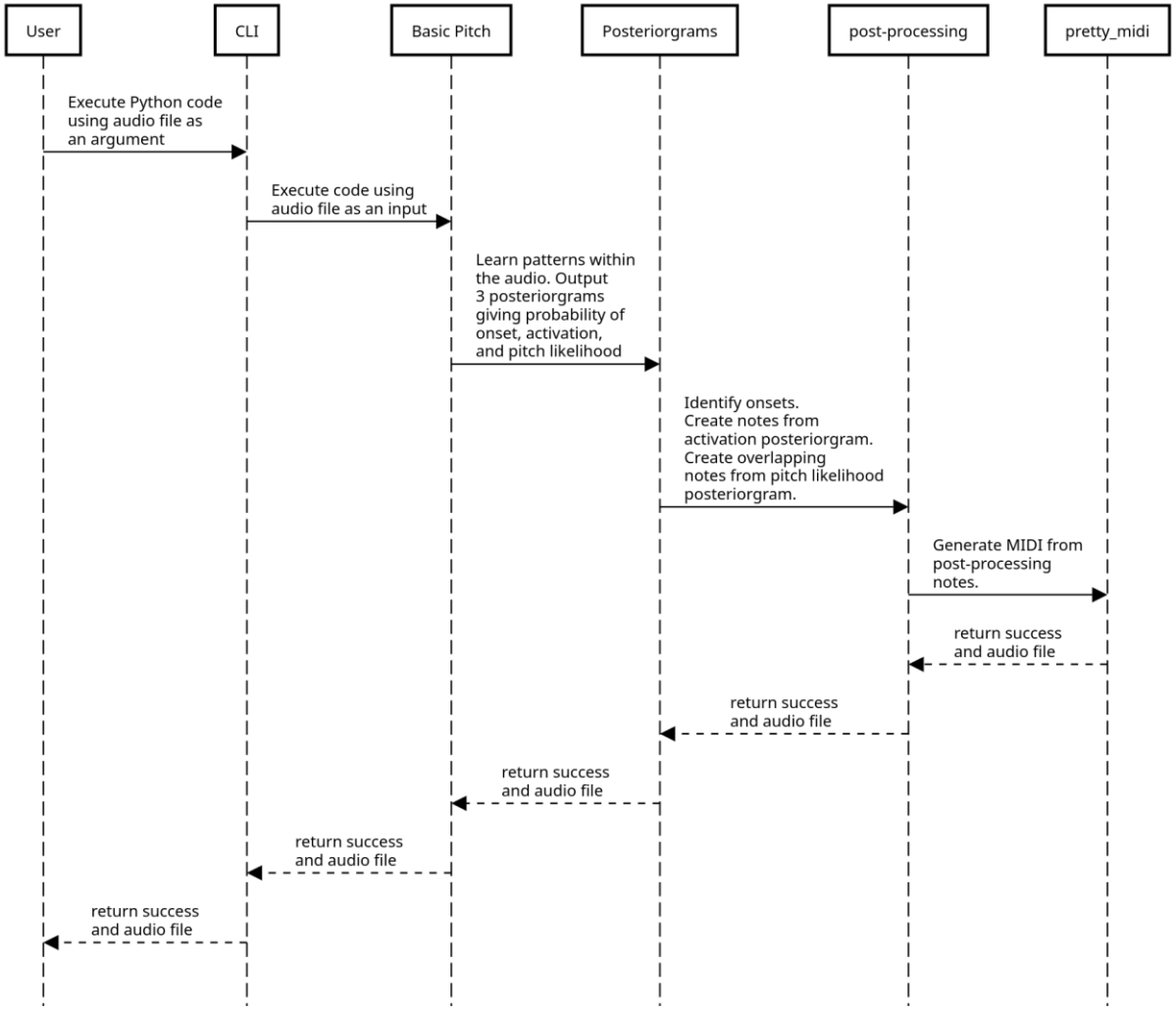
This prototype will focus on one of the big challenges of Automatic Music Transcription which is handling onsets and offsets. Determining onset and offset is difficult because small changes lead to very different results. Tackling this problem early on in development will allow me to confirm that the project is feasible.

6.2 Prototype Development

The first step in creating notes would be to identify note start times, known as onsets. To do this requires peak picking. Peak picking is a process where you extract frequencies of peaks [8]. Onsets with more than a certain threshold are added to a group of candidate onsets and onsets that don't meet this threshold are discarded. These candidates will be iterated through and music notes will be created by tracking forward in time until the likelihood of note activation falls below a certain threshold. This is how we get our probabilities of note ending times also known as offsets.

After each onset has been used, more notes will be created by iterating through each note in the note activation posteriorgram that is above the note activation threshold. This process is the same for the onset note creation except it traces both forward and backward in time until the probability that the note is active falls below the activation threshold. After creating a note, the corresponding probability value in the note activation posteriorgram is set to 0. Notes that were created that run less than 120ms are removed. To get the probabilities of overlapping pitches, peak pick the frequency of the pitch likelihood and keep everything that has a likelihood over the activation threshold. [5]

Sequence Diagram



6.3 Results

Expected Results

Module	Test case	Expected results
Note creation	1	Pitch is accurate. Note onsets and offsets are slightly off. Some missing notes.
Note offsets	9	A mixed result. Some offsets are fine while others are slightly inaccurate.
MIDI creation	2	MIDI is created without issues and sounds fine.
Note creation	14	Notes are accurate. Onsets and offsets are slightly early or late.

These are the results that I am expecting based on known AMT struggles. I am predicting that pitch detection will be accurate across different instruments and songs but onsets and offsets could be inaccurate due to things like harmonics cancelling each other out, amplifying each

other or lingering. These predicted struggles are where I am going to be putting most of my focus on during development, as they are the most likely components to fail.

6.4 Evaluation

My evaluation of prototype test results will follow my evaluation methodology from section 5. By using test cases that match how the system is intended to be used for both validation and verification, I can ensure that the system is able to meet its requirements and behave as designed. There is currently no implementation of code or tests, therefore this will all be based on research and intuition.

Tests for my prototype will cover note offsets, note creation, and MIDI generation. These components are essential to get correct and require extensive testing as they will be some of the hardest components to deal with in my system. I predict that note creation will detect accurate pitches but will struggle for onsets and offsets. This will be due to several factors such as the issue of false positives at the end of notes due to lingering harmonics [7]. MiDI creation should not be a problem as it is much less complex than note creation.

6.5 Conclusions

For my prototype, the goal will be to process the posteriorgram output from the Basic Pitch model into data that can be used for note creation and to create MIDI from those notes. There are no current implementations of code. However, from researching AMT, I am able to predict the challenges that I will be facing in post-processing, which are offsets and onsets.

7. Issues and Future Work

7.1 Introduction

The purpose of this section is to highlight the challenges I will be facing along the way as well as the limitations of the current AMT technologies. I will go over how I will handle these issues and limitations, as well as my plan for the future and how I will allocate my time for each part of the project.

7.2 Issues and Risks

A part of the reason why AMT is so difficult is because of the amount of issues and limitations that come with the technology. The current strategy is to use harmonically stacked Constant-Q Transforms, where each layer is shifted to where harmonics of musical notes reside. This is great because it forces frequencies to be mapped onto the pitch of a music note.

However, this can also be looked at as a limitation. Forcing pitches to only make up musical notes of the chromatic scale completely removes microtones. Microtones are the sounds that are in between musical notes. They are commonly used in eastern music, or as a stylistic choice by some musicians. CQTs completely remove this and instead maps the pitch to the nearest frequency bin [5]. This is also a problem for instruments that produce sounds between the notes in the chromatic scale, such as a violin or guitar bends. It will not be possible to capture these microtones unless some other strategy that replaces the use of CQTs is implemented.

This will be an issue for my system if a user decides to use a song with heavy use of microtones. I have already tested this myself using the Basic Pitch web demo and Songscription [2], and the result was the least accurate transcription I have gotten from using these sites.

Other issues of audio detection primarily stem from harmonics. There seems to be 3 main issues in this area which. Accompanied harmonics in a pitch may linger after the main pitch has already died. This is a problem because it can lead to non-existent notes or extended offsets. This issue was seen in Seinn, a previous FYP. Another issue is that it is possible for harmonics to cancel each other out. This can lead to missing notes or notes that end early. The inverse of this issue is harmonic amplification. This is when harmonics add up with each other, which leads to false positives. Because of the sensitivity in how harmonics are handled, certain techniques such as vibrato would not be handled well by the system because of how the frequency is constantly moving. This is another big problem because vibratos are a common technique in music.

In the context of my project, it means that note creation will have to be done precisely, small changes in thresholds could lead to a completely different transcription. This is because of how onset identification and offset creation are extremely sensitive and rely on small thresholds.

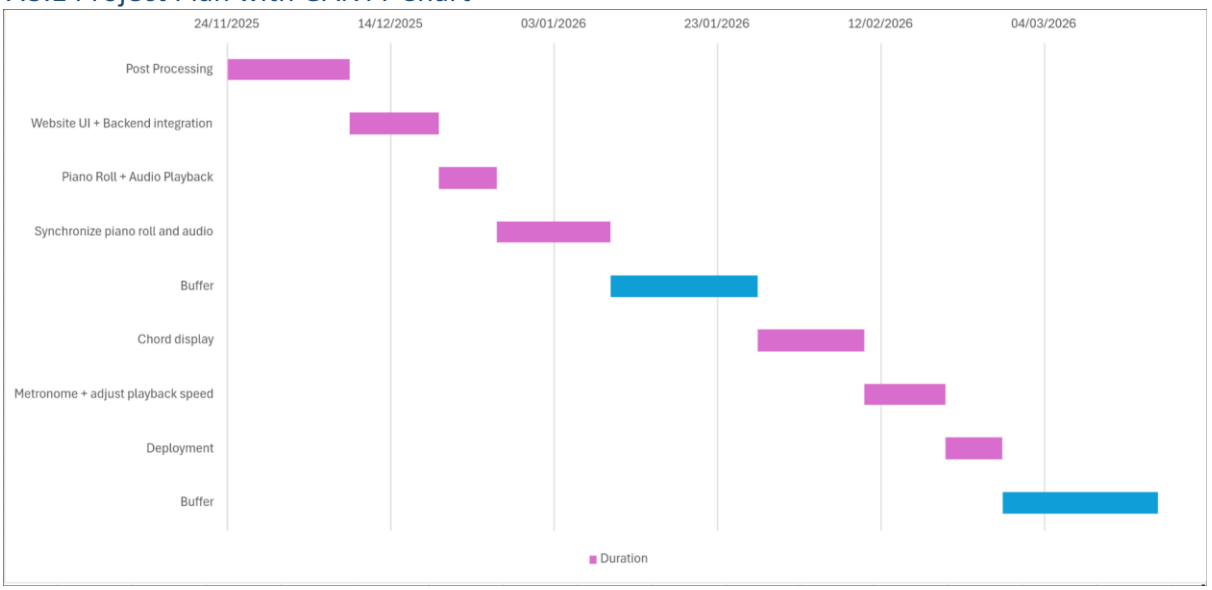
7.3 Plans and Future Work

My immediate plan is to create the post-processing part of the system. This is the area that directly deals with the issues talked about in the previous subsection. The only realistic way to address these issues would be through trial and error to see which thresholds work best to capture the most accurate pitch information. As for the microtonal detection limitation, it is not something that is feasible for me to fix as it is a technological limitation.

I will be using Feature Driven Development to develop each part of my system one by one. I will only be focusing on a single feature at a time with no overlap on different features. This is in line with the typical FDD workflow. After post-processing is created, I will develop the website UI and integrate it with the backend. This would be built using React, where I will be implementing the designs from section 4. After the website works with the backend, I will implement the piano roll with audio playback. These 2 components would have to be synchronised with each other. This makes up the essential requirements of my project and should be completed around the middle of January, giving me plenty of time to implement other features.

After the essential components are developed, I will make the chord display. This is another part of my project that will be a great challenge. The final components to be implemented are the metronome and playback speed adjustment. Once the entire system works, I will deploy it. This should be all finished at the start of March. I have also given myself 2 buffers, both roughly 20 days each. This is necessary because work from exams or other assignments could slow down development for my final year project.

7.3.1 Project Plan with GANTT Chart



References

[1] “Basic Pitch: An open source MIDI converter from Spotify - Demo,” *basicpitch.spotify.com*. https://basicpitch.spotify.com/

[2] “Songscription AI,” *Songscription.ai*, 2025. https://www.songscription.ai/

[3] Bytedance, “GitHub - bytedance/piano_transcription,” *GitHub*, 2025. https://github.com/bytedance/piano_transcription

[4] Whitney Cole, “What are harmonics?,” *www.youtube.com*. https://www.youtube.com/watch?v=znbfY-tXROk

[5] R. Bittner, J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, “A LIGHTWEIGHT INSTRUMENT-AGNOSTIC MODEL FOR POLYPHONIC NOTE TRANSCRIPTION AND MULTIPITCH ESTIMATION,” 2022. Accessed: Nov. 13, 2025. [Online]. Available: https://arxiv.org/pdf/2203.09893

[6] “A guide to fundamental frequency and harmonics in music,” *Izotope.com*, 2025. https://www.izotope.com/en/learn/fundamental-frequency-harmonics

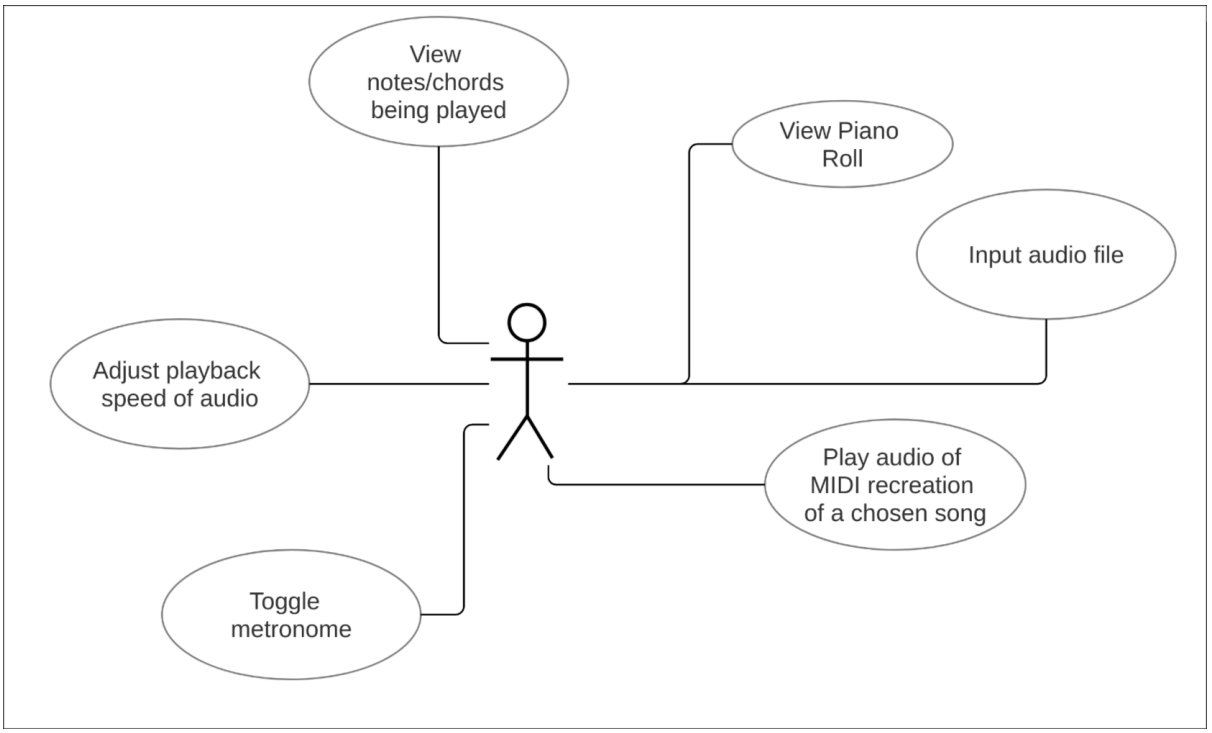
[7] Y. Telila, T. Cucinotta, and D. Bacciu, ”Automatic Music Transcription using Convolutional Neural Networks and Constant-Q transform” 2023. Accessed: Nov, 17, 2025. [Online]. Available: https://arxiv.org/pdf/2505.04451

[8] “Peak-Picking,” *Inmr.net*, 2025. https://www.inmr.net/Help3/ref/picking.html (accessed Nov. 20, 2025).

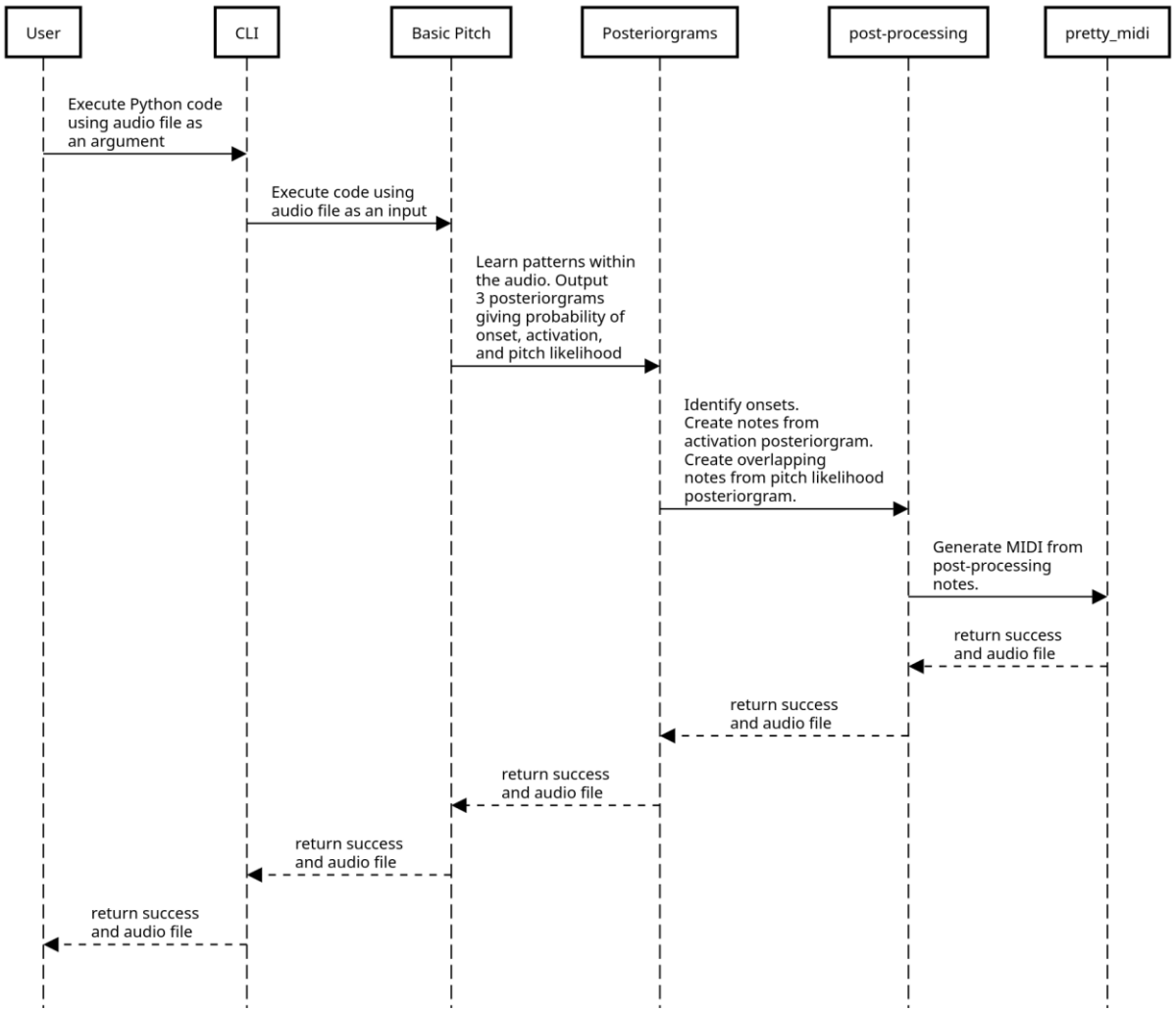
A) Appendix A: System Model and Analysis

To gather requirements, I looked at existing solutions and analysed what would work for my own system and what wouldn't. Examples of systems that I analysed are Basic Pitch [1], Songscription [2], and Piano transcription by Bytedance [3].

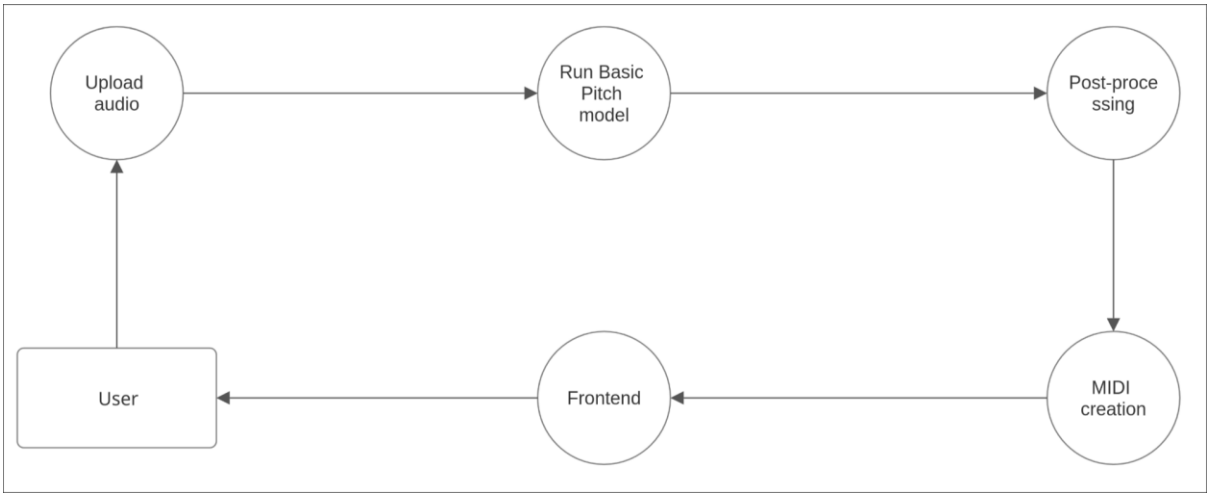
Use Case Diagram



Sequence Diagram



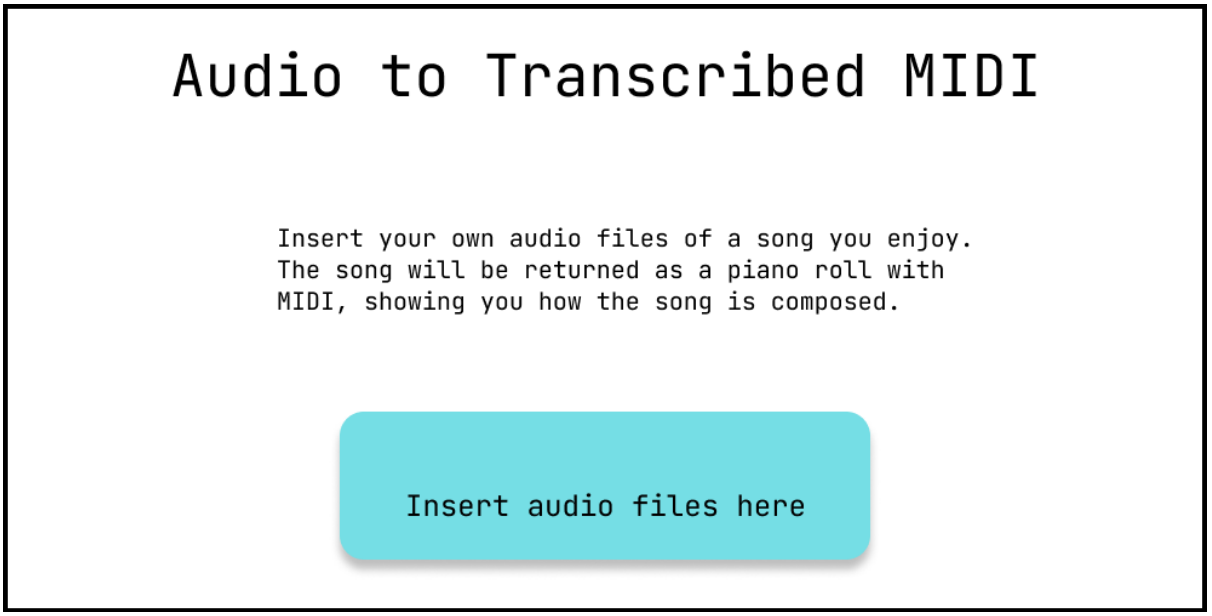
Data Flow Diagram



B) Appendix B: Design

To approach design, I used the existing system requirements as a base to create the initial designs. Figma was used to create the prototypes for the website screens. I decided to keep the design minimal, ensuring users aren't confused by unnecessary elements. This aesthetic is also in line with popular UIs today.

Audio Input Screen



Transcription and Audio PLayback Screen