Ollscoil
Teicneolaíochta
an Atlantaigh

Atlantic
Technological
University

Pitch Detection

Project Engineering

Year 4

Oluwamayowa Praise Ojofeitimi

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2024/2025

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technological University.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

_____

Acknowledgements

I would like to extend my gratitude to Analog Devices for Sponsoring my Final Year Project and in particular my Conrad Collins who helped and guided me during my Project development.

I would also like to thank my project supervisors Paul Lennon and Patt Hurney for their support and assistance.

Finally, I would also like to thank my classmates for their personal support, as we worked together helping each other in our peer reviews.

Table of Contents

1 Summary

In my final year project, I created an application Pitch Detection which would enable a user or a musician to detect notation, recognize major and minor chord notation, chord name detection, key detection, along with an Instrument tuning program.

Users are allowed to record or upload an audio file that is saved, and a Fast Fourier Transform(FFT) is performed on the audio file and peak frequencies are then gathered from the file, with this we are able to identify notation by converting frequency values to notational values with the help of the notation conversion formula. With machine learning algorithms, chord concepts along with the name can be predicted.

For project management, we utilized Jira for monitoring work while developing the projects, along with OneNote weekly logs.

This project is made using a python environment PyCharm, the various frequency functions were also developed here since I used PyCharm when I was training my models for machine learning features. Technologies like React.js for front end and also FastAPI for front end are being used.

2 Poster

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Pitch Detection

Pitch Detection

Project Summary

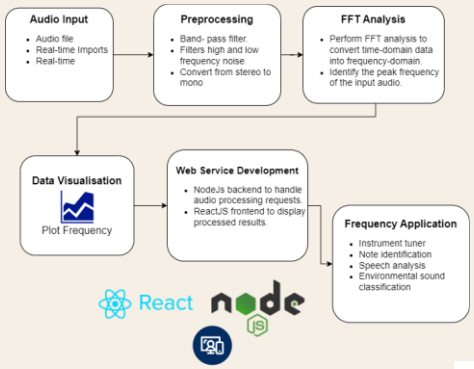
This Project is a real-time pitch detection application in which analyses audio frequencies for various applications, including and not limited to instrument tuning, note identification, speech analysis, and environmental sound classification.
The application processes audio input from pre -recorded files, live microphone audio, and external frequency tones, it applies Filtering and Fast Fourier Transforms(FFT) analysis and visualises the result in real time.
This project includes a web interface with React and Node.js to provide interactive controls for adjusting frequency ranges and displaying real -time data update.

What My Project Does

•Real-Time Audio Processing:
The system processes various audio inputs in a python environment using a .Wav file, it then applies a band-pass filter and normalizes the signal for consistent volume throughout the audio sample.
•Frequency Detection & Analysis:
Using FFT, the system converts audio to frequency data, detects peak frequencies, and identifies musical notes.
•Visual Real-Time:
An interactive graph displays real-time frequency variations, allowing users to adjust settings like frequency range.
•Web-Based Interface:
The backend of the system is built using Node.js, which processes the data, and React is used for the frontend, providing a user interface. It updates the data in real -time using a communication protocol, allowing users to interact with and visualize the pitch detection results.
Frequency-Based Applications:

- **Instrument Tuner:** Provides real-time tuning information such as 'Sharp', 'Flat'.
- **Note Identification:** Detects musical notes and displays corresponding frequencies.
- **Speech Analysis:** Analyses speech pitch and variation.
- **Sound Classification:** Differentiates between sound sources and classifies notes with machine learning.

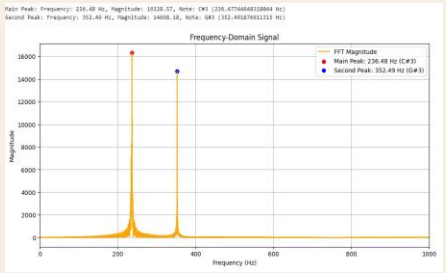
Project Architecture



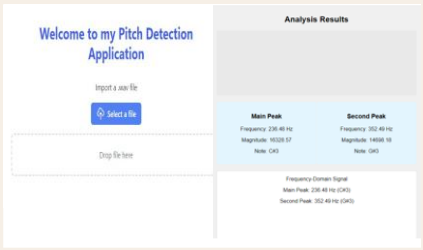
Technologies and Tools

•Python pandas – for processing data and generating CSV reports.
•Matplotlib – for plotting graphs and visualizing frequency spectrum in Python.
•NumPy – for implementing the FFT algorithm and processing frequency -domain data.
•Librosa – for pitch detection and musical note identification.
•Git – version control system for managing codebase.
•React Native – for building mobile apps.
•FastAPI – backend server to handle requests and audio processing.
•React.js – frontend library for building dynamic user interfaces.

Data Visualisation



Web Server



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Introduction

This Project falls under the Engineering discipline of Digital Signal Processing, for my project I merged my present interest in signal processing and my passion for music to be of help to primarily young musicians who are not as advanced as others. This project not only benefits them, but also benefits users to collect information needed when they have an instrument to play and is of no use to users who don't have an instrument available with them since music/audio can be recorded and information needed will be available to them. Also, this project would expose me to a challenge of actually working on machine learning, as I am used to C types of coding venturing deeper into machine learning presented a challenge

The project's primary objective was to create a working music assistant. The project's purpose is to expose pitch detection through a webservice, where the user can upload or record audio and then get in-depth analysis, such as FFT output (frequencies), notational values, chord prediction and instrument tuning.

3 Background Technologies and Research

In preparing for this project, it was of utmost importance that I use the right tools and technologies for what I wanted my project to look like, using current compatible tools that can coexist together.

3.1 Audio Signal Processing

Audio Signal Processing is the manipulations of audio signals through computational means to analyse, modify and or to enhance then. In this projects case it was used to analyse signals.

What signal processing does is convert continuous analog audio signals into discrete digital samples that a computer can work with. This conversion can be and is done in this project's case by utilizing FFT's.

When we are looking at domain graphs, we see that the time domain is plotting amplitude against time whereas the frequency domain is plotting the signal as a summation of sinusoidal components

3.2 Machine Learning

Machine Learning is a subset of artificial intelligence that allows computers to learn from data patterns and make decisions or predictions without being programmed specifically. machine learning systems get better over time as they're subjected to additional data.

As seem in my project when a dataset of chords is trained using the chords training model when a chord is played it can predict said chord value.

Machine learning algorithms learn from examples in training data to make predictions or decisions on new input data.

3.3 Web Technologies

3.3.1 React.js

React.js is a popular open-source JavaScript library for building user interfaces.

React creates a lightweight version of the actual DOM in memory and uses it to determine the most efficient way to update the DOM in the browser when data changes.

React is utilized in my Frontend to show FFT analysis outcomes.

3.3.2 Axios

Axios is a JavaScript library that is utilized for making http requests from the browser. Axios is a promise-based HTTP client utilized for simplifying the process of sending async requests to Rest endpoints.

3.3.3 FastAPI

FastAPI is a high-performance, modern web framework to build APIs using Python. FastAPI is highly suitable for machine learning. I use FastAPI as my backend for my server side.

3.3.4 Uvicorn

Uvicorn is a Python web server that conforms to the ASGI standard. It is designed to host Python web applications that are standards-compliant ASGI and enable the utilization of asynchronous code. Uvicorn is a central piece of the new Python stack and especially for asynchronous applications like the one that I am creating that make use of the popular FastAPI.

3.4 Python Libraries

3.4.1 Soundfile

A Python audio library for reading and writing various audio file types (WAV, FLAC, OGG). Provides a simple API for audio I/O in Python.

3.4.2 Io

A typical Python module that handles the different types of I/O (input/output), including text I/O, binary I/O, as well as raw I/O.

3.4.3 NumPy

The central Python package for numerical computation providing support for big multi-dimensional arrays and matrices and mathematical operations on these arrays and matrices.

3.4.4 SciPy

A library of scientific and numerical computations augmenting NumPy's capabilities with features like optimization, signal processing, and statistics in addition to linear algebra.

3.4.5 Sounddevice

Sounddevice.py offers bindings into the PortAudio open-source audio I/O API as a simple and Pythonic entry point for sound playback and recording on multiple platforms. Sounddevice is particularly suitable for audio applications involving real-time audio.

3.4.6 Matplotlib

A powerful plot and visualization library that produces static, animated and interactive visualizations in Python. It's most often used for creating graphs, charts and figures for scientific computing.

3.4.7 Librosa

A Python library focused on music and audio analysis. It includes audio signal feature extractors, signal visualizers, and signal process tools, and the common applications are music information retrieval.

3.4.8 Pickle

A Python standard library for serializing and de-serializing Python objects that allows you to save complex structures on disk and reuse them later.

3.4.9 Sklearn

A strong Python machine learning library offering simple and effective tools for modelling as well as data analysis.

3.4.10 Pandas

A powerful library for data manipulation and analysis that includes such data structures as DataFrame and Series and structured data-handling structures. Ideal for transforming, exploring, and cleaning the data.

4 Project Architecture

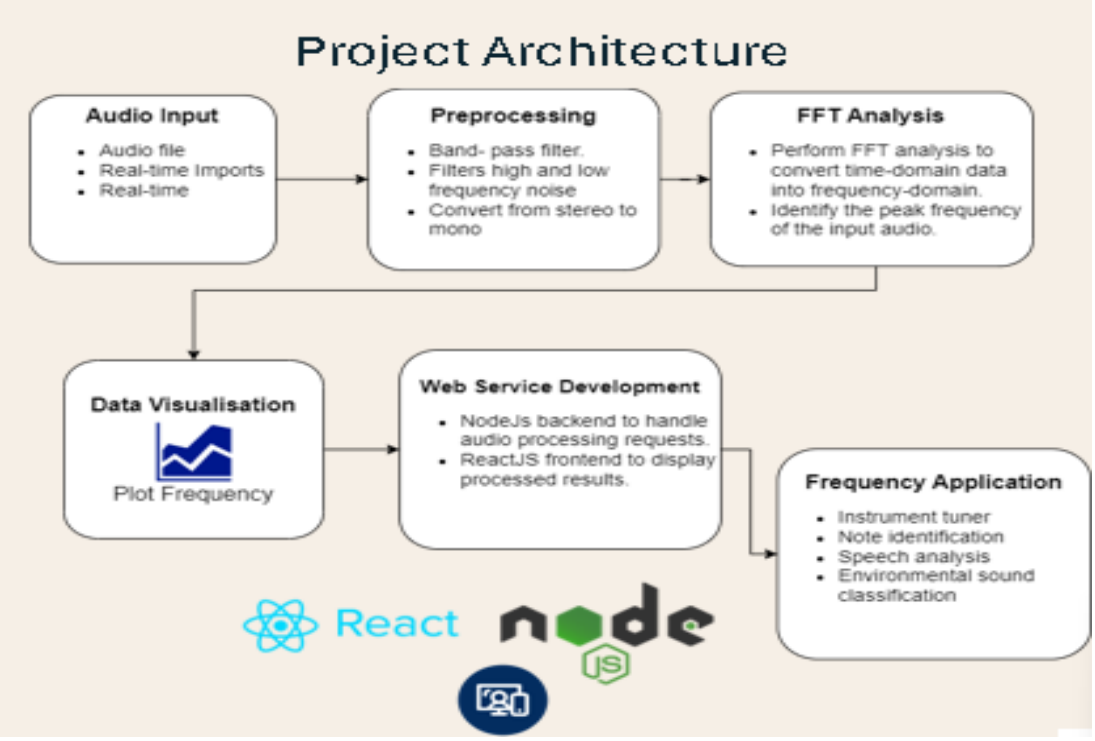


Figure 4-1 Architecture Diagram

5 Project Flow

1. The user is allowed to upload a file or record live audio and save as a file →
2. The audio is passed through a band pass filter to filter noise, static and extreme frequencies. The audio is converted from stereo to mono →
3. FFT analysis is then performed on the audio collected calculating its peak frequencies →
4. We then receive our data visualisation prints and graphs →
5. We then move over to the Web development where backend server and Frontend is created and we out our results on the frontend browser where we can perform step 1 within the browser→
6. Finally, in the webservice browser we can select from out different frequency Applications.

6 Project Plan

The project plan went as follows:

September → December

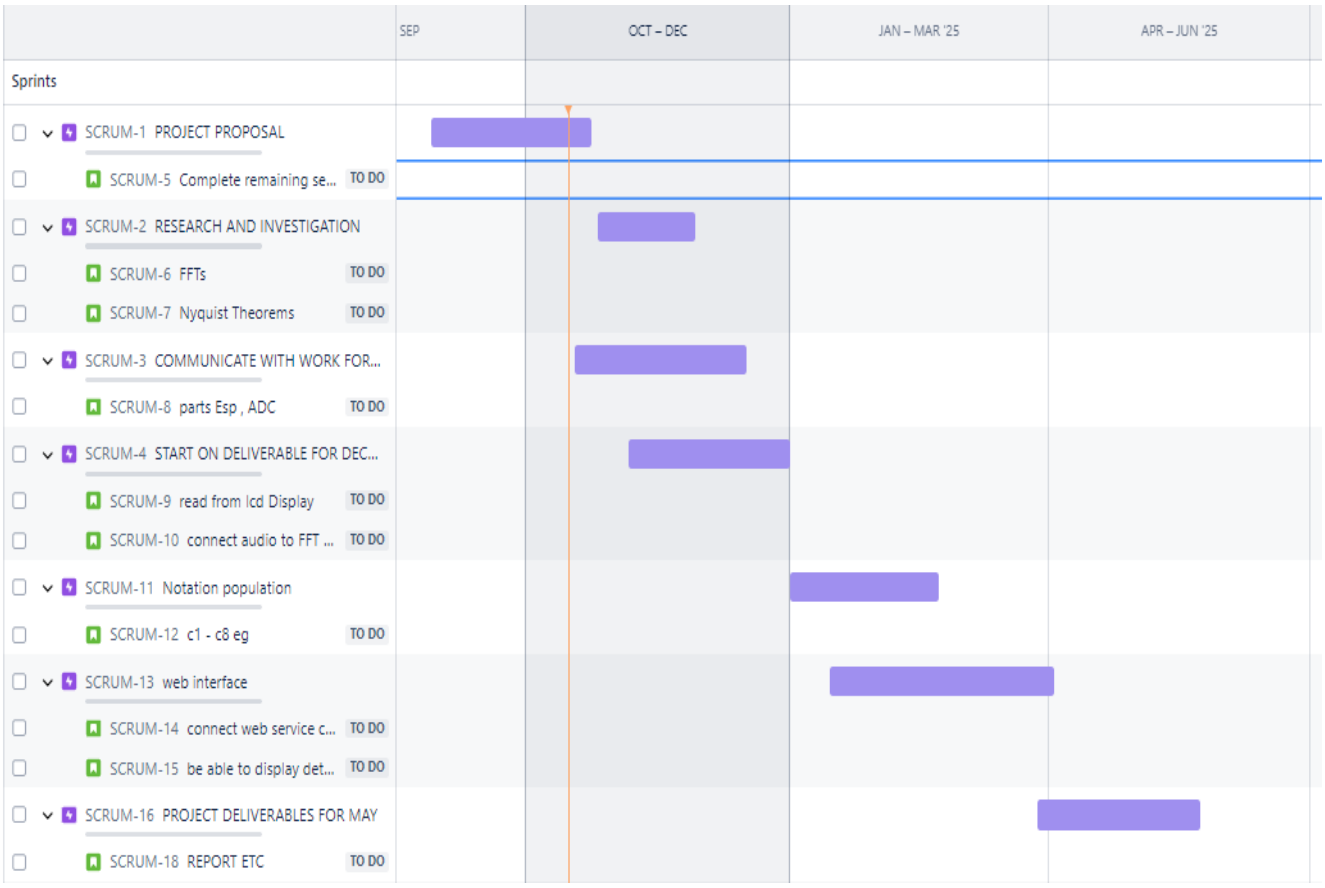
- Drew up a project plan.
- Completed Project Proposal.
- Communicated with work placement to get advice and Components (which were originally part of the project).
- Did the research and investigation aspects discovering project capabilities.
- For December completed Phase 1 in Python file upload then perform FFT.

December → March

- Complete Phase 2 and Phase 3.
- Phase 2 record live audio then save as file.
- Phase 3 machine learning for predictions.

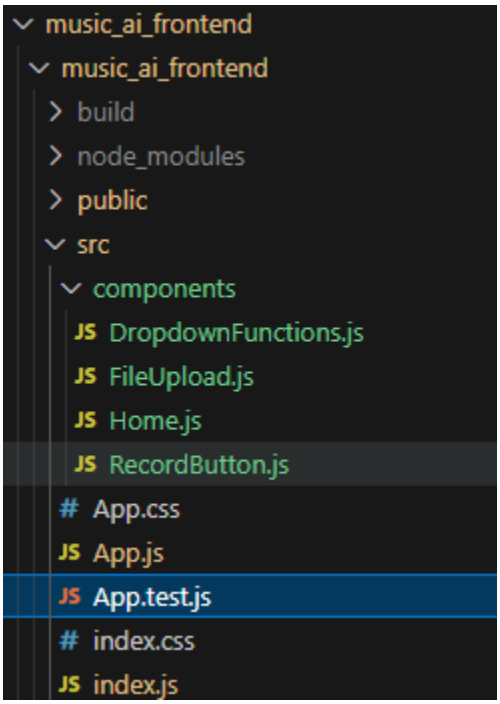
March → May

- Web development
- Report
- Video
- PowerPoint
- Prepare Demonstration


SEP
OCT – DEC
JAN – MAR '25
APR – JUN '25
Sprints
SCRUM-1 PROJECT PROPOSAL
SCRUM-5 Complete remaining se... TO DO
SCRUM-2 RESEARCH AND INVESTIGATION
SCRUM-6 FFTs TO DO
SCRUM-7 Nyquist Theorems TO DO
SCRUM-3 COMMUNICATE WITH WORK FOR...
SCRUM-8 parts Esp , ADC TO DO
SCRUM-4 START ON DELIVERABLE FOR DEC...
SCRUM-9 read from lcd Display TO DO
SCRUM-10 connect audio to FFT ... TO DO
SCRUM-11 Notation population
SCRUM-12 c1 - c8 eg TO DO
SCRUM-13 web interface
SCRUM-14 connect web service c... TO DO
SCRUM-15 be able to display det... TO DO
SCRUM-16 PROJECT DELIVERABLES FOR MAY
SCRUM-18 REPORT ETC TO DO

7 Code Implementation

7.1 Frontend Application



The frontend was built using React.js to create a fast web interface.

Each React component has a task:

Home.js

- Displays a welcome message and a brief description of what the application does.
- Helps the user understand they can either upload audio or choose an analysis function.

FileUpload.js

- Allows users to upload a .wav file from their device.
- Uses axios to send a POST request to the /upload_fft endpoint on the backend.
- Handles form data and ensures the proper content type is used (multipart/form-data).

```
try {
  const response = await axios.post('http://127.0.0.1:8000/upload_fft', formData, {
    headers: { 'Content-Type': 'multipart/form-data' },
```

Inside:

- handleFileChange sets the selected file.
- handleUpload posts the file to the server.

```
const handleFileChange = (e) => {
  setFile(e.target.files[0]);
};

const handleUpload = async () => {
  if (!file) return;
  const formData = new FormData();
  formData.append('file', file);
```

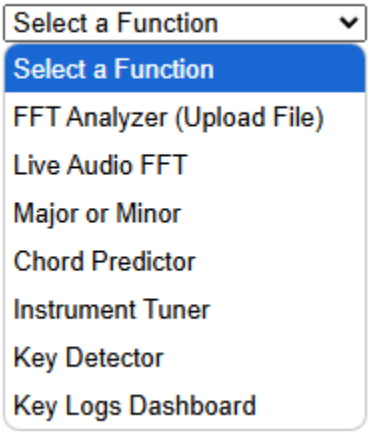
DropdownFunctions.js

Displays a dropdown menu listing all available analysis functions:

- FFT Analyzer
- Major/Minor Prediction
- Full Chord Prediction
- Instrument Tuner
- Instrument and Key Detector
- Dashboard Logs

```
return (
  <div style={{ textAlign: "center", marginTop: "30px" }}>
    <select value={selected} onChange={handleChange}>
      <option value="">Select a Function</option>
      <option value="uploadFFT">FFT Analyzer (Upload File)</option>
      <option value="liveFFT">Live Audio FFT</option>
      <option value="major_minor">Major or Minor</option>
      <option value="chord_predictor">Chord Predictor</option>
      <option value="tuner">Instrument Tuner</option>
      <option value="key_detector">Key Detector</option>
      <option value="key_logs">Key Logs Dashboard</option>
    </select>
  </div>
);
}
```

- Updates the selected function state, triggering which form or upload method is shown.

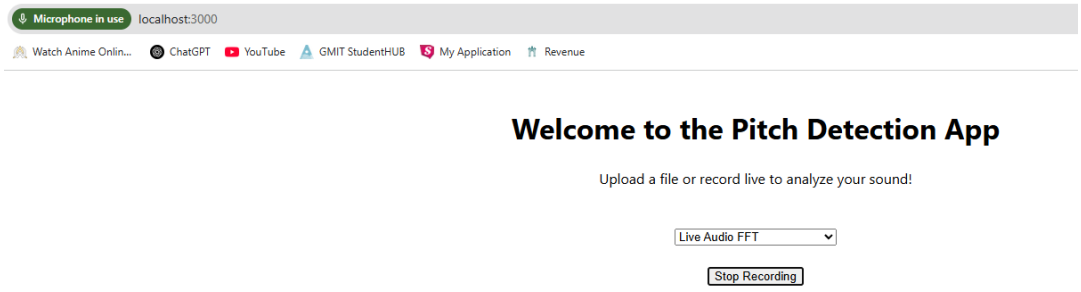


RecordButton.js

- Initially used for recording live audio using react-media-recorder.
- However, due to recording issues and cross-browser complications, it was removed.

```
  try {

    const response = await axios.post('http://127.0.0.1:8000/record_live_audio', formData, {
      headers: { 'Content-Type': 'multipart/form-data' },
    });

    setResult(response.data);

  } catch (error) {
    console.error('Error uploading live recording:', error);
  }
};

return (
  <div style={{ textAlign: 'center', marginTop: '20px' }}>
    <ReactMediaRecorder
      audio
      blobPropertyBag={{ type: 'audio/wav' }}
      onStop={handleStop}
      render={({ startRecording, stopRecording, mediaBlobUrl }) => (
        <>
          {!isRecording ? (
            <button onClick={() => { setIsRecording(true); startRecording(); }}>
              Start Recording
            </button>
          ) : (
            <button onClick={() => { setIsRecording(false); stopRecording(); }}>
              Stop Recording
            </button>
          )}
        </>
      )}
    />
  </div>
);
```

- It takes in the recorded audio and sends using endpoint to the backend to perform FFT



7.2 Backend Server

The backend was developed using FastAPI, a modern Python web framework known for speed and simplicity.

Each backend endpoint has a specific purpose:

Endpoint	**Purpose**
/upload_fft	Accepts file upload, performs FFT, returns peak frequency and magnitude
/predict_chord_simple	Upload a file → Predict Major or Minor chord using ML model
/predict_chord_full	Upload a file → Predict exact musical chord using ML
/instrument_tuner	Upload a file → Detect peak musical note
/instrument_key_detector	Upload a file → Predict instrument and musical key
/dashboard_logs	Fetch CSV logs of key predictions for later analysis

Each endpoint workflow accepts a file UploadFile.

```
@app.post("/upload_fft")
async def upload_fft(file: UploadFile = File(...)):
```

Then reads binary contents using await file.read()

```
contents = await file.read()
```

Load and reads the audio using soundfile.

```
data, samplerate = sf.read(audio_file)
```

7.3 Machine Learning

Two pre-trained ML models are passed into the backend:

Instrument Classifier

```
new_instrument_classifier.pkl
```

This model predicts the type of musical instrument based on audio features (MFCC, Chroma, Tonnetz).

Chord Classifiers

```
chord_classifier.pkl
full_chord_classifier.pkl
```

Predict whether the input chord is:

- Major or Minor
- Specific Chord (chord name)

How did I Train these Models?

To train these models I populated a database with audio files of instrument audio from Bass Guitars, Electric Guitar, Pianos, and Violins along with chords of every variation. These models were trained using sklearn algorithms in a Python Environment

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
model = RandomForestClassifier(n_estimators=200, random_state=42)
model.fit(X_train, y_train)
```

7.4 Feature Extraction

In machine learning audio just like in my project raw sound waves are too advanced to be directly fed into a model.

So, we get numerical summaries of features that capture important characteristics of the sound. I use three Extraction Features in my Project.

1. MFCC (Mel-Frequency Cepstral Coefficients)

MFCCs are a compact representation of the timbre texture the colour or texture of a sound. It looks at the distribution of the sound's energy over different frequencies. Frequencies are warped to a Mel scale this scale approximates the way humans perceive sound in general we are more sensitive towards lower frequencies. Logarithmic transformation and discrete cosine transform (DCT) are used on the spectrum to compress the information. This is important because different instruments or chords have different timbre. MFCC helps the machine learning model to differentiate between a variety of sounds like a major chord and a minor chord.

```
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
mfcc_mean = np.mean(mfcc.T, axis=0)
```

2. Chroma Pitch Class Representation

Chroma features indicate which musical notes (C, C#, D) are playing at a moment in time, independent of octave. What it does is apply the Fourier Transform (FFT) to the audio. Reduces all frequencies to twelve bins, one per semitone of the musical scale (C, C#, D,). Concentrates on the musical content and not the raw frequency. In music analysis like chord or key identification, it is less important which frequencies are sounded and more about the notes. Chroma helps the model to understand harmonic structure, whether a C major or an A minor.

```
chroma = librosa.feature.chroma_stft(y=y, sr=sr)
chroma_mean = np.mean(chroma.T, axis=0)
```

3. Tonnetz (Tonal Centroid Features)

Tonnetz stands for tonal network it captures how notes relate to each other harmonically. It is based on music theory specifically neo-Riemannian theory, and it maps notes into a 6-dimensional space representing:

- Consonant relations (major/minor thirds, fifths).
- Tonal distance between chords.

Tells us not just what notes are played, but how they are harmonically structured.

It is particularly useful for chord classification and key detection because different chord types have distinct positions in the Tonnetz space.

```
harmonic = librosa.effects.harmonic(y)
tonnetz = librosa.feature.tonnetz(y=harmonic, sr=sr)
tonnetz_mean = np.mean(tonnetz.T, axis=0)
```

8 Ethics

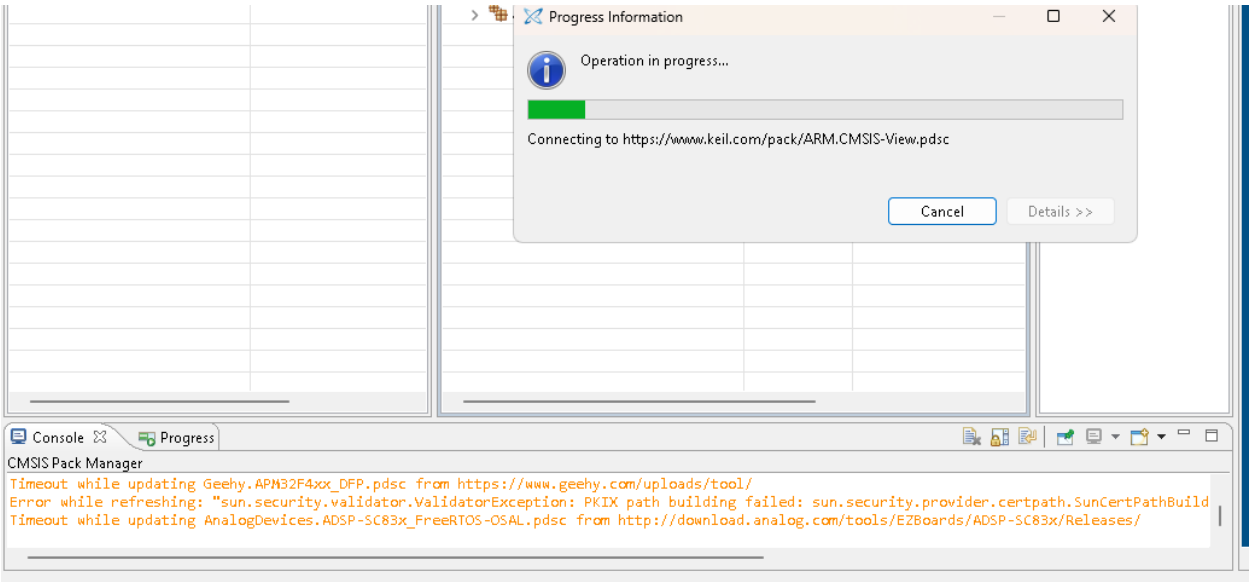
My Pitch Detection Project is part of a field of research where ethical considerations are taken into account. Any audio uploaded or recorded is only temporarily processed for analysis. There is no storage of user recordings permanently, this is to protect user privacy and prevent misuse of personal recordings. The app is not to be used to record or analyse copyrighted musical content illegally without permission. Users should upload or record only audio for which they have rights. My Instrument and chord recognition models were trained on mixed datasets to minimize model bias. The system is meant to assist musicians, All predictions such as key detection and chord identification are returned as suggestions and not final results.

9 Problem Solving

9.1 Hardware Packet installation

The first issue I saw in the development stage of my project when I was still using ADI hardware was when I loaded the ADI crossCore embedded IDE it would not allow create a project. For this IDE on startup u have to press a button which loads packages from a package website keil.com but the issue I saw was some of the necessary packages needed to configure the IDE were not being downloaded so the IDE did not recognise when I plugged in my Evaluation board the ADICUP 3029.

I was able to fix this issue by manually going onto the webservice searching through the packages and installing them manually, once downloaded the project was then created and the IDE recognised the board



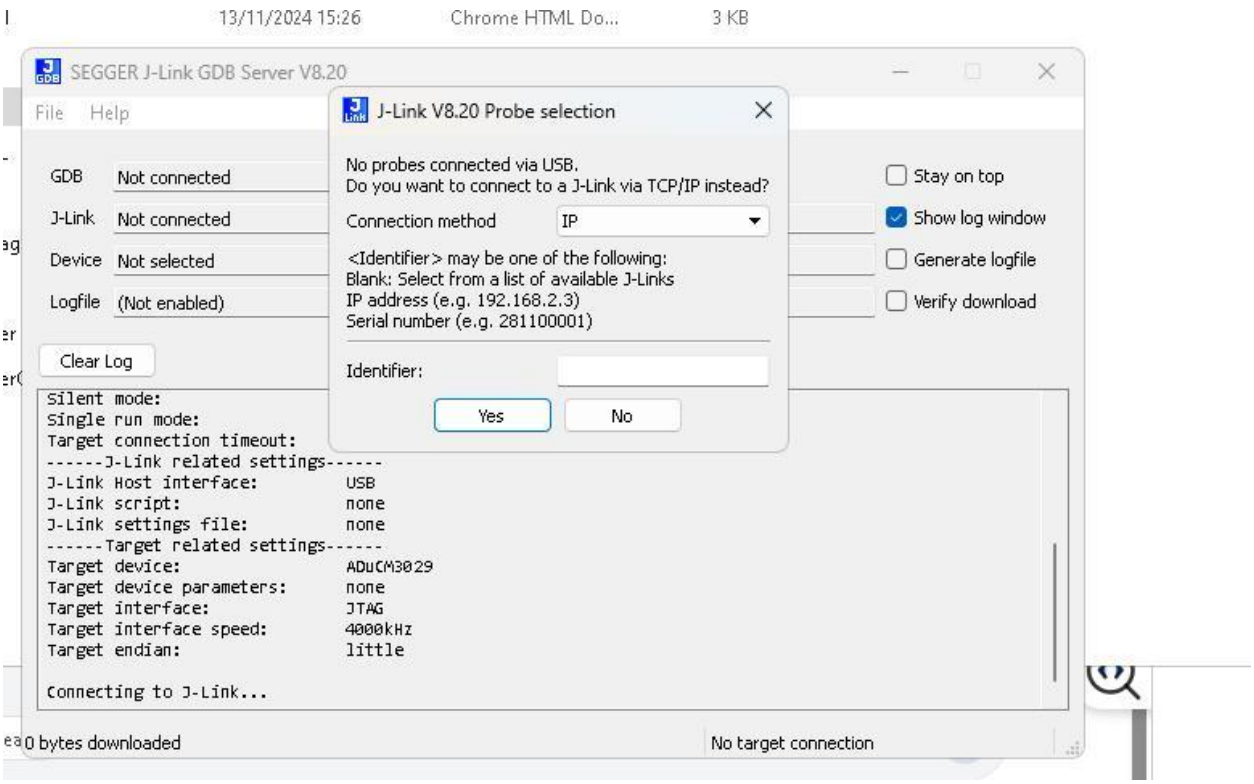
9.2 Hardware Dugger

The second issue I had with the Hardware in which I wasn't able to fix even with assistance from work placement supervisor in Analog Devices was a debugger issue. This issue was extremely persistent remedy after remedy it would not function the issue was an external style debugger is needed, SEGGER J-LINK is needed to connect, what it needed to do was connect to

the 3029 via the J-link and then select the J-LINK option in the debugger settings and from there the debugger would be set up.

Instead of this the J-LINK would not connect while debugging this I found that the J-Link was not being recognised on my PC it responded as if it did not exist.

As a result of this I decided to change my project path to a Python heavy Project.



9.3 Format not Recognised

For this issue when users uploaded audio blobs recorded live issues appeared, when the audio is recorded the browser tell it is in an unsupported format even when the content type was audio.wav, minor deviations in how browsers wrote the audio caused loading problems.

Results:


```
{
"error": "An error occurred: Error opening <_io.BytesIO object at 0x0000019DEF69B3D0>: Format not recognised."
}
```

Conclusion

My Pitch Detection was successful in achieving its core goal to provide a web-based platform that is capable of processing musical audio files through various methods like Fast Fourier Transform (FFT), chord prediction, key detection, and instrument identification. The users can upload.wav files, and the backend, powered by FastAPI and machine learning models, processes the audio and produces useful musical information. The frontend: implemented in ReactJS, straightforward user experience, with components such as file upload and function selection.

Key deliverables are:

• FFT analysis and peak frequency extraction accurately.

• Chord classification into major/minor and full chord names.

• Instrument tuning support by determination of the closest musical note.

• Instrument type and musical key determination from audio files uploaded.

• Real-time logging and dashboard generation for monitoring historical predictions.

The project culminated in a working prototype that demonstrates how machine learning, digital signal processing, and web technologies are combined.

Although live audio recording was technically challenging and was not included in the final system, the current design provides space for future additions such as enhancing live recording, employing machine learning models to handle a broader range of instruments, and offering mobile devices app features. In general, my project combines frontend and backend technologies to generate a music analysis tool.

10 References

[1] FastAPI, "FastAPI Documentation," [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: 07-April-2025].

[2] React, "React Documentation," [Online]. Available: https://react.dev/. [Accessed: 21-January-2025].

[3] Axios, "Axios Documentation," [Online]. Available: https://axios-http.com/. [Accessed: 05-February-2025].

[4] Uvicorn, "Uvicorn Documentation," [Online]. Available: https://www.uvicorn.org/. [Accessed: 07-April-2025].

[5] Librosa, "Librosa: Audio and Music Signal Analysis in Python," [Online]. Available: https://librosa.org/. [Accessed: 23-January-2025].

[6] SciPy, "SciPy FFT (Fast Fourier Transform) Documentation," [Online]. Available: https://docs.scipy.org/doc/scipy/reference/fft.html. [Accessed: 02-March-2025].

[7] React Media Recorder, "React Media Recorder Documentation," [Online]. Available: https://www.npmjs.com/package/react-media-recorder. [Accessed: 07-April-2025].

[8] A. Aaron, "FastAPI: The Modern Web Framework for Python," RealPython, 2021. [Online]. Available: https://realpython.com/fastapi-python-web-apis/. [Accessed: 07-Apr-2025].