**MetroGnome  
Software Requirements Specification, Software Design Specification, and Project Plan**

Miles Anderson, Dax Lynch, Harry Robertson, Ryan Helms 5-12-24

**System Requirements****1**

[1.1 System Overview](#_l4d4kg34xn3w) 1

[1.2 Justification for a New System](#_thwzn3pbgkxx) 1

[1.3 Operational Features of the Proposed System](#_43h5e9uloszr) 2

[1.4 User Classes and Modes of Operation](#_rfc0i5zawhtd) 2

[1.5 Operational Scenarios or Use Cases](#_sdq7l9k4sie) 2

[1.6 Real-World Users](#_14zcfvny4tq4)

**System Design**

[2.1 System Overview](#_1fob9te) 2

[2.2 Software Architecture](#_3znysh7) 3

[2.3 Software Modules](#_2et92p0)

**Project Plan**

3.1 Overview

**Acknowledgements**

**Revision History**

| **Date** | **Author** | **Description** |
| --- | --- | --- |
| 5/12/24 | All | Initial document creation |
| 5/14/24 | MA / RH | Updated project plan and SRS, post interviews with musicians |
| 5/19/24 | All | Updated SDS and SRS, created models for various systems |

**1. SRS**

# 

# ConOps

## 1.1 System Overview

The workflow for a typical musician includes a tuner and a metronome. Often, at the beginning of a practice session, a musician will tune their instrument, and then rely upon a separate tool to configure their metronome. To maximize performance and output, we propose conjoining the two processes into a single tool.

## 1.2 Justification for a New System

Musicians alternating between applications creates an opportunity for distraction. Studies have shown that switching apps, in instances separate from musical training, can decrease productivity (Madore & Wagner, 2019). By combining these components, musicians will be able to focus on their practice rather than their technology, increasing the efficacy and productivity of their practice sessions.

## 1.3 Operational Features of the Proposed System

The proposed system will provide a combination of a tuner and metronome, seamlessly integrated so that users can switch between the two. The system allows the opportunity for the addition of future music-related technologies, such as transcription. The system will be designed to support such additions.

## 1.4 User Classes and Modes of Operation

Our system allows for only one user class: musicians. The application requires no configuration of a database, nor admin setup. Rather, the web-application focuses on curating the skills of musicians. The modes of operation are binary and simple: The musician is using the metronome; the musician is not using the metronome; the musician is using the tuner; the musician is not using the tuner. The system will not allow for the synchronous use of both the tuner and the metronome, as the microphone input may be disturbed by the sound of the metronome.

## 

## 

## 

## 

## 1.5 Operational Scenarios or Use Cases

**Description:** This use case describes how a musician would use the app.

**Actors:** A musician with a microphone.

**Preconditions:**

1. The musician has access to the internet, a web browser, and a microphone.
2. The student wishes to use the tuner and the metronome.

**Steps to Complete the Task:**

1. The student will use the tuner portion of the app, playing a note on their instrument, and reading off the frequency reported from the tuner, and adjusting their instrument until the note is within an acceptable range of the expected note.
2. Then the student will launch the metronome from the landing page, input their desired BPM and time signature, and click play.

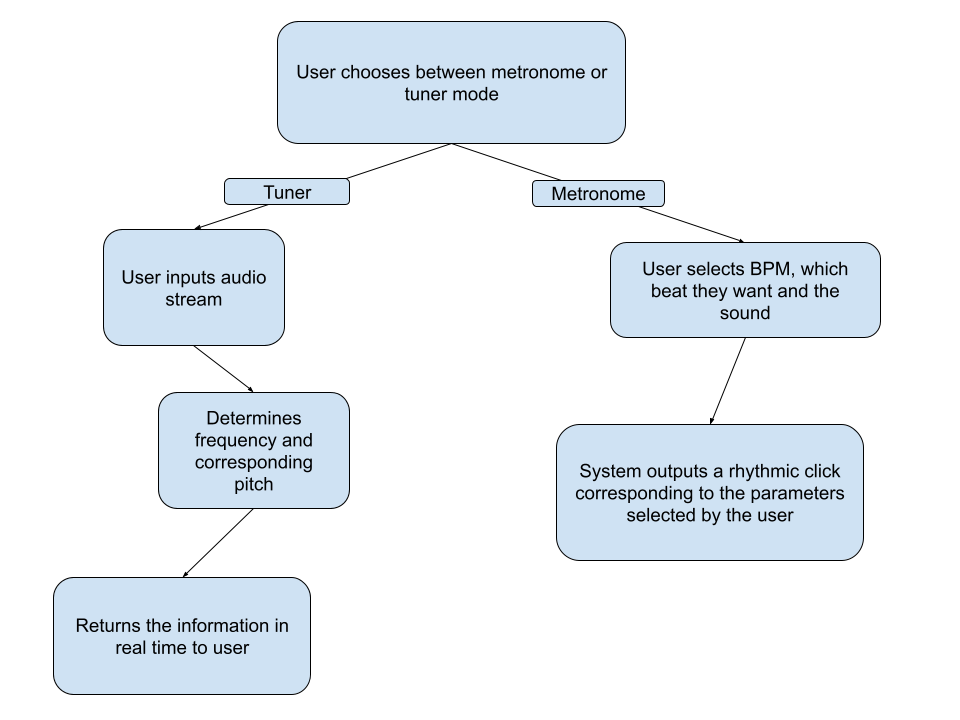
## 1.6 Real-World Users

To find real world users we can poll music students entering the music buildings on campus and ask if they are willing to be interviewed. We can then poll them about which tools or apps they use to tune their instruments, and which metronomes they use. We will ask about what features they like and which features they dislike.

**2. System Design Specifications**

# 2.1 System Overview

We intend to create a program for musicians with dual functionality, to assist the user in various ways. They will be able to choose between using the program as a metronome, to keep track of time whilst they perform/practice, and the other function will be a tuner, which will take in an audio stream, and return to the user the pitch, allowing for quick adjustments to their instrument. The system will be able to function with various different instruments allowing a wide variety of users to find use in this software.



# 2.2 Software Architecture

**Components:**

* Tuner / Visualizer
* Metronome

**Design Rationale:**

We intend to create our project using JavaScript, as the simplicity of our technology would benefit from the front-end stylings allowed to the language. We will use CSS to enhance the UI. A possible addition would be a visualizer, providing further appeal to users. The software will rely upon a client-server model, wherein the user sends requests to a remote server that then displays the corresponding components.

After doing brief interviews with various musicians, we determined that creating a good fully flushed out metronome would be quite the undertaking. To be on par with an industry standard app our metronome will have to function in various ways, having a variable speed, subdivision of notes, a tap feature where the user taps out the tempo, accent on the downbeat, gradual speed up etc. Creating an all in one music app that allows users to break free from multiple individual apps and keep it simple was the driving idea behind our project, combining functionality with a slick and simple design was paramount.

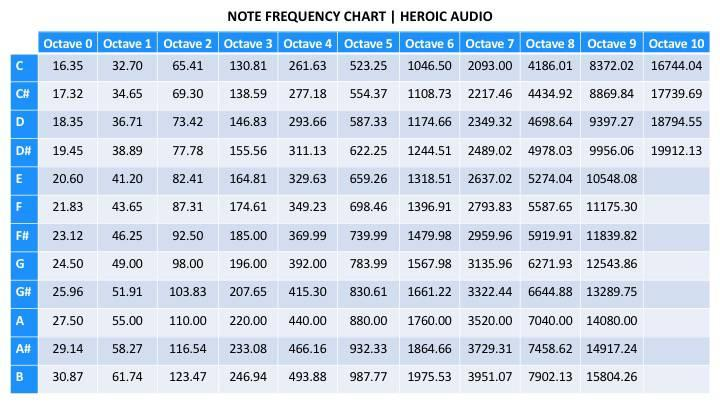
# 2.3 Software Modules

**Metronome**

* Program will take a user-inputted BPM, and produce a sound in rhythm with the chosen BPM
* Program will also let users choose what beats they want their sound to be on, (1st beat of the bar, 2nd, etc.), as well as the time-signature of the beat.
* Program will provide functionality for polyrhythms, i.e. multiple concurrent time signatures.

**Tuner**

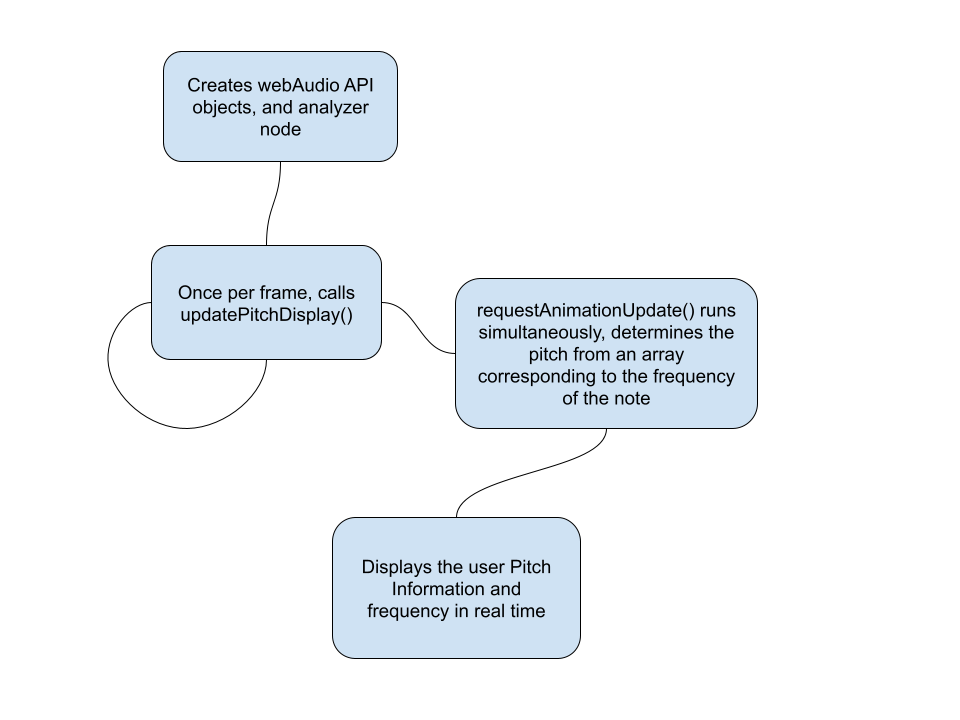
* The program will process a user-input audio stream into numerous arrays, in sequence. The program will then determine the maximum value from the array and display the corresponding frequency, as well as its closest adjacent note.



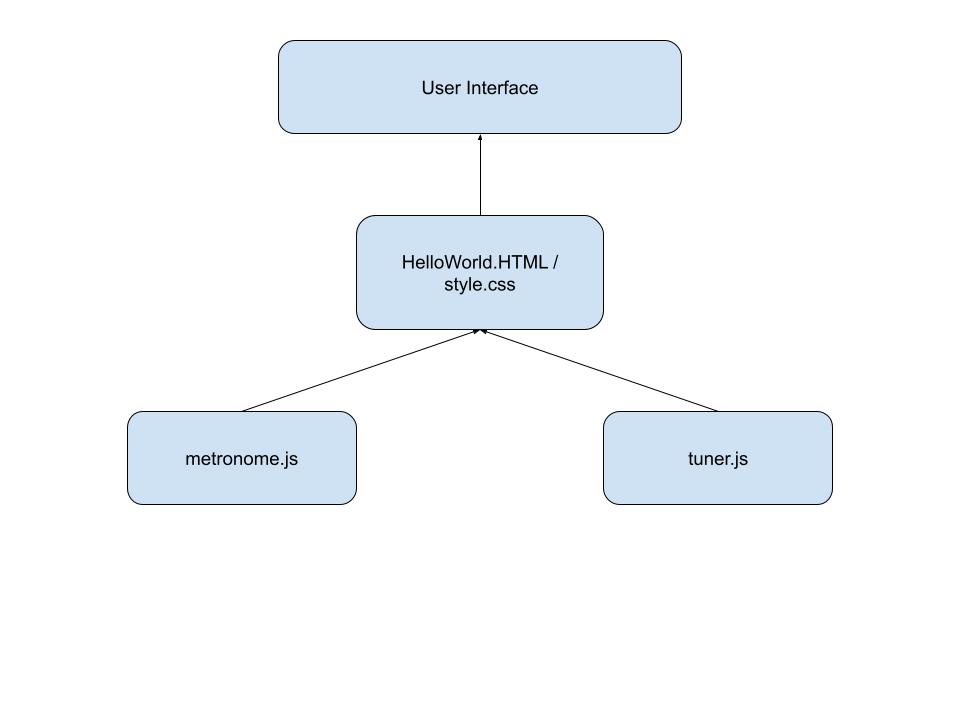
**Fig 2.3.1 Frequency Chart (Heroic Audio, nd.)**

* The audio stream will be coming in real time, and the user will receive the pitch information (nearly) instantly.

**System Models:**

****

The chart above shows a simple representation of how the tuner interacts with our webpage through JavaScript



The model above gives a simple representation of how the backend is handled and information is displayed to the user. Metronome.js and tuner.js handle the calculations, and they are embedded in the html file where with the help of css to provide some style, the information is sent back to the user in real time.

**Interaction between modules**

* The tuner and metronome are hosted on the same homepage, giving easy access to both of them for the user
* As they are on the same page, both the tuner and the metronome are able to function at the same time, regardless of the state the other is in, this allows the user to use the system more efficiently then having an independent tuner or metronome separately
* Tuner and Metronome are each individual JavaScript programs, “HelloWorld.html” and “style.css” (names are subject to change) are the hub which links the two main modules together

**3. Project Plan**

\

**3.1 Overview**

Our project plan consists of three main components, with those components leading into prototypes. Those prototypes, then, will be iterated upon until a final product is ready for delivery. We are using the iterative development approach, allowing for agile development and adjustments depending on the state and needs of our system and stakeholders. We also intend to leave ample time at the end to iterate upon our design and perhaps add additional functionality. Currently, we are waiting for input from music students. The three phases are as follows:

**3.2 Guitar Tuner**

At the crux of the application is the tuner. It will be built in parallel with the metronome (see section 3.2). As the technologies behind it are relatively simple, it will be built first.

**3.3 Metronome Module**

The second functionality of the web-app will be the metronome. We will develop the metronome in parallel with the guitar tuner, integrating both with the landing-page/front-end

**3.4 MetroGnome Landing Page**

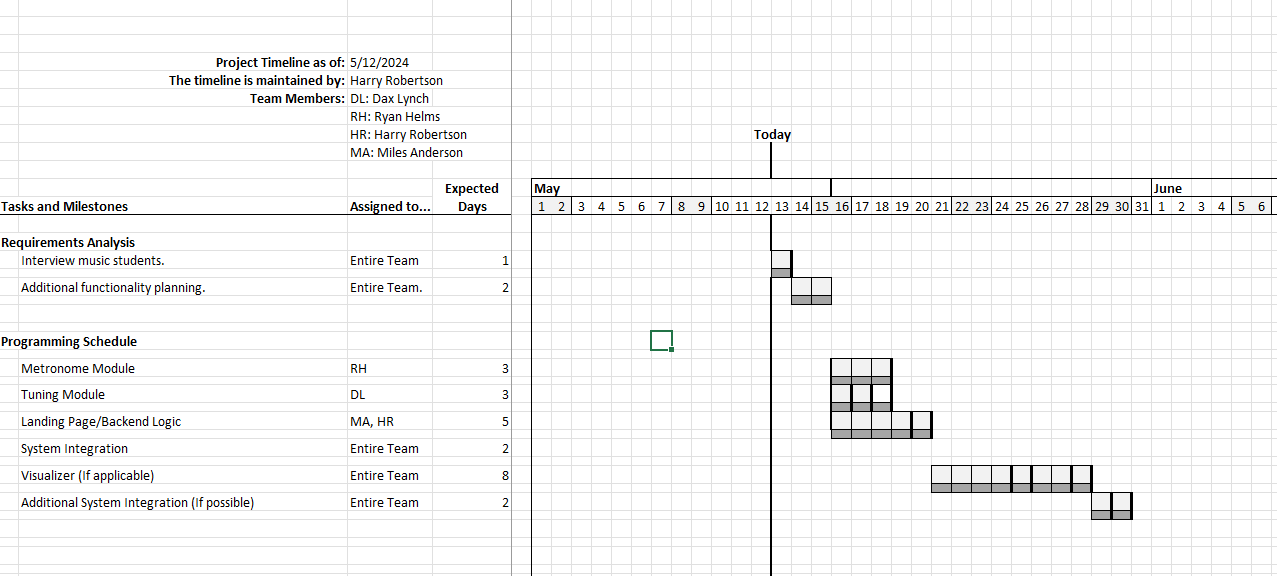
The landing page will be the front-end hub in which users can launch both the metronome and the tuner. The tuner and metronome will be developed first, as the landing page is simply a means to access said components.

**3.5 MetroGnome Server**

JavaScript server built to handle user input and interaction. Will be hosted on a local machine or ix. Built following the landing page. Prior, we will be running on a local machine to limit coupling during development.

**3.6 Additional Functionality**

Time is left within the project plan to iterate upon the initial plan. Our software is intentionally modular to allow for such augmentations.



**Fig 3.5.1 (Optimistic) Gantt Chart of Project Timeline**

**4. Acknowledgements**