**MetroGnome**

**Software Design Specification**

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

[2.1 System Overview 1](#_Toc167017577)

[2.2 Software Architecture 2](#_Toc167017578)

[2.3 Software Modules 3](#_Toc167017579)

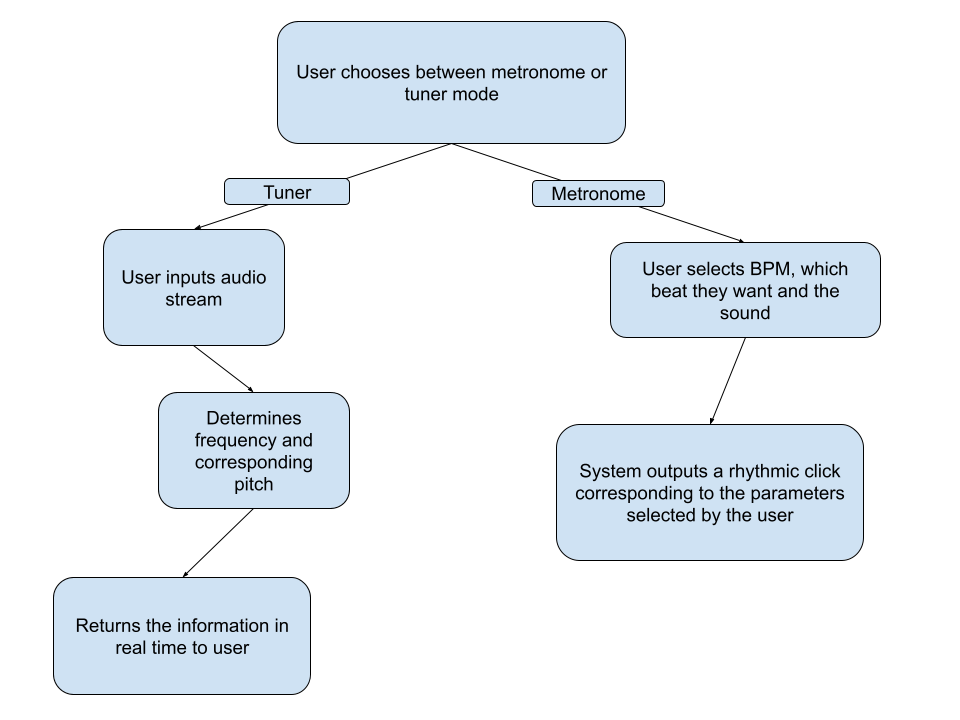
**Revision History**

|  |  |  |
| --- | --- | --- |
| **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 |

**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.

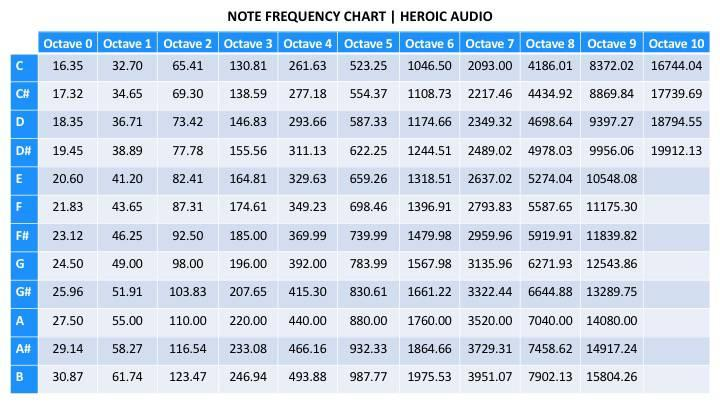
# 2.3 Software Modules

**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**

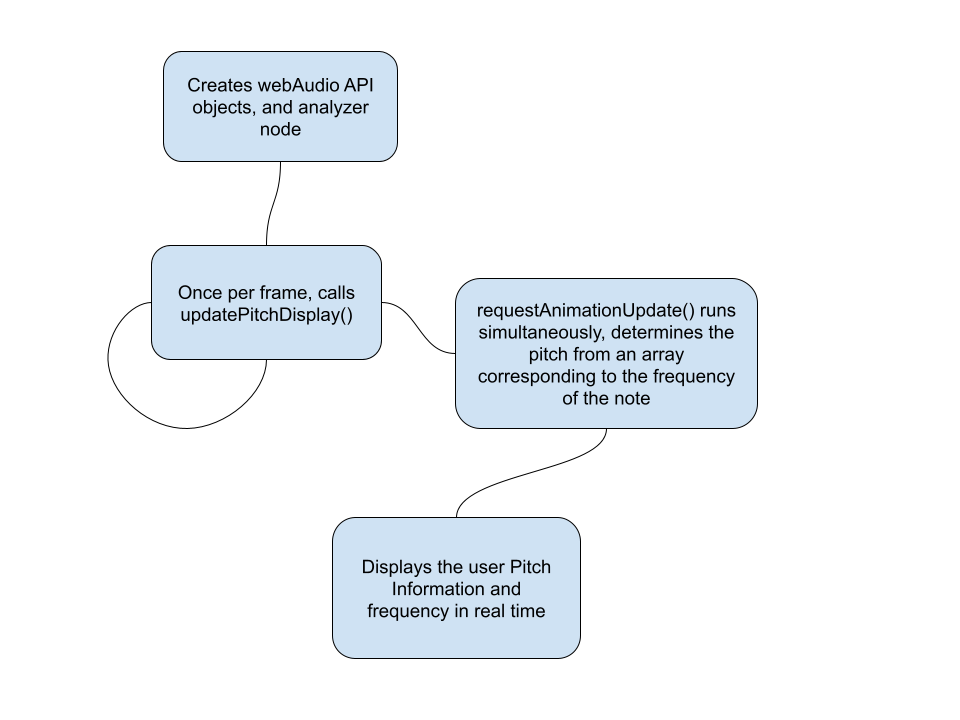
* The program will process a user-input audio stream into numerous arrays, in sequence. The program will then determine the maximum value from the array and display the corresponding frequency, as well as its closest adjacent note.



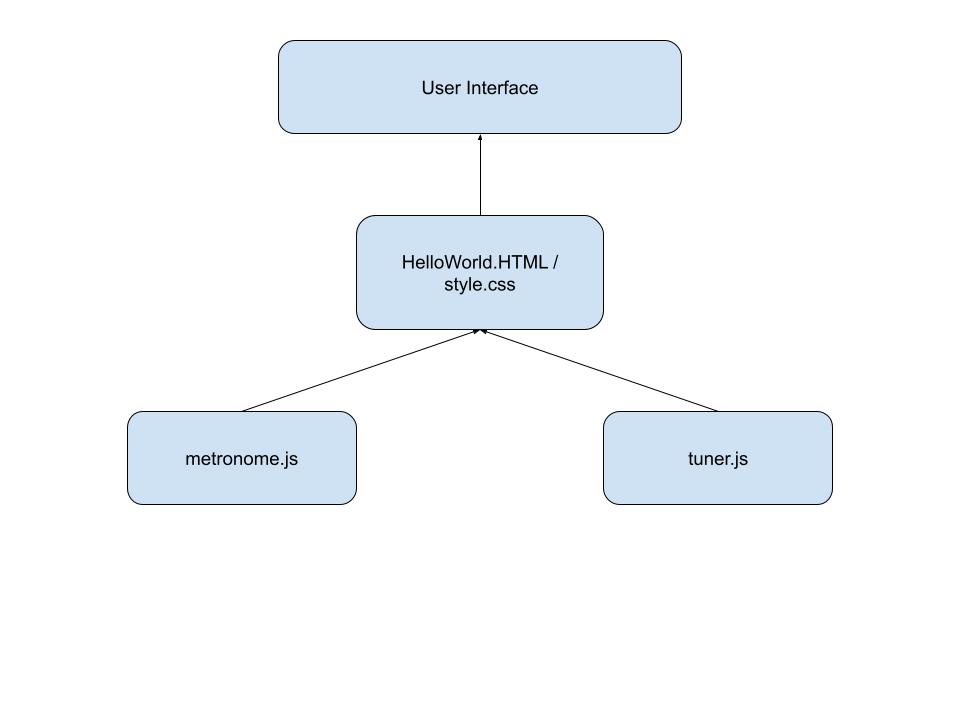
**Fig 2.3.1 Frequency Chart (Heroic Audio, nd.)**

* 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