PROJECT NAME: TunerFish
GROUP MEMBERS: Chris Greenup, Quynh Tran, Gavin Werenczuk
REVISED REQUIREMENTS:

1. UI
    1.1. Button for each of the application function
2. Tuning
    2.1. Tuner will have a visual indicator
        2.1.1. The indicator will be animated
        2.1.2. Animation shows sharpness or flatness
        2.1.3. Animation will give a visual indication of how much the instrument is in tune
    2.2. Tuner will have a digital indicator
        2.2.1. Indicator will show how sharp or flat the instrument is
    2.3. Optional displays
        2.3.1. Instrument select
            2.3.1.1. There will be a menu to select the different types of displays
                2.3.1.1.1. Have an instruments option for
                    2.3.1.1.1.1. Guitar
                    2.3.1.1.1.2. Ukulele
                    2.3.1.1.1.3. Bass Guitar
                2.3.1.1.2. Have an option for the default tuner
        2.3.2. A separate menu for selecting predefined tunings
        2.3.3. A drop-down menu under each string to make custom tunings
        2.3.4. Display will show strings corresponding to selected instrument
            2.3.4.1. When tuning, only the closest note will light up
3. Metronome
    3.1. Box that allows the number of beats in a measure
        3.1.1. Assumes beats are the same denomination as the tempo given[1]
            3.1.1.1. I. E. 4/4 and ♩= 60 is the same thing as 4/8 and ♪ = 60 is the same thing
    3.2. Box that allows the user to type in the beats per minute (BPM)
    3.3. Activation button
        3.3.1. Play button when inactive
            3.3.1.1. Starts the metronome at beat 1
        3.3.2. Stop button while active
            3.3.2.1. Stops the metronome
    3.4. Animation to show what beat the metronome is on
    3.5. Sound
        3.5.1. Play a sound whenever a beat is hit
        3.5.2. Play a slightly different sound for beat 1
        3.5.3. Should stop playing for:
            3.5.3.1. When the bpm is changed
            3.5.3.2. When the activation button is clicked again

[1] There is no easy way to write this concisely

3.5.3.3.

4. Note Player
    4.1. Menu for note layout
        4.1.1. A menu to select which layout to use
            4.1.1.1. Layouts:
                4.1.1.1.1. Keyboard-like
                4.1.1.1.2. Simple left to right in ascending order
                4.1.1.1.3. Simple left to right in descending order
    4.2. Sustain toggle
        4.2.1. A toggle switch to hold the note clicked until clicked again
        4.2.2. Will show it's state visually
        4.2.3. Sustains the note until
            4.2.3.1. it is turned off
            4.2.3.2. The same note is clicked again
            4.2.3.3. The user clicks on a different tool in the application
    4.3. Buttons
        4.3.1. When clicked, note displayed on button will play
            4.3.1.1. If sustain is off, it stops playing when click is released
            4.3.1.2. If sustain is on, keep playing note
                4.3.1.2.1. If the same note is clicked again, it will stop playing
                4.3.1.2.2. If a different note is clicked,that note will play and the old note will stop playing
    4.4. Sound:
        4.4.1. Plays a pure sine wave at the frequency of whatever note is selected
        4.4.2. Supports octaves 2 through 7
            4.4.2.1. These octaves correspond to the octaves on a piano
        4.4.3.
5. History
    5.1. Buttons
        5.1.1. One for clearing the history for that day
        5.1.2. One for clearing all note history
    5.2. History
        5.2.1. Button to advance date forward
        5.2.2. Button to reverse date backward
        5.2.3. Date selected will be shown
        5.2.4. Scrollable wall to see what notes where played and the average it was off by
        5.2.5. Display information should say how many cents off
    5.3. Record
        5.3.1. Keeps track of notes tuned
        5.3.2. Keeps track of the average tuning per note
            5.3.2.1. If a note is held, and at any point is no longer heard, that will be the end of that note's history

5.3.2.2. If the same note is then played after, that will be recorded as a new note with its own history entry and average tuning

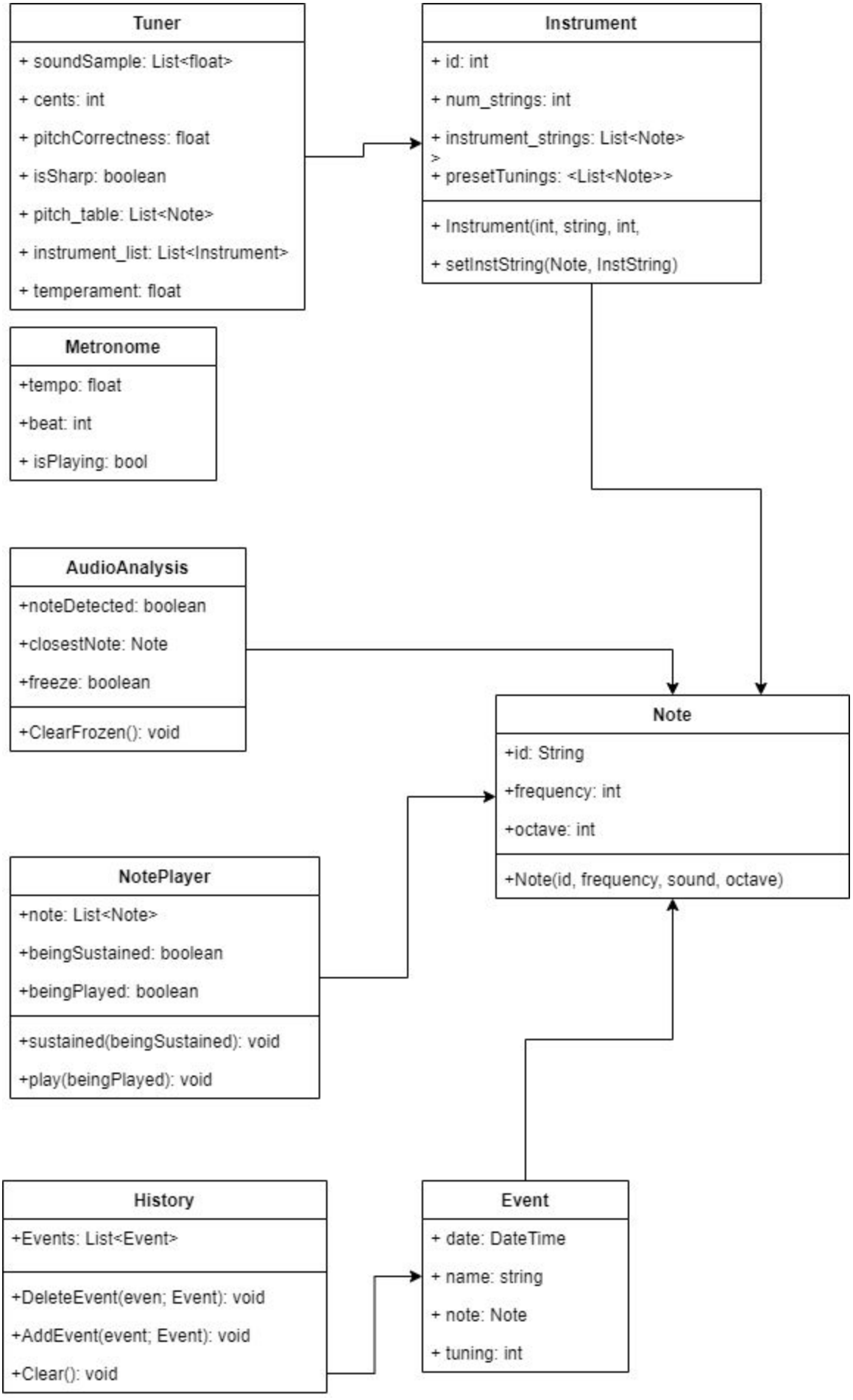
5.3.3. Keeps track of notes averages per note, not as a whole

5.3.4.

6. Audio Analysis
   6.1. A simple display showing the result of the FFT on a graph in real time
      6.1.1. Will process all of the sounds it can hear with the microphone
         6.1.1.1. Does not need anything form of trigger
   6.2. Buttons:
      6.2.1. A button to freeze the current graph of the FFT
         6.2.1.1. The frozen graph should be translucent and be displayed on top of the real time graphs
         6.2.1.2. Only one frozen graph should be displayed at a time
         6.2.1.3. The frozen graph will be a different color than the real time graph
      6.2.2. A button to clear the frozen graphs from the screen.
   6.3. A vertical bar to easily locate loudest pitch
      6.3.1. This indicator bar should be overlaid on the graph
      6.3.2. The note that this closest to should be displayed
   6.4. Detection
      6.4.1. Will detect whatever note is loudest above the threshold
   6.5. Visual
      6.5.1. Graph will be a XY plot, with 0 hz on the left, and ascending

DESIGN DESCRIPTION:

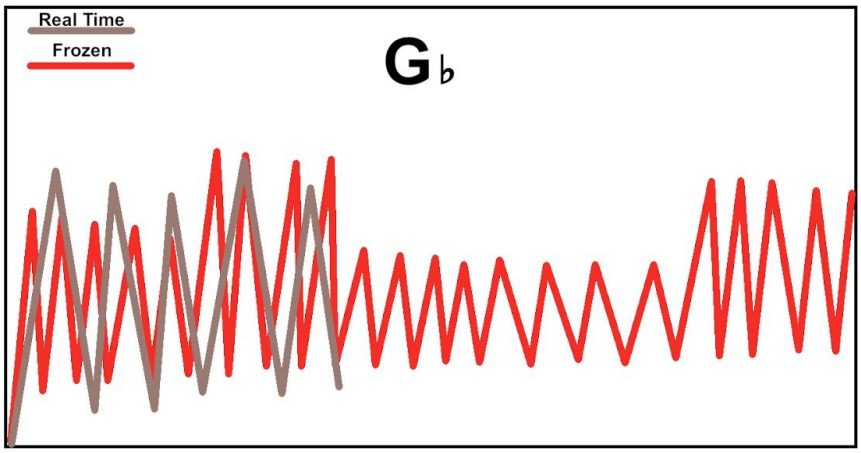
1. **~~~**
2. **APPENDIX-CLASS/MODULE DIAGRAM:**


Tuner
+ soundSample: List<float>
+ cents: int
+ pitchCorrectness: float
+ isSharp: boolean
+ pitch_table: List<Note>
+ instrument_list: List<Instrument>
+ temperament: float
Instrument
+ id: int
+ num_strings: int
+ instrument_strings: List<Note>
>
+ presetTunings: <List<Note>>
+ Instrument(int, string, int,
+ setInstString(Note, InstString)
Metronome
+tempo: float
+beat: int
+ isPlaying: bool
AudioAnalysis
+noteDetected: boolean
+closestNote: Note
+freeze: boolean
+ClearFrozen(): void
Note
+id: String
+frequency: int
+octave: int
+Note(id, frequency, sound, octave)
NotePlayer
+note: List<Note>
+beingSustained: boolean
+beingPlayed: boolean
+sustained(beingSustained): void
+play(beingPlayed): void
History
+Events: List<Event>
+DeleteEvent(even; Event): void
+AddEvent(event; Event): void
+Clear(): void
Event
+ date: DateTime
+ name: string
+ note: Note
+ tuning: int

- Tuner Class:
    - Properties:
        - soundSample: List<float>
        - Cents: int (what cent is the note on?)
        - pitchCorrectness: float (how correct is the pitch?)
        - Pitch_table: List<Note> (list of pitches to compare with the current pitch)
        - Instrument_list: List<Instrument> (List of standard instruments that is used)

3. VISUAL DESIGN:

TunerFish



Freeze

Clear Frozen

History	Note Player	**Tuner**	Metronome	Audio Analysis

TunerFish

Sort by:

- [x] Newest Date
- [ ] Oldest Date
- [ ] Custom Date

"MM-DD-YYY"

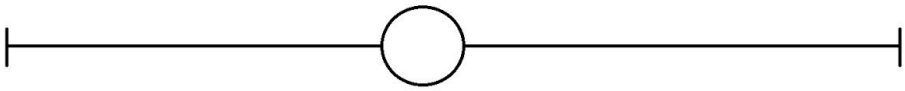
Clear all History

Clear Today's History

Date	Note	Cents Off
May 5, 2021	G	15¢ flat

History	Note Player	**Tuner**	Metronome	Audio Analysis

TunerFish



Tempo

"Enter a Numeral"

Beat

"Enter a Numeral"

History	Note Player	**Tuner**	Metronome	Audio Analysis

- Guitar △
- **Guitar**
- Bass Guitar
- Ukulele

TunerFish

G♭

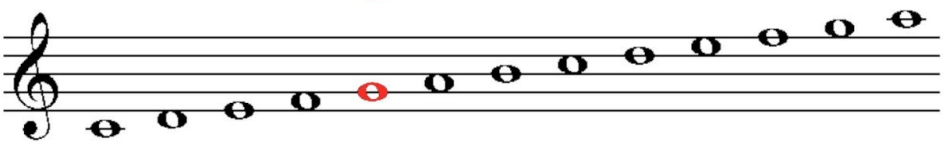
Note | String

History	Note Player	**Tuner**	Metronome	Audio Analysis

TunerFish

Ascending △
Keyboard
Ascending
Descending

C D E F G A B C D E F G A



Loop Note

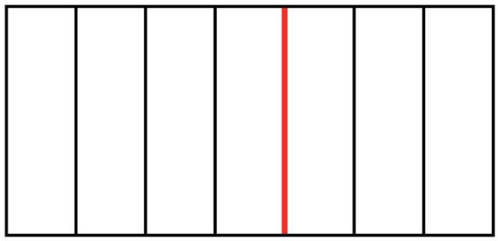
Off **On**

History | Note Player | **Tuner** | Metronome | Audio Analysis

Guitar △
Guitar
Bass Guitar
Ukulele

TunerFish

E A D G B E



Note **String**

History | Note Player | **Tuner** | Metronome | Audio Analysis