Milestones:

1. Create a tuner

The first milestone is to create a simple tuner that is similar to existing solutions. The tuner will accept a played note and output the closest note letter in real time. Additionally, the proximity of the played note to the closest note will be displayed in +/- cents.

2. Research Key Identification Implementations

There are many solutions existing for key identification. Some of the implementations for these could range from highly elaborate to simple to construct. It will take some trial and error to see which of these algorithms provides the functions we need. It would be preferable to modify an existing implementation in C++ or a library rather than construct our own from a whitepaper.

3. Implement Key Identification GUI elements

The GUI component of the Key identification will display a dynamically generated bar graph indicating the proximity to a key. Working real-time in a similar fashion to the tuner. As more notes are played, the ranking of the keys should grow, shrink and change order based on information received from the Key Identification algorithm. The associated notes will display on a piano roll with colors indicating their relationship to the key. The playback functions will include a start, stop and reset button. Lastly, a dropdown box to select the minimum note length may be needed for accuracy.

4. Implement Note Identification Algorithm

Use, modify, or construct from reference an algorithm that accepts MIDI and/or audio input (depending on difficulty) and processes it into a proximity rating to a key to be periodically polled by the GUI. Accepts start, stop and reset signals from the GUI.

5. Animate GUI according To Key Identification

Create real time links according to the identified key. The bar graph for key identification should dynamically grow with changes in Key proximity. The piano roll will highlight played notes with one color, predicted notes of the key in another, and notes played out of key in a third color.

6. Touch ups

Do edge case testing, refactoring, implement any identified stretch goals if time permits. Clean up UX.

Timeline

| Milestone No. | Task | Task Start Date | Task Finish Date |
|---|---|---|---|
| **1** | **Create a tuner** | **10-18-2022** | **11-18-2022** |
| | Create an animated box displaying the currently played note letter | 10-18-2022 | 11-16-2022 |
| | Design a general layout for GUI elements | 9-18-2022 | 9-22-2022 |
| | Research a way to identify a fundamental frequency from a Fast Fourier Transform output. | 10-21-2022 | 11-01-2022 |
| | Implement logic to assign notes to a frequency | 11-02-2022 | 11--11-2022 |
| | Develop real time link between GUI and DSP subsystem for note identification | 11-12-2022 | 11-18-2022 |
| **2** | **Research Key Identification Implementations** | **11-19-2022** | **12-10-2022** |
| | Investigate existing implementations of key identification algorithms for reference. | 11-19-2022 | 12-10-2022 |
| | Accumulate audio and midi data to test accuracy and edge cases for key identification algorithm | 12-01-22 | 12-10-2022 |
| **3** | **Implement Key Identification GUI elements** | **12-11-2022** | **1-15-2023** |
| | Create piano roll | 12-11-2022 | 12-14-202 |
| | Implement note highlighting for piano roll | 12-15-2022 | 12-17-2022 |

| | | | |
|---|---|---|---|
| | Develop a static box indicating the identified key ranking | 12-18-2022 | 12-20-2022 |
| | Create a dropdown box for note quantization interval | 12-29-2022 | 12-31-2022 |
| | Create a start/stop and reset button | 1-02-2023 | 1-15-2023 |
| **4** | **Implement Note Identification Algorithm** | **1-15-2023** | **2-15-2023** |
| | Create handling to store midi notes | 1-15-2023 | 1-22-2023 |
| | Implement algorithm for key identification | 1-22-2023 | 2-15-2023 |
| **5** | **Animate GUI According to Note Identification** | 2-15-2023 | 3-15-2023 |
| | Animate the static box for key ranking | 2-15-2023 | 3-1-2023 |
| | Develop real time link between GUI and DSP subsystem for key identification | 3-1-2023 | 3-7-2023 |
| | Develop real time link between GUI  DSP subsystem for handling start and stop signals | 3-7-2023 | 3-14-2023 |
| **6** | **Touch Ups** | **3-14-2023** | **4-10-2023** |
| | Edge case handling/testing | 3-14-2023 | 3-20-2023 |
| | Implement Stretch Goals (If applicable) | 3-21-2023 | 4-10-2023 |
| | Refactor | 3-14-2023 | 4-10-2023 |

Effort Matrix:

| Task | Effort (Hours) | Assigned To: |
|---|---|---|
| Create an animated box displaying the currently played note letter | 2 | Joe |
| Design a general layout for GUI elements | 2 | All |
| Research a way to identify a fundamental frequency from a Fast Fourier Transform output. | 1 | Klayton |
| Implement logic to assign notes to a frequency | 3 | Klayton |
| Develop real time link between GUI and DSP subsystem for note identification | 8 | Klayton |
| Investigate existing implementations of key identification algorithms for reference. | 18 | JP, Klayton |
| Accumulate audio and midi data to test accuracy and edge cases for key identification algorithm | 2 | Joe |
| Create piano roll | 2 | Joe |
| Implement note highlighting for piano roll | 8 | Joe |
| Develop a static box indicating the identified key ranking | 4 | Joe |
| Create a dropdown box for note quantization interval | 1 | Joe |
| Create a start/stop and reset button | 1 | Joe |

| | | |
|---|---|---|
| Create handling to store midi notes | 8 | Klayton |
| Implement algorithm for key identification | 18 | JP, Klayton |
| Animate the static box for key ranking | 3 | Joe |
| Develop real time link between GUI and DSP subsystem for key identification | 6 | Klayton |
| Develop real time link between GUI  DSP subsystem for handling start and stop signals | 3 | Klayton |
| Edge case handling/testing | 4 | All |
| Implement Stretch Goals (If applicable) | 4 | All |
| Refactor | 4 | All |