Music Master

A musician's practice buddy.

Paul Lee

Alexis Garcia

Alex Gerulis

Nathaniel Trujillo

Amari West

Problem Statement

- There is a **lack** of options for beginner musicians to practice their craft in a meaningful way that directly correlates to their **unique skill level.**
- 151,300 jobs in the field as of 2021

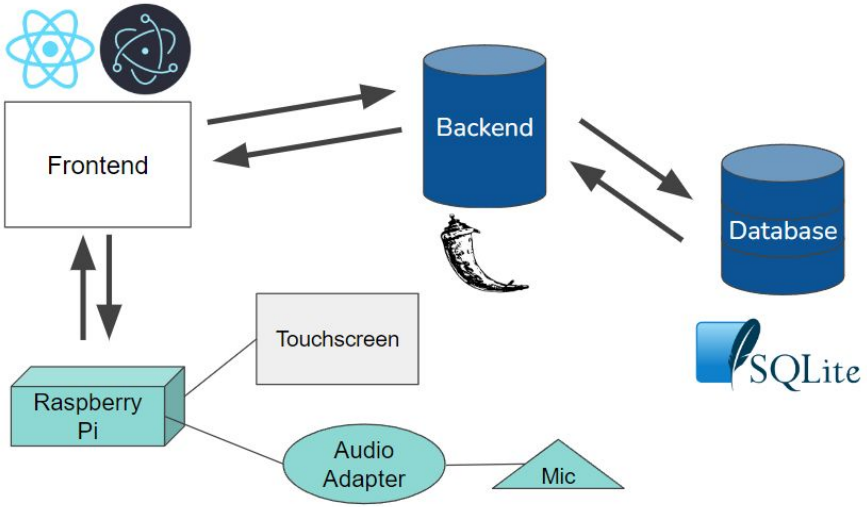
Requirements

- **Records and receives** musician live performance
  - Notes, Tempo, Pitch
- Gives **accurate and precise** feedback
  - Missed Notes, Tempo Accuracy, and Pitch Accuracy
- Provides the user with **comprehensive** feedback

Design Alternatives

- Digital Metronome
  - Can tick indefinitely and provides the user with a tempo to follow
  - Doesn’t provide the user with immediate feedback
- Basic Tuner App
  - Provides the user with the pitch they are playing
  - Doesn’t provide the user with tempo feedback
- Audio Recording Player
  - Let’s the user playback their recording
  - Doesn’t give the users direct and easy to see feedback
    - User’s have to listen and come up with their own feedback points

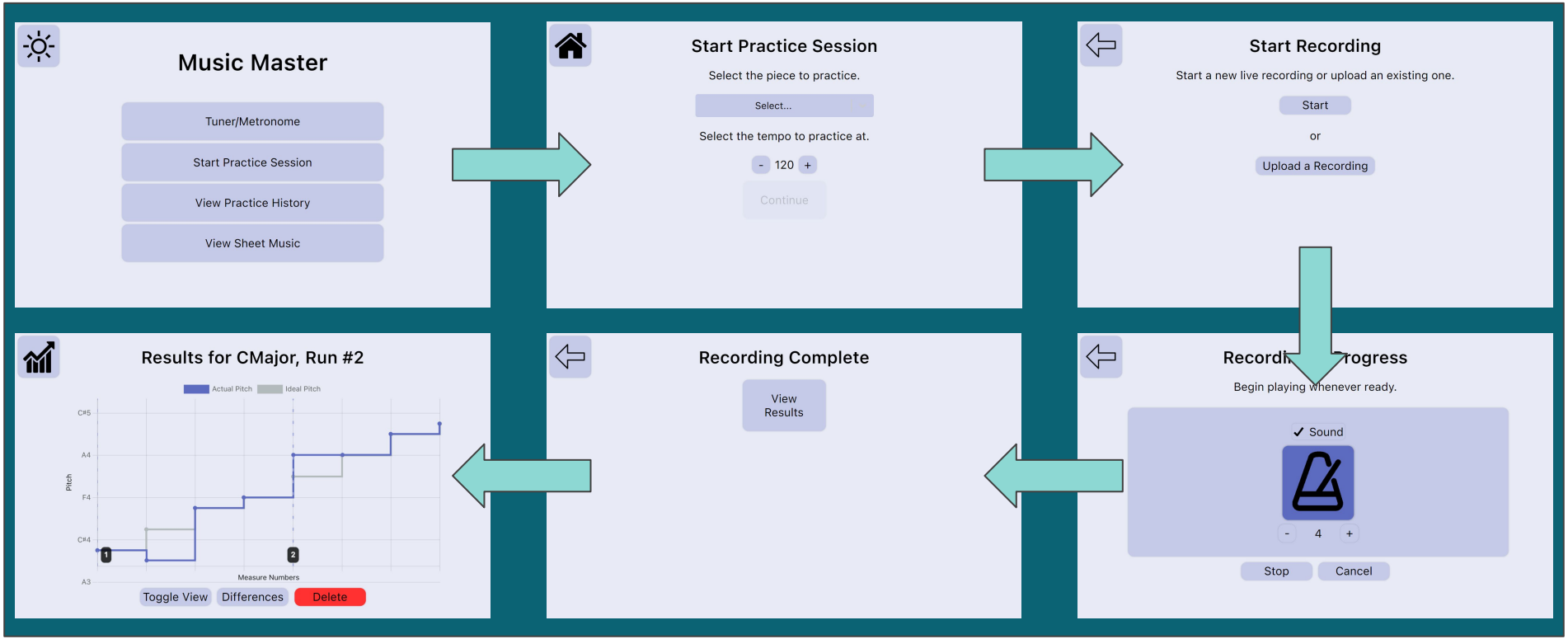
System-Level Design



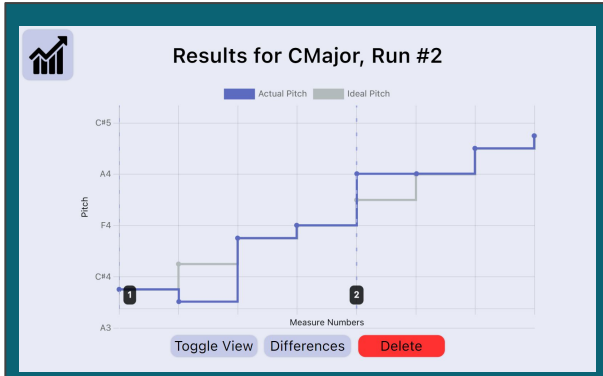
Component-Level Design: Hardware

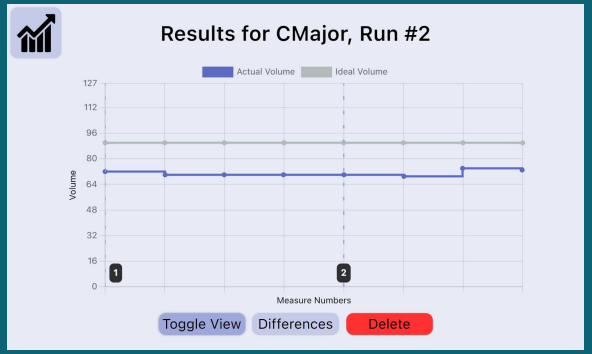
- Raspberry Pi
  - Main hardware component for this project
  - Perform all of the computational tasks
- 7 inch Touch-Screen Device
  - Display and interaction
- Lavalier Clip-on Microphone
  - Record audio from user at decent quality
- USB Audio Adapter
  - Required for microphone compatibility
- 3D Printed Case
  - House the unit without exposing internals

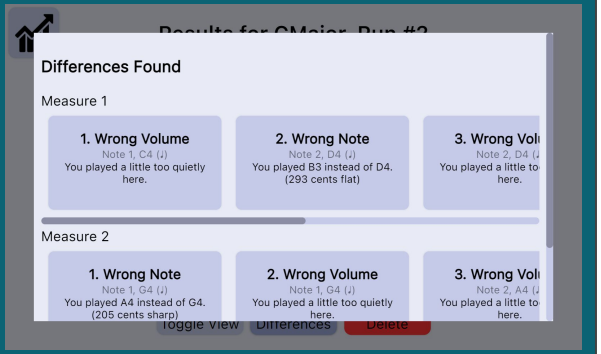
Component-Level Design: Frontend

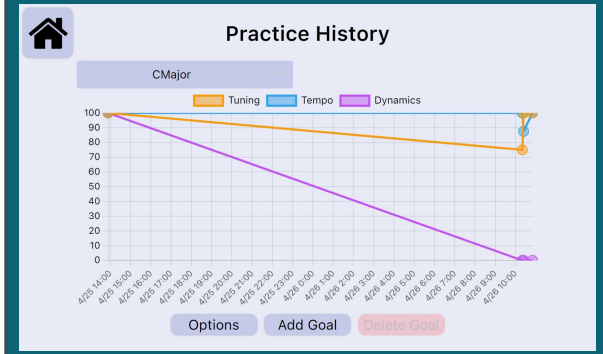


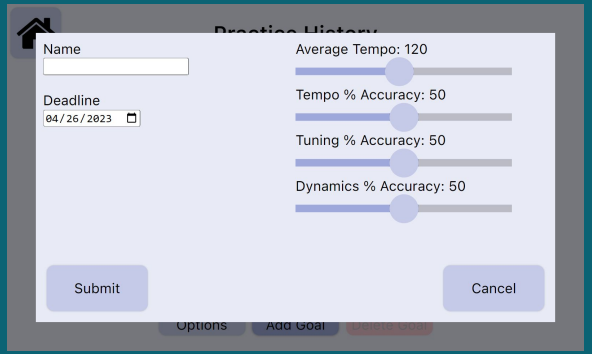
Component-Level Design: Frontend (cont.)

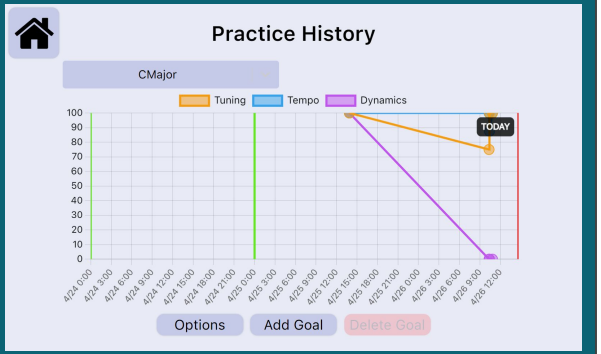




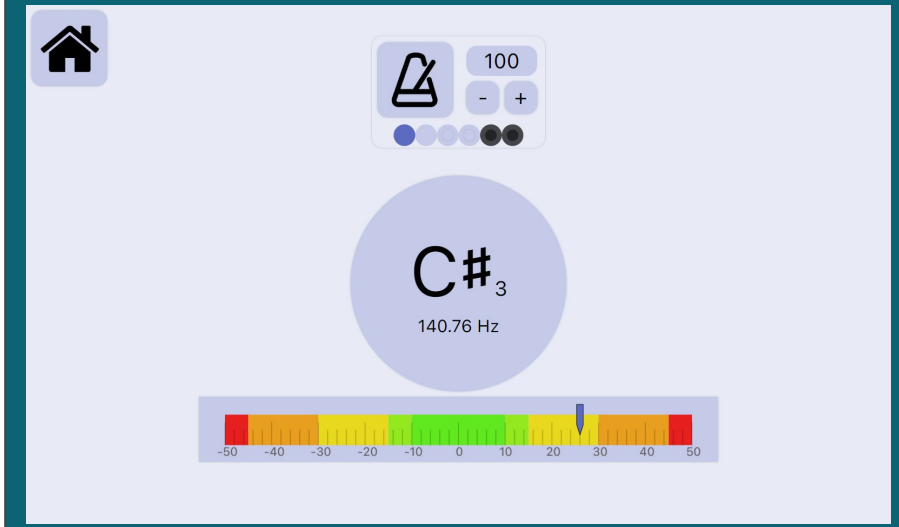


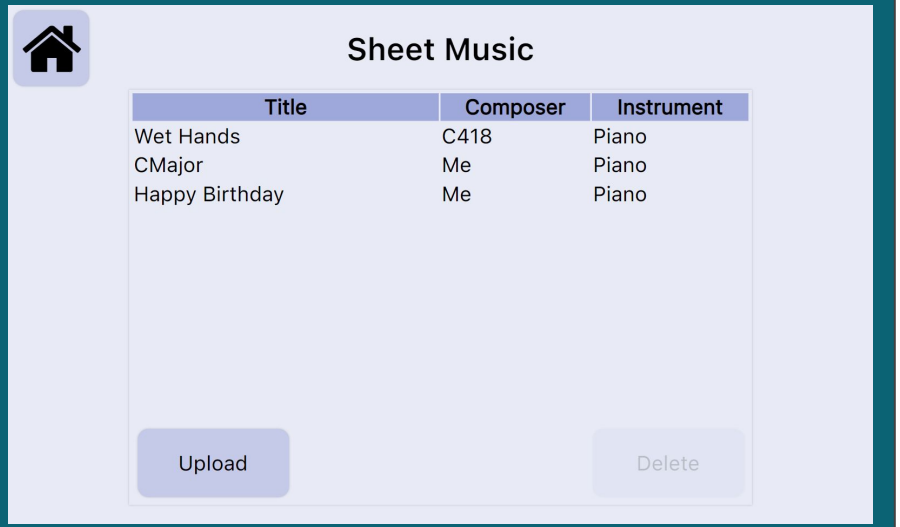






Component-Level Design: Frontend (cont.)





Component-Level Design: Backend (API)

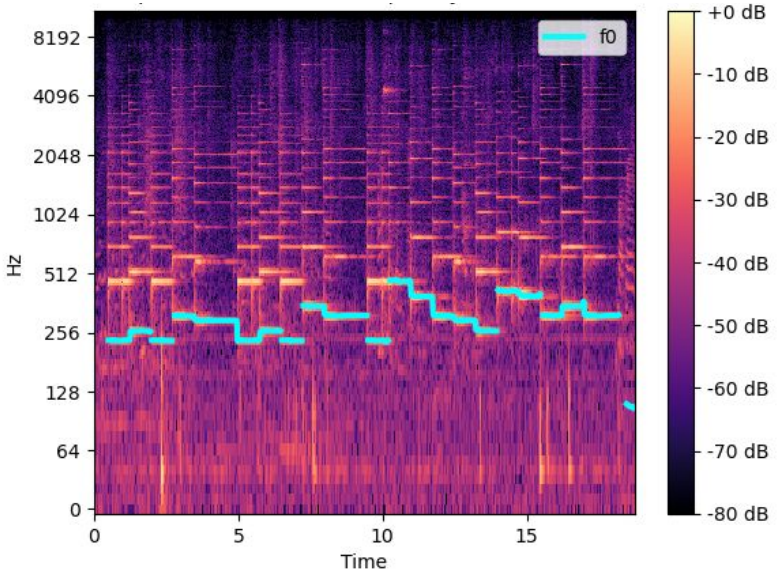
- **/goal**
  - POST - create a new goal and add it to the database
  - GET - read all user-created goals from the database
- /goal/<id>
  - GET - get matching goal from database
  - DELETE - delete a goal from the database

- **/performance**
  - POST - create a new performance and add it to the database
  - GET - read all performances from the database
- /performance/<id>
  - GET - get matching performance from database
  - DELETE - delete a performance from the database
- /performance/<id>/notes
  - GET - get expected and actual notes arrays for performance
- /performance/diff/<sheet_music_id>/<run_number>
  - GET - get difference file for performance

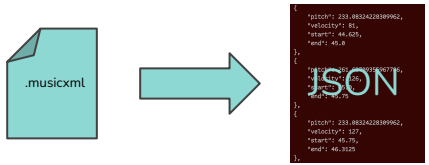
- **/sheetmusic**
  - POST - add a new sheet music to the database
  - GET - read all sheet music pieces from the database
- /sheetmusic/<id>
  - GET - get matching sheet music from database
  - DELETE - delete a sheet music piece from the database

Backend (Signal Processing)

- Uses the YIN algorithm to find the frequencies in a WAV recording
  - In the Librosa library
  - Obtained the following from the WAV:
    - Pitch (Frequency)
    - Velocity (Loudness)
    - Start Time
    - End Time
- Sends the notes found to the comparison algorithm in a JSON format



Backend (MusicXML Reader)



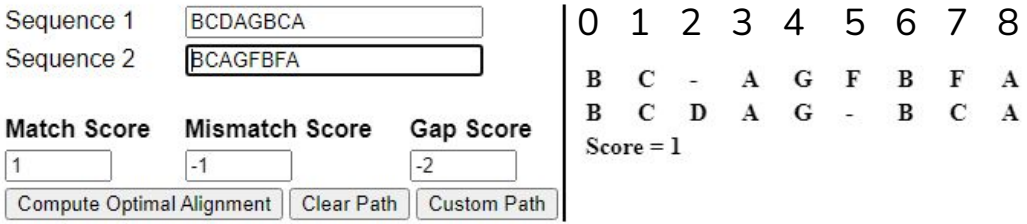
- Object that stores .musicxml files (pieces) and parses them into their individual components.
  - Individual notes
    - Pitch
    - Velocity
    - Start Time
    - End Time
  - Measure numbers
  - Tempo

```python
def get_notes(self, part_index=0):
    # Get a list of notes for the specified instrument
    notes_list = []
    notes = self.pretty_midi.instruments[part_index].notes
    for note in notes:
        notes_list.append({
            "pitch": pitch.Pitch(midi=note.pitch).frequency * # use nameWithOctave for A4
                (SEMITONE_RATIO**FREQUENCY_OFFSETS.get(self.instrument)), # apply instrument pitch offset
            "velocity": note.velocity,
            "start": note.start,
            "end": note.end
        })
    return notes_list
```

- Stores and separates notes played by different instruments
- Sends json data containing all of the note information to the comparison algorithm.

Backend (Comparison)

Needleman-Wunsch
Global Alignment Algorithm

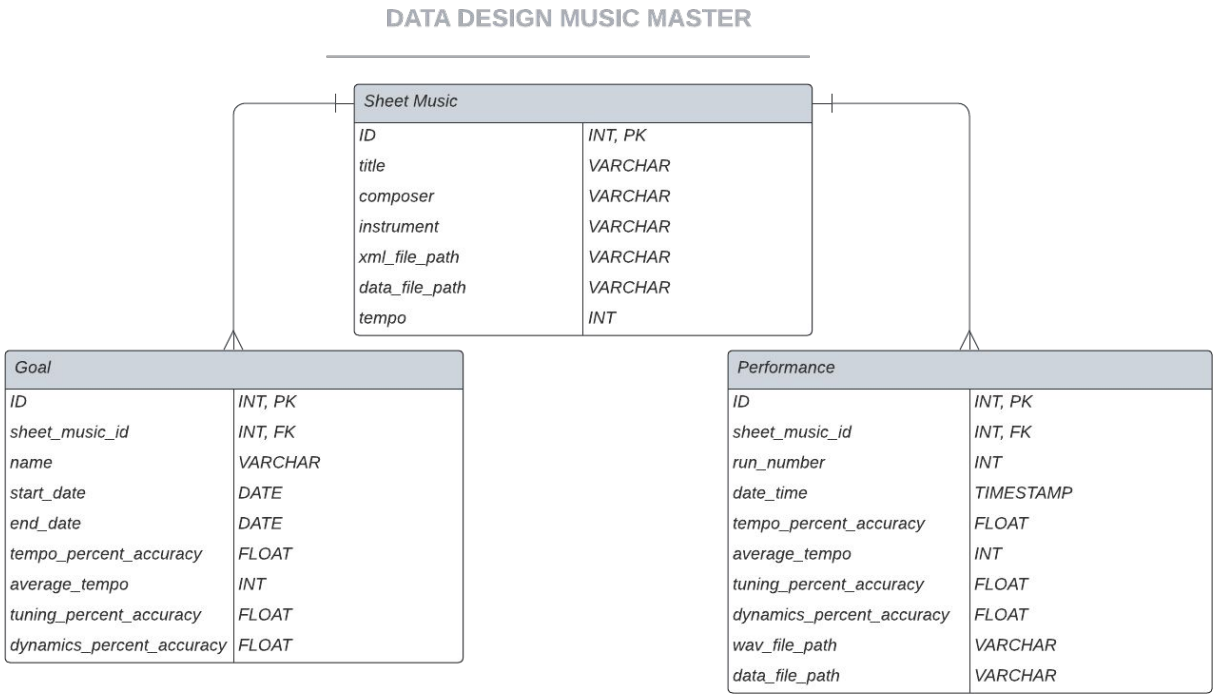


Equality Function Between
Note Objects

- **Pitch** in Hz (+/- 50 cents) 70%
- **Start Time** in seconds (+/- 0.125 s) 20%
- **End Time** in seconds(+/- 0.25 s) 5 %
- **Velocity** in MIDI velocity units (+/- 20 Units) 5%


```
"notes": [
    {
        "pitch": 261.6255653005985,
        "velocity": 90,
        "start": 0.0,
        "end": 0.5
    },
    {
        "pitch": 293.66476791740746,
        "velocity": 90,
        "start": 0.5,
        "end": 1.0
    },
    {
        "pitch": 329.62755691286986,
        "velocity": 90,
        "start": 1.0,
        "end": 1.5
    },
    {
        "pitch": 349.2282314330038,
        "velocity": 90,
        "start": 1.5,
        "end": 2.0
    },
```

Component-Level Design: Database



Demo

https://drive.google.com/file/d/116AtwQKb_JJZtEbDIEjEX8457wBadWRJ/view?usp=share_link

Test/Validation Results

- Single-note extrapolation algorithm accuracy
  - 100% accuracy with piano master files (up to 80 bpm, eighth notes)
  - 98.5% accuracy with clarinet master files (up to 120 bpm, sixteenth notes)
- Performance grading algorithm accuracy
  - Comparison algorithm highly accurate
- Unable to complete chord extrapolation under time constraints
- Performance issues with Raspberry Pi
  - No issues when running program on PC or Mac

Economic Analysis and Budget

Final components list:

- 4GB Raspberry PI 4B - $55 (pre-owned)
- Touch Screen - $50
- Lavalier Microphone - $40
- USB Audio Adapter - $8 (pre-owned)
- 3D Printed Case - $3 (FEDC funds)

The cost of purchasing all these components is $155 at the moment, but factoring in pre-owned items and FEDC's team allotment, only $90 of the budget was actually spent.

Manufacturability and Sustainability

- Packaged software easily **runs on different platforms** besides a Raspberry Pi
  - Pre-compiled binaries for macOS and Raspberry Pi (ARM64) included with release
- Only a **computer and a decent microphone are required**
- Free **OSS (Open Source Software)** can be contributed to by other developers
- **No long-term sustainability concerns**

Social and Political Concerns

- Employment: **Possibility of reduced demand for human music teachers** due to MusicMaster adoption.
- **Social isolation**: Increased focus on individual practice with MusicMaster could lead to decreased opportunities for group learning and social interaction among musicians.
- **Performance anxiety**: Increased emphasis on achieving technical perfection may contribute to performance anxiety and stress for musicians.
- **Creative limitations**: Musicians may feel constrained by the device's feedback, limiting their willingness to explore unconventional techniques or styles.

Ethical Concerns

- **Human connection loss:** Over-reliance on MusicMaster may diminish mentorship value.
- Intellectual property: Audio recording and data storage **may raise questions about ownership and protection**.
- Accessibility: Despite being **more affordable than traditional music lessons,** MusicMaster may still be out of **reach for low-income individuals**, perpetuating existing inequalities in access to music education.

Team Productivity and Workflow

- Main branch had 320 commits in the finalized source code
- Split work into several different branches (i.e. Main, Backend, Frontend, and other feature branches)
- Organized user stories via Jira
- Held weekly stand-up meetings
- Total of four sprints

- Alexis, Amari, Nathaniel (Backend)
- Alex, Paul (Frontend)

Works Cited

"Musicians and singers : Occupational Outlook Handbook," *U.S. Bureau of Labor Statistics*, 16-Sep-2022. [Online]. Available: https://www.bls.gov/ooh/entertainment-and-sports/musicians-and-singers.htm.[Accessed: 15-Feb-2023].

Questions?