The Music Theory Mentor

Hanna Suzuki

Bedford High School
Bedford, MA, 01730, USA

Abstract

The Music Theory Mentor (MT Mentor) is a web application that automatically generates practice problems with immediate feedback for students studying music theory and preparing for the AP Music Theory exam. Although numerous resources exist for general music theory study, materials specifically designed for the AP Music Theory curriculum are severely limited. Unlike other AP subjects such as Calculus, Physics, and Economics, there is no dedicated AP Music Theory textbook, and only two books provide reliable, AP-aligned practice problems at the time of writing. As a result, students lack sufficient opportunities for systematic training and practice. MT Mentor addresses this issue by producing unlimited practice problems for core AP Music Theory topics that many students struggle with. It is designed to support repeated, hands-on practice, which is essential for mastering theoretical concepts.

X.1 Introduction

Music theory is the study of the fundamental elements and structures of music. It serves as a set of principles, guidelines, and conventions used to analyze how musical elements like notes, keys, scales, melody, rhythm, dynamics, and timbre work together to create and organize music.

Music theory is a core and mandatory requirement for almost all college music majors. For example, New England Conservatory of Music requires seven music theory courses (four 100-level and three 200-level courses) for all applied/performance majors [1]. Similarly, music theory is often a required component of the curriculum for high school students in conservatory preparatory programs and music-oriented boarding schools. For example, New England Conservatory Preparatory School requires an eight-year sequence of music theory courses (eight year-long music theory courses) for high school students to earn its highest-level certificate [2, 3]. The music theory component in this highest-level certificate is considered equivalent to introductory (100-level) collegiate music theory education [2].

As it is central to music education, music theory is designated as a subject in the Advanced Placement (AP) program, which is administered by the College Board in the US and Canada [3]. The AP program offers college-level introductory curricula and examinations to high school students across a variety of subjects including Calculus, Physics, English, History, foreign languages, Economics, Psychology, art history, and music theory. By obtaining qualifying scores on AP exams, students can earn advanced placement and college course credit. For example, at many state universities, a score of 3 or 4 in the five-point grading scale can often waive one equivalent course, and a score of 5 may waive two courses [5]. In 2025, over three million students took over six million AP exams [6]. Approximately 18,000 students took the AP Music Theory exam [6].

The AP Music Theory exam is one of the most challenging AP exams in that the percentage of students earning scores of 3 or higher is low. In 2025, the percentage was 60%, making it the second lowest after Latin (59%) and tied with Statistics [6]. In the past five years, the percentage has been consistently low between 60% and 62% [6].

AP Music Theory is challenging for many students because its questions emphasize listening, not just reading or problem-solving, and simultaneously assess aural perception, analytical reasoning, written composition, and real-time performance (sight-singing) skills. For example, students are required to identify the rhythm, pitch, and chord progression while listening to a musical piece, compose a bass line for a given melody, and sight-sing a musical score.

This challenge is compounded by the scarcity of resources specific to the AP Music Theory curriculum. Unlike many other AP subjects such as Calculus, Physics, and Economics, there is no dedicated AP Music Theory textbook, and only two books provide reliable, AP-aligned practice problems at the time of writing [7, 8]. Moreover, in 2025, the College Board restricted access to past exam questions, limiting their availability to only the most recent three years. As a result, students lack sufficient opportunities for systematic hands-on training and repetition-based practice.

This project addresses this issue and proposes a web application, called the Music Theory Mentor (MT Mentor), which algorithmically generates unlimited practice problems with immediate feedback. Currently, MT Mentor targets a few types of questions that many students struggle with. By supporting repeated, hands-on practice, which is essential for mastering theoretical concepts, MT Mentor has the potential to lower barriers to ensure success and improve outcomes for students preparing for the AP Music Theory exam.

X.2 Background

This section overviews the format of the AP Music Theory exam and summarizes the questions that many students struggle with.

The exam is divided into two sections: the first section for multiple-choice questions, and the second part for free-response questions. Each section has multiple parts, as described below.

Section 1: Multiple choice, 75 questions (1 hour 20 minutes), 45% of exam score

- **Unit 1: Music Fundamentals 1.** Pitch, major scales, scale degrees, key signature, time signatures, rhythm, meter, tempo, dynamics, and articulation.
- **Unit 2: Music Fundamentals 2.** Minor and other scales, relative and parallel keys, key relationships, intervals, melody, timbre, texture, and transposing instruments.
- **Unit 3: Music Fundamentals 3.** Triads, chord qualities, Roman numerals, chord inversions and figures, and seventh chords.
- **Unit 4: Harmony and Voice Leading 1.** Soprano-bass counterpoint, SATB voice leading, harmonic progression, functional harmony, cadences, and voice leading with seventh chords.
- **Unit 5: Harmony and Voice Leading 2.** Chord functions to melodic phrases for the IV and ii chords, the vi chord, predominant seventh chords, the iii chord, cadential second inversion chords, and other second-inversion chords.
- **Unit 6: Harmony and Voice Leading 3.** Non-chord tones, embellishing tones, motives, motivic transformation, melodic sequence, and harmonic sequence.
- **Unit 7: Harmony and Voice Leading 4.** Tonicization through secondary dominant chords, tonicization through secondary leading tone chords, part-writing of secondary dominant chords, and part-writing of secondary leading tone chords.
- **Unit 8: Modes and Forms.** Modes, phrase relationships, and common formal sections.

Section 2A: Free response (written), 7 questions (1 hour 10 minutes), 45% of exam score

- **Questions 1 and 2:** Melodic dictation
- **Questions 3 and 4:** Harmonic dictation
- **Question 5:** Part-writing from figured bass
- **Question 6:** Part-writing from Roman numerals
- **Question 7:** Composition of a bass line/harmonization of a melody

Section 2B: Free response (sight-singing), 2 questions (10 minutes), 10% of exam score

- **Questions 1 and 2:** Sight-singing of a diatonic melody of about 4 to 8 bars

In the past 11 years (2015–2025), students have consistently struggled with Units 4, 5, and 7 in Section 1, Questions 1 to 4 in Section 2A, and Section 2B [9].

Section 2B is arguably the most challenging and intimidating part for students. It requires students to sight-sing two melodies, one at a time. They are expected to read each melodic line and practice it aloud for 75 seconds. Then, they have an additional 60 seconds to sing and record their performance using any syllable. Students find both questions quite difficult and similar in difficulty. In 2021, only about 10% of students earned the full 9 points on either question [9]. On the first question, approximately 12% of students earned 0 or 1 point; on the second, approximately 20% of students earned 0 or 1 point [9].

Another challenging area is musical dictation in Section 2A (Questions 1 to 4). In Questions 1 and 2, students are asked to listen to a one-part, four-bar melody three or four times and notate it on the answer staff. The clef, key signature, time signature, and the first note are given on the staff. One melody is in a major key and a compound meter, and the other is in a minor key and a simple meter. One melody is written in the treble clef, and the other is in the bass clef. In Questions 3 and 4, students listen to a four-part harmonic excerpt four times and notate its soprano and bass lines on the answer staff. The clef, key signature, time signature, and the first chord are given for each line. Students are also asked to write the Roman numerals that indicate chord functions and inversions. Usually, Question 4 is more difficult than Question 3 because it includes secondary function chords.

In Section 2A, students found the melodic dictation part (Questions 1 and 2) the most difficult in nine of the past 11 years (2015–2025), while the harmonic dictation part (Questions 3 and 4) was the most difficult in two years. For example, in Question 2, only 14% of students earned most or all of the 9 available points in 2025, and 27% scored 0 or 1 point in 2024 [9].

In Section 1, the most challenging multiple-choice questions are on harmony and voice leading, particularly in Units 4, 5, and 7. In Unit 4, the average score was as low as 49% in 2023. Unit 5 was the most difficult multiple-choice unit in 2024, 2022, and 2021, with 35% of students earning zero point (2022). Many students also struggled with Unit 7 in 2025 and 2021, with only 32% of students earning more than half of the available points in 2025.

Given these observations, MT Mentor initially focuses on the sight-singing questions in Section 2B, which is one of the most challenging parts of the exam. In addition, it also covers multiple-choice questions on keys, chords, and cadences.

X.3 The Music Theory Mentor

This section describes the design and implementation of the Music Theory Mentor (MT Mentor). It is developed in Python, using NumPy (a numerical computation library) [10], music21 (a computational musicology library) [11], and Streamlit (a web application development library) [12].

MT Mentor's source code is available in the `code` folder at [13]. This paper focuses on the generator of sight-singing questions, which is implemented in `sight_singing_gen.py`. To run the generator, MuseScore must be installed in advance [14]. MuseScore is a music notation and synthesis application used to display the generated sight-singing scores.

X.3.1 Clef, Time Signature, and Key Choices

In the AP music theory exam, Section 2B has two sight-singing questions. One of them uses a major key with three or fewer accidentals, and the other uses a minor key with three or fewer accidentals. One is written in the treble clef, and the other in the bass clef. One melody is in 4/4 time, and the other in 6/8 time.

Therefore, MT Mentor randomly selects the key, clef, and time signature from these constraints to generate a melody. The key-selection process is described below.

```
from music21 import *
import random

keyLettersList = ["C","G","D","F","B-","E-",
                  "a","e","b","d","g","c"]
keyLetter = random.choice(keyLettersList)
k = key.Key(keyLetter)
```

The list `keyLettersList` contains 12 possible major and minor keys with three or fewer accidentals. Uppercase letters indicate major keys while lowercase letters indicate minor keys. A minus sign indicates a flat. For example, B- means B-flat major. One of the 12 keys is randomly chosen with the `choice()` function, and its corresponding `Key` object is initialized with music21.

With the key selected, MT Mentor determines the key's diatonic scale as follows.

```
if k.mode == "major":
  sc = scale.MajorScale(keyLetter)
else:
  sc = scale.HarmonicMinorScale(keyLetter)
```

The variable `sc` contains a `MajorScale` or `HarmonicMinorScale` object from music21. In AP sight-singing, the harmonic minor scale is consistently used for minor keys.

The clef-selection process works as follows.

```
scalePitchNames = []

randomFloat = random.random()
if randomFloat < 0.5:
    cl = clef.TrebleClef()
    for p in sc.getPitches(keyLetter+"4"):
        scalePitchNames.append(p.nameWithOctave)
else:
    cl = clef.BassClef()
    for p in sc.getPitches(keyLetter+"3"):
        scalePitchNames.append(p.nameWithOctave)
```

The treble or bass clef is chosen randomly, and its corresponding object (`TrebleClef` or `BassClef`) is initialized with muic21. The list `scalePitchNames` contains the pitch names for the selected key and clef; for example, `[“C4”,“D4”,“E4”,...,“C5”]` if the key is C major. MT Mentor uses the fourth octave for the treble clef and the third octave for the bass clef, which aligns with AP sight-singing expectations.

The time signature (4/4 or 6/8) is randomly chosen as follows.

```
randomFloat = random.random()
if randomFloat < 0.5:
    timeSig = "4/4"
else:
    timeSig = "6/8"
```

X.3.2 Rhythm Generation

Based on the selected time signature (4/4 or 6/8), MT Mentor generates the rhythm for a melody. Each melody is four measures long in the AP sight-singing questions. MT Mentor randomly selects one of the pre-defined rhythm patterns for each measure. For example, the following list shows the rhythm patterns used for the first measure when the time signature is 4/4.

```
measureOneFF = [
    [1, 1,   0.5, 0.5, 1],
    [1, 1,   1,   0.5, 0.5],
    [1, 0.5, 0.5, 1, 1],
    [1, 1,   1,   1],
    [1, 1,   0.5, 0.5, 0.5, 0.5],
    [1, 0.5, 0.5, 0.5, 0.5, 1]]

if timeSig == “4/4”:
    m1Rhythm = random.choice(measureOneFF)
    ...
```

Each nested list in `measureOneFF` represents a particular rhythm pattern. In this example, there are six possible rhythms for the first measure. The total number of elements in a rhythm pattern indicates the number of notes in the measure, and each element (number) indicates the note duration measured in quarter-note units. For example, `[1,1,0.5,0.5,1]` means that the first measure has five notes: the first two are quarter notes, followed by two eighth notes and one quarter note. In the above code fragment, the variable **`m1Rhythm`** stores a randomly-selected rhythm pattern for the first measure.

Given a rhythm pattern, MT Mentor creates a score as a **`Part`** object in music21 and assigns the duration of each note accordingly, as follows.

```
melody = stream.Part()
m1 = stream.Measure()
m1.append(cl)
m1.insert(0, meter.TimeSignature(timeSig))
m1.keySignature = k
melody.append(m1)

for index, noteDuration in enumerate(m1Rhythm):
    ...
    newNote = ...
    newNote.quarterLength = noteDuration
    ...
    m1.append(newNote)
```

The variable **`m1`** represents the first measure. The **`for`** loop iterates over the note durations in the chosen rhythm pattern, adds each note to the measure, and configures its duration. The same procedure is repeated for the remaining measures.

Currently, MT Mentor uses six, eleven, eleven, and three different rhythm patterns for the first, second, third, and fourth measures, respectively, when the time signature is 4/4. When the time signature is 6/8, eight, thirteen, thirteen, and three rhythm patterns are used for the first to the fourth measures. Fewer rhythm patterns are used in the last measure to favor longer note durations, reduce rhythmic subdivision, and promote rhythmic stability toward the end of the phrase.

X.3.3 Melodic Line Generation

Given a four-measure rhythmic sequence, MT Mentor assigns a pitch to each note. Since AP sight-singing questions assume diatonic melodies, MT Mentor generates a melodic line with the seven pitch classes in the selected key. For example, in C major, the melody sticks to C, D, E, F, G, A, and B. To emphasize tonal clarity, MT Mentor uses the tonic (e.g., C in C major) for the first and last notes in the melody. The melody starts and ends in the fourth octave when the treble clef is selected and in the third octave when the bass clef is selected. MT Mentor also prioritizes stepwise melodic motion, in which notes often move by steps rather than by large leaps. These constraints for melodic line generation align with AP sight-singing expectations.

MT Mentor implements these constraints with a pitch-to-pitch transition matrix. This matrix is used to generate a melodic line (i.e., a sequence of pitches) as a first-order Markov chain [15, 16]. Currently, MT Mentor uses the following 8x8 transition matrix.

```
import numpy as np
...
P = np.array([
    [0,   0.4,  0.4, 0.2,  0,    0,    0,    0],
    [0.3, 0,    0.4, 0.3,  0,    0,    0,    0],
    [0.1, 0.4,  0,   0.4,  0.1,  0,    0,    0],
    [0,   0.15, 0.4, 0,    0.4,  0.05, 0,    0],
    [0,   0,    0.2, 0.65, 0,    0.1,  0.05, 0],
    [0,   0,    0,   0.3,  0.5,  0,    0.2,  0],
    [0,   0,    0,   0.1,  0.1,  0.3,  0,    0.5],
    [0,   0,    0,   0,    0.6,  0,    0.4,  0]])
```

Each row represents the current pitch (scale degree), and each column represents the probability of transitioning to the next pitch (scale degree) in the melodic line. For example, in C major, the rows correspond to C, D, E, F, G, A, B, and C: scale degrees 1 to 7, plus scale degree 1 (C) in the upper octave. If the current pitch is scale degree 1 (C), the next pitch is scale degree 2 (D), scale degree 3 (E), or scale degree 4 (F) with the probabilities of 40%, 40%, and 20%, respectively. Figure 1 visualizes the transition probabilities from each scale degree. (It focuses on scale degrees 1 to 4 for readability.)

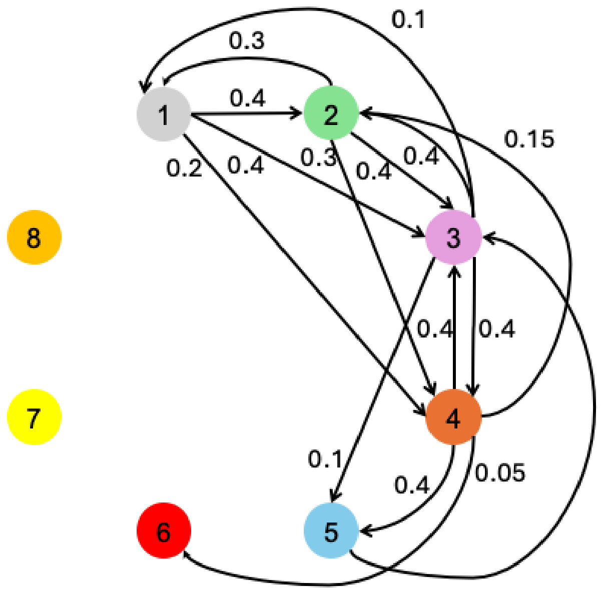

Fig. 1: Transition Probabilities from each Scale Degree

Mathematically, the probability distribution for the next pitch choices is computed with the pitch-to-pitch transition matrix P, as follows:

$$\mathbf{p}_{i+1} = \mathbf{p}_i \mathbf{P}.$$

P_0 denotes the probability distribution for the first pitch in a melody. Since the first pitch is fixed as the tonic in MT Mentor (e.g., C in C major), $\mathbf{p}_0=[1,0,0,0,0,0,0,0]$. Then, the probability distribution for the second pitch is determined as:

$$\mathbf{p}_1 = \mathbf{p}_0\mathbf{P} = [0,\ 0.4,\ 0.4,\ 0.2,\ 0,\ 0,\ 0,\ 0].$$

If the second pitch is sampled as scale degree 3 (e.g., E in C major) based on p_1, the probability distribution for the third pitch is computed as:

$$\mathbf{p}_2 = \mathbf{p}_1\mathbf{P} = [0.1,\ 0.4,\ 0,\ 0.4,\ 0.1,\ 0,\ 0,\ 0],$$

where $\mathbf{p}_1=[0,0,1,0,0,0,0]$. This way, the current pitch depends on the previous pitch, forming a first-order Markov process. MT Mentor implements this process as described below.

```
rng = np.random.default_rng()
...
newNoteSD = rng.choice(["1", "2", "3", "4", "5", "6", "7", "8"],
                       p=P[scalePitchNames.index(prevNote.nameWithOctave)])
newNote = note.Note(scalePitchNames[int(newNoteSD)-1])
```

The variable `rng` represents a NumPy random number generator. Its `choice()` function selects a scale degree based on the probability distribution computed with the previous scale degree. A `Note` object is then initialized using the selected scale degree.

If the interval between the newly created note and the previous note is seven or more semitones, MT Mentor adjusts the octave of the new note with the `adjustOctave()` function to avoid a large melodic leap and maintain singable melodic motion, as shown below.

```
if (newNote.pitch.midi - prevNote.pitch.midi) >= 7:
    newNote.octave -= 1
if (temp.pitch.midi - prevNote.pitch.midi) >= 7:
    newNote.octave += 1
```

Every time `sight_singing_gen.py` is executed, MT Mentor randomly samples a new sight-singing score from its underlying probabilistic model. This randomness enables repeated practice with varied material and supports effective skill development. Figure 2 shows example scores generated by MT Mentor.

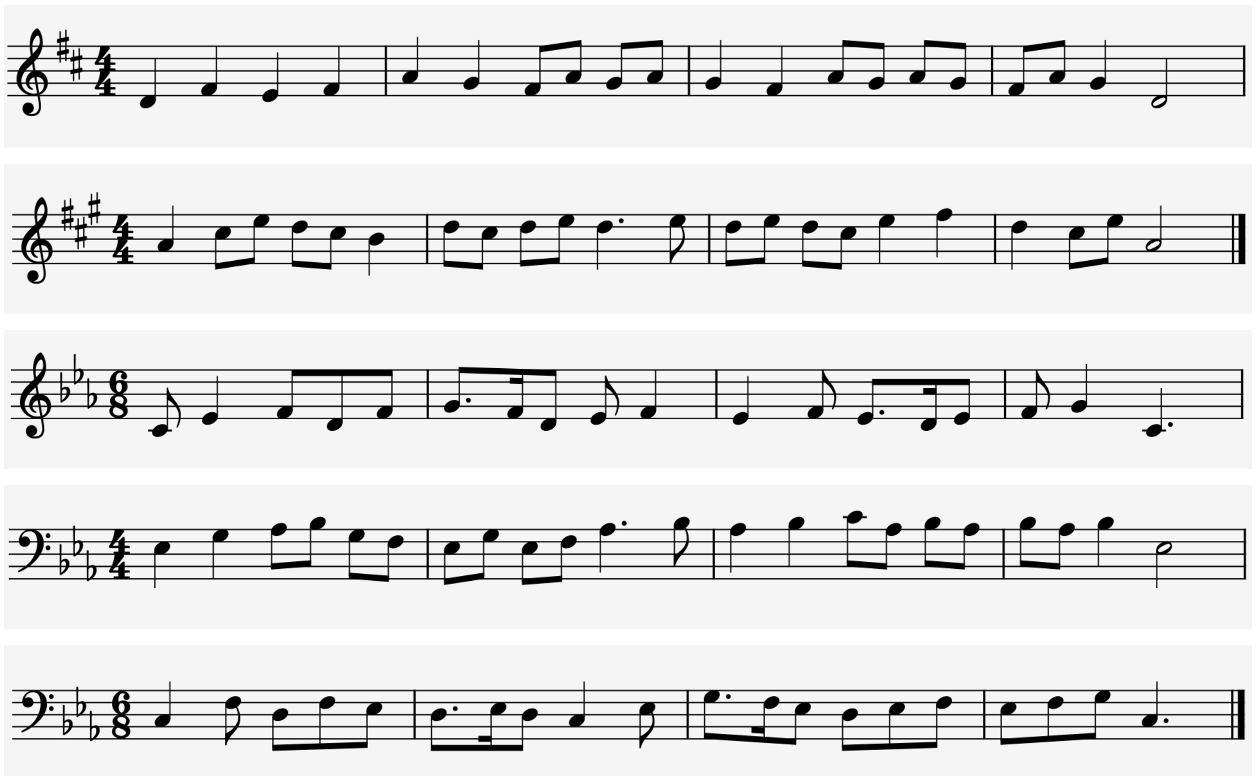

Fig. 2: Example Scores generated for Sight-singing Questions

X.3.4 Webpage Generation for a Local Streamlit Server

MT Mentor can run as a web application as well as a terminal-based application. The following command starts a local Streamlit server and launches a web interface that displays a sight-singing question.

- `streamlit run sight-singing-local.py`

The red part of the code fragment below specifies the path to the MuseScore executable.

```
MUSESCORE_PATH = "/Applications/MuseScore 4.app/Contents/MacOS/mscore"
```

In this example, the path corresponds to the default MuseScore installation folder on MacOS. If MuseScore is installed in a different location, replace the red part with the correct path. For Windows and Linux systems, locate the MuseScore executable and revise the red part accordingly, following each operating system's standard path notation.

Using music21, MT Mentor saves a generated melody in MusicXML format [17] and converts it to a score image (PNG) and musical performance data (MIDI), as shown below.

```
score.write("musicxml.png", fp = "melody-image.png")
score.write("midi",         fp = "melody.mid")
```

MT Mentor then synthesizes a piano performance of the melody by passing the MIDI file to MuseScore and saves the performance as an MP3 audio file. The MP3 file serves as the audio solution for the sight-singing question.

```
subprocess.run(
    [MUSESCORE_PATH, "melody.mid", "-o", "melody.mp3"],
    check=True)
```

Finally, MT Mentor uses Streamlit to generate a web page that displays the score image and its corresponding audio solution (Figure 3).

```
import streamlit as st
from PIL import Image
...
st.title("Section 2B: Sight-singing")
st.image(Image.open("melody-image-1.png"))
st.audio("melody.mp3")
```

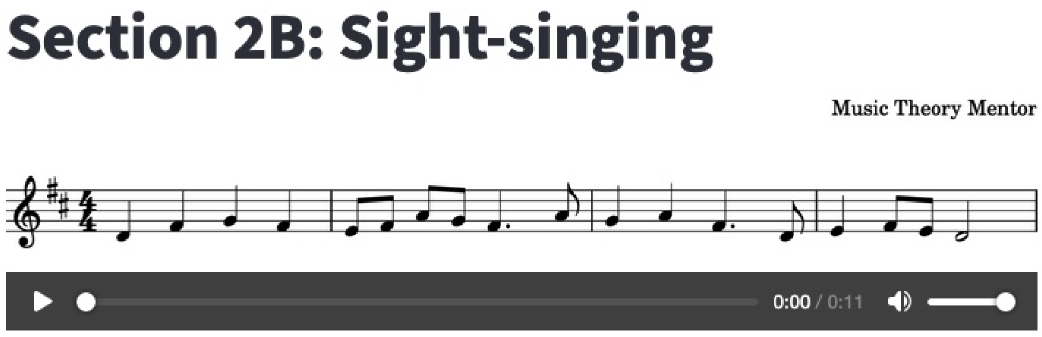

Fig. 3: Web Interface for a Sight-singing Question

X.3.5 Webpage Generation for the Streamlit Community Cloud

MT Mentor can be deployed on the Streamlit Community Cloud [18]. Create a Streamlit account, place `sight-singing.py`, `sight_singing_gen.py`, `packages.txt`, and `requirements.txt` in a GitHub repository, and set up a cloud application at [19] by connecting the repository to the Streamlit account and specifying `sight-singing.py` as the "main file path." A deployed version of the sight-singing question generator is available at **`https://mt-mentor.streamlit.app/`**.

The **`packages.txt`** file lists the external packages (applications and tools) required to run MT Mentor: MuseScore, **`ffmpeg`**, and **`xvfb`**. **`ffmpeg`** is required to generate MP3 files. **`xvfb`** allows MuseScore to run in a headless environment like the Streamlit Community Cloud by simulating a graphical display. The **`requirements.txt`** file lists the required Python libraries: NumPy, music21, and Streamlit.

The programs **`sight-singing.py`** and **`sight-singing-local.py`** implement the same core logic–converting a generated score to an image and synthesizing its audio performance–but differ in their implementation strategies. These differences are due to the older version of MuseScore available on the Streamlit Community Cloud. At the time of writing, the Streamlit Cloud uses MuseScore version 2.4, while the latest available version is 4.6.

What **`sight-singing.py`** does first is to find the name of the MuseScore executable with the function **`findMuseScoreCmd()`**. It should be **`musescore`** (version 2.x), **`musescore3`** (version 3.x), or **`mscore`** (version 4.x), depending on the version of MuseScore on the Streamlit Cloud. The **`xvfb-run`** command is used to run MuseScore as a terminal application without GUI.

Older versions of MuseScore often do not crop generated score images and do not set the background to white, while more recent versions handle both automatically. As a result, a web page displays a large blank area containing only a single bar, which is visually awkward. In addition, a transparent image background makes the score unreadable when a web browser is in dark mode. Therefore, **`sight-singing.py`**, implements image cropping and background whitening in the **`cropHeight()`** function.

Older versions of MuseScore often cannot convert MIDI files directly to MP3, while more recent versions can. Therefore, **`sight-singing.py`** first converts a MIDI file to a WAV file with MuseScore and then converts the WAV file to an MP3 file with **`ffmpeg`** in the function **`midi2mp3()`**.

X.4 Future Work

MT Mentor is planned to enhance and expand its features further. First, the sight-singing question generator will be extended to implement melodic minor practice, which occasionally appears in the second of two sight-singing questions. The generator will also support additional rhythm patterns to increase the diversity of the generated melodic rhythms.

In addition, MT Mentor will explore second- and higher-order Malkov processes beyond the current first-order model. In the second-order process, the pitch (scale degree) of each note depends on the two preceding pitches (scale degrees), using a 56 x 8 pitch-to-pitch transition matrix (Figure 3). In the shaded cell, *$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=1,2)$* means the probability that the next pitch is scale degree 1 given that the two preceding pitches are scale degrees 1 and 2.

SD (Scale Degree)	1	2	3	4	5	6	7	8
1, 1	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=1,1)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=1,1)$						
1, 2	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=1,2)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=1,2)$						
⋮	⋮	⋮						
1, 8	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=1,8)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=1,8)$						
2, 1	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=2,1)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=2,1)$						
⋮	⋮	⋮						
2, 8	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=2,8)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=2,8)$						
3, 1	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=3,1)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=3,1)$						
⋮	⋮	⋮						
3, 8	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=3,8)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=3,8)$						
⋮	⋮	⋮						
7, 1	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=7,1)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=7,1)$						
⋮	⋮	⋮						
7, 8	$P(SD_i=1 \mid SD_{i-2},SD_{i-1}=7,8)$	$P(SD_i=2 \mid SD_{i-2},SD_{i-1}=7,8)$						

Fig. 4: Pitch-to-pitch Transition Matrix for a Second-order Markov Process

Currently, the pitch-to-pitch transition matrix is manually configured. Future work will focus on training it with real musical pieces from music corpora (e.g., [20]).

MT Mentor will support additional types of AP exam questions that students often struggle with; for example, questions related to melodic and harmonic dictation in Section 2A (Questions 1 to 4) and questions on harmonic progressions and cadences in Section 1 (Units 4 and 5). By supporting these areas, MT Mentor will be able to cover a majority of the most challenging topics in the AP Music Theory curriculum.

Another planned enhancement is to log and track practice performance, so learners can clearly identify the areas they need to improve and repeat the questions they missed. Upon repetition, MT Mentor can generate essentially the same questions with slight variations to assess and reinforce proficiency in those areas.

X.5 Conclusion

Music Theory Mentor (MT Mentor) algorithmically generates an unlimited number of practice problems with immediate feedback for students who study music theory and prepare for the AP Music Theory exam. This paper focuses on the design and implementation of the sight-singing question generator in MT Mentor. By supporting repeated, hands-on practice, MT Mentor has the potential to lower barriers to success and improve learning outcomes for students. Its sight singing question generator is available at **`https://mt-mentor.streamlit.app/`**.

References

[1] New England Conservatory of Music, Course Catalog: 2024–2025. https://necmusic.smartcatalogiq.com/en/2024-2025/nec-academic-catalog-2024-2025/

[2] New England Conservatory Prep Certificate Programs. https://necmusic.edu/expanded-education/certificates/

[3] New England Conservatory Prep Classes. https://necmusic.edu/expanded-education/nec-prep/classes/

[4] College Board AP Program. https://ap.collegeboard.org/

[5] College Board, AP Credit Policy Search. https://apstudents.collegeboard.org/getting-credit-placement/search-policies

[6] College Board, AP Score Distributions, May 2025. https://apcentral.collegeboard.org/media/pdf/ap-score-distributions-by-subject-2025.pdf

[7] Nancy F. Scoggin, *AP Music Theory Premium*, Fifth Edition, Barron’s, 2023.

[8] Julie M. Johnson, *Julie Johnson's Guide to AP Music Theory*, Second Edition, Julie Johnson Music, 2019.

[9] Total Registration, AP Exam Score Distribution. https://www.totalregistration.net/AP-Exam-Registration-Service/AP-Exam-Score-Distributions.php

[10] https://numpy.org/

[11] https://www.music21.org/music21docs/

[12] https://streamlit.io/

[13] https://github.com/HSSBoston/music-theory-mentor

[14] https://musescore.org/

[15] P. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*, Wiley, 2017.

[16] A. Ankan and A. Panda, *Hands-On Markov Models with Python*, Packt Publishing, 2018.

[17] https://www.musicxml.com/

[18] https://streamlit.io/cloud

[19] https://share.streamlit.io/

[20] M. Gotham, G. Micchi, N. N. López, and M. Sailor, "When in Rome: A Meta-corpus of Functional Harmony," *Trans Int'l Society for Music Information Retrieval*, 6(1):150–166, 2023.

Biography

Hanna Suzuki is a 10th grader who loves music, tennis, and hanging out with her friends. She co-founded the InnovArt hackathon (**`https://innovart-hack.com`**) and co-chaired its inaugural edition in 2026. As a long-time music student, Hanna is passionate about integrating music, science, and technology–using technology for music when she was younger and using music for science recently. Her early projects included experimenting on the Pythagorean tuning method, monitoring humidity to keep the piano well-conditioned in winter, and visualizing piano sound with LEDs and wearable devices. Currently, Hanna works on sonification research, which uses sound to represent scientific data. Her projects have received a dozen regional, national, and international awards. Hanna studies piano performance, chamber music, and music theory at the New England Conservatory Preparatory School. She has won dozens of prizes at prestigious competitions, including the Elite International Music Competition (Best Performer Award, 2026), Miclot International Music Competition (Grand Prize, 2025), and American Protégé International Piano and Strings Competition (1st place, 2025). She has been invited for recitals at internationally-renowned venues such as Carnegie Hall, Jordan Hall, Schumann House, and Liszt House. Her technical and research work can be found at **`https://github.com/HSSBoston`**, and her LinkedIn profile is at **`https://www.linkedin.com/in/hanna-suzuki-/`**.