User Stories (sprint4)

1. User Story: As a basic user, I want to save music so I can listen to it again.

Discussion: The save button should only be active after the music has finished playing

Testing for this should be repeatable.

How to test it: Generate a scale, write that to a file, play the file. Hear that it plays the same scale.

1. User Story: As a basic user, I want to specify an emotion for the music so that it can induce me to feel a certain way.

Discussion: Emotions are subjective, Make sure all music generated follows rules and specifications derived from a dataset… For example, 50 songs using Xylophones in a ‘spooky’ context that a large percent of listeners perceive as a representation of ‘skeletons’ is stronger data as compared to 1 song using a Xylophone to represent ‘sand’. The problem is that it is subjective, nothing in music is perceived the same by 100% of people.

How to test it: Testing of this will be the testing of the generation that uses the ruleset. Can’t really test the emotional subjectivity.

E ach ruleset must have its own specific test to make sure generation conforms to the rules laid out for it.

1. User Story: As a novice pianist, I want to generate simple songs so that I can practice

Discussion: “Simple” is NOT a ‘pattern’ that users have an option to select; the user selects these specifications from a list of pattern complexity specifications.

UI must have options for rhythms complexity, tempo, and number of voices or instrument selection similar to MuseScore. (MuseScore defaults to just the right hand on the piano, but through user specifications you can add the left hand and even add accompanying instruments.)

How to test it: Simple: one note at a time in one voice, non-fractional (quarter and above, no ties) rhythms, slow tempos (< 80 bpm).

The outputted music should match the specification above.

Test the generated string for conformity to the ruleset, make sure the tempo is below the specified limit.

1. User Story: As a user, I want to ‘generate’ a tune to replay it so that I can reiterate and then save it.

Discussion: Requires new specifications from the user, new specifications are entered into the UI, and the reiterate button is pressed, giving a browse window for file selection

In theory, extends to any midi file. (Not just songs our software generates.)

Possible high level implementation, read in a song, break down into patterns, mutate patterns until they reach new requirements, construct new song out of them

How to test it: Requires the ability to parse a file back into music ‘notation’.

The ability to take a file as an input parameter and produce output with similar ‘data’ as the input. For example, only 10% of specifications changed, like “Same Instruments, Tempo, Harmony, Volume… But slightly different melody.”

1. User Story: As a singer, I want to hear a specific scale so I can practice singing that scale.

Discussion: This is an ‘extra feature’, not part of the cohesive whole of the software.

The user clicks a button for ‘scale’ then the user enters the starting note for the scale and what type of scale… for example major or minor. The program executes a simple algorithm to return the scale requested.

How to test it: The program plays scales to the user’s requested specifications. This is not subjective. For example, the user selects ‘piano middle C major’. There is a correct answer to this. [C D E F G A B C]

1. User Story: As an instrumentalist, I want to be able to tune my instrument.

Discussion: This is an ‘extra feature’, not part of the cohesive whole of the software.

How to test it: The user selects the instrument to be tuned from a list of instruments, and requests a song with one note in it to be generated and saved. If the user plays an instrument that we do not support, for example Shakuhachi, the program will default to piano.

1. User Story: As a user, I can generate chord tones and passing tones in the melody and a chord progression in the harmony in order to generate songs usable in teaching piano.

Discussion: Estimate on simple songs on which is stronger.

Right hand: {1 3 5 3 > 1 2 3 5 > 5 1 2 3 > 1 2 5 4 > 1 4 3 X … > (avoid) 7 2 4 6}

Left hand: {Cmaj}

An absolute rule is that the first note, in 4/4 with right hand melody and left hand harmony, will be a note that is also sounding in the chord in the left hand.

How to test it: (Check this by brute forcing like… 50 songs, to make sure this is true.)

“Stereotypical pop song - left hand chords, right hand melody”, melody has something to do with the chords.

1. User Story: As a user, I want to be able to read in a midi file and change it to meet my goals, so that I can modify music.

Discussion: Read in music, change tempo, key, emotion etc...

How to test it: Must be able to mutate patterns that are given

1. User Story: As a composer, I want to generate many pieces so that I can be inspired.

Discussion: This is an ‘extra feature’, not part of the cohesive whole of the software.

How to test it: Generate anything. Allow for the pieces generated to have a diverse set of potential themes and emotions. ‘Pieces’ are subjective. What inspires this user is also subjective.

1. User Story: As a basic user, I want to transpose a piece I wrote into a different key.

Discussion: This is an ‘extra feature’, not part of the cohesive whole of the software.

Requires the ability to parse a file back into music ‘notation’.

This is NOT subjective.

How to test it: Input: A file containing the user’s piece.

Output: A ‘new’ piece consisting of all the notes in the old piece shifted up or down the same interval; this interval is the one that is requested by the user.

1. User Story: As a basic user, I want to pause the music when I need a break

Discussion: Pause button, definitely want this

How to test it: Self explanatory testing

1. User Story: As a basic user, I want to exit the application when I want to stop

Discussion: Should be able to exit when music is playing

How to test it: Save should complete if user hits save then exit

1. User Story: As a basic user, I want the interface to be as organized as possible for me to operate

Discussion: Limit screens to certain numbers of options,

How to test it: Use screens to split into different categories

1. User Story: As a basic user, I want to hear as many different melodies as possible for a single emotion setting so that I won’t get bored

Discussion: Avoiding repetition

How to test it: Make sure melodies are different enough, patterns should have wide variations

1. User Story: As a user, I would like to generate songs with different melodies, with the same parameters, so that there can be variation listening to multiple songs

Discussion: Avoid hard-coding

How to test it: Self explanatory. Easily testable

1. User Story: As a basic user, I want to combine multiple melodies in a single song, so that I can create music with multiple themes

Discussion: Multi-movement pieces

Maybe different themes?

Does the user input this information?

How to test it: User does NOT input melodies themself, instead picks parameters

1. User Story: As a basic user, I want to accelerate or decelerate the playing speed of music

Discussion: Can be done with a combination of other features

Should it be its own feature though? Seperate from others?

How to test it:: Self explanatory. Easily testable

1. User Story: As a basic user, I want to control the volume of the generating music

Discussion: Important feature

How to test it: Self explanatory. Easily testable