Overall Test Plan

Our tests come from different sources depending on the function of the code. GUI tests will consist of a description of inputs, either programmatic or user provided, and expected visual outputs. The programmer will manually provide the input and verify that the observed output is within the parameters of the written test description Testing for audio classification and manipulation will be done with the included test cases from our chosen key detection library, libkeyfinder. If the included tests with the library function correctly, more advanced testing will be conducted with real instruments providing input with expected key or note outputs..

Test Case Descriptions

GUI Test 1

GUI1.2 This test will ensure that display for the key identification indicates a list of correct keys
GUI1.3 Different mocked inputs for the key identification GUI element will be provided and the desired output should be displayed properly.
GUI1.4 Inputs: All 12 different keys of the circle of fifths will be supplied to the key identification input listener.
GUI1.5 Outputs: Spline will indicate the key on a circle of fifths.
GUI1.6 Normal
GUI1.7 Blackbox
GUI1.8 Functional
GUI1.9 Unit

GUI Test 2

GUI2.2 This test ensures the keyboard GUI displays keys from the key identification algorithm are displayed properly.
GUI2.3 Mocked inputs for each key will be provided to a listener on the keyboard.
GUI2.4 Inputs: All 12 different keys of the circle of fifths will be supplied to the keyboard input listener.
GUI2.5 Outputs: Keyboard should highlight with the correct colors displaying the correct key.
GUI2.6 Normal
GUI2.7 Blackbox
GUI2.8 Functional
GUI2.9 Unit

GUI Test 3

GUI3.2 This test ensures the play/pause button takes audio input to be used for key identification.
GUI3.3 A set of different recordings will be provided and the play/pause button will be toggled manually.
GUI3.4 Inputs: A recording for each of the 12 different keys.
GUI3.5 Outputs: The given audio stream from the input will display on a command line indicating data has been collected.
GUI3.6 Normal
GUI3.7 Whitebox
GUI3.8 Functional
GUI3.9 Unit

GUI Test 4

GUI4.2 This test ensures the quantization drop down displays and manipulates the rate of audio digestion.
GUI4.3 Drop down for quantization options will be clicked and each interval tested on one audio input after the play button is selected.
GUI4.4 Inputs: A recording of only one scale.

GUI4.5 Outputs: The rate at which audio is captured and is manipulated to help in the key identification is displayed on a command line.
GUI4.6 Normal
GUI4.7 Whitebox
GUI4.8 Performance
GUI4.9 Integrational

Key Identification Test 1
KI1.2 This test will ensure the Fast Fourier Transform library is properly integrated.
KI1.3 The libKeyFinder library depends on the FFTW library. Our implementation will use JUCE's FFT library instead. libKeyFinder provides tests for black box testing of other FFT implementations.
KI1.4 Inputs: Program generated audio data in the form of sine waves, with known outputs.
KI1.5 Outputs: Depending on the test, the data is either properly or improperly set and retrieved. The results of real and complex numbers are taken as equal to a sufficient precision.
KI1.6 Normal
KI1.7 Blackbox
KI1.8 Functional
KI1.9 Unit Test
KI1.10 Results:

Key Identification Test 2
KI2.2 This test will ensure the Key Finder library correctly identifies the key of an audio sample.
KI2.3 libKeyFinder provides test data for various stages of the detection process, including downsampling, chromogram generation, and filtering.
KI2.4 Inputs: Program generated audio data in the form of sine waves, with known outputs.
KI2.5 Outputs: An integer value corresponding to an identified key.
KI1.6 Normal
KI1.7 Blackbox
KI1.8 Functional
KI1.9 Unit Test
KI1.10 Results:

Key Identification Test 3
KI3.2 This test ensures that the tuner component functions correctly.
KI3.3 A person will play a musical note into the recording device and observe that the full range of notes are correctly identified by their letter and frequency. The output should be stable, and fast enough to be used for tuning purposes.
KI3.4 Inputs: Sound data from a real instrument. The available notes must be identified from A0 (27.50hz) to G9 (12543.85 Hz), which is the range of the midi standard keyboard.
KI3.5 Outputs: Either a GUI output if available, or debug outputs showing the associated frequency.
KI1.6 Normal
KI1.7 Blackbox
KI1.8 Functional
KI1.9 Unit Test
KI1.10 Results:

Key Identification Test 4
KI4.2 This test ensures the interfaces to the key identifier function correctly

KI4.3 If a button is not available, the start, stop, clear, and sensitivity functions will be activated and data checked for proper function. The start function begins recording data, stop pauses the recording, and clear resets the state of libKeyFinder. The sensitivity function should cycle from 1/32nd notes to ¼ notes.
KI4.4 Inputs: A button press from the user and sound input from the user with a known result.
KI4.5 Outputs: Debug outputs verifying data has been created and cleared in the expected quantities
KI1.6 Edge Case
KI1.7 Blackbox
KI1.8 Functional
KI1.9 Unit Test
KI1.10 Results:

Full System Test 1
FS1.2 This is a test for long term usage and stability in a Audio Workstation environment
FS1.3 The plugin will be left running for an extended period of time (>5 minutes) in a record mode. Different audio settings such as buffer size and sample frequency will be changed during runtime. The program should not output any pops or clicks, and should provide the expected output along the duration. The memory usage should not grow linearly with the running time of the program..
FS1.4 Inputs: Any audio data from a microphone.
FS1.5 Outputs: Debug statements and pop-free sound from the speakers when used in an audio workstation environment alongside other programs.
FS1.6 Abnormal
FS1.7 Whitebox
FS1.8 Performance
FS1.9 Integrational
FS1.10

Full System Test 2
FS2.2 This test is for determining the accuracy of the displayed key on the scale estimator and keyboard.
FS2.3 The plugin will receive live audio after pressing the play button and the accuracy of the key displayed will be evaluated based on what is displayed on the screen.
FS2.4 Input: Live audio from someone playing an instrument.
FS2.5 Output: The proper key is displayed on the scale estimator and keyboard.
FS2.6 Normal
FS2.7 Black box
FS2.8 Functional
FS2.9 Integration test
FS2.10

Full System Test 3
FS3.2 This test is for determining how well the system handles audio that does not fit into the scales handled by the key identification (i.e chromatic scales, non western scales, noise).
FS3.3 The plugin will receive live audio of the unsupported keys and system performance will be monitored.
FS3.4 Inputs: Live audio of unsupported audio.
FS3.5 Outputs: Debug statements of system performance.
FS3.6 Abnormal
FS3.7 Blackbox
FS3.8 Performance
FS3.9 Integrational

FS3.10

Test Case Matrix

	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
GUI1	Normal	Blackbox	Functional	Unit
GUI2	Normal	Blackbox	Functional	Unit
GUI3	Normal	Whitebox	Functional	Unit
GUI4	Normal	Whitebox	Performance	Integrational
KI1	Normal	Blackbox	Functional	Unit
KI2	Normal	Blackbox	Functional	Unit
KI3	Normal	Blackbox	Functional	Unit
KI4	Abnormal	Blackbox	Functional	Unit
FS1	Abnormal	Whitebox	Performance	Integrational
FS2	Normal	Blackbox	Functional	Integrational
FS3	Abnormal	Blackbox	Performance	Integrational

Results

Test	Final Result as of 3/25/2023
GUI1	Pass
GUI2	Pass
GUI3	Pass
GUI4	N/A: Quantization is no longer needed.
KI1	Pass
KI2	Pass
KI3	Pass
KI4	Pass
FS1	Pass

FS2	Pass
FS3	Pass - Unsupported key data receives a lower confidence rating