TABTOMXL

TEST DOCUMENT

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PURPOSE OF THE TEST PLAN DOCUMENT

The purpose of test plan document is to provide details on how testing process will be conducted for a given project. It identifies amongst the test items, the features to be tested, the testing cases implemented, etc. and its intended audience is for the group members, manager and other stakeholders to demonstrate the many tests in the project.

2. GUI TESTING

2.1. TESTED COMPONENTS

Item to Test	Test Description
TestmakeFade()	Tests to see if the fade transition was made
TestBrowse ()	Tests to see if the Browse button prompts the FileChooser
TestConvert ()	Tests the software's ability to upload a file and append it to the textarea
TestHelp ()	Tests to see if the software produces an XML file after the convert button has been pressed
TestLoadScreen ()	Tests to see if the next screen is loaded after Convert button has been pressed
TestErrorMessages ()	Tests if the prompt appears when user makes a mistake
TestSaveChanges ()	Tests if the software is saving the changes.
loadNextScene()	Loads a new application view if input was successful and revert back if new conversion was clicked

2.2. TESTING DERIVATION

The unit tests were derived from the need to check the functionality of the GUI. The testing input is provided by pasting a sample tab into the text area (whether through the file explorer, drag and drop or manually) and performing numerous tests upon it.

2.3. TESTING SUFFICIENCY

These test cases prove to be sufficient as they cover what must be covered, from pasting/editing/drag and drop into the text area to the functionality of the different buttons. Thus this testing is sufficient as it covers most if not all of the features of the GUI and shows that they are performing as they should.

3. GUITAR TAB PARSING/XML TESTING

3.1. TESTED COMPONENTS

Test Name	Test Description
readTab()	Clean and reads the tablature and determine if tab is guitar, bass, or drum
parseGuitarTab()	This function takes the given drum tablature and calls the generateDrumXML before returning the whole XML translation if tablature is valid
generateGuitarXML()	This function is called and generated the XML translation using the variables from the method that called it.
createHammerPulloffHelper()	This function processes hammer and pull offs
stepFromGuitarString()	This function processes the steps of each notes.
createGuitarTuningPitches()	Creates a 2D array of pitches to be used as a standard tuning reference.

3.2. TEST CASES IMPLEMENTED

- Given a valid tablature, it will give the next function a Tab variable that contains an array of all the notes in each line and will receive and return the XML translation
- Given an invalid tablature, it will assert an empty array which will be returned and outputted as an error message.

3.3. TEST DERIVATIONS

This test was derived from the possible inputs this function will receive based on the information of the notes, measures and techniques like hammer ups to check for problems. It is important to make sure that the parser does not encounter issues parsing the various features of a guitar tablature like the different types of playing techniques that could possibly be normally present in a tab. The tablature would have already been checked from a previous function so we know that it is definitely a guitar/bass tab.

3.4. TESTING SUFFICIENCY

The test is able to confirm that this parser is able to parse an entire tablature that contains various features such as pull offs, etc. and does not encounter any error. Also it is

sufficient since the only possible inputs that this function can receive are either a valid or invalid tablature.

4. DRUM TAB PARSING/XML TESTING

4.1. TESTED COMPONENTS

Test Name	Test Description
parseDrumTab(String drumTab)	This function takes the given drum tablature and calls the generateDrumXML before returning the whole XML translation if tablature is valid
generateDrumXML(DrumTab drumTab)	This function is called and generated the XML translation using the variables from the method that called it.

4.2. TEST CASES IMPLEMENTED

- Given a valid tablature, it will give the next function a Tab variable that contains an array of all the notes in each line and will receive and return the XML translation
- Given an invalid tablature, it will assert an empty array which will be returned and outputted as an error message.
- After tablature has been parsed, the music XML file has been inputted into SoundSlice to determine if the parser had any issues that needed to be resolved.

4.3. TEST DERIVATIONS

These cases were derived from the possible inputs the function will get. The tablature would have been checked in a different function so we will know whether the input was a guitar/bass/drum tablature.

4.4 TESTING SUFFICIENCY

These are sufficient since the only possible inputs that this function will receive is either a tablature that is valid (or empty) or an invalid tablature thus these tests cover the possible cases.