# Software Requirements Documents

for

# <TAB2XML_G14>

Prepared by Abdelrahman Altamimi

Hieu Le

Mahdiar Shoraka

Prabjot Dhaliwal

Yongjie Ba

Date: March 6th, 2022

## Table of Contents

**1. Introduction……………………………………………………….………..…..3**

**2. System Requirements.…………………………………..…...……………4**

**a. Functional…………………………………………………………..4**

**b. Non-Functional…………………………………..….………………..6**

**3. Use Cases……..……………………………………….……......………7**

**4. User Stories……..……………………………………....……………..10**

**5. Future Development …………………………………..…………………...11**

# 1. Introduction

TAB2XML is a software tool used to convert text-based tablature files to MusicXML files with its corresponding visual representation in downloadable sheet music, and playable audio of the music itself. Many other music programs can easily use the MusicXML file due to its high compatibility. The visual sheet music feature is currently supported for Guitar, Bass and Drums tablatures.

The intended user can start using the software by selecting the text-based tablature on their computer then the program will output the MusicXML. Before using the software it is necessary for the user to check that the text-based tablature is written in the standard musical notations to ensure that the program successfully converts the file.

This program is mainly aimed to help musicians or any individual who’s looking to translate their tablature file to a music XML file which can easily be played or edited on various music software.

# 2. System Requirements:

## 2.1 Functional:

- The System can convert the tablature txt file to a MusicXML file and the System should support the user to view the converted MusicXML code.
- The System should allow users to save the converted MusicXML code.
- The System should allow users to convert tablature to sheet music:
  - The System should support text guitar/bass tablature to staff-based guitar/bass tabs.
  - The System should support converting text drum tablature to standard musical notation.
- The System should allow users to preview the Sheet Music of the given tablature.
- The System should allow users to export the converted sheet music as a PDF version.
- The System should notify the user of input(By highlighting their input).
  - Gray-highlight: will be ignored.
  - Yellow-highlight: warning.
  - Red-highlight: Error
- The System should support the display of specified measures on the sheet music.
- The System should provide an option for users to display tablature underneath Sheet Music
- The System should support viewing sheet music as one continuous page
- The System should support playing sheet music from the start or from a specified measure.
- The System should support pausing the music playback and resume the playback if paused.
- The play function must allow the user to change the tempo of the music, i.e. how fast or slow it is played back. This must be expressed in beats per minute (bpm). 60 bpm means that there are 60 quarter notes in a minute, i.e. each quarter note lasts one second.
- The System should support three different instruments (Guitar, Bass, Drums).

- The System should support users to customize the display of the music sheet, such as changing the line space, changing the font, changing the note space and changing the table width.
- The previewer should allow the user to see what measure number they are looking at, as well as support a Go-To Measure function as in the text input.
- The System should support repeat versions of the music file
- There must be a connection between the play function and the visual output as follows: The user must be able to select a measure (or a note) and start playback from that point. Also, when the music is playing, the visual output must highlight the note that is currently being played

## 2.2 Non-Functional:

- Reliability:  The System should keep features working correctly
- Reliability: The System should keep no background working when the application is closed. or all activity must be stopped when the window is closed.
- Reliability: The file should be built in order to make cohesion as high as possible.
- Reliability: The System should have a Plan-B if the system crashed or not working
- Reliability: Notify immediately (low latency) when:
    - Invalid input is provided.
    - Prohibited actions are attempted.
    - An error has occurred.
- Security: No user file will be accessed, written, or read unless explicitly done by the user themself.
- Performance: The sheet music should render within 30 seconds given the tablature is of reasonable length.
- Dependencies: Run entirely locally on the user's device without requiring any network connection once installed. The device can be a PC, laptop or Ipad.

# 3. Use Cases:

## Case 1:

**Title:** Show MusicXML

**Primary Actor:** User

**Pre-condition:** valid txt file uploaded or valid music text pasted on the text file of main app GUI

**Post-condition:** the text file is converted to a music XML file, and the user can see it.

**Success Scenario:** The user inputs a text file containing tablature for supported instruments, like a guitar.txt file. The tablature is converted to a MusicXML sheet. The MusicXML sheet is displayed to the user.

**Failed Scenario:** The User inputs a text file containing tablature for unsupported instruments, like piano, the text failed to convert.

## Case 2:

**Title:** Preview sheet music

**Primary Actor:** User

**Pre-condition:** valid txt file uploaded or valid music text pasted on the text file of main app GUI

**Post-condition:** A new window popped out (preViewGUI)

**Success Scenario:** User inputs a text file containing tablature for supported instruments(guitar.txt etc.). The user chooses to view the sheet music. The tablature is converted to sheet music and the sheet music is displayed to the user.

**Failed Scenario:** The user input a text file containing tablature for an unsupported instrument(Piano.txt etc.). The system failed to show the music sheet in the scroll pane. and the user sees an error message on the scroll pane.

## Case 3:

**Title:** Save MusicXML

**Pre-condition:** valid txt file uploaded or valid music text pasted on the text file of main app GUI

**Primary Actor:** User

**Post-condition:** musicXml file is successfully saved

**Success Scenario:** User inputs a text file containing tablature for supported instruments. The user selects to save the converted MusicXML. The user selects the location of the file on their device. The user specifies the name of the score and the author. The MusicXML is saved on the user's device at the specified location.

**Failed Scenario:** The user input a single character in the text field, and the button for saving music XML can not be clicked.

## Case4:

**Title:** Play the music

**Pre-condition:** valid txt file uploaded or valid music text pasted on the text file of main app GUI and music is not playing

**Primary Actor:** User

**Post-condition:** music is playing

**Success Scenario:** User inputs a text file containing tablature for supported instruments. The user selects to click the preview music sheet button. Then the user clicks the play button, and the music starts playing.

**Failed Scenario:** User inputs a text file containing tablature for supported instruments. The user selects to click the preview music sheet button. Then the user clicks the play button, and the music starts playing, but the user wants to click the play button again. but only the pause and stop buttons are displayed. (play button is not visible during the music is playing)

## Case4:

**Title:** Customize display

**Pre-condition:** valid txt file uploaded or valid music text pasted on the text file of main app GUI, preview GUI loaded, and the customize display button clicked.

**Primary Actor:** User

**Post-condition:** Music sheet style changed

**Success Scenario:** User inputs a text file containing tablature for supported instruments. The user selects to click the preview music sheet button. Then the user clicks the customize display button, and then a new design window popped out, the user changes the font, and line space, then clicks the apply&Close button, the sheet style changes correctly and the design window closed.

**Failed Scenario:** User inputs a text file containing tablature for unsupported instruments.  The user selects to click the preview music sheet button. Then the user clicks the customize display button, and then an error message shows out. customize display failed.

**Note:** The “User” can be any human (or programmed bot) as anyone can access music tablatures online. But the most pertinent user will be a music composer.

## 5. <u>User Stories:</u>

- As a *musician*, I want to *play music*, so that *I can listen to the music I just edited.*
- As a *musician*, I want to *print sheet music for my tablature* so that *I can share my sheet music while only having to write tablature which is easier for me.*
- As a *composer*, I want to *listen to specific measures in my music* so that *I can check if my changes to the tablature sound good.*
- As a *student*, I want to *preview sheet music without saving it to my computer* so that *I have a good reference for the sheet music that should be printed by the application I am developing which should convert MusicXML files to sheet music.*
- As a *user*, I want to *play the music*, so that *I can listen to the music I just downloaded.*
- As a *musician*, I want to *save my work*, so that *I can save the file on my computer.*
- As a *musician*, I want to *open the file I just downloaded*, so that *I can convert it to XML format.*
- As an *upcoming guitar player*, I want to *play guitar tabs at slow speeds* so that *I can easily learn how to play new songs.*
- As a *customer*, I want to *preview the sheet music*, so that *I can see the tablature of that .txt format file.*
- As a musician, I want to export the pdf files, so I can use them to play guitar anywhere.

- As a user, I want to customize the display, so I can design a good look music sheet.

## 6. Future Development

### 6.1 Support more Instrument

**For Playing music**

All play features are included in the musicPlayer.java file which is saved in the custom_player package.

- If we want to add a new notification about playing music, only need to add a new notification in the musicPlayer.java class.
- If we want to change the condition for the play, pause or exit feature, only need to change musicPlayer.java class.
- If we want to support new features like playing music from the measure that the user input or the notion user clicked, also the musicPlayer.java is the only class that needs to be changed.
- If we want to support more instruments such as piano, celesta, clavinet and so on, we only need to change musicPlayer.java class.

**Modification of XML Parsing Unit**

- If a new MusicXML element must be supported, simply create a new class ***X*** which has the attributes corresponding to the content/attributes of this element.

- Then, create an object of this element within the corresponding class for the new element's parent.
- Make sure to store this object within the class it is initialized in, so it can be referenced via an object of this class.
- The initialization of the attributes of class ***X*** must be handled by the constructor the class itself which has the single parameter(**doc. Element)** containing the XML element in the tag: <new-supported-element-tag-name>.
- If previously unsupported attributes/contents of music XML elements must be supported, then add these attributes to the corresponding Java class in the custom_component_data package. Make sure to initialize them within the class's constructor.

## Updating the Music Sheet Graphics Generation

- If a new graphical music element must be supported, and this element applies to musical notes, then follow the steps below:
  - Create a new class simply generating the graphical component without any translations. Make sure its constructor has the appropriate arguments and it extends a JavaFX class.
  - Set and initialize relevant data that may be used (such as its width, or its enumerated occurrence within a measure)
  - Accordingly, initialize an object of this note using appropriate parameters within the DisplayNote or DisplayChord class for drums, and the

BoxedText or BoxedChord class for guitars. Then add this object to the display by “this.getChildren().add(element);” for the new element created.

- Translate the element horizontally and horizontally based on its position on the note (The top-left corner of the square around the notehead is the origin (0,0)).
- Modify the relevant attributes of the note (if this new element causes the note to take up a greater width, then increment the width accordingly). Such values can be “spacing type, width, preceding, trailing” which are further documented in the project code.