Requirements Document

TablatureConverter

EECS 2311 – Group 6

Tuan Dau

James Le

Temiloluwa Odunfa

Mohammad Amin Mohammadi

02/28/2021

Table of Contents

1. Introduction

1.1 Purpose of Document

The purpose of this document is to outline the description of the Tablature Converter software, the project's functionality, the technologies being used, some common use cases, and functional requirements that the project must support.

1.2 Background

There are many ways to learn how to play a song using the internet, including following along with video tutorials or finding sheet music online. However, if you are playing guitar, bass or drums, the most common method is to search for tablature represented using text characters.

Tablature, or tab, is a way of writing down an instrument's parts in an accessible manner. In the context of this project, tablature will refer to guitar, bass, and drum tablature. Although tablature is widely used due to its ease of use, most tablature exists in a text-based format. Currently, there does not exist an easy way to turn tablature into a digital file format, such as MusicXML.

Although using text-based tablature is a great way to learn a new song, it cannot be easily transposed, edited, or played back to the user. There are many alternative formats that do not have these issues, but the majority of songs are found as text-based tablatures. The music research community developed a format called MusicXML where it can accurately denote tablature.

1.3 System Purpose

The software product will consist of a desktop application that, upon the upload of a tablature in *.txt format, will detect the instrument that is being described by the uploaded tablature, and convert the tab to a MusicXML file format. The outputted MusicXML file can be opened and used in programs that support the viewing of MusicXML files, such as MuseScore.

2. System Features

2.1 System Features

Users will be able to upload a text file of tablature for conversion to MusicXML.
Users will be able to paste their text tab into a textbox for conversion to MusicXML.
The application will output a correct MusicXML file from the tablature input.
Users will be able to select to save where to store their newly converted files on their local machine.

2.2 User Interfaces

Front-end software: JavaFX
Back-end software: Java, JAXB

2.3 Hardware Interfaces

Windows and Mac operating systems

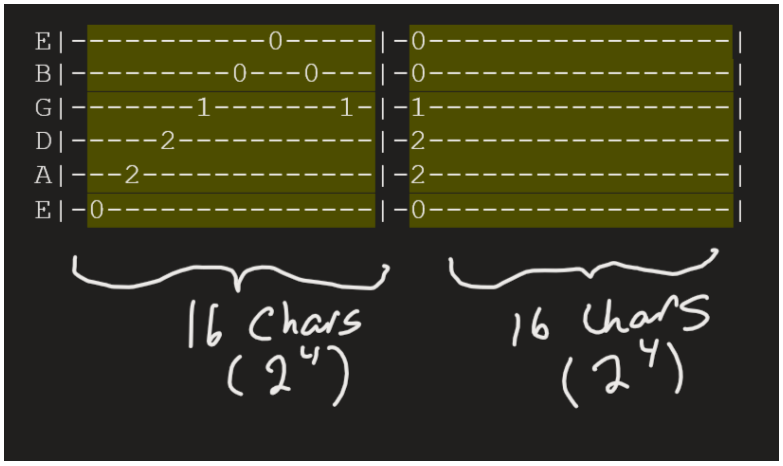
3. Functional Requirements

3.1 Functionality

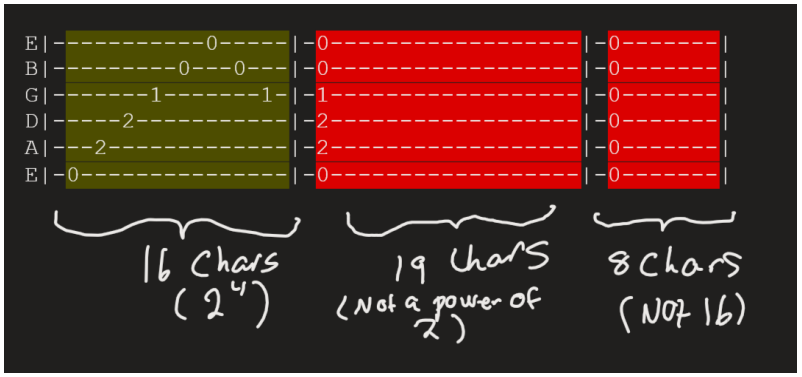
1. The application must convert a text file (either uploaded or pasted into the UI's textbox) of guitar, bass, or drum tablature into a MusicXML file.
2. The application must automatically detect which instrument the uploaded tablature is describing.
3. The application must always produce an accurate MusicXML file, with the exception of note timings, granted that the tablature input is valid. A valid tablature format is defined below:
    - Each line of the tablature starts with a letter indicating the note that the string is tuned to (or for drum tablature, the designated drum kit piece) immediately followed by a lowercase 'L' or vertical bar (|).
    - A new measure is defined by vertical lines, indicated by either a lowercase 'L' (l) or with the vertical bar glyph (|). Note that multiple measures are not required for a tab to be converted properly, ie, tabs with one measure will be converted properly.
    - All lines in a measure must have the same number of characters.
    - If a tab is multiple lines, the strings or drum pieces must be redefined for each new line.

3.1 Functionality – Continued

4. The application must produce appropriate note timings if given a “perfect input file”. A perfect input file is defined as the following:
    - All conditions stated in the above Requirement 3) must be satisfied.
    - The number of characters in each line from the first note in each measure to the end of the measure must be a power of two. This condition must be consistent for all measures in a tab.
    - For example, the following tab is a perfect input file:



    - In the case of imperfect inputs, the following tab is an example of an imperfect input file:



3.2 Usability

1. The application must allow users to view the converted MusicXML file format after conversion.
2. The application must allow users to save the MusicXML file to a specified file path after conversion.
3. The application must allow users to upload a tab text file from their machine to upload for conversion.
4. The application must allow users to copy tablature and paste it into the system as an alternative to uploading a text file.

4. Non-Functional Requirements

4.1 Performance

1. The application should operate quickly, with a text to MusicXML conversion time of less than 1 second for any humanly playable guitar tab under 5000 lines. Performance is measured once the “Convert” button is clicked, and ends when the MusicXML file is written to the user’s computer.

4.2 Usability

2. The application should have an intuitive user interface.
3. The application should only convert the latest file uploaded.

5. Use Cases

5.1 – File Conversion

Use Case Name:	File Conversion
Primary Actor:	User
Summary:	The user's uploaded text file (.txt) should be converted into a MusicXML file.
Basic Flow:	1. The use case starts on program start up. 2. The user selects and uploads a text file containing tablature. 3. The system detects which instrument the tablature is describing. 4. The system converts the text file into a MusicXML file for the appropriate instrument. 5. An output preview of the conversion can be displayed if the user presses the "Open MusicXML" button in the "Conversion Complete" menu.
Alternative Flows:	Step 3: If neither a guitar, bass, or drum tablature is detected, the program throws an exception to the user and goes back to the program start-up menu.

5.1.1 – User Stories

Primary Actor: John Smith

Goal: John Smith is a music teacher who has guitar tablature in a text file and wants to convert it to a MusicXML file.

1. John starts the program.
2. John selects the text file with the built-in file explorer and starts the conversion process. The system detects that John has uploaded a guitar tab, and converts the text file to a MusicXML file.
3. John decides to save the MusicXML file to a specified file path on his local machine.
4. John closes the program.

5.2 – User File Editing

Use Case Name:	User File Editing
Primary Actor:	User
Summary:	Allows the user to edit the text tablature before processing the file.
Basic Flow:	1. The use case starts when a text file is uploaded to the program. 2. The file contents are displayed in the textarea. 3. Users can edit tablature within the textarea.
Alternative Flows:	Step 1. If an invalid file format is given, the program will throw an exception to the user and the program will go back to the program start-up menu.

5.2.1 – User Stories

Primary Actor: Ashley Martin

Goal: Ashley Martin is a beginner musician who has guitar tablature in a text file and wants to convert it to a MusicXML file.

1. Ashley uploads a tablature file
2. The system displays the text within the textarea
3. Ashley notices a missing note, so she changes the tablature in the textarea

5.3 – Save Converted XML File

Use Case Name:	Save Converted XML file to a specified file path
Primary Actor:	User
Summary:	Allows the user to save the converted XML file to their local machines through a file explorer.
Basic Flow:	1. The use case starts when the user is done with the conversion process. 2. The user presses the “Save File” button to choose a file path to save their converted MusicXML file to.
Alternative Flows:	Step 2. If the conversion does not compile then the program will throw an exception and go back to the program start-up menu.

5.3.1 – User Stories

Primary Actor: Dr. Gonzalez

Goal: Dr. Gonzalez is a music professor who has guitar tablature in a text file and wants to convert it to a MusicXML file for a future lecture.

1. Dr. Gonzalez has successfully converted a text file.
2. Dr. Gonzalez clicks “Save File” and saves the XML file to a local folder.

5.4 – Open Outputted XML File in Browser

Use Case Name:	Open Outputted MusicXML File in browser
Primary Actor:	User
Summary:	Allows the user to view the MusicXML file after conversion.
Basic Flow:	1. The use case starts when the user finishes the conversion process. 2. The user presses the “Open MusicXML” button in the “Conversion Complete” menu, opening the MusicXML file in a compatible web browser for preview.
Alternative Flows:	No alternative flows.

5.4.1 – User Stories

Primary Actor: Peter Parker

Goal: Peter Parker is a music researcher who has guitar tablature in a text file and wants to convert it to a MusicXML file.

1. Peter has successfully converted a text file.
2. Peter clicks “Open MusicXML”
3. The system opens his web browser containing the MusicXML file