Requirement documents

Team 1 – Tab2Pdf

Introduction

This document specifies the required features of the tablature-to-pdf conversion software. The Tab2PDF software exists to make it easier for guitar players to convert their ASCII formatted tabs into beautiful and easy to read PDF files. This program should be capable of accepting an ASCII formatted guitar tab and changing it to the desired symbols and notes. By following the simple input syntax, the user can create guitar tabs which can then be converted into an elegant PDF file. The sections covered include: Input Handling, Conversion Requirements, Acceptance Cases, Customization features, and a Graphical User Interface (GUI).

Brief requirements

1. Program must accept an input file that is formatted as a guitar tab in ASCII format and convert it into a PDF file.
2. The program should be able to convert user inputs into a more professional tablature format. To do this, the program should be able to read individual symbols from input and convert them to their equivalent representation.
3. The user should be able to interact with the system using a Graphical User Interface(GUI). The GUI will allow the users to choose the input file to read, where to save the generated output and customize the look of the PDF file.
4. The user will be able to customize the title, subtitle, horizontal spacing, and vertical scaling using the GUI.

Input Handling

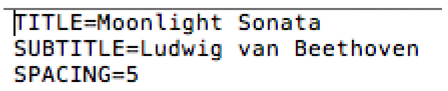
1) Input files should be acceptable if they are represented in ASCII format. If the input is not in guitar tab format, a message will pop up in the preview panel saying “Couldn’t render tab”. The program will read all text files but will only convert to pdf if it is formatted as a guitar tab.
2) If a file is a guitar tab but has many elements not following the specified syntax, the program should look past these extra elements and print the rest of the tabs.

Conversion Requirements

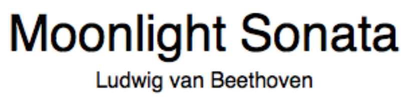
This section outlines what kinds of symbols the program should convert and display in a PDF.

Title, Subtitle and Spacing

3) The program will allow the user to specify the title, subtitle and horizontal spacing of frets at the top of the input using the keywords “Title=” “Subtitle=” and “Spacing=”.
4) When the user chooses not to include the title, subtitle and spacing in their input, the program will provide a way to add these values through the GUI features.



```
TITLE=Moonlight Sonata
SUBTITLE=Ludwig van Beethoven
SPACING=5
```



Moonlight Sonata

Ludwig van Beethoven

Arrangement of bars:

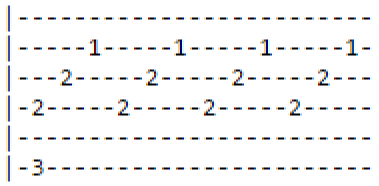
5) If the space is insufficient to fit a full bar on a line, the program will move the individual bars to the next line to make it easier for the user to understand the music.

String lines:

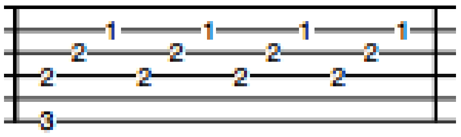
6) The program will interpret lines of dashes as strings. By aligning the fret numbers along the dash entries, the program will determine when the user intends to play the notes and translate that order onto the PDF.

7) The program will change these dash symbols into solid string lines on the PDF making the tabs more readable.

8) The program will give the user freedom to modify the input by changing notes, bar length, number of repeats, etc.



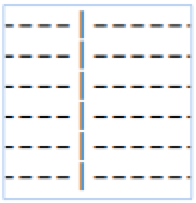
```
|-------------------------
|-----1-----1-----1-----1-
|---2-----2-----2-----2---
|-2-----2-----2-----2-----
|-------------------------
|-3-----------------------
```



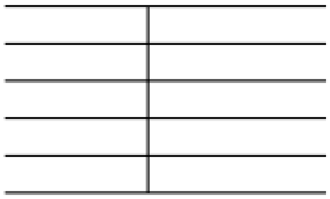
Vertical Bar Lines:

9) The program will read the text file and find where there are six “|” lines located, a solid bar line extending all six strings will be printed on the PDF.

10) The program should ignore any improper placements of ‘|’ within the program and only translates those that are arranged together row by row.



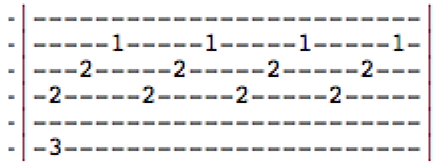
```
----|------
----|------
----|------
----|------
----|------
----|------
```



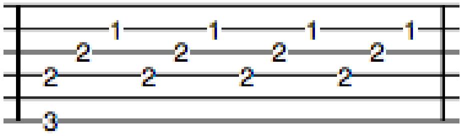
Fret numbers:

11) The program should read all single or double-digit numbers from the input file within range of 0 - 24, these digits represent the fret numbers of a guitar tab.

12) The program will maintain the placement and order of the fret notes when it translate the tabs to a PDF.

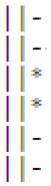
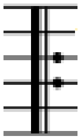
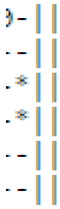
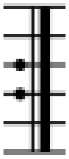
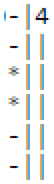
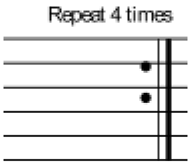


```
-|-------------------------|
-|-----1-----1-----1-----1-|
-|---2-----2-----2-----2---|
-|-2-----2-----2-----2-----|
-|-------------------------|
-|-3-----------------------|
```



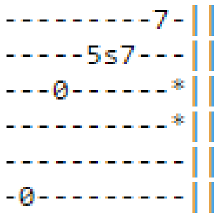
Repeats:

13) The program will allow the user to input tab repeat bars in multiple formats and convert them to a standard musical notation in the PDF. The following types of repeats should be supported:

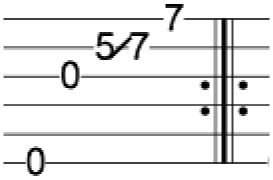
Tab Repeat Type	Sheet Music Repeat
 ||- ||- ||* ||* ||- ||-	
 --|| --|| *|| *|| --|| --||	
 -|4 -|| *|| *|| -|| -||	

Slide to and from fret numbers:

14) The user will be able to represent a slide by inputting a character “s” in form “xsx”, where X represents a valid fret number.The output will create a slide as shown below.



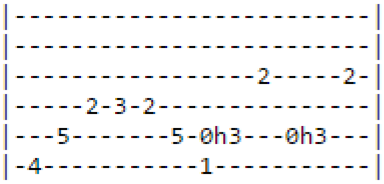
```
---------7-||
-----5s7---||
---0------*||
----------*||
-----------||
-0---------||
```



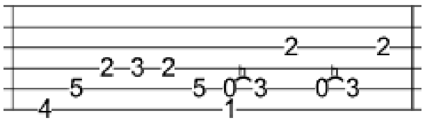
Hammer-on, Pull-off:

15) The program will support the creation of hammer-ons and pull-offs. When there is a “xhx” in the input, where x is a digit that represents a fret number, the program should write the fret numbers normally with an arc over them.

16) Pull offs should also be supported with input of “xpx” and will print a ‘p’ above the arc.



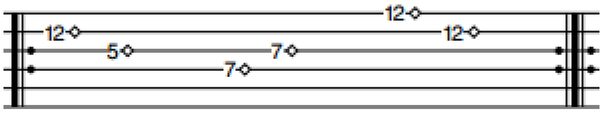
```
|-------------------------|
|-------------------------|
|-----------------2-----2-|
|-----2-3-2---------------|
|---5-------5-0h3---0h3---|
|-4-----------1-----------|
```



17) The program will also allow the user to have any amount of “-” in between the characters of a slide, hammer-on and pull-off, and still produce the same output.

Harmonics:

18) The program should be able to create harmonics when the user places ‘<x>’ in the input. This should create a diamond next to the number of the fret as shown below.

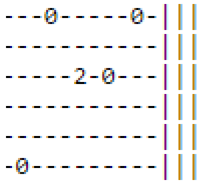


Muted String:

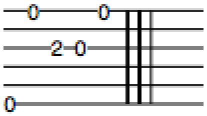
19) The program should allow the user to represent muted strings by using ‘x’ in place of a fret number.

Triple Bar:

20) The program should let users represent triple bars in their tabs by placing “|||” lines in their input. The output will be as shown below.



```
---0-----0-|||
-----------|||
-----2-0---|||
-----------|||
-----------|||
-0---------|||
```



Acceptance Cases

21) Should pass test cases which simulate the type of inputs a user might enter and how the user will interact with the system.
22) Should take properly formatted tabs and output a proper translation as a PDF.
23) When the input contains extra symbols that are not supported by the system, the program should ignore the extra symbols and only output the rest of the tabs.
24) Program must be capable of error handling, by taking inputs with errors and still produce a valid guitar tab.

Usability

25) The program should be tested by people who don't know the high level details of the project in order to study the programs user experience. Depending on the outcome of these tests, the program should be changed to fit the needs of the user.
26) The program should be intuitive and simple enough for the user to understand the program with little guidance. If there is any ambiguity in the design of the GUI or confusion in the workflow of the program, then revision must be undertaken.

Graphical User Interface (GUI)

The GUI will be designed to be easy to use and allow immediate visualization of the desired output. The GUI will allow the user to easily customize and change the input file they have chosen to handle.

Easy-to-use:

27) The GUI will be designed to be easy to use for people of all experience with computers.
28) If the user chooses not to customize any other features within the GUI, then the program assumes those specifications are already in the text file or the program assigns a default value for the spacing, while title and subtitle should be left blank.
29) The GUI will allow the user to select any input file from within a browse window. The user will also be able to browse the location they want to save the output file.
30) The GUI should feature an editor for the input file on screen to allow the user to edit the file before exporting it as a PDF.
31) The program will have a save button that will allow the user to store all the changes they have made within the input editor.
32) The user will be able to preview if the input file was read correctly through the input editor.
33) The program should contain a PDF preview panel that allows the user to see a rendered version of their tab after every change made to the text file.
34) Once the user is satisfied with the file, they can export the tab to PDF which will open up the user's default PDF viewer with the tabs PDF displayed.

35) The GUI should contain tooltips that will appear after the user hovers over a button for a few seconds, this will provide a short description of what the button does.
36) A help menu will provide the user with resources to understand how to use the program and how to properly format the input files. The help menu will contain the user manual and some sample tab files that are properly formatted.

Customization:

37) The user will be able to change the title, subtitle and spacing value that is specified in the text file through the editors and sliders in the customization panel.
38) The user will be able to alter any details within the input by using the on screen text box displaying the input file contents.
39) The user will be able to control the appearance of the output file by changing the horizontal spacing and vertical scaling within the GUI by a range of assigned sizes.