Software Requirements Specification

for

TAB2XML

Prepared by: Ziqi Zhou 214726283
Amer Alshoghri 214992291
Hargovind Singh 217303025
Uwais Kazi 215263940
Ali Yamany 216513269

Lassonde School of Engineering

Group 13

April 12, 2021

Table of Contents

Revision History

Date	Reason For Changes	Version
2021-04-12	Final Submission	V1.1
2021-02-03	Submission for Peer Assessment 1	V1.0

1. Introduction

1.1 Purpose

The purpose of this document is to cover all the software requirements for TAB2MXL for it to function as described. TAB2MXL is a standalone desktop application designed to convert different formats of ASCII music tablatures to the popular music description file format called MusicXML. This software will be able to convert different types of tablature (guitar, drums, and bass) into platform and instrument independent musicXML.

1.2 Document Conventions

The word “User” is used as an inclusive term that refers to anyone who is going to use the application.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, and shareholders. Potential users, such as hobbyists and professional musicians, can also use sections 1.1-1.3 to gain a deeper insight into the software description and its intended goals.

1.4 Product Scope

The main goal of this product is to provide a convenient local software for tablature-to-MusicXML conversion. This product will include a simple graphical interface, convert formats accurately, and provide intuitive ways for users to manage input and output. The software must run locally and must be platform-independent.

1.5 References

MusicXML website : https://www.musicxml.com/

2. System Features

2.1 Input ASCII music tablature

Priority: high

2.1.1 Description and Priority

This feature allows the *user* to input ASCII music tablature for guitar, bass, or drums into the input text area.

2.1.2 Stimulus/ Response Sequence

The system will ask the user to input the ASCII text tablature. The *user* will be able to upload the tablature or write it in the text input area. The *user* should also be able to edit the tablature inside the text input area.

2.1.3 Functional Requirements

2.1.3-REQ-1: The system must be able to input ASCII music tablature as a *.text file or through the text input box.
2.1.3-REQ-2: The system must verify that the tablature does not contain illegal characters that do not belong in guitar, bass, or drum music tablature.
2.1.3-REQ-3: In case the system detects unknown tablature elements, it should display a warning message to the *user* highlighting the error-prone portion of the input tablature.
2.1.3-REQ-4: Currently, the system must only accept a single instrument tablature.

2.2 Instrument Auto-detect

Priority: high

2.2.1 Description and Priority

This feature detects the type of instrument associated with the tablature provided as input to the system.

2.2.2 Stimulus/ Response Sequence

The *user* will be displayed with an “Instrument Detection” button above the text input area that allows the *user* to use this feature.

2.2.3 Functional Requirements

2.2.3-REQ-1: The system must be able to recognize the instrument associated with the tablature.
2.2.3-REQ-2: The system must show an error message to the user in case it is not able to detect the correct instrument.
2.2.3-REQ-3: The system must allow the *user* to override this feature and manually select the type of instrument associated with the tablature.

2.3 MusicXML Conversion

Priority: high

2.3.1 Description and Priority
This feature music tablature in ASCII form to MusicXML
2.3.2 Stimulus/ Response Sequence
1) The *user* will be displayed with a "Translate" button above the text input area that allows the *user* to utilize this feature.
2) The *user* should be able to view the generated musicXML file in the text area after the conversion process is over.
2.3.3 Functional Requirements

2.3.3-REQ-1: The *user* must confirm the tablature instrument type before generating the musicXML file.
2.3.3-REQ-2: The system must ask the user to input the composer, title, and time signature associated with the tablature.
2.3.3-REQ-3: The system must allow the *user* to edit the generated MusicXML output before saving the file.

2.4 Navigating through the System

Priority: high

2.4.1 Description and Priority
This feature allows the *user* to return to the text input menu in case the user wants to make changes or input new tablature.
2.4.2 Stimulus/ Response Sequence
1) The *user* will be displayed with a "Restart" button that allows the *user* to return to the text input screen.
2) The *user* will be displayed with a "Clear" button that deletes the content of the text input area.
2.4.3 Functional Requirements

2.4.3-REQ-1: The system should be able to take the *user* from the screen that displays the generated musicXML to the initial text input screen.

2.5 Saving Files

Priority: high

2.5.1 Description and Priority
This feature allows the *user* to save the generated musicXML file on his/her computer.
2.5.2 Stimulus/ Response Sequence
The *user* will be displayed with a "Save" button that allows the user to save the generated musicXML output as a *.txt or *.musicxml file.

2.5.3 Functional Requirements

2.5.3-REQ-1: The system should be able to save the final musicxml tablature with the appropriate format.

3. Use Cases

3.1 Product Perspective

The music research community has developed a free format, called MusicXML, that can be used to precisely denote a piece of music. This format is supported by many music apps to display the song in an easy-to-read fashion, transpose to another key, play the song, etc. Conversion from tablature to MusicXML is still performed manually. This product is designed to break the barrier of tablature-to-MusicXML conversion. TAB2XML is designed to be a brand-new and self-contained product coded in Java suitable for all platforms with Java Runtime Environment installed.

3.2 User Stories

3.2.1 User story of professional guitarist Max (persona).

Story Identifier	TAB001
Story Name	Guitarist
Description	As a Guitarist, I need to convert a text file containing guitar tablature into a MusicXML file.
Functional Acceptance Criteria examples	- Can I go back to the initial text tablature without clearing the text area? - Can I edit the text tablature after uploading?
Non-functional Acceptance Criteria example	- Can an unauthorized person view my text tablature or converted MusicXML file?

3.2.2 User story of professional conductor Jessica (persona).

Story Identifier	TAB002
Story Name	Conductor
Description	-As an orchestra conductor, I work with multiple instruments. I need to convert a text file containing guitar, bass, or drum tablature into a MusicXML file.

Functional Acceptance Criteria example	- Can the software auto-detect whether the tablature I entered is for a guitar, bass, or drum?

3.2.3 User story of professional songwriter Andrew (persona).

Story Identifier	TAB003
Story Name	Songwriter
Description	-As a songwriter, I write and sell music tablature for multiple instruments. I want the musicXML file to store my name as the 'composer' of the document.
Functional Acceptance Criteria example	- Can the software add my name as the composer to the musicXML file?

3.3 Use cases

1. **Save translated MusicXML file**

Use-case field	**Description**
Use case name	Output translated MusicXML file
Primary actor	User
Overview	The Actor inputs the TAB2XML and should be able to save correctly translated MusicXML files
Preconditions	● The input tablatures must be that of guitar, bass, or drum ● The input tablatures must conform to a valid format.
Main Success Scenario	1. The user inputs a music tablature file 2. The parsing service asks the user to enter the Composer, Title, and Time signature associated with the tablature. 3. The parsing service reads the file 4. Parsing service classifies the file as one of Guitar, Bass, or Drum tablature 5. XML Generation Service generates a correctly translated file 6. The system asks the user for a location to save the musicXML file 7. User selects an appropriate location 8. MusicXML file is saved to the system
Extensions	4a. Translation error:

	The translation is inaccurate, resulting in an unusable MusicXML file. The program should display an informative error to the user 6a. Output error: The user does not have sufficient permissions to store the file in the selected directory. The system displays an error and asks the user to upload the file to another location

2. Display translated MusicXML

Use-case field	**Description**
Use case name	Display translated MusicXML
Primary Actor	User
Overview	Instead of saving the result to a file, the XML Generation service can display the resulting musicXML content directly
Preconditions	● The input tablatures must be that of guitar, bass, or drum and the correct format ● The input tablatures must conform to a valid format. ● the Parsing service successfully parsed the tablature and saved the appropriate values in memory
Main Success Scenario	1. The MusicXML translation is generated 2. XML Generation Service outputs the contents of the musicXML to the display
Extensions	1a. Translation error: The translation is inaccurate, resulting in incorrect musicXML translation

3. Upload tablature files

Use-case field	Description
Use case name	Upload tablature files
Primary Actor	User
Use case overview	Actors are able to upload tablature files to TAB2XML and have them processed for translation
Preconditions	The input file must be a text file. The text file must exist on actors' local computers.
Main Success Scenario	1. User selects the option to upload a tablature file 2. User selects the file from their local system 3. The Parsing service reads, classifies, and translates the file contents 4. The XML Generation Service Generates correct MusicXML 5. The system saves the file or displays the output depending on the option the user selected
Extensions	2a. File I/O error due to insufficient permissions to read the indicated file: The system should display an informative error message and ask the user to upload another file

4. Input tablature as text

Use-case field	Description
Use case name	Type in the tablatures directly in the input
Primary Actor	User
Use case overview	Actors are able to directly type in the tablatures in the TAB2XML input and have them processed for translation.
Preconditions	None
Main Success Scenario	1. User inputs tablature text into the display 2. Parsing service classifies and translates the tablature into musicXML format 3. XML generation service will save the resulting translation into a file or display the translated text based on the user-selected option
Extensions	2a. Input error: User did not input correctly formatted tablature. The program should display an informative error and ask the user for an appropriate input

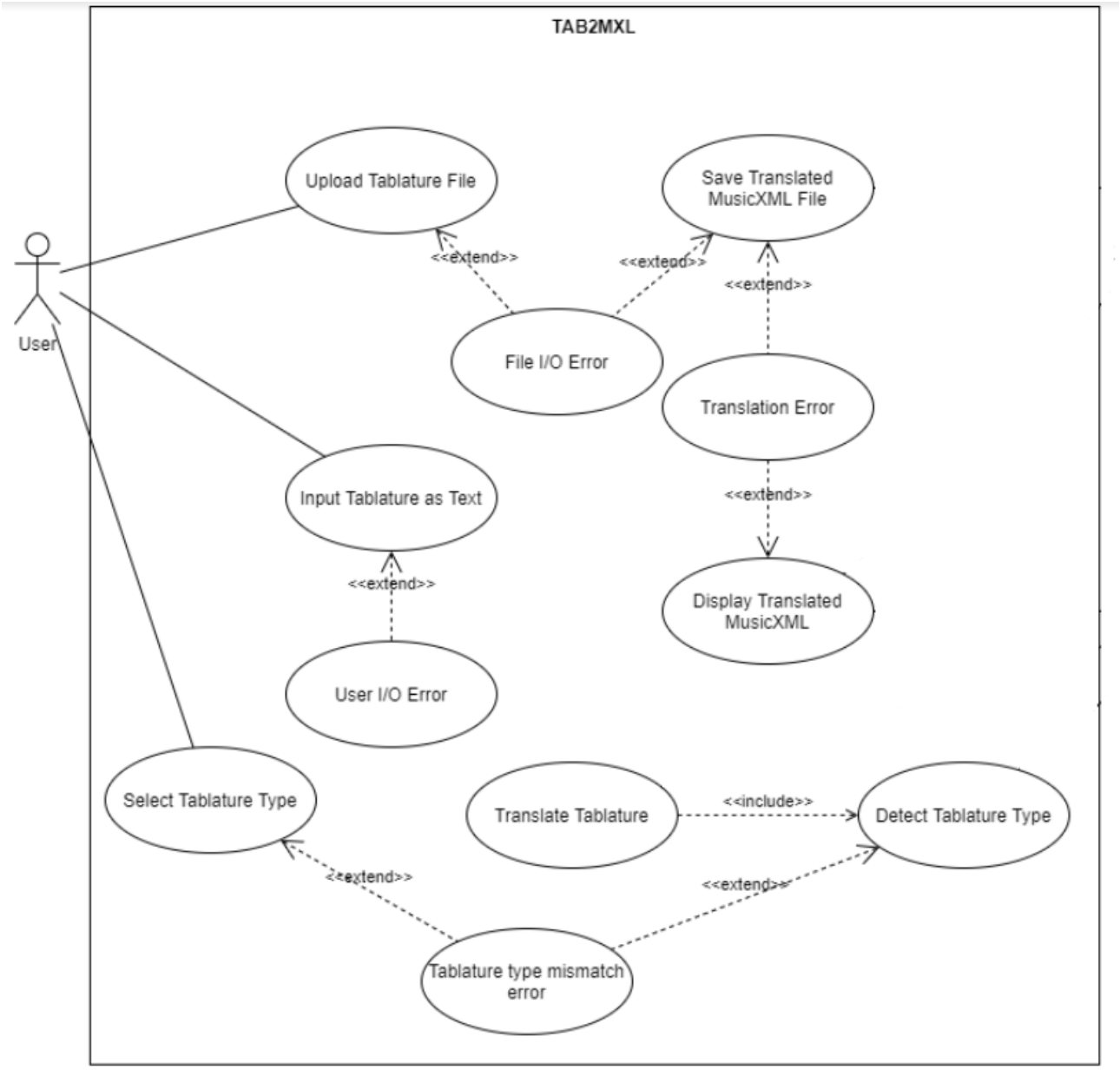
5. Select tablature type

Use-case field	Description
Use case name	Select tablature type
Primary Actor	User
Use case overview	Actors are able to choose the tablature types manually either preemptively, or if the system classifies them incorrectly
Preconditions	The desired tablature type is that of bass, guitar, or drum
Main Success Scenario	1. The user chooses the desired tablature type from the display 2. User inputs tablature through text or by uploading a file 3. Parsing service attempts to translate given tablature using the given tablature type 4. XML generation service will save the resulting translation into a file or display the translated text based on the user-selected option
Extensions	2a. Tablature type mismatch Error: The selected tablature type does not match the type of the input tablature

6. Detect Tablature Type

Use-case field	Description
Use case name	Detect tablature type
Primary Actor	User
Use case overview	The program detects the tablature type directly to educate MusicXML learner and to make the process easier for musicians
Preconditions	The input tablature type is that of bass, guitar, or drum. The input content/file is valid tablature.
Main Success Scenario	1. User inputs valid tablature through a file upload or text input 2. Parsing Service reads the file into memory 3. Parsing Service detects the tablature type and prepares to apply MusicXML translation to the tablature type 4. XML Generation Service generates valid MusicXML and stores it in memory 5. XML Generation service outputs the resulting translation or saves it into a file depending on user preference
Extensions	3a. Invalid tablature type: The program cannot detect some available tablatures

3.4 Use case diagram



3.5 Design and Implementation Constraints

A GUI interface is essential to the system. The application is built as a standalone application, and in the future, it can be implemented as a web or mobile application, which at this point is beyond the scope of the project. The system will be designed, implemented, and deployed by mid-April. There will be no communication between the system and any external server.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

As this is a music-based application, this system should be able to handle long tablatures efficiently without shutting down or being overwhelmed. Most communications should take place within milliseconds to at most 3 seconds.

4.2 Usability Requirements

Software must offer the users the ability to access all available functionality in 4 clicks or less. Software interface must behave evenly spaced buttons no less than 10px on a 1920x1080 resolution screen

4.3 Maintainability Requirements

Software components must be modular and maintainable to support the addition, removal, or overhaul of core features. Software must use the git fork-pull request workflow to help developers maintain the project.

4.4 Supportability Requirements

Software must perform critical functionality on Windows 10 and macOS operating systems. Software must render the user interface correctly on 1920x1080 displays. Software must not consume more than 500 megabytes of RAM storage on the device it’s running on.