Four-part Writing Machine Requirements Document

v1.0

By: Steven Yeung

In partial fulfillment of the requirements for the completion of CSCI 497 at Charleston Southern University, Spring 2021

Advisor: Dr. Sean Hayes

## Table of Contents

**Table of Contents**

1. **Introduction**
   1. Purpose of Document ……………………………………………………………… 3
   2. Intended Audience …………………………………………………………………. 3
   3. Project Scope…………………………………………………………………………3
   4. Project Goals………………………………………………………………………….3
   5. Rationale………………………………………………………………………………3
2. **Requirements Scope**
   1. Functional Requirements……………………………………………………………3-4
   2. Non-Functional Requirements……………………………………………………….4
   3. Constraints……………………………………………………………………………4
3. **Functional Requirements**
   1. Drag notes to staff based on time signature…………………………………………..4
   2. Build chords based on key signature…………………………………………………4
   3. Check if the notes have gone out of range……………………………………………4
   4. Enter roman numerals based on chords built…………………………………………4
   5. Correct users if they have incorrect chords based off key signature and/or roman numerals……………………………………………………………………………..4-5
   6. Correct users if they have incorrect roman numerals based on chords written………5
   7. Correct users if they any four-part harmony rules are broken………………………..5
4. **Non-Functional Requirements**
   1. Personalization Requirements…………………………………………………………5
   2. Learning Requirements………………………………………………………………..5
   3. Understandability Requirements………………………………………………………5
   4. Accessibility Requirements……………………………………...……………………5
   5. Convenience Requirements…………………………………………………………..5
   6. Performance Requirements……………………………………………………….....5-6
   7. Release Requirements…………………………………………………………………6
5. **Interface Requirements**
   1. Ease of Use Requirements……………………………………………………………6
   2. Aesthetics Requirements……………………………………………………………..6
6. **Constraints**
   1. Solution Constraints………………………………………………………………….6
   2. Schedule Constraints…………………………………………………………………6
   3. Platform Constraints………………………………………………………………….6
   4. Extension Constraints…………………………………………………………………6
7. **Approval**………………………………………………………………………………….7
8. **Introduction**
   1. **Purpose of Document**

The purpose of this document is to describe the requirements needed to create a four-part harmony machine that checks if user has written their SATB melody correctly. Other non-functional requirements like UI design and constraints are described in this document.

* 1. **Intended Audience**

Intended audience for this document is mainly for the program creator and his advisor. When program is fully developed, this document may be edited and be intended for mainly musicians that are learning how to SATB and/or musicians who want to create melodies but forgot the rules.

* 1. **Project Scope/Description**

The scope of the project is to create a fully functional music notating program that will allow the user to create their very own SATB melody while in line with the harmony rules. Users should be able to access the software by downloading a repository from GitHub. The project will implement quarter notes, half notes, whole notes, any simple meter for rhythm. It will not implement compound meters. The project will allow users to select from a variety of chords, but no modern chords will be available for choosing. The project will aim to have no budget with everything being free open-source code that can be found online and visual studio to build.

* 1. **Project Goals**
* The project will allow the user to create up to eight measures worth of melodies in SATB form.
* The project will check if the user has any four-part rules broken.
* The project will check if the user has entered a roman numeral first, they must have the correct notes in that chord in the correct key signature.
* The project will check if the user has entered notes first, the roman numeral must be correct according to the notes.
* The project will check if the user has gone out of range for any parts.
* The user will be able to check their work when they have a melody in mind.
* The project should reduce stress for anyone using it to learn or to review rules of theory.
  1. **Rationale**

The reason behind this project is to create a successful four-part notation platform/error detection device on an open-source repository that is already existing. The project creator wants to research on this because they are a musician themselves and are interested in simple compositions that can be error prone. This project will help alleviate unknown errors that might be in the melody the user is trying to create.

1. **Requirements Scope**
   1. **Functional Requirements**

Functional Requirements are the project’s requirements to function properly. These are essential features that must be implemented first before all other requirements. These are not limited to correct logic and non-faulty functionality. Without a functioning product, the product is a failure. These should carry a fit criterion that is measurable. Once the requirement is implemented, the solution should be tested and determine if it fits in that requirement.

* 1. **Non-Functional Requirements**

Non-Functional Requirements are properties of a project that must be implemented that make the project easier to use and efficient to use. These are not limited to performance and user friendliness. These functions are just as important as the functional requirements for the success of the project. As in functional requirements, fit criterion is needed because for non-functional requirements, the user should be able to mess with the interface in a certain number of minutes.

* 1. **Constraints**

Constraints are “roadblocks” that hinder the progress of the project. They are not limited to time, knowledge, research, and/or platform support. Knowledge and research are the biggest issues as the skill level of the student developing this project is classified as intermediate.

1. **Functional Requirements**
   1. **Drag notes based on time signature.**

Based on what the user picked for their time signature (no compound meter), logic will be used to not allow users to enter less than or more than the correct beats per measure. Attempting this will result in the user to not be able to compose a melody properly. Musicians shall be able to compose a standard SATB melody in at least 10 minutes.

* 1. **Build chords based on key signature.**

The key signature will be the foundation of what the chords will be in the four-part harmony. The chord must be written based on the key signature that is selected. Logic will be implemented to not allow non-related chords to be in key signatures. Musicians shall be able to determine what chords belong to a specific key signature in less than 5 minutes.

* 1. **Check if notes have gone out of range.**

Soprano, Alto, Tenor and Bass parts in four-part harmony have specific ranges. The user will not be able to go out of range when working on specific part of the harmony. Musicians should already know that SATB have a certain range. Users will realize if they have gone out of range as soon as they drag the note in.

* 1. **Enter roman numerals based on chords built.**

When the user has built their chord, they must assign that chord a roman numeral of some sort. The roman numeral may be inverted. Logic will be used to check if the notes of the chords match the roman numeral of the key signature. It will also check for inversions as well. No applied chords allowed. After musicians have composed their melody, they should be able to perform a roman numeral analysis. Time may vary on how quickly musicians accomplish this because of varying theory knowledge.

* 1. **Correct users if they have incorrect chords based off key signature and/or roman numerals.**

If the users have notes in a chord that do not match the key signature and/or roman numeral they selected, the program will mark the wrong note and prompt the user to correct it.

* 1. **Correct users if they have incorrect roman numerals based on chords written.**

If the user has a correct chord, but incorrect roman numeral, the program will mark the roman numeral and prompt the user to correct it. Once roman numeral analysis is complete, musicians should immediately see their errors and correct them in 10 minutes.

* 1. **Correct users if they have any four-part harmony rules broken.**

If the user has any rules of four-part harmony broken (i.e., parallel fifths, octaves, leading tone etc.) , the program will mark the error made and prompt the user to correct it. As mentioned above, the users can correct these errors in 10 minutes or less.

1. **Non-Functional**
   1. **Personalization Requirements**

The project is using most of the personalization from Muse Score, which is the source of the project. The user will be able to customize from Muse Score’s library in 5 minutes.

* 1. **Learning Requirements**

The finished product should be easy to use. If the user is not familiar with music notation software, they can infer by hovering over the buttons they have. They can learn in 5 minutes.

* 1. **Understandability Requirements**

Within 3 minutes of using this product, the user should realize it is a music notation device for SATB form.

* 1. **Accessibility Requirements**

The user should be able to access this program with a modern computer. They need the programming platform to run this project as it will not be public. Depending on their technological background, accessing this project may take 20-30 minutes.

* 1. **Convenience Requirements**

If the user has the project on their own device, they should be able to access it without the presence of internet. They will also be able to write their melodies as soon as the project is launched.

* 1. **Performance Requirements**

The program must have efficient runtimes because a bad program that can timeout is bad for the user.

* 1. **Release Requirements**

The project will be kept private and property of its creator and the institution. When the product is more developed, the project will be launched publicly.

1. **Interface Requirements**
   1. **Ease of Use Requirements**

The product should be easy to use to musicians without much technical experience because it is notating music on a device. The learning curve should be low as it is a user-friendly program. The user should be able to start notating within their first 30 minutes of using the product.

* 1. **Aesthetics Requirements**

The product should have the buttons for notes in one place, time signature in one place, etc. By having an appealing interface, the user will navigate the notation software with ease.

1. **Constraints**
   1. **Solution Constraints**

The solution of the project will take place in C++. However, not only C++ will be used, other factors like QT and JavaScript will play a role in compiling the project as one. Also, C++ logic can be difficult.

* 1. **Schedule Constraints**

The project timeline will take approximately 12-14 months in designing, building, and testing. The project designer may have other events occur during the 12-14 months. The designer is not uninterrupted during the time, so time must be managed well.

* 1. **Platform Constraints**

The project will not be able to support mobile devices. It can only support laptop and desktop platforms that have the required applications and accounts to access the source code of the project. Mobile support may be designed in the future. Upwards of an extra year may be required to create mobile support.

* 1. **Extension Constraints**

As mentioned above, the four-part notation project will not include compound meter, applied chords, and non-related chords based on key signature. The extra extensions will take upwards of 3 extra months.

1. **Pending Approval**

This document is written in partial fulfillment of CSCI 497, Senior Project Design and is a prerequisite for CSCI 498, Senior Project Construction.

Approval of this document is pending. Once corrections to this document are made, this section will be approved to being construction of project. Changes will be made if necessary.