Software Requirements Specification
for
MusicMaker

Version 1.0 approved
Prepared by <author>
<organization>
<date created>

Table of Contents

Revision History

1.Introduction

1.1Purpose

The purpose of this document is to describe the implementation the the Music Maker application. The Music Maker applications is designed to make musical notes from a range of 12 different instruments. This application also allows the user to save music in the form of a text file, load music files, record a tune created and also play back the notes that the user creates.

1.2Document Conventions

There is no particular odd/difficult terminology used in this document.

1.3Intended Audience and Reading Suggestions

This document is intended for a developers and users as the implementation details are explained as well as how to use the actual software in a simple manner. This document can be read from the start to end if the reader is a developer if he/she wishes to know the intricate details of the implementation of this software. If the reader is a user then the reader can directly jump to section 3.1 which explains the user interface and then to section 4 of this document which explains the system features.

1.4Product Scope

This Music Maker software will consist of 5 major functions. Playing music based on the instrument of choice and the notes that are played, record the notes played by the user, playback the tune, save the recorded tune into a text file and load a saved tune.

1.5References

This document refers many documents and web pages which has been used throughtout this software project in it's implementation. The implementation of the Jfugue package has been referred from the documents SunJava Jfugue and Jfugue.org/book.
Online reference sources have been used for the UI implementation. The main source has been stackoverflow.com and Java documentation on Jframe, and Swing package.

2.Overall Description

2.1Product Perspective

This software is original and not a part of any bigger system or component.

2.2Product Functions

This software is mainly for entertainment purposes though composing music from 12 different instruments.

2.3User Classes and Characteristics

This software is easy to use and hence there is no set or subset of users that are going to use this product. Again, this product can be studied upon to improve the UI and functionality, but it's main purpose is for entertainment and hence everyone is of equal importance wrt this product's appeal.

2.4Operating Environment

Since this is a Java application, it it platform independent, and hence this application will work on any hardware that supports Java and any operating system I.e Windows, Linux and Unix.

2.5Design and Implementation Constraints

The main constrain is the amount of knowledge in notes, keys, voices etc to actually make a good music generating application may be hard for the developer to obtain in a short period of time. The language that it was implemented in, i.e Java is not exceptionally difficult, but the various intricacies of the JFugue package should be explored. The programming standards should be top quality, especially maintaining personal data to calculate metrics from the PSP sheets that are required to be filled. The maintenance is not of great importance here, but as the software gets less interesting, it may need to go through an evolution phase in order to keep it relevant.

2.6User Documentation

User can understand the internal workings of this product by reading up on JFames in java documentation to understand the flow of the software and user can also read up on the package JFugue to understand how the music is being generated from the system.

2.7Assumptions and Dependencies

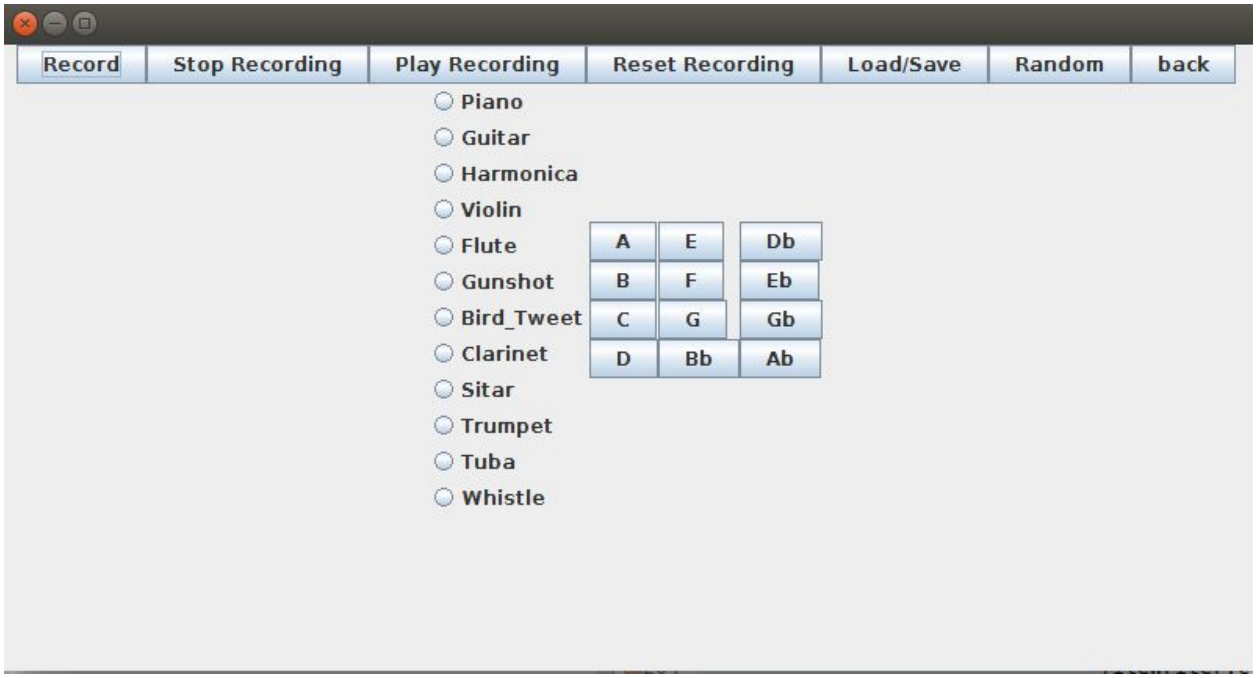
The music only plays when the user has selected an instrument. Also assumptions is that if a user loads a file into the software, the data in the file will be read as one tune and will be played as one tune. This tune should only contain notes from “A” to ”G” and “Bb”, “Db”,”Eb”, “Gb”,”Ab”.

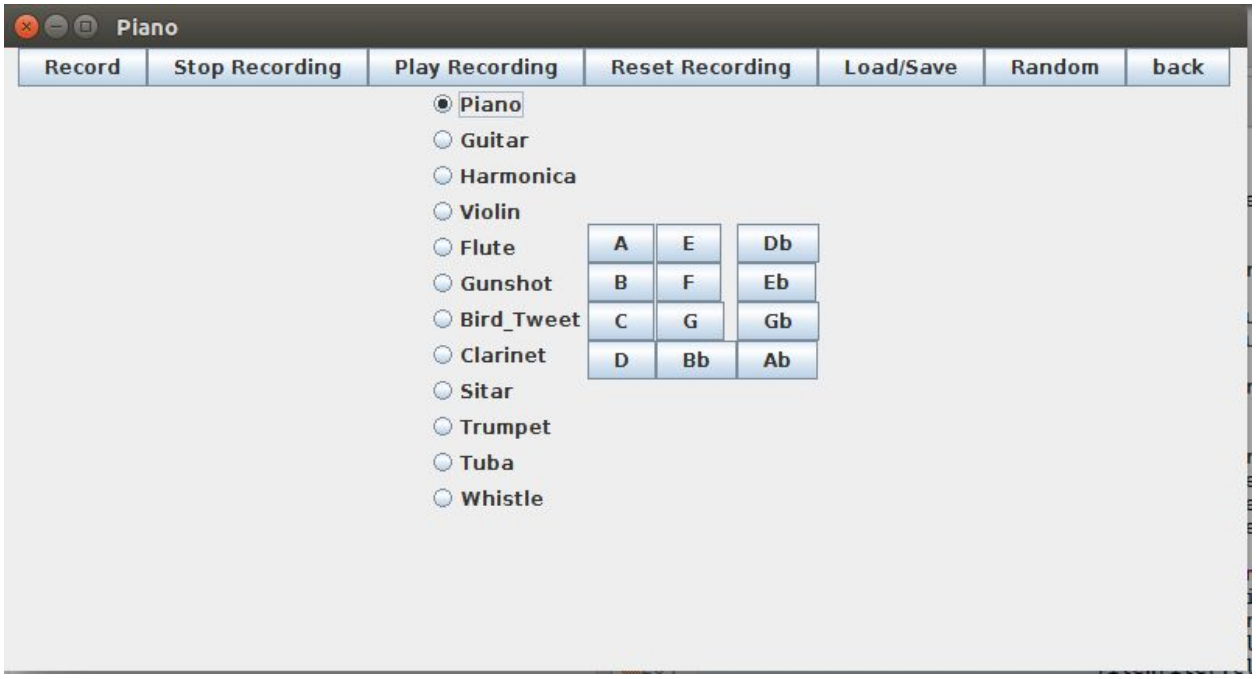
3.External Interface Requirements

3.1User Interfaces

The user interface is shown as below. This application first opens with the main Frame which displays the title of the software which is Music Maker. This frame has 2 buttons. The button “Start Here” will display the next screen which is shown below.

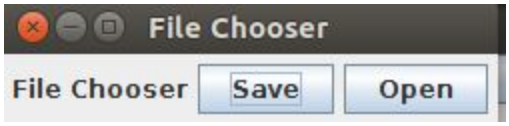
Selecting any one of the radio buttons will change the frame name to the corresponding instrument's name and the user can start playing tunes.



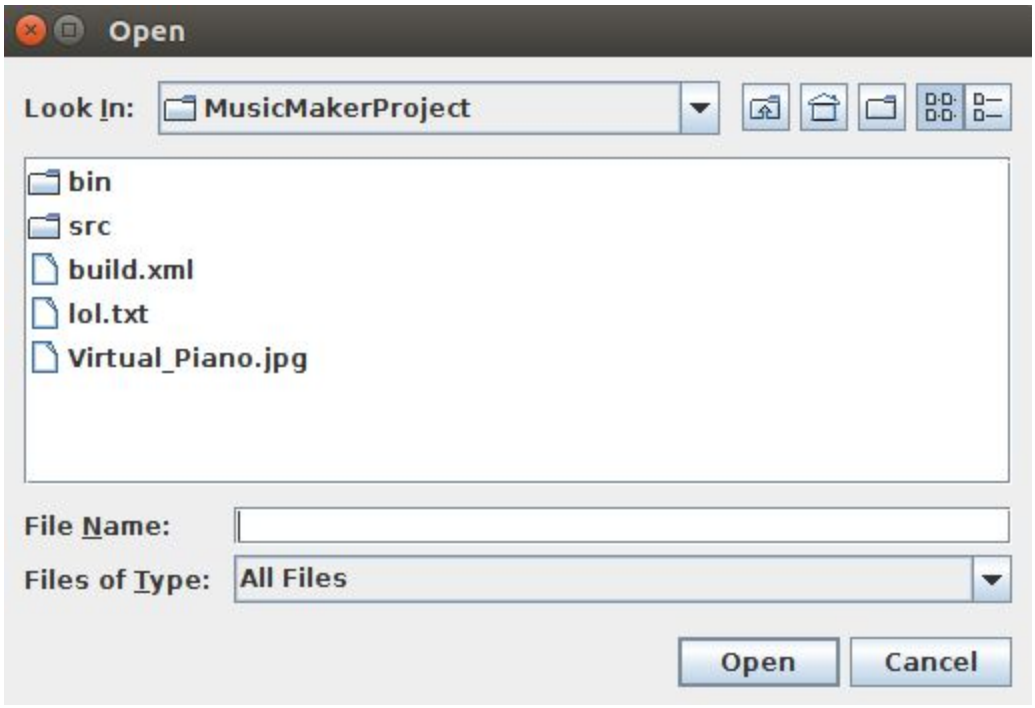


The tunes are played by appending the character that is present on the button the user clicked on to the sequence and the corresponding instrument plays it. A user can record the tune by pressing the record button. Similarly they can stop recording by pressing stop recording. They can play the recorded sequence by clicking the play recording. The user also has a choice of

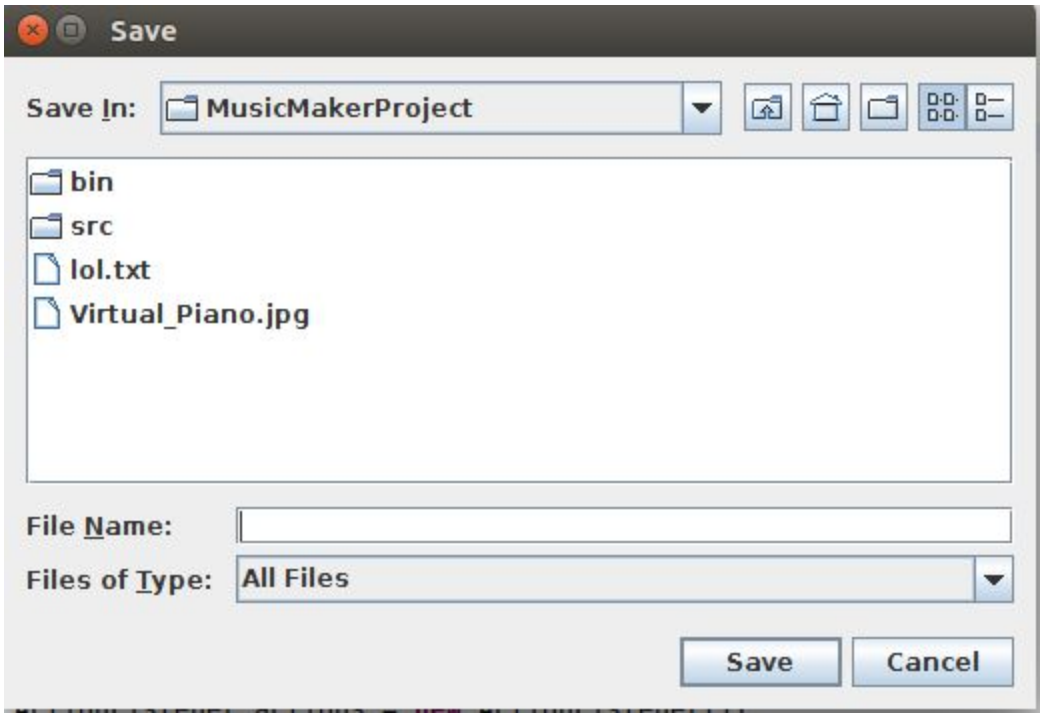
saving the recorded sequence to a file by pressing the Load/Save button on the top. The following frame is opened when the button is pressed.



If the user presses the open button, the following window opens.



If the user presses save, the following window opens.



3.2Hardware Interfaces

Since this is a java program, this is completely platform independent. And hence can work in any hardware environment. There are no particular communication protocols to be used. The data is stored in the form of text files which are also universal in any operating system.

3.3Software Interfaces

The database used is the system provided text files. One tune, no matter how big is stored in one file. The only data that is generated is the sequence of the tunes. Hence this data is shared b/w the 2nd frame and the save screen. Also the save window will send the name of the file to the program such that that file is created and the particular tune is created.

4.System Features

4.1 Play Music

4.1.1 Description and Priority

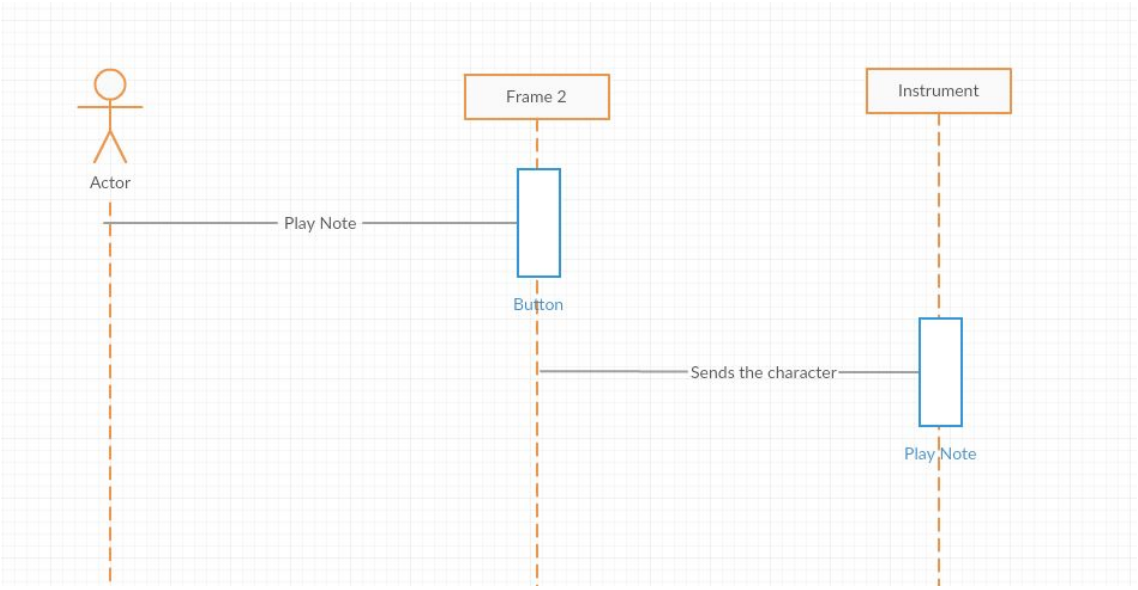
This is the main feature of this software. It's priority is at a high of 9. This is achieved through the java package JFugue. The music is played by the Player class reading the string of tunes that the user has selected.

4.1.2 Stimulus/Response Sequences

The list of sequences can be the player pressing the buttons corresponding to the tunes he desires, and that tune being played from the system.

4.1.3 Functional Requirements

An error can occur when there is no instrument selected and then the user is trying to play a note. In this case, no event will occur as in no sound or tune will be produced.



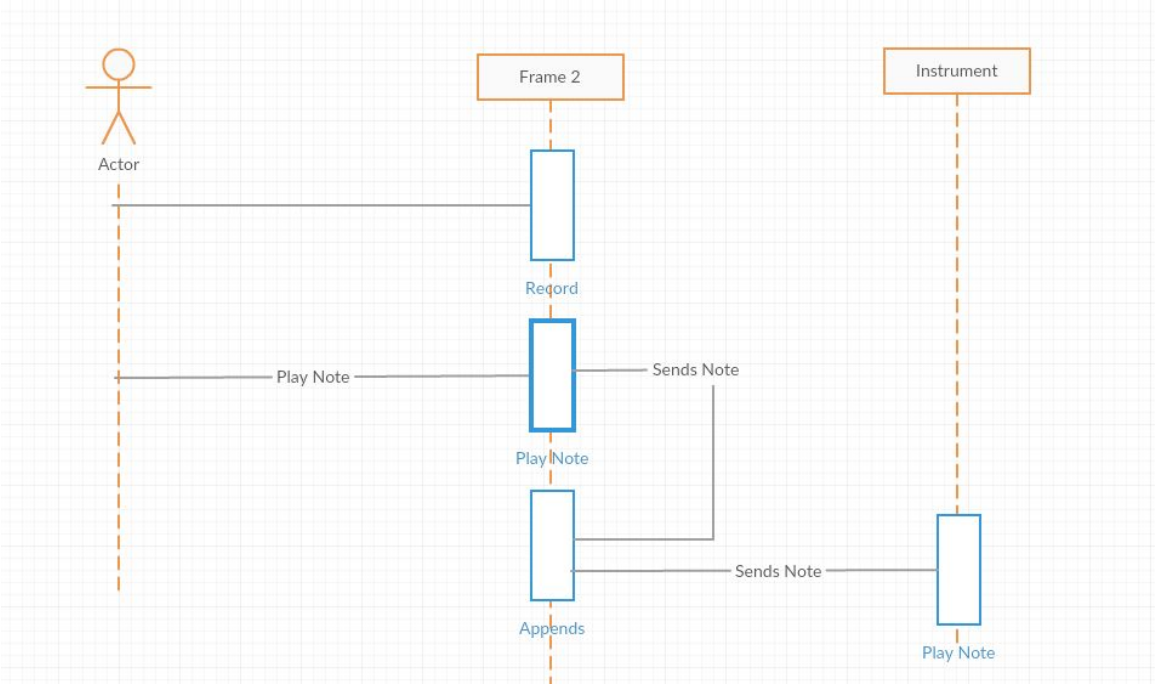
4.2 Record Music

4.2.1 Description and Priority

This feature is not as important and hence can be given an above average priority of 6.

4.2.2 Stimulus/Response Sequences

This occurs when the user presses on the record button and then the notes and the instrument that the user is currently on gets appended to a string which can be either saved into a file or can be played back.



5.Other Nonfunctional Requirements

5.1Performance Requirements

This application is fairly light on resources and hence there are no strict performance requirements.

5.2Safety Requirements

This application is fairly light and simple and hence there are no particular safety requirements.

5.3Security Requirements

There is no database that stores user information, nor does a user need to identify him/herself while using this application and hence the security requirement is minimal.

5.4 Software Quality Attributes

The development of this software has been based on PSP model and hence the software itself is of a high quality. It is highly portable because it being a java program. Usability is simple because of a simple UI and ease of flow.

5.5 Business Rules

There are no restrictions placed on user based on any sort of administrative privileges in this application.

6.Other Requirements

“Not applicable”

Appendix A: Glossary

The SRS itself does not have complicated vocabulary. The vocabulary mainly used for this software mainly includes musical vocabulary. They are tempo-indicates speed of music, voice-Channel and Instrument- the type of instrument.