Guitar Teacher App

Software Requirements Specification

<1>

<30-10-2013>

<Netanel Keidar Levi>

Lead Software Engineer

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 30-10-13 | Creation | Netanel Keidar Levi | <First Revision> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | <Your Name> | Lead Software Eng. |  |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.3 Use Cases 3

3.3.1 Use Case #1 3

3.3.2 Use Case #2 3

3.4 Classes / Objects 3

3.4.1 <Class / Object #1> 3

3.4.2 <Class / Object #2> 3

3.5 Non-Functional Requirements 4

3.5.1 Performance 4

3.5.2 Reliability 4

3.5.3 Availability 4

3.5.4 Security 4

3.5.5 Maintainability 4

3.5.6 Portability 4

3.6 Inverse Requirements 4

3.7 Design Constraints 4

3.8 Logical Database Requirements 4

3.9 Other Requirements 4

4. Analysis Models 4

4.1 Sequence Diagrams 5

4.3 Data Flow Diagrams (DFD) 5

4.2 State-Transition Diagrams (STD) 5

5. Change Management Process 5

A. Appendices 5

A.1 Appendix 1 5

A.2 Appendix 2 5

# Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotates are largely taken from the IEEE Guide to SRS).

## Purpose

*This document is intended for all programmers of this projects (thousands in a few months ;) ).*

## Scope

*The objective is to create an app meant to teach the user how to play the guitar, in an interactive way, providing tools needed for him to get started.*

## Definitions, Acronyms, and Abbreviations

## References

## Overview

*We describe all views In the app, it’s core modules, and all required parts in them.*

# General Description

* 1. *The app has a few screens:*
  2. ***The main screen*** *– Will provide a quick entry (button) to the current exercise, another for picking a field, another button to enter the tuner, and another for the settings.*
  3. ***The exercise picker screen*** *– Will let the user choose a field to practice (Basics, Chords, Rhythms…) , after that an exercise stage (Major chords, minor chords…), that will switch to the Exercise Plan screen.*
  4. ***The Exercise Plan screen*** *– Lets the user pick the exercise plan to play (He chooses a field, then an exercise). Both in the fields picking screen and afterwards, in the stage picking screen you can pick a “Pick for me” button which will create an automatic plan.*
  5. ***The Game Screen*** *– This is where most of the logic will take place. For this moment, we refer to the basic playing screen where a guitar fretboard is displayed. On this screen, the user will see the acts to be played.(A popup window saying what to do, a displaying of notes pending user play…).*
  6. ***The Tuner –*** *A tuner, allowing the tuning of the guitar. The tuner will have an information screen to assist newcomers. It will have a needle showing the current played frequency whereas the closest frequency will be in the center.*

## User Characteristics

We are focusing on making the experience as simple as possible while providing an interactive experience.

## General Constraints

*We know that planning far ahead to save later coding is important. However, in each aspect of the app, we should seriously consider the costs of this forward support, to make sure we are putting all of our efforts on getting as fast as we can to an early release.*

# Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

## External Interface Requirements

### User Interfaces

### Tuner Activity

* + - * 1. The screen’s background will be the common background image – a wooden background.
        2. Pressing back from the tuner activity will lead you to the calling screen, which has to be the main screen. Note that you don’t have much to do here.
        3. On the top, there will be a horizontal list of buttons displaying the guitar notes (of the type *OctavedNote*), from highest to lowest.
        4. A needle will be presented in bottom to mid-screen (An image, supplied by designer).
        5. There will be an off/on switch (image) button on the right bottom of the screen to switch between Auto and Manual mode. These two modes will be specified below.
        6. In manual mode, pressing of the aforementioned buttons will play their tone. The needle’s frequencies will disappear and the needle will be on 0.5 opacity.
        7. In Auto mode, when the user plays a note, The closest actual note (OctavedNote)’s frequency will be displayed above the needle. On the other two sides of the “speedometer” the next and previous notes’ frequencies will be displayed.
        8. In Auto mode, when the user plays a note, the needle will rotate to an angle corresponding to the difference between the central frequency and the frequencies on the side.
        9. In Auto mode, after a note is played and the needle rotated as requested, it will slowly shift back to the 0 angle.
        10. In Auto mode, if a new note is played before the needle got back to 0, the new rotation animation will begin from the needle’s current angle.
        11. The top buttons mentioned above will be images and react to a click – a really small decrease in size (0.9) and back to original size.
        12. In auto mode, the tuning will happen from the left most (lowest) to right-most string. The current string button will be highlighted (different image).
        13. In auto mode, two leds on both sides will indicate how accurate the string is.

### ExercisePicker Activity

1. The screen’s background will be the common background image – a wooden background.
2. For all the following modes, a list of buttons (images) inside a horizontal scroll view will be displayed.
3. The items shown in each mode are the children of the previous mode so (All of the fields -> Stages of selected field -> Options of selected stage). An additional button for “Get me started” will pick an automatic plan varying the current options in the current mode, as will described in the ExercisePlan chapter.
4. For all the following modes, the buttons will appear, starting from the first in an Overshooting linear scaling interpolation.
5. For all the following modes, upon clicking, the buttons will disappear starting from the one clicked, the ones next to it, and those next to them and so forth… The delay between object disappearances is 0.5 of the disappearing duration.
6. The user transits between the following modes (Main activity <-> Field <-> Stage <-> Option (Only in debug mode) <-> ExercisePlan activity ). Transition to the next mode is by clicking on an item (button). Transition to the previous mode is by clicking on the back button.
7. The first mode that is displayed is a Field picker. Inside every button will be written the name of the field, and an image representing it.
8. In the second mode, Stage, we’ll display the stages of the selected field. A text of the stage name will be inside the button image.
9. The third mode, option, will look like the second mode. Note that this mode is for debug purposes only.
10. While in transition between modes, further clicks will do nothing, pressing the back button will get you back to the pre-transition mode.
11. Once the user chose from the last mode, a transition to the exercise plan will occur, given the parameters (Option(Debug)/Stage(Release)) chosen.
12. A bar containing a simple one line instruction will be on the top of the screen. Upon transition into a mode the bar will appear from above the activity. Upon exit it will go up.
13. Before exiting the activity all objects will disappear: The bar will go up, the button list will finish disappearing.

### Game Activity

1. The screen’s background will be the common background image – a wooden background.
2. On the screen will be presented an image of a part of a guitar neck (6 frets need to be visible), with 6 strings visible.
3. On the right side of the ‘bridge’, within a fret’s distance, will be circles with the letter of the note representing the open string note.
4. The activity will receive the StageOption to be played in the parameters (intent).
5. The screen will know to display a scenario by implementing the IPlayingDisplayer interface:
6. The DisplayInformation operation will be implemented by displaying a popup window with the background darkened. The window will have the title on the top, and the message in the center. A button to proceed will be visible on the right bottom corner of the window.
7. The displayNotes operation will take the given IHandPositioning and display each note in its’ destined fret-string by a circle containing the number of the finger to play.
8. The circle just stated will have a revolving dotted circle surrounding it.
9. If the frets-strings to be played are off the visible amount of frets, a vertical line will be displayed alongside the first fret (behind the strings), and above it a number indicating the offset.
10. The WrongNotePlayed operation will be implemented by an according image displayed on the predicted note played. The mark will appear for 0.5 seconds by an alpha transition, stay there for 0.5 seconds, and disappear during 0.5 seconds.
11. The activity will implement the setWaitingPlay by a microphone inside a circle with ‘waves’ surrounding it from right and left in an alternating number: 1,2,3,1,2,3…
12. A bar on the lower left will display the current Act. Upon entering the activity the bar will enter from the left.
13. A button on the top left (image with the pause mark on it) will be visible. It will appear from the top left corner when entering the activity, and stay there until leaving the activity.
14. Clicking the pause button will open a pause menu displaying the current task on the top and the following buttons on the bottom in a row: Resume. Restart. Exercises, Home. The buttons will appear all together in an overshoot interpolation. The buttons will be evenly spaced horizontally.
15. A click on any of the four buttons will cause the hiding of the pause menu and then execute the task clicked.
16. On the top right corner of the screen will be a yellow star and left to it the total number of points the user got so far.
17. The IPlayingDisplayer’s endExercise operation will be implemented by bringing up a popup, similar to the one of the pause menu. On the top the name of the exercise will be displayed. In the center, 3 stars indicating how successful the player was. The stars will be evenly spread horizontally.

### Main Activity

* + - * 1. The screen’s background will be the common background image – a wooden background. On top of it, an image of a guitar frame.
        2. There will be buttons (images) on the screen: Start button, larger than the rest. Tuner button on the right. Exercises on the left. A settings button somewhere.
        3. Mouse down on the buttons shrinks them to 0.9 size, mouse up puts them back, while disappearing all of them.

### ExercisePlan Activity

* + - * 1. The screen’s background will be the common background image – a wooden background.
        2. There will be a popup covering 0.95 of the screen that will be overshoot for 0.5 sec.
        3. A title will be on the top saying what exercises we have to do
        4. A list of exercise names with a checkbox(not real, image) next to them indicating whether that exercise is done.
        5. A button saying start will be on the right bottom of the popup.
        6. After 5 seconds, if the button wasn’t pressed, all elements beside the next stage will disappear (opacity animation for 1 sec) and a sec afterwards the popup will disappear altogether.

### Metronome Activity

* + - * 1. The screen’s background will be the common background image – a wooden background.
        2. The screen will have a settings button (image) at the bottom right. Pressing on it will lift it above the screen, while it drags the settings view.
        3. The screen will have a play/pause button on the bottom left.
        4. At the center of the screen, a circle (image) will be displayed. On top of it a number displaying the current beat subdivision (1... 2… 3… 4… 1… 2… 3… 4…).
        5. Each change of subdivision the circle will scale up to 110% and then down in a duration of a quarter of the subdivision length.
        6. Two lines will be drawn in the circle. First one will be static and the second one will go from that one and rotate around the circle making a full round over each note (4 beat subdivisions by default).
        7. Each beat subdivision a tick sound will be played. Each note, an ascended sound will be played.

### Settings Activity

* + - * 1. The screen’s background will be the common background image – a wooden background.
        2. There will be a popup covering 0.95 of the screen that will be overshoot for 0.5 sec.
        3. A title will be on the top saying ‘Settings’
        4. If possible, android’s native settings mechanism will be used.
        5. The possibility to enable/disable the notifications will be there. If enabled, the notifications mechanism described in the next chapters will be enabled.

### Software Interfaces

### Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

### 3.2.2 <Functional Requirement or Feature #2>

…

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

### 3.4.2 <Class / Object #2>

…

## 3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

### 3.5.1 Performance

### 3.5.2 Reliability

### 3.5.3 Availability

### 3.5.4 Security

### 3.5.5 Maintainability

### 3.5.6 Portability

## 3.6 Inverse Requirements

State any \*useful\* inverse requirements.

## 3.7 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

## 3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

## 3.9 Other Requirements

Catchall section for any additional requirements.

# 4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

## 4.1 Sequence Diagrams

## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)

# 5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

# A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

## A.1 Appendix 1

## A.2 Appendix 2