

**Software Requirements Specification**  
for  
Ubuntu Touch Audio Mixer (working name)

Prepared by:

Steven Lares

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Purpose . . . . .	2
1.2	Intended Audience . . . . .	2
1.3	Intended Use . . . . .	2
1.4	Product Scope . . . . .	3
1.5	Risk Definition . . . . .	3
<b>2</b>	<b>Overall Description</b>	<b>4</b>
2.1	User Classes and Characteristics . . . . .	4
2.2	User Needs . . . . .	4
2.3	Operating Environment . . . . .	4
2.4	Constraints . . . . .	4
2.5	Assumptions and Dependencies . . . . .	4
2.6	Code Style . . . . .	4
2.7	Test Style . . . . .	4
<b>3</b>	<b>Requirements</b>	<b>5</b>
3.1	Functional Requirements . . . . .	5
3.2	External Interface Requirements . . . . .	5
3.3	System Features . . . . .	5
3.4	Non Functional Requirements . . . . .	5
	<b>Appendices</b>	<b>6</b>
<b>A</b>	<b>Glossary</b>	<b>7</b>

# Chapter 1

## Introduction

The aim of this document is to provide in-depth insight of the Ubuntu Touch Audio Mixer software by defining the problem statement in detail.

It describes expected capabilities from end users while defining high-level product features.

This document was built using references from:

1. <https://www.overleaf.com/latex/templates/cse355-software-requirements-specification-layout/pvjpzxthtngc>
2. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>
3. <https://www.geeksforgeeks.org/software-engineering-quality-characteristics-of-a-good-srs/>
4. <https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/?ref=lbp>
5. utdallas.edu -SRS4.0 doc

### 1.1 Purpose

This document is used to "kick off" the project with a strong functional base. I intend for this to be open source later on.

### 1.2 Intended Audience

### 1.3 Intended Use

The app will be downloaded by the user through Ubuntu Touch's Open Store. It will not be available elsewhere, other than the GitHub repo which hosts source code and possibly test builds of the app.

It is intended to be used in two major ways:

1. It can be used as a portable external audio mixer.
  - (a) This means that the app will act as a bridge between:
    - i. An external audio playback / microphone device feeding audio input into the phone running the app.
    - ii. An external audio playback device receiving mixed audio output from the phone running the app.

- (b) This is the main strength of this app for the following reasons:
  - i. As of 10/21/2022, there are no Android or iOS alternatives that provide this functionality.
- 2. It can also be used as an internal audio mixer.
  - (a) This means that the app will be able to mix audio across the system.
  - (b) This functionality is comparable to other apps on the Android and iOS stores, usually with the terms "equalizer", "EQ", "Mixer" in their name.
  - (c) This functionality leads this app into having desktop counterparts:
    - i. EqualizerAPO for Windows (<https://sourceforge.net/projects/equalizerapo/>)
    - ii. PulseEffects for Linux (<https://github.com/wwmm/easyeffects>).
  - (d) Why use this Ubuntu Touch implementation instead of Android or iOS? Well:
    - i. This app will behave closer to its desktop counterparts in that it will not contain ads, subscriptions, and other scummy money-grubbing schemes. If donations are added to support the app, they will stay on the official Open Store page and out of the user's way.
    - ii. The app is open source, and is not a mysterious black box.
    - iii. Will be implemented using as much native Ubuntu Touch and Linux functionality as possible.
    - iv. It will also be written with a faster and more memory efficient programming language compared to Java (Android) or Swift (iOS). This is especially important as real time audio mixing can be a CPU and memory intensive process.
  - (e) **NOTE:** This is also the "fall-back" in case functionality #1 is not possible in the UT environment. However, it will still retain its strength over Android / iOS counterparts.

**NOTE:** The above list may be revised over time depending on how requirements can be fulfilled in the UT environment.

## 1.4 Product Scope

## 1.5 Risk Definition

## Chapter 2

# Overall Description

### 2.1 User Classes and Characteristics

### 2.2 User Needs

### 2.3 Operating Environment

### 2.4 Constraints

### 2.5 Assumptions and Dependencies

### 2.6 Code Style

<https://mitcommlab.mit.edu/broad/commkit/coding-and-comment-style/>

### 2.7 Test Style

Will also implement a Test Driven development style, using Unit Testing Principles, Practices, and Patterns by Vladimir Khorikov

TDD will likely be implemented using concepts from parts 1 and 2 of the book. Part 3 is on an as-needed basis.

By combining TDD with this SRS, I will be able to better test outline units of behavior and individual requirements (like Ubuntu dependencies separately)

## Chapter 3

# Requirements

### 3.1 Functional Requirements

### 3.2 External Interface Requirements

### 3.3 System Features

### 3.4 Non Functional Requirements

# Appendices

# Appendix A

## Glossary

- Audio Input:
- Mixer:
- Mixed:
- Audio Output: