Software Requirements Specification

for

Ubuntu Touch Audio Mixer (working name)

Prepared by:

Steven Lares

Contents

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
6. https://ieeexplore.ieee.org/document/278253

It uses Prototype Outline 1 for SRS Section 3 from the IEEE link

1.1 Purpose

This document is intended to be read by volunteer developers and testers, as well as anyone else curious enough to learn more about the project.

Admittedly, this SRS is more developer/tester oriented since it relies on git version control to track its development. Git is used instead of maintaining a version history table within the SRS itself; the latter of which is typical for SRS documents created in Microsoft Word or SRS-creation software.

Therefore, learning how to compare git commits and branches will help with understanding the SRS development over time.

NOTE: This document is meant to aid a first official release. It will likely not be maintained once the project has it's official release, and will stay for archival purposes.

1.2 Product Scope

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.
      ii. PulseEffects does provide similar functionality, however there is a major limitation: You must run a server on an Linux machine, and have another Linux machine (with the same package installed) connect to this machine in order to receive the mixed audio output
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 closely initial requirements can be fulfilled in the UT environment.

1.3 Risk Definition and Management

At this early stage within the project, there are no risks. This may change as the project matures.

Chapter 2

Functional Requirements

2.1 System

1. File IO
   (a) load_template
   (b) save_template
2. Device Setup Per Selected Mode
   (a) switch_between_bridge_and_internal_modes
   (b) in_bridge_mode_set_the_audio_source_input
   (c) in_bridge_mode_set_the_audio_source_output
   (d) in_internal_mode_block_the_audio_source_input
   (e) in_internal_mode_set_the_audio_source_output
3. Audio Routing
   (a) chain_audio_effects_in_series
   (b) split_audio_stream_into_two_streams
4. Audio Control
   (a) adjust_audio_effect_parameters_during_audio_stream
   (b) audio_control_modules_adjust_the_paramters_of_audio_effect

2.2 Audio Effect Modules

1. Equalization
   (a) apply_15_band_EQ_to_audio_stream
   (b) apply_30_band_EQ_to_audio_stream
   (c) apply_variable_band_EQ_to_audio_stream
2. Filtering
   (a) apply_all_pass_filter_to_audio_stream
   (b) apply_band_pass_filter_to_audio_stream

   (c) apply_low_pass_filter_to_audio_stream
   (d) apply_notch_filter_to_audio_stream
   (e) apply_high_pass_filter_to_audio_stream
   (f) apply_high_shelf_filter_to_audio_stream
   (g) apply_low_shelf_filter_to_audio_stream
   (h) apply_peaking_filter_to_audio_stream
3. Miscellaneous
   (a) apply_compressor_to_audio_stream
   (b) apply_limiter_to_audio_stream
   (c) apply_delay_to_audio_stream
   (d) apply_loudness_correction_to_audio_stream
   (e) apply_preamp_to_audio_stream

2.3 Audio Control Modules

1. LFO
2. Math calculator

Chapter 3

Non Functional Requirements

Chapter 4

Ubuntu Touch Specifics

This chapter contains items that are specific to the development in the UT environment. There is some decision making that is listed here.

General docs: https://docs.ubports.com/en/latest/

TODO: Decide on a license

4.1 IDE and Debugger Choice

- Use QtCreator for IDE since it has debugging and testing:
  - https://docs.ubports.com/en/latest/appdev/code-editor.html
- Use C++... maybe use UT Specific libraries:
  - https://docs.ubports.com/en/latest/appdev/nativeapp/index.html

4.2 App Suspension

Apps are suspended whenever not in the foreground, or when the device is locked. When an app is suspended, it cannot receive location data. For this reason, apps will not be able to track your location whenever they are not in use or the device is locked.

1. https://docs.ubports.com/en/latest/userguide/dailyuse/location.html
2. https://docs.ubports.com/_/downloads/ro/latest/pdf/

NOTE: May need the user to use UT Tweak to remove app suspension for full usage. However, there may be a way around it.

4.3 Shared Dependencies

This is the section that outlines external and out-of-memory dependencies that the project will have.

As it stands, this project is shared dependency heavy.

Since I will follow the classical approach to TDD, I need to know these "shared" dependencies so that my unit tests can mock these out.

Below are results of learning how to import external libraries and linux packages into the project.

- NOTES:
  - You can include C++ dependencies (libraries) in your project through clickable.
  - From what I currently understand, Clickable can compile ARM software into desktop architecture as it uses a container. It may not be able to go the other way $x86/x86\_64 -> ARM$
  - Interestingly, it can run the app in a continuous integration environment due to the use of the ARM -> Desktop container. There are guides on how to do CLCI
  - The consensus on Quora seems to be that the difficulty will range from: rebuilding with an ARM compiler to rewriting from scratch. It all depends on the type of software

- Links
  - https://docs.ubports.com/en/latest/appdev/guides/dependencies.html
  - https://ubports.com/blog/ubports-news-1/post/introduction-to-clickable-147
  - https://clickable.bhdouglass.com/en/latest/
  - https://www.quora.com/How-hard-is-it-to-port-an-application-from-x86-to-ARM

TODO: Create a small list of external dependency candidates for each function in your proposed app. Favor dependencies that: already have an ARM build, that fit within your license model, is a C++ library, and requires the least amount of work to adapt.

TODO: Once the candidate list has been created, create a prototype project in which you try to import each dependency and use them one by one. Write down results for the results of each.

Contains the largest list of open source audio software that I could find: https://github.com/webprofusion/OpenAudio