**Software Requirements Specification (SRS) Document**

**for**

**C++/Java Project**

**Prepared by:**

The Kitchen

**Date:** 10th October 2024

# Table of Contents

[Table of Contents 2](#_Toc183716558)

[1. Introduction 3](#_Toc183716559)

[1.1. Team Details 3](#_Toc183716560)

[1.2. Project Overview 3](#_Toc183716561)

[1.3. Scope 3](#_Toc183716562)

[2. Objectives 4](#_Toc183716563)

[3. System Overview 5](#_Toc183716564)

[3.1. Technical Specifications 5](#_Toc183716565)

[3.2. Input/Output Requirements 6](#_Toc183716566)

[4. Functional Requirements 6](#_Toc183716567)

[4.1. Detailed Features 6](#_Toc183716568)

[Use Case 7](#_Toc183716569)

[5. Non-functional Requirements (Optional) 7](#_Toc183716570)

[6. Development Setup 7](#_Toc183716571)

[7. Workflow 8](#_Toc183716572)

[8. UML Diagram: 9](#_Toc183716573)

[9. Important Files & Folders 9](#_Toc183716574)

[10. Testing 10](#_Toc183716575)

[11. Conclusion 10](#_Toc183716576)

# Introduction

## Team Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll No.** | **Email** |
| M S Dheeraj Murthy | IMT2023552 | [ms.dheerajmurthy@iiitb.ac.in](mailto:ms.dheerajmurthy@iiitb.ac.in) |
| Mathew Joseph | IMT2023008 | [Mathew.joseph@iiitb.ac.in](mailto:Mathew.joseph@iiitb.ac.in) |
| Ayush Tiwari | IMT2023524 | [Ayush.tiwari524@iiitb.ac.in](mailto:Ayush.tiwari124@iiitb.ac.in) |
| Lesin | IMT2023565 | [Lesin.565@iiitb.ac.in](mailto:Lesin.565@iiitb.ac.in) |
| Priyanshu Pattnaik | IMT2023046 | [Priyanshu.Pattnaik@iiitb.ac.in](mailto:Priyanshu.Pattnaik@iiitb.ac.in) |

## Project Overview

The audio editor will provide essential multimedia playback functionalities such as Amplitude scaling, Frequency scaling, Pitch scaling, looping, clipping, clip superimposition, etc.

## Scope

* **In-scope:**
  + Implementing playback functionalities (play/pause, looping).
  + Amplitude and frequency scaling of audio clips.
  + Audio clipping and superimposition of clips.
  + Cross-platform compatibility (Windows, macOS, Linux).
  + Java-based UI and C++ backend logic using JNI for media handling.
* **Out-of-scope:**
  + Advanced audio effects like equalization.
  + Implementing an undo tree.
  + Video editing or support for visual media.
  + Real-time streaming functionality.

# Objectives

* Develop a **cross-platform audio editor** using **Java** for UI and **C++** for backend media processing.
* Implement the following functionalities in C++ that can be applied on the audio files:
  + Amplitude Scaling
  + Compression
  + Frequency Manipulation
  + Looping
  + Reverb
  + Time Scaling
  + Clip Trimming
  + Clip Superimposition
  + Audio Filter
  + Multi Track Editing
* Ensure seamless **JNI integration** between Java and C++.
* Handle exceptions such as:
  + Illegal parameter exception
  + Exceptions related to each effect
* Optimize media decoding and playback for smooth performance across platforms.

# System Overview

## Technical Specifications

* **Frontend:** Java (Swing)

The Java portion of the project will involve Java Swing and other classes to create a user-friendly UI that allows users to seamlessly interact with the application’s core functionality. This includes providing intuitive controls, responsive layouts, and visually appealing components.

Features such as dynamic updates, and error handling will be implemented to enhance usability. The UI will also incorporate proper validations and feedback mechanisms to guide the user through various tasks effectively.

Also, each wave file can be is drawn as an amplitude vs time chart for easy visibility during editing.

* **Backend:** C++

The C++ portion of the project involves implementing the main logic for the audio effect. Each effect has been implemented through functions that take the minimal amount of information from the user as parameters.

The functions implemented to edit the audio have been written from scratch and do not use any external libraries.

* **Middleware:** JNI for communication between Java and C++

Creates a Java wrapper to bridge the Java frontend with the C++ backend. This is done by declaring native methods to call C++ functions.

* **Development Platform:** MacOS, Ubuntu
* **Tools:** OpenJDK 22.0.1, g++, NeoVim, IntelliJ, VS Code, Git

## Input/Output Requirements

* **Input:** Audio files in “.wav” format.
* **Output:** Modified audio clips are saved in the same format.
* **User Input:** UI controls for playback, amplitude/frequency adjustments, and clip editing.
* **Output Display:** Visual feedback on the audio waveform, playback status, and applied effects.

## Functional Requirements

* **Amplitude Scaling:** Allow users to modify the amplitude of audio clips.
* **Compression:** Allow users to apply dynamic range compression based on a specified threshold and ratio.
* **Frequency Scaling:** Enable frequency changes for pitch adjustment.
* **Looping:** Users can loop selected portions of audio clips.
* **Reverb:** Allow users to modify the persistence factor of an audio clip.
* **Time-Scaling:** Allow users to alter the duration of the clip.
* **Clip Trimming:** Provide options to cut or trim parts of the audio.
* **Clip Superimposition:** Merge multiple audio clips into one.
* **Audio Filter:** It helps in setting a threshold for the frequency of the audio

## Use Case

* **Play and Pause Audio:** The user plays or pauses audio clips through the UI.
* **Multi-Track Editing:** Edit Multiple audio files and layer them onto a single output file.

# Non-functional Requirements (Optional)

* **Performance:**
  + Audio playback should not exceed a **100ms response time**.
  + It can process a **30min audio file** within **1 second**
* **Usability:**
  + The UI should be **intuitive and responsive** with clear feedback.
* **Security:**
  + The application should ensure **file safety** and prevent corrupt audio files.

# Development Setup

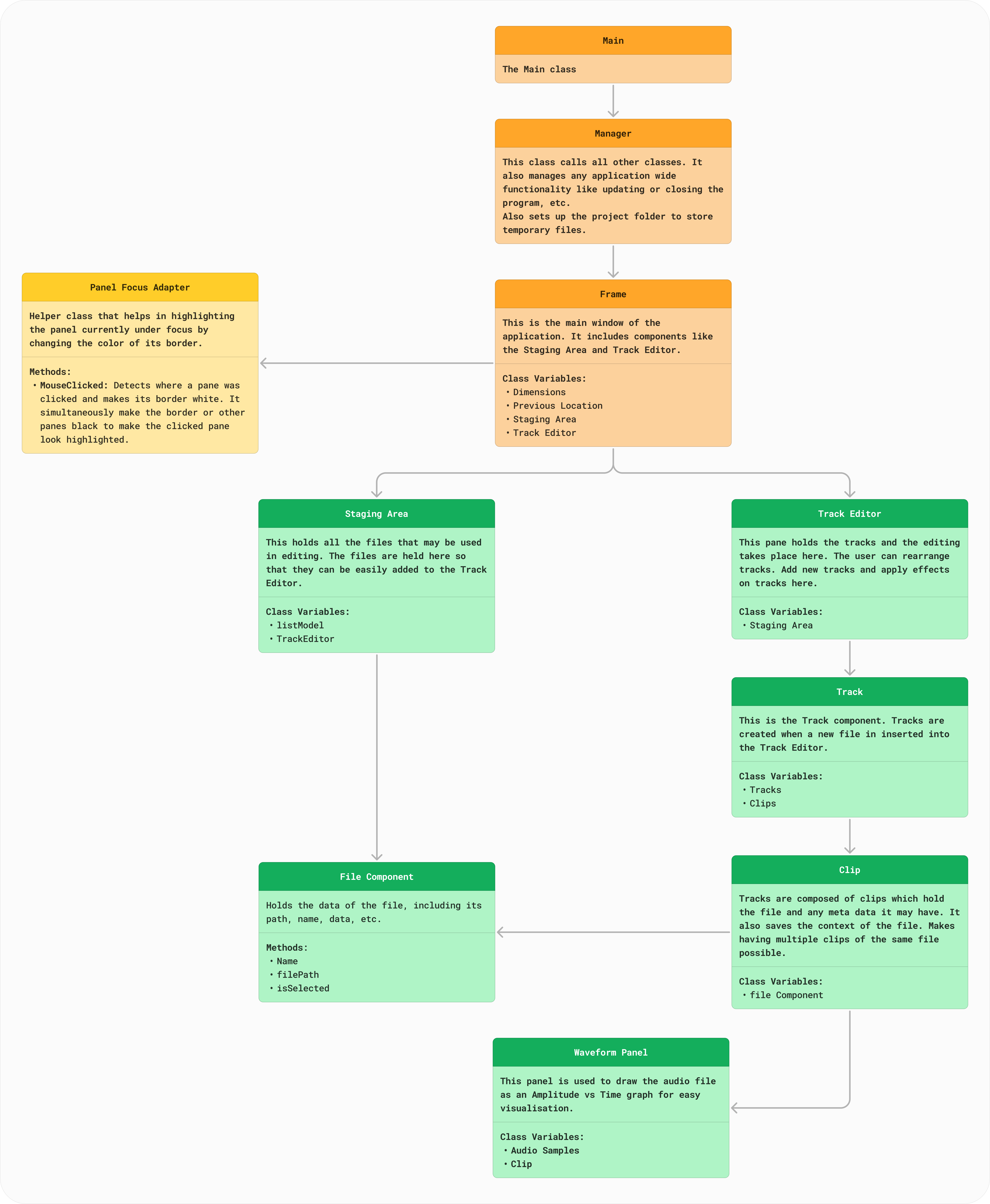
* Repository Information:
* URL:[*The\_Kitchen\_Project*](https://github.com/Dheeraj-Murthy/AudioEditor.git)
* SSH: *git@github.com:Dheeraj-Murthy/AudioEditor.git*
* *S*teps to Clone:
* Run the command: *$ git clone https://github.com/Dheeraj-Murthy/AudioEditor.git*
* Configure the build environment, ensuring java and g++ are installed and the sdk is set up.
* Open the Project in IntelliJ. Run Main.java.
* You might need to compile the c++ code if you are using a windows system or a linux distribution other that ubuntu.

Refer to [link](https://www.baeldung.com/jni) for detailed steps to do so.

# Workflow

* **UI Development**:
* Build the frontend in Java using Swing.
* Create user-friendly controls for playback, amplitude scaling, and clip editing.
* **Audio Processing Implementation**:
* Implement audio manipulation functionalities in C++.
* Example functions include amplitude scaling, frequency manipulation, and reverb.
* **JNI Bridge**:
* Use JNI to enable communication between Java and C++.
* Write Java native methods that invoke the C++ backend logic.

# UML Diagram:



# Important Files & Folders

* **native:**It contains the entire C++ implementation of the functions that can be used to edit the audio tracks.
* **src/main/java/:** This folder contains all the java code.

# Testing

* **Unit Testing**:
  + Test individual functionalities like amplitude scaling, trimming, and looping.
* **Integration Testing**:
  + Validate the interaction between the Java frontend and C++ backend through JNI.
* **Performance Testing**:
  + Measure the time taken to process large audio files.
* **Error Handling**:
  + Test the robustness of exception handling mechanisms for invalid inputs.

# Conclusion

The project successfully combines the strengths of Java and C++ to create a feature-rich audio editor. With an intuitive user interface and efficient backend logic, it provides essential audio editing functionalities for users.

Future extensions may include advanced audio effects, real-time processing, and support for additional file formats. By leveraging JNI, the project bridges the gap between platform-independent UI development and high-performance native processing, ensuring a robust and extensible system.