Functional Specification

Name: Darragh Connaughton **Student Number:** 13469978

Contents

1 Introduction

1.1 Overview

The musical notepad is an application aimed at the avid musician who wishes to document and store musical ideas, whenever they may arise. Keeping track of musical ideas can be a time expensive process. A fleeting phrase may be lost in the time it takes to gather up the necessary resources to capture it. A musically illiterate musician is at a distinct disadvantage here, with an inability to effectively capture and transmit their ideas in written form.

The musical notepad aims to cross this divide, bridge the gap between music in its purest sense and its technical notation. Offer musicians with any level of musical theory knowledge the ability to capture and share their ideas. The musical notepad will record audio through a smartphone microphone, process the recorded audio and convert it into musical notation.

1.2 Business Context

The target audience would be traditional musicians. The tradition of Irish music is often passed down by ear or using letter notation, leaving many musicians unable to read musical notation. The application could find a platform for any musician who wishes to document their progress.

1.3 Glossary

Fast Fourier Transformation:
Fast Fourier Transformation is an algorithm that computes discrete Fourier Transformation of a sequence. It essentially converts a signal from its original domain to a representation in the frequency domain.

Linear PCM:
Linear PCM is a lossless compression data format. Lossless in the sense that it is not compressed. It is the audio format used in standard CDs as it retains all the detail from the recording.

WAV:
WAV stands for Waveform Audio File Format. It is an audio file format for storing an audio bit stream.

Harmonic Product Spectrum:
The harmonic product spectrum is a spectrum containing only frequency components whose frequencies are whole number multiples of the fundamental frequency.

2 Description

2.1 Product / System functions

The application will be available on android, designed with a smartphone in mind. The layout shall be as easy to use as possible. Upon opening the application the user will be presented with a simple start menu offering two options:

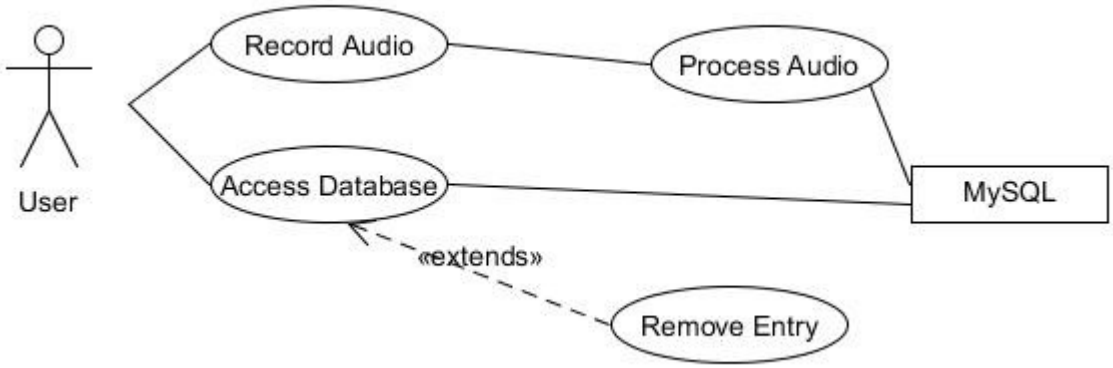
- **Record.**
  - This option will allow our user to record their next song. The interface here will be a record and stop button. This interface will also contain a menu option which will return the user to the home page.
  - Recorded audio will be converted into the frequency domain by the PCM->Frequency Domain Converter and subsequently converted into its respective pitches. This output is then stored within a database for future use.
- **Access previous recordings.**
  - This button will prompt an interface that will show the user all of their previously recorded material. This section will be connected to a database that will be used to store all of the user data.

2.2 User Characteristics and Objectives

My target audience is Irish traditional musicians, particularly those who are unable to read musical notation. Although this is my main focus, the scope of this application extends towards any musician who is interested in documenting their progress. There are two main aims for the musical notepad, them being speed and accuracy. The musical notepad is here to replace pen and paper. If this goal is to be achieved it must be able to capture an idea quicker than a pen and paper can. The speed difference here must be notable in order for the musical notepad to have a worthwhile purpose. I expect my target audience will want to capture their idea as quickly and efficiently as possible and continue on with their creative process.

Accuracy is the main focus of this application. Although speed is a key component in the musical notepad, it would be worthless if the output is inaccurate. Our users want to capture their idea in one go, quick and effortlessly. Time and patience will be lost if multiple attempts are required to capture a single melody line.
The end product will be connected to a database,. This will allow our musicians to store their fleeting ideas, which may be used at a future date. Over a series of months a user may have gathered enough musical segments to create a full song.

2.3 Operational Scenarios



Examples of some potential use cases:

- User records a segment of audio. Backend processes the sound, passing it to each individual component. The user has the option on screen to save the recording.
- The user opens the application and accesses a previously stored recording. The database is queried and the appropriate value is returned. The user, not pleased with this recording, decides to delete it from the database. The database is updated and the unwanted input is removed.

2.4 Constraints

Time Constraint
With the limited time provided for the project and the complexity it contains, time may prove to be a fleeting resource. Time management will be essential in order to fully complete this project.

Hardware
The musical notepad is developed for the microphone present on your standard smartphone. The sound quality of the recording is limited by this factor. Since the accuracy of the final pitch detection is dependent on the quality of the recording, this may prove debilitating.

3 Functional Requirements

3.1 Lossless audio recording

Description:
In order to accurately extract information from the audio recording, we will need the most detailed audio recording we can find. Due to the size of an audio file, they are often stored compressed to save room, this is known as lossy compression. A lossless recording will retain the maximum amount of information possible.

Criticality:
This section is rather important for receiving maximum accuracy. The higher the detail of the audio file, the more data that will be extracted from it. A compressed audio file may leave very little to work off.

Technical issues:
Android studio supports Linear PCM, which is a lossless audio recording format. Reading audio in through Linear PCM should guarantee a highly detailed, uncompressed audio file.

Dependencies with other requirements:
Converting PCM to WAV is fully dependent on this step. Without this format, another new step would be required to fill this void. The conversion from to the frequency domain is reliant on this step. A lossless compressed audio format will yield greater detailed output in the frequency domain. A lossy compression audio format will lead to substantially less detailed output in the frequency domain. The level of detail

within the frequency domain will determine the accuracy of the results we receive from the pitch detection step.

3.2 Convert PCM into the frequency domain using FFT

Description:
Using Fast Fourier Transform, the PCM audio file recorded will be converted into the frequency domain. FFT will compute the discrete fourier transform sequentially over the PCM file. This will convert the signal from the original domain into the frequency domain.

Criticality:
This is a mandatory step. Without this transformation it will not be possible to extract the frequencies from the data.

Dependencies with other requirements:
The step following this is fully dependent on the output from this section. We require our audio to be represented in the frequency domain if we are to perform pitch detection.

3.3 Pitch detection within frequency domain

Description:
Extract a sequence of pitches from the frequency domain. This will be achieved using the harmonic product spectrum. This section will be computed using the FAUST programming language.

Criticality:
This is the primary feature of the application. Without the successful extraction of the pitches from the frequencies, the application is practically worthless.

Technical Issues:
Accurately detecting pitch from the frequency domain is where the majority of the complexity for this project lies. Background noise removal is essential for isolating the frequency produced by the musician.

Dependencies with other requirements:
Pitch detection can’t occur unless all previous steps have been completed successfully. Likewise, all previous sections hold no value if this section isn’t complete.

3.4 MySQL Database

Description:
A MySQL database will store all previous recordings made by the user for future use. The database may be accessed by the user to retrieve or delete their previous recordings.

Criticality:
This is a mandatory feature. The musical notepad is a documentation and archiving tool. For this purpose, it needs a form of storage.

3.5 User Interface

Description:
An easy to navigate user interface that allows the user to catalogue their previous converted audio and the option to record new audio. The user interface is clutter free and intuitively easy to understand.

Criticality:
This functional requirement is essential. Recording an audio track and accessing previously processed audio must be seamless. Development is made with a non-technical user in mind. The user interface brings all the components together into an easy to handle package.

Technical Issues:
Connecting the backend to the frontend will prove challenging. Design is another factor that needs to be taken into account here. There needs to be a concise layout that must be accessible for all users.

Dependencies with other requirements:
The user interface is essential for bringing all components together. The initial recording is initiated through a button on screen. Processed data is cataloged and accessed through a convenient user interface as well.

4 System Architecture

4.1 Android Studio

I have decided to use Android studio as my integrated development environment. Android studio is google official IDE for developing android. Android studio is based off IntelliJ IDEA and builds using gradle. Android studio allows development in C++. The FAUST programming language compiler compiles FAUST code into C++, making both compatible with each other.

4.2 FAUST Programming Language

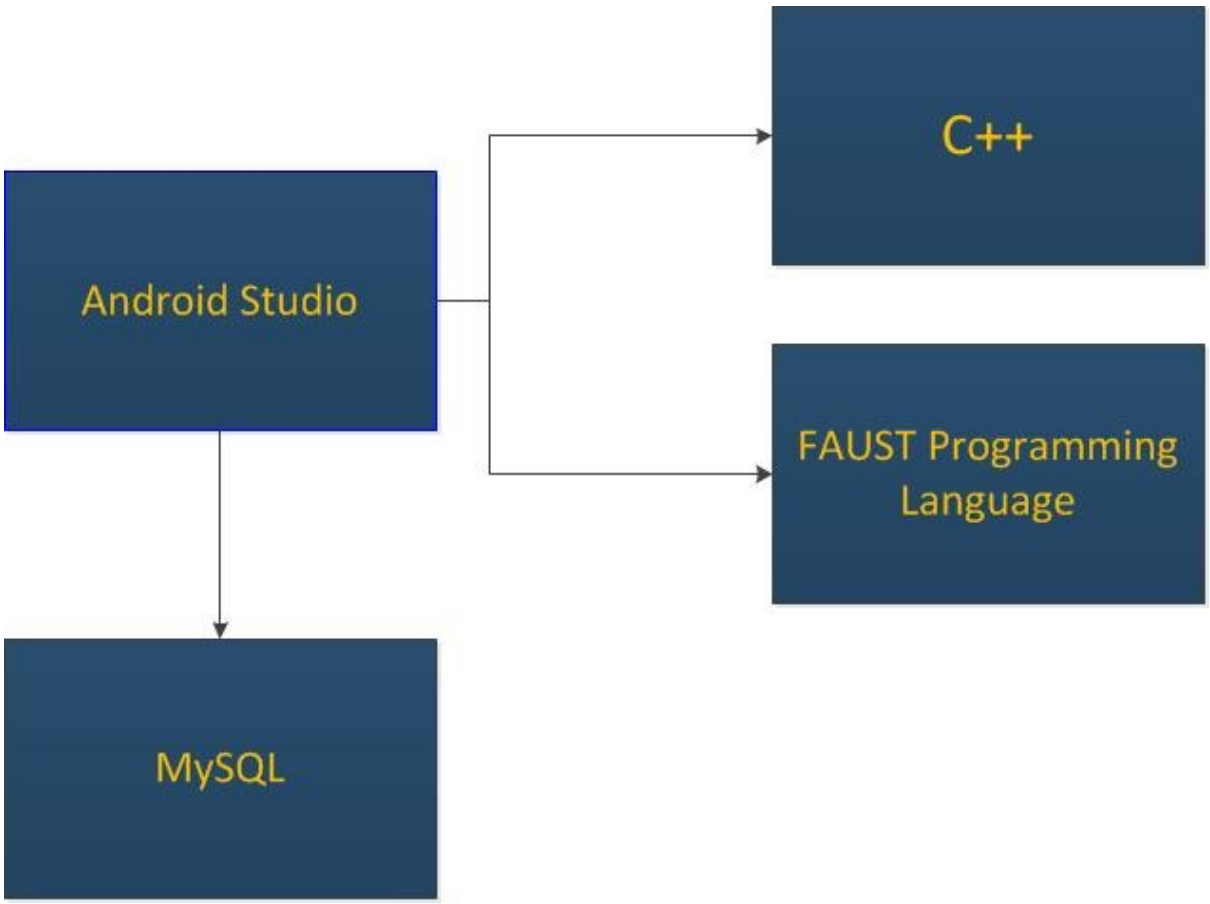
FAUST is a high level functional programming language for digital signal processing. It supports real-time audio applications and plugins from Android, as well as many other software platforms. FAUST is domain-specific language for audio and signal processing. A FAUST program describes a signal processor, a function applied to some input signal.

4.3 C++

C++ is a generic, imperative and object-oriented programming language.

4.4 MySQL

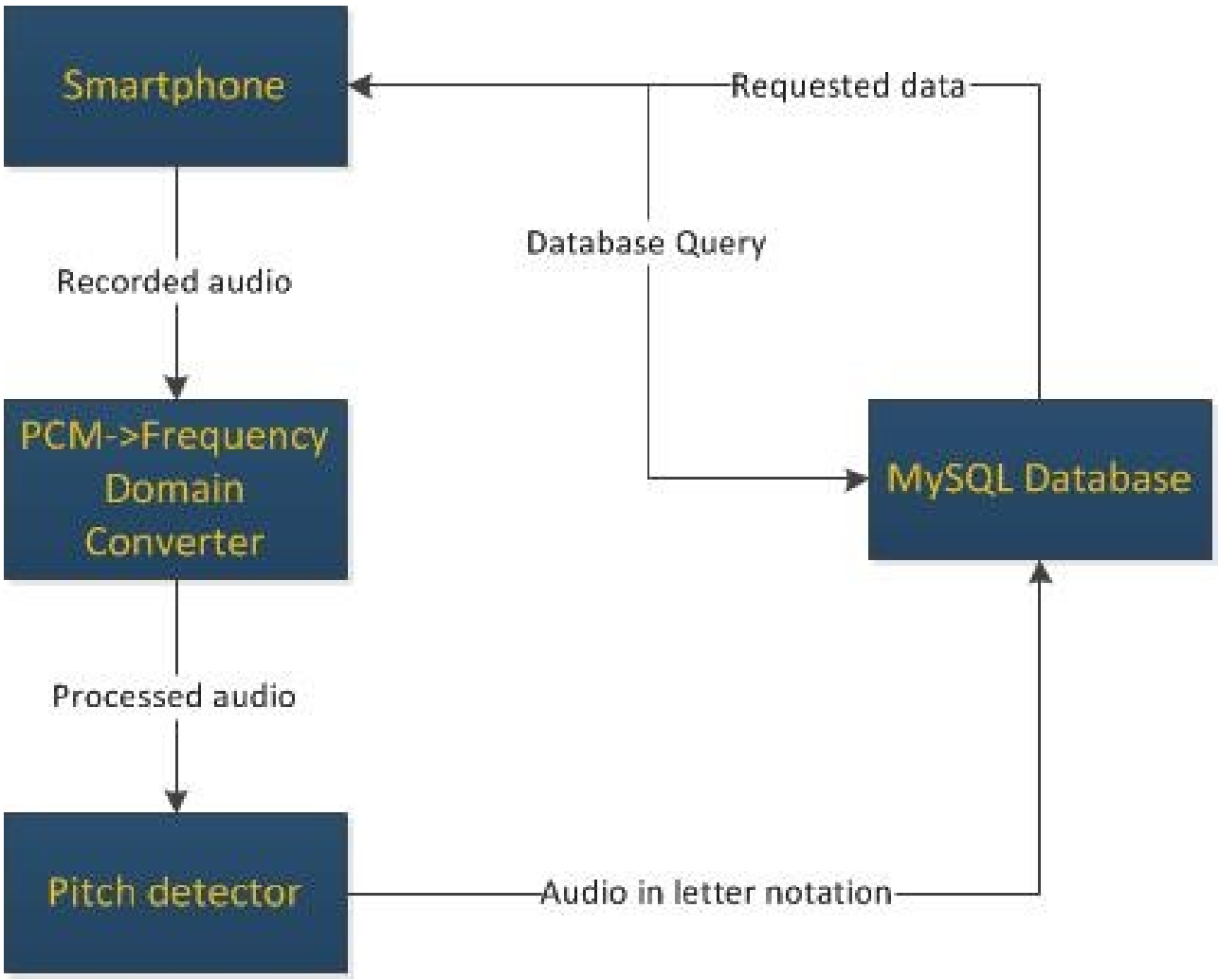
MySQL is an open source relational database.



5 High-Level Design

This section will outline the higher-level features of the musical notepad.

Data flow diagram



6 Preliminary Schedule

The development schedule is as outlined:

- Research: 2 weeks.
- Android template that reads in lossless data: 3 weeks.
- PCM->Frequency domain converter: 3 weeks.
- Pitch Detection: 16 February - 3 weeks.
- Interface and Design: 2 weeks.
- Documentation: 1 week.

Testing time is included within the estimated times above.