**Senior Design Writing Assignment 2: Functional Specification  
The following functional specification is not complete and is subject to several modifications as the project progresses.**

**Overview**

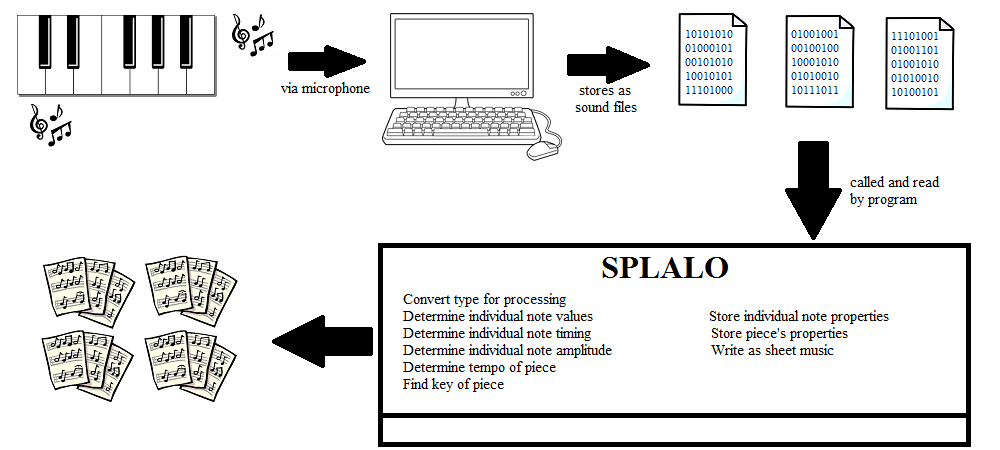
The software under consideration is intended to be a computer centered program capable of “listening” to some instrument being played and transcribing the music piece’s properties as sheet music. To implement this, the program (nicknamed SPLALO) will only make use of four basic I/O devices: one standard microphone to capture audio data, a mouse and a keyboard for in-program commands (like starting a recording or setting the recording time) and a monitor for displaying the program interface to the user.

To effectively run SPLALO one needs to place the microphone within 5 ft. of the instrument of choice and set the length of recording time before clicking the “Record” button. Once the predefined time has elapsed, the program will automatically stop recording and display some message to alert the user of this. It will store the recording as an audio file in some centralized location on the host computer (which can be played back independently from SPLALO). At this stage, the program will access the file and analyze it to determine the various properties necessary for generating sheet music. These properties include (but are not limited to) the overall tempo of the piece, individual note values, individual timing values and individual amplitude values. Once the necessary attributes have been determined, the appropriate sheet music equivalent will be generated and displayed on the screen. For pieces spanning multiple pages, the user will need to scroll through the program’s main display to view the rest of the piece.

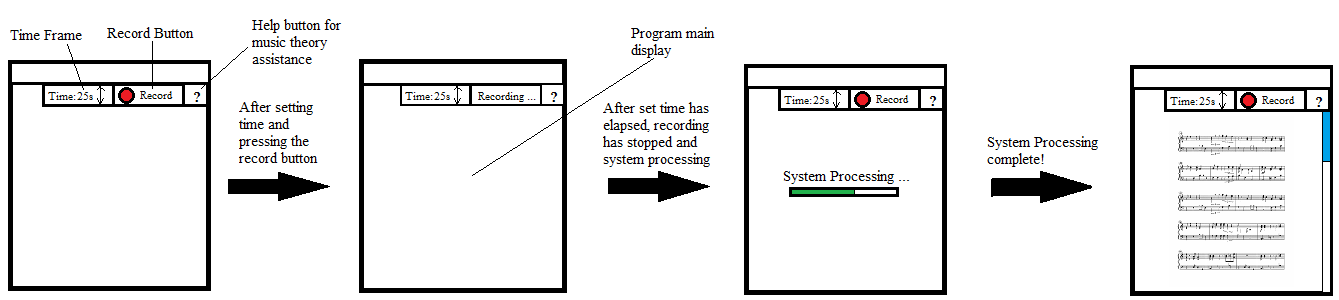
SPLALO shall be developed with a focus on users with a basic knowledge of music theory. This lower bound of knowledge is only necessary to understand the output of the system. A help guide may be provided to assist in the interpretation of sheet music for beginners to music theory.

**Graphical Representation of Processes**

The diagram below is a simplified run-through of how the program shall operate (as described in the overview):



The diagram below is a rough draft of the screen layout for the program. It shows, to a limited level of detail, how the program’s User Interface should change during standard execution.



**Recording Phase   Processing Phase**

**Scenarios**

To imagine the program in a realistic case some scenarios of use have been provided below:

Case 1: Sarah is sophomore in college with a passion for music and some knowledge of music theory. She has known how to play the piano and violin for nearly a decade. For this reason, she recently declared a major in music. One of her courses requires that she compose a piece of original music, but Sarah doesn’t know where to begin. She is used to freely playing until something comes to her as opposed to writing out ideas immediately. She also worries that disrupting this free play to jot down notes as sheet music would break her concentration.

Sarah has heard a few classmates talk about SPLALO and how it assisted with their class assignments. Sarah decides to try out the program and is impressed by the accuracy and assistance the program provides. She gets her work done more efficiently that semester and improves her grades in the class.

Case 2:

John is a young man who loves to read about brain training exercises. He has an interest in puzzles, logic games and the sort. He stumbles upon an article that cites mastery of music theory and playing instruments as effective training exercises. Although he loves listening to music, John has no idea how to play any instrument nor does he have any knowledge in music theory. He decides to start with the piano and learns a bit by watching piano tutorial videos for some of his favourite songs. Although this gets him somewhere, it does not provide the training he hoped for.

In his research into music learning tools John stumbles upon SPLALO. He tests it out by recording audio from a piano cover that he found online and is presented with a series of music sheets. Confused by the technical notation he enlists the assistance of SPLALO’s help tool which explains the notation to him. John becomes an avid user of the program in his learning process and finds himself practicing more than he did prior to discovering the program.

**Details**  
 The following section provides a high level description of the individual tasks executed by SPLALO. These may serve as guides to programmers during the development stage or as stepping stones for technical writers when manuals and other documents are being drafted. It will not include an exploration of the algorithms and data structures implemented nor will it include time and space complexity analyses. For notes on these concepts and their relation to the program a technical specification may be drafted.

Recording the Audio Data (Recording Phase):

The default recording time will be set to 0 seconds to ensure user interaction. If the user does not change this time and clicks the Record button, the program will not run and may print some message to the screen. If the user attempts to set the recording time to a negative value the system will not run.  
 Assuming a valid recording time is entered, SPLALO will attempt to enter its recording phase. In this state it aims to store the detected amplitude values at periodic times (based on programmed sample frequency). If some error occurs with reading or storing these values the system will methodically handle it and alert the user of the issue.If the system is able to read and store the audio input, it will enter its processing phase (as illustrated by the 3rd screen in the screen layout diagram).  
  
Reading and Processing from Sound File (Processing phase):  
 As part of the processing phase, the program will attempt open the recently created audio file for processing. If it fails to open the file, it will securely restart this opening process and do so until the file is safely accessed. It will go on to convert the data from the file into a processable format (from big endian/little endian byte format to double format). If done successfully, the procedures carried out from this point should yield accurate results.

SPLALO will simultaneously process properties for portions of audio data (which will represent notes); these properties are the frequency for individual note values, time durations for timing values and amplitude for additional notation. Each note’s properties will be stored collectively to simplify the writing process. With each individual note processed, SPLALO will use the collection of note values to determine the key of the piece and the collection of timing values for tempo and time signatures. This will continue until the end of the file has been reached.

The sheet music will be written by writing out the tempo, time signatures and key first. Then it will transcribe each notes properties appropriately, starting from the first note processed until the end of file has been reached.  
 As these processes occur, the main program screen will display some status bar to notify the user of where the system is in processing the audio file.

**Non-Goals**

The non-goals mentioned below may be implemented in further stages or future version once the core functionality of the program has been established:

**1)** SPLALO will not implement real time audio processing and transcription for audio input. Alternatively, it will “listen to” and save all input as sound files from which processing may be executed.

**2)** The recording portion of the program will not run indefinitely. This is a step taken to appropriately initialize audio buffers and effectively test the core functions of the program.

**3)** SPLALO will not simultaneously process input from multiple sources. It will only analyze and write out sheet music for one sound file upon user request. When a user requests the analysis of another sound file, the one currently being displayed (if any) will be cleared and results from the new one shall replace it.