A-Level Coursework

# Analysis

# Documented Design

## Overall System Design

My solution will consist of 3 main components, the practice section, where the student is shown: the sheet music for the piece and the fingering for the instrument that they are playing. The other component will be the testing section where the student plays along, like in the practice section, however the program will listen to the audio and mark the student biased on how well they have played the song. The 3rd component will be the database and the system where the music is stored for the student to practice with or test themselves with

A possible extension would be a database for the students to track their scores over time, however due to time constraints, it is unlikely that this will be implemented in this version of the program.

### Componant 1 – Practice

* The student selects a piece of music to play from a list of catalogued pieces of varying difficulties
* The student is shown the sheet music and a selectable amount of learning aids. These may include the fingering for each note, a metronome to keep time, a cursor to show at what point in the song they should be on, ect. Having a selectable amount of help, will allow students who are just starting off to get used to playing the instrument. These can then be removed to allow the student to push themselves.
* The student can then play the music as many times as they please until they feel that they have mastered the piece

### Componant 2 – Testing

* The student is again allowed to select the music they wish to play from the list of pieces
* They select how many learning aids they wish to use to help them and then they play along
* The program uses the microphone to record the audio from the instrument. This allows the program to process the sound from the instrument to work out what note was being played, what time the note started and what time it ended. This can then be compared to the note that should have been played as stated by the music that the user selected to play. If the right note was played, at the right time until the right time, then the user would be given a higher than if the note was played wrong, or at the wrong time.
* At the end of the song, the user is shown the score they reciveced for each metric and shown which notes in the song that needs improving.
* The user would then be able to store in a simple database what score they got for each metric, the song they played and what helping aids they used. This database can be accessed later to see how they have progressed.

### Componant 3 – Database of music

This component will be responsible for storing the songs that are available to the student to play and practice with. These will be specifically written MIDI files that will follow the MIDI Specification, but also meet a more specific set of criteria to be compatible with my program. Eg, all with only consist of one track chunk and only 1 instrument. This will make it easier for the program to deal with and allow the program to more easily process the file and display it to the user.

In future it is hoped that these files will be able to be created by a music teacher using existing midi software to generate these files which can then be transferred for the student to get a more tailored learning experience. However, for the time being, the program should be able to import midi files that I can create and then make downloadable.

### Componat 4 – Database of Scores

Once a student has completed a task, they should be able to store their score into a database to compare their results to later on. This is not a major component is is more of a nice to have. If a student wanted a record, they could just write it down.

### IPSO chart

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Process | Storage | Output |
| Midi File to display selected | Read MIDI File in | Store the MIDI file in memory | Show the user the sheet music for the midi file |
| Audio from the michropone |  | Store the input audio in a wav file not in ram |  |
| Wav file | Analyse what note was being played at each time interval | Store what notes were being played at each time in a list in RAM |  |
| List of what notes were being played at what time | Compare the list of notes to the midi file to see if the right notes were being played at the right time |  | Show the user a score to show how well they did playing the piece |

### Modular Design

* **First Form** (Main Menu)
  + Select instrument
  + Select music
  + View sheet music
    - **Second Form** (Display sheet music)
      * Next page
      * Previous page
      * Main menu
  + Select practice
    - **Third Form** (Practice)
      * Previous page
      * Previous line
      * Play/Pause
      * Next line
      * Next page
      * Main menu
  + Select Test
    - **Fourth Form** (Test form)
      * Start
      * Pause
      * Stop
      * Retry
      * Main Menu
      * Mark
        + **Fifth Form** (Result Form)

Retry

Main Menu

Quit Program

### Data Requirements

The program will need to store quite a bit of data in order to function.

This only convers the main, bulky, bits of data that will need to be processed. Is not an exhaustive list of all variables.

|  |  |  |  |
| --- | --- | --- | --- |
| What is being stored | Why it is being stored | Where is it being stored | How long will it be stored |
| Selection of songs for the student to play | The songs will need to be stored for the user to play back when they need them | The songs will be stored in a series of folders which will be linked to through a small database | The songs will need to be stored for as long as the user wants, across sessions. |
| Audio input from the microphone | The audio from the microphone needs to be stored so that the program can run through and process it to find what notes were being played at what time to give the student a score | The recorded audio will be stored in a wav file in the main program storage to make sure a long song does not use up excessive amounts of ram | The audio will only need to be stored from the start of the recording to the end of the processing of the audio |
| The song that is currently being played | The will need to be stored to allow the program to generate the images of the sheet music what will be displayed to the user and also be compared to the recorded audio to find out if the student was playing the right notes at the right time | The current song will be stored in a custom class that will allow the program to access the data it needs quickly. It will be stored in the RAM | The current song will need to be stored from when it is first loaded in by the user until the program is closed or another song is selected and loaded into memory |
| The notes that were played in the recorded audio | After the audio has been processed to find out what notes were being played when, the notes that were being played need to be stored to be compared against what the midi files says should have been being played | This will be stored in a list in the RAM for quick access and easy processing. | The notes that were being played will only need to be stored from when the audio has been processed until the score has been given to the user |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Data Validation

The program will need to validate the MIDI file as it is being inputted into the program to make sure it is the right format and only contains 1 instrument track to make sure that a error is not thrown later down the line when the file is being processed. If the file is the correct type and is not corrupt, then the program will load the file into memory and then allow the program to continue.

For anything the else that the user will be selecting, I will be used controls where the user does not get to type in their own values. Eg, drop down lists and sliders. This limits the amount of validation that I will have to perform on the data and make the program more robust.

## Preliminary Form Sketeckes

### Form 1 – Main menu

btnOpenFile

lblInstument

btnTest

btnDisplayMusic

btnQuit

btnPractice

btnOpenFile

Select a song:

Instrument: \_\_\_\_\_\_\_

Test

Quit

Practice

Look at music

Open File

###### Process

* Shows the open file dialogue
* Allows the user to select a Midi File to open
* Converts the MIDI file to a byte stream
* Reads the head chunk to make sure the file is compatible, displaying an error if it is not
* Reads the rest of the file parsing it into the class that with store the file, catching any errors that occur
* If it’s successful, then the instument that the file is written for will be displayed on “lblInstument”

###### Pseudo Code

*Show OpenFileDialogue*

*byteStream = selectedFile*

*FOR i = 0 to 13 LOOP*

*headChunk = headChunk + readByteFromStream(i)*

*ENDFOR*

*IF verifyFormat(headChunk) = true THEN*

*midiObjectParse(byteStream)*

*ELSE*

*Message “Format not compatible”*

*ENDIF*

btnDisplayMusic

###### Process

* If file has been selected,
* Open DisplayMusicForm and pass through midiFileObject
* Hide main Menu

btnPractice

###### Process

* If file has been selected,
* Open Practice form and pass though midiFileObject
* Hide main Menu

btnTest

###### Process

* If file has been selected,
* Open Test form and pass though midiFileObject
* Hide main Menu

btnQuit

###### Process

* End the program