**DESIGN OF TYPING AND LISTENING TEST APPLICATION USING LabVIEW**

Submitted in the partial fulfilment

of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**Electronics and Communication Engineering**

By

P.SUSEEL KOUSIC 190040415

under the guidance of

**Madhav Khanal Sir**

**ASSOCIATE PROFESSOR**

### 

### **DEPARTMENT OF ELECTRONICS AND COMMUNICATION**

**KLEF**

**K L E F**

**DEPARTMENT OF ECE**



**DECLARATION**

I P.SUSEEL KOUSIC (190040415) student of B. Tech under Department of Electronics and Communication, K L University, Vaddeswaram , hereby declare that all the information furnished in this report is based on our own intensive research and is genuine.

This report does to the best of my knowledge, contain part of our work which has been submitted for the award of our degree either of this university or any other university without proper citation.

NAME ID

P SUSEEL KOUSIC 190040415

**K L Education Foundation**

**Department of ECE**

­



**CERTIFICATE**

This is to certify that the project based laboratory report entitled “ **DESIGN OF TYPING AND LISTENING TEST APPLICATION USING LABVIEW**” submitted by P.SUSEELKOUSIC 190040415 to the **Department of Electronics and Communication Engineering, KL University** in partial fulfilment of the requirements for the completion of a project based Laboratory in “SKILLING” course in II year B Tech II Semester, is a bonafide record of the work carried out by him/her under my supervision during the academic year 2021-22

**Signature of the Supervisor Signature of the HOD**

# **ACKNOWLEDGEMENT**

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satya Narayana**, for giving the opportunity and platform with facilities in accomplishing the project report.

We express the sincere gratitude to our principal **Dr. K. Subba Rao** for his administration towards our academic growth.

We record it as my privilege to deeply thank our pioneer **Dr. Suman Maloji,** Head of the Department for ELECTRONICS & COMMUNICATION ENGINEERING for providing us the efficient faculty and facilities to make our ideas into reality.

We express sincere gratitude to our research head for his leadership and constant motivation provided in successful completion of our academic semester.

We express my sincere thanks to our project supervisor **Mr. Madhav Khanal** for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

NAME ID

P. SUSEEL KOUSIC 190040415

**List of Content**

1. **Introduction**
2. **Functions Used**
   1. **Flat Sequence**
   2. **While loop**
   3. **Boolean Control**
   4. **Local Variable**
   5. **Comparison Functions**
   6. **Numeric Functions**
   7. **Array Functions**
   8. **File I/O, String, Sound File Functions**
3. **Mode of Operations**
   1. **Point Counter Vi**
   2. **Choose File Vi**
   3. **The main Vi**
4. **Front and Block panel**
5. **Results and Conclusion**
   1. **Results**
   2. **Conclusion**
   3. **Reference**
6. **Introduction**

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments. The graphical language is named "G"; not to be confused with G-code. The G dataflow language was originally developed by LabVIEW, LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of operating systems (OSs), including Microsoft Windows as well as various versions of Unix, Linux, and macOS. This Graphical programing is quite use full when it comes to real-time and large applications as LabVIEW consists of highly complex functions like signal simulation, manipulation, data analysis. The one of the useful feature is it can be integrated with mat lab and code can be analysed.

In this project we will be making any application which will be helpful for our practice of on board examination which has listening text and we can improve our typing speed with typing speed test. So basically we have designed a application which will be providing us the score based on the data that we have entered which is displayed in the display box and the second type is we will be given a text with blanks and we need to fill them by listening to the audio description that would be played and the final score will be displayed.

**2. Functions Used**

**2.1. FLAT SEQUENCE**

Consists of one or more sub diagrams, or frames, that execute sequentially. Use the Flat Sequence structure to ensure that a sub diagram executes before or after another sub diagram. Data flow for the Flat Sequence structure differs from data flow for other structures. Frames in a Flat Sequence structure execute from left to right and when all data values wired to a frame are available. The data leaves each frame as the frame finishes executing. This means the input of one frame can depend on the output of another frame. When you add or delete frames in a Flat Sequence structure, the structure resizes automatically. You cannot drag tunnels across the frames of a Flat Sequence structure. To avoid overusing Flat Sequence structures, attempt to control the data flow of your VI by establishing data dependency or using flow-through parameters.

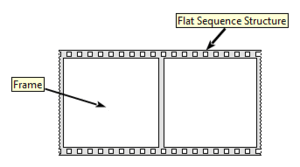


Figure 2.1.1(Basic image of Flat sequence)

**2.2. WHILE LOOP**

Repeats the code within its sub diagram until a specific condition occurs. A While Loop always executes at least one time.

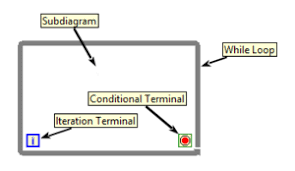
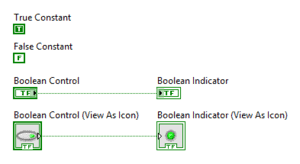


Figure 2.2.1 (Basic Image of While loop)

**2.3. BOOLEAN CONTROLS**

Use the Boolean controls located on the System, Classic, Express, and Modern palettes to create user interfaces.



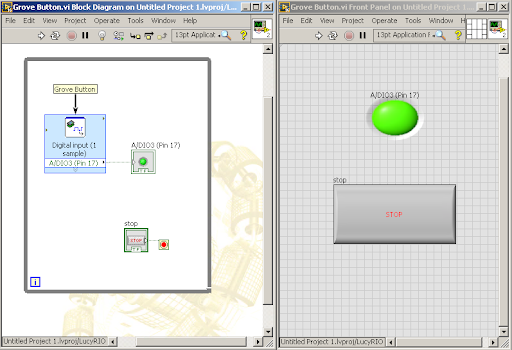


Figure 2.3.1(Basic image of boolean control and indicator)

**2.4. LOCAL VARIABLE**

Use local variables to read or write to one of the controls or indicators on the front panel of a VI. When you create a local variable, a local variable icon for the object appears on the block diagram. Writing to a local variable is similar to passing data to any other terminal. However, you can write to a local variable even if it is a control or read from a local variable even if it is an indicator. In effect, with a local variable, you can access a front panel object as both an input and an output.

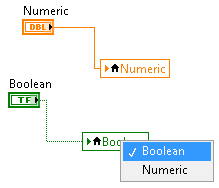


Figure 2.4.1(Image of local variable of a Boolean and numeric in block panel)

**2.5. COMPARISION FUNCTIONS**

Use the Comparison functions to compare Boolean values, strings, numeric values, arrays, and clusters. The Comparison functions treat Boolean, string, numeric, array, and cluster values differently. You also can use the Comparison functions to compare characters. You can change the comparison mode of some Comparison functions.

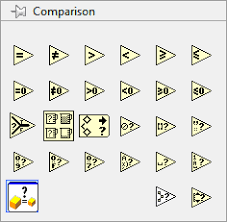


Figure 2.5.1(All the functions present in comparison palate)

**2.6. NUMERIC FUNCTIONS**

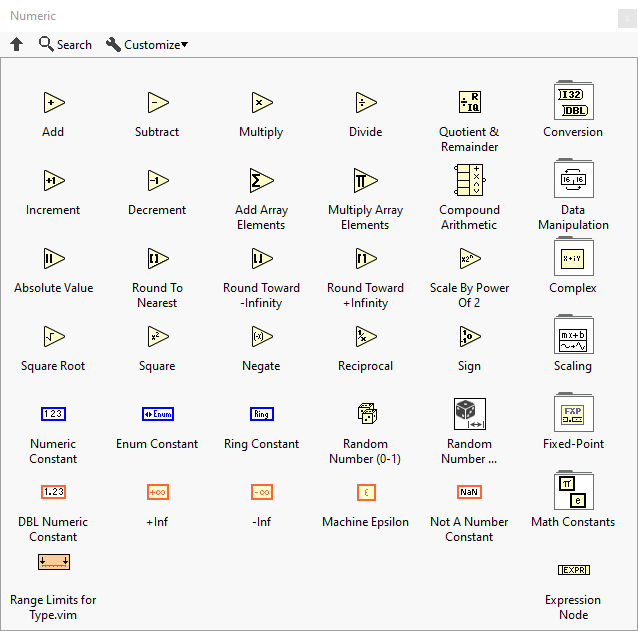
Use the Numeric functions to create and perform arithmetic and complex mathematical operations on numbers and to convert numbers from one data type to another. Use the VIs and functions on the Elementary and Special Functions and VIs palette to perform trigonometric and logarithmic functions.

Figure 2.6.1

(All the numeric functions available in numeric function palate)

**2.7. Array Functions**

Array Constant: Use this constant to supply a constant array value to the block diagram.

Array Size: Returns the number of elements in each dimension of array.

Array to Cluster: Converts a 1D array to a cluster of elements of the same type as the array elements. Right-click the function and select Cluster Size from the shortcut menu to set the number of elements in the cluster.

Build Array: Concatenates multiple arrays or appends elements to an n-dimensional array.

Index Array: Returns the element or subarray of n-dimension array at index.

**2.8. File I/O, Sound File, String and cluster Functions**

Play waveform: This vi plays the data that has been passed it as input

Sound File Read Simple VI : Reads data from a .wav file into an array of waveforms. This VI automatically opens, reads, and closes the .wav file.

Read from Text File Function: Reads a specified number of characters or lines from a byte stream file. This function does not work for files inside an LLB.

One Button Dialog Function: Displays a dialog box that contains a message and a single button

String To Path Function: Converts a string, describing a path in the standard format for the current platform, to a path.

Scale Delta t VI: Multiplies the delta t component of the waveform by the specified scale factor. Generally, this lengthens or shortens the sample rate of the waveform. The data type you wire to the waveform in input and the data type of the Y component of the analog waveform determine the polymorphic instance to use.

Match Pattern Function: Searches for regular expression in string beginning at offset. If the function finds a match, it splits string into three substrings. A regular expression requires a specific combination of characters for pattern matching. This function gives you fewer options for matching strings but performs more quickly than the Match Regular Expression function.

Unbundle By Name Function: Returns the cluster elements whose names you specify.

Invoke Node: Invokes a method or action on a reference. Most methods have associated parameters

Property Node: Gets (reads) and/or sets (writes) properties of a reference. Use the property node to get or set properties and methods on local or remote application instances, VIs, and objects. You also can use the Property Node to access the private data of a LabVIEW class.

**3. Mode of Operations**

This project has been divided into 3 VI’s in which 2 are used as sub vi’s and 1 is main vi.

The 2 sub vi’s are 1)Point Counter vi 2)Choose File vi

So I will explain the working principle in simple terms.

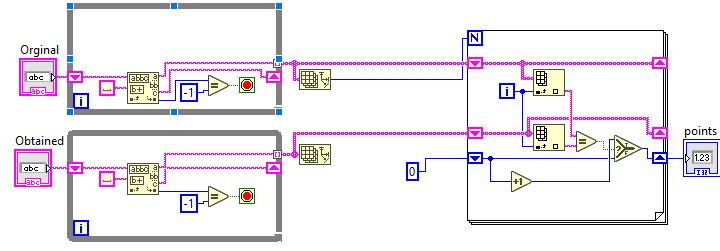
The basic logic of the typing test is to compare the two inputs and give us the score. In which one of the input is default text that has be chosen by the user and the next one is the user entered data within the timer time if they opted. These two data will be passed to the point counter vi and the score will be produced.

The basic logic of the listening test is the user will be chosen the audio type and the mode and the designated text paragraph with gaps will be displayed and the answers for the gaps will be in the audio description that runs in the background and we need to enter the gaps in the answer box. These two will be again passed through the point counter and score will be provided as same as typing test. Here we have given total of 9 different audios with 3 tones and user need to choose one of them by checking the tone at the start. The process of playing chosen tone and audio will be done by choose file vi. This will take some input arguments and provide us the path of the file to the main vi.

**3.1. Point Counter**

The main task of the point counter is to take 2 strings and delimited those strings with space constant and make an array of those splited words. So from the figure below we can understand that the size of original array with delimited words will be passed as N value for a for loop and the two arrays will be compared by taking each word from the arrays by using array index method. We will be having a shift register for the for loop and default value is 0 and the value of the register increases when each word is matched otherwise the value will be the previous value. Please refer fig.3.1.1

The point counter works differently for the typing and listening. As for listening program the vi will choose the correct answers text file form the provided path and then verifies the arrays same as the above it did for typing evalution. This will be done with the help of audio number integer. This indicator is hidden in the front panel and the value will be assigned when the user chooses the tone and difficulty level. Please refer fig.3.1.2



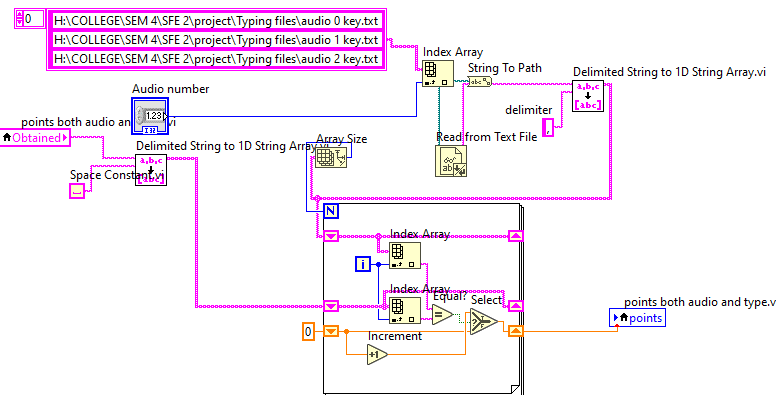
Figure 3.1.1 (Points counter for typing test)

Figure 3.1.2(Point counter for listening test)

The text file locations need to be defined as per the program location and to run the program in other source we do need to have the text files and there location need to be updated in the string array.

**3.2. Choose file**

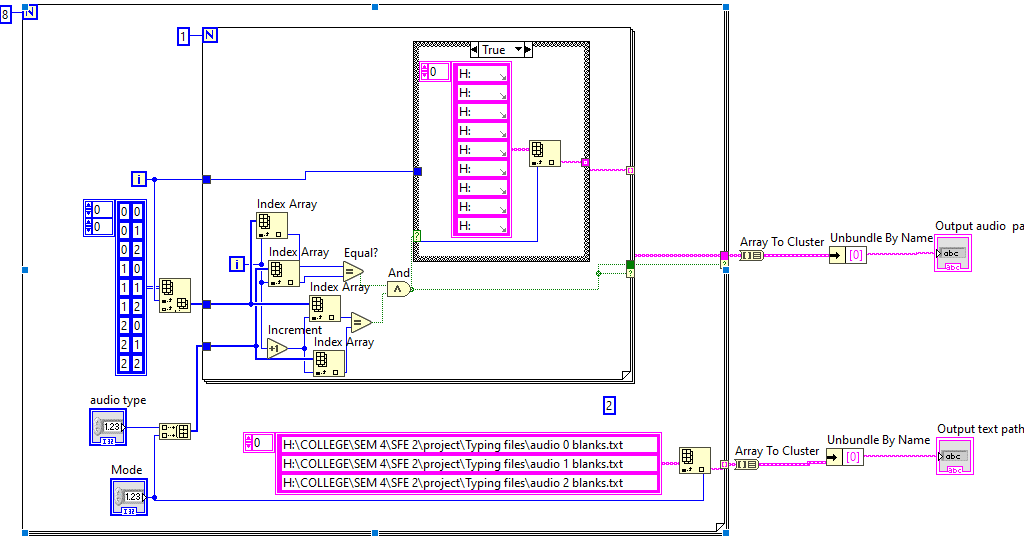
This Vi need to be understood carefully in simple manner we have created an default 2d array with 00,01,02,10,11,12,20,21,22 as we have provided total of 9 audio files with 3 different tones. Refer the fig 3.2.1

Figure 3.2.1(Choose file Block panel)

Let’s see the fig 3.2.1 and we can see there are two for loops or nested for loop in the main loop the constant 2d number array will be present. As the user proceeds through the program they would select the audio tone and difficulty these are audio type and mode. These two will be concatenated with build array function and passed through the inner loop along with the row wise output from the index array function. We have total of 9 iterations from 0 to 8 and in each iteration the inner loop iterates once and checks whether the two 1d arrays that have passed into the loop have completely equal values. If the values are same then the case structure inside the inner loop will pass the predefined path of the audio wav file. This will run all 9 times but we will have an output 1d array with one value and 8 empty strings in it. That array is converted to cluster and unbundled by name for the first index. This will be passed as the required audio path for the main Vi.

While the iterations were going on in the outer loop the text that need to be displayed will be taken from the predefined paths which are in string format in an 1d array.

**3.3. The main Vi**

This Vi is runs in a while loop which has a case structure for which input of typing or listening will be chosen form the enum ring and after choosing the mode we need to start the program and if we have selected

**3.3.1. Typing mode**

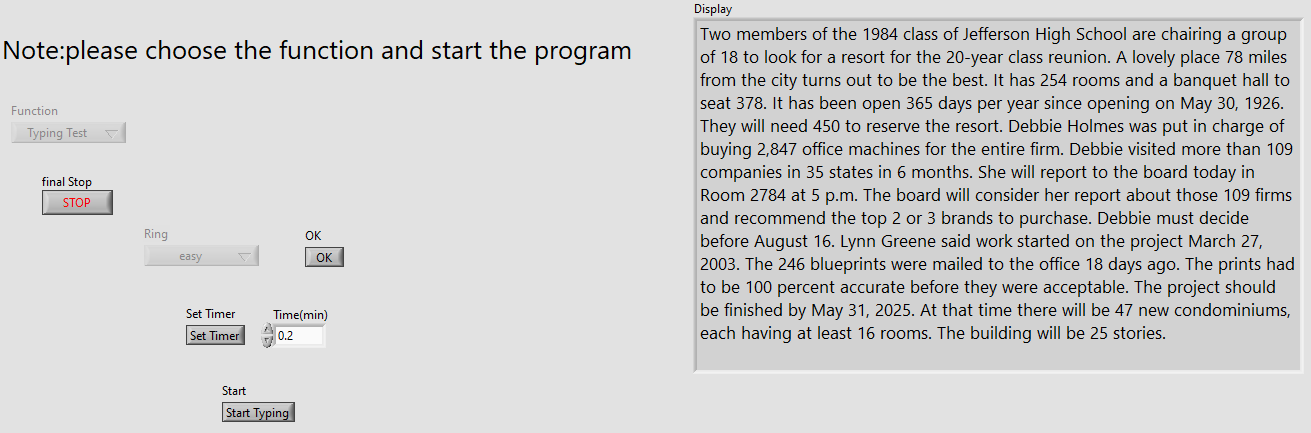
So initially the front panel will be empty by as the user goes on the option will be appeared on their required time. First we will be choosing the difficult level when we press ok button timer option and start button and display box with text will appear. Setting up the timer is upto the users wish but when the user sets the timer it will be running in the background when we start typing.

Figure 3.3.1a (Front panel of typing before starting the test)

So as from the fig.3.3.1a we can understand upto it. So when we press start typing button we will be getting an empty data entry box where we need to type the displayed text within the settled time (if activated) and after completion we can check our score of how many words were correct by pressing submit button. Refer fig 3.3.1.b .You will see your score on the right side of the panel as shown in fig 3.3.1.c

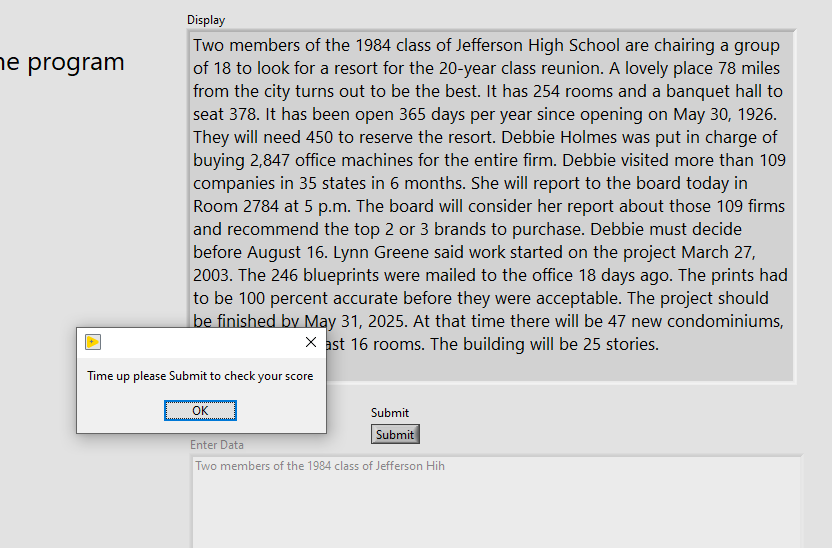
Figure 3.3.1.b (Timer disabling the entry field)



Figure 3.3.1.c (Score displayed on the front panel)

**3.3.2. Listening Mode**

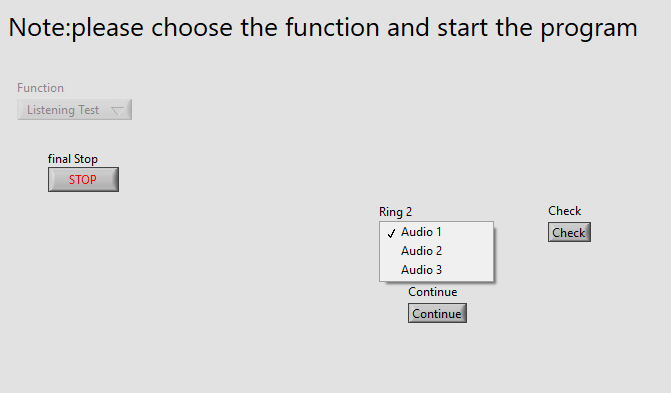
 In general listening mode is a bit complex than the typing mode as here we will be choosing the tone we want to hear by listening to it before we start the test and the program need to play and display the same file that user has chosen. As of the typing even the front panel here is empty and fields will be visible when they are required.

Figure 3.3.2a (Front panel of listening and choosing the tone we want)

After choosing the tone we are notified to select the difficulty level and speed of the audio. (Refer fig 3.3.2b) and after starting the test all the remaining fields will be disabled except the data entry field. As the audio clip with gaps will be played which will be displayed in the display box with gaps. Refer fig 3.3.2c after the audio clip completes we get an option to submit and check our score. Fig 3.3.2d

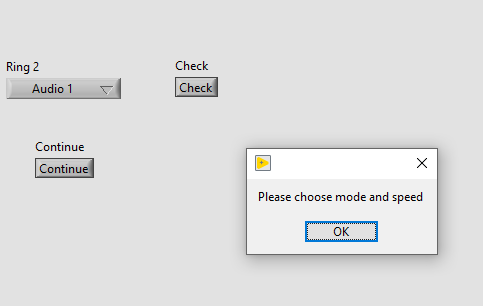


Figure 3.3.2b (FP of listening notifying to choose mode and speed

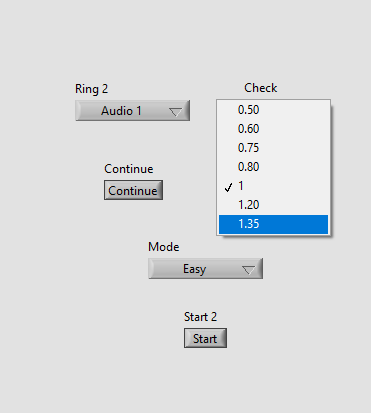


Figure 3.3.2c

(FP of the listening where we choose the speed and difficulty level)

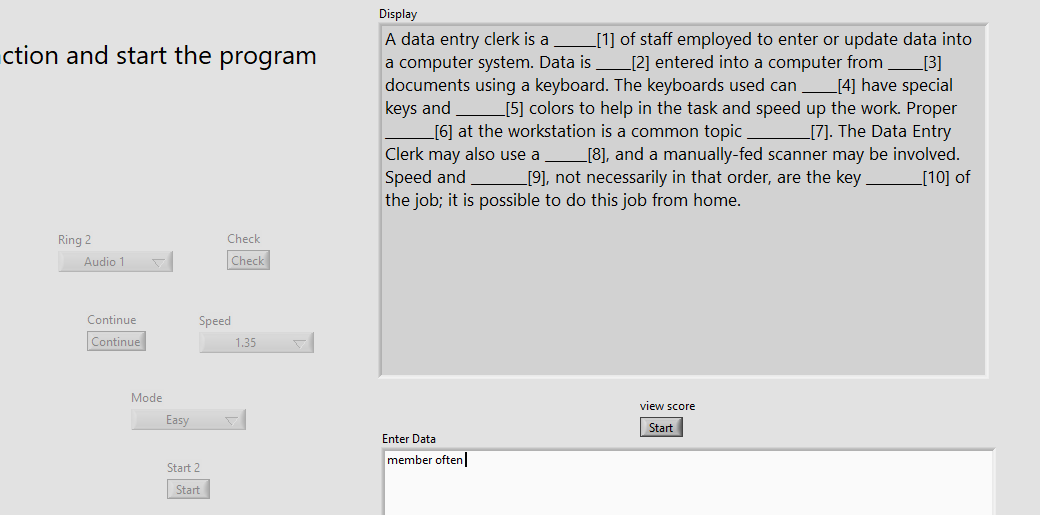


Figure 3.3.2c (FP of listening we got the submit option to view our score)

**4. Front and Block Panel**

In this chapter we will see the figures of the entire block and front panel

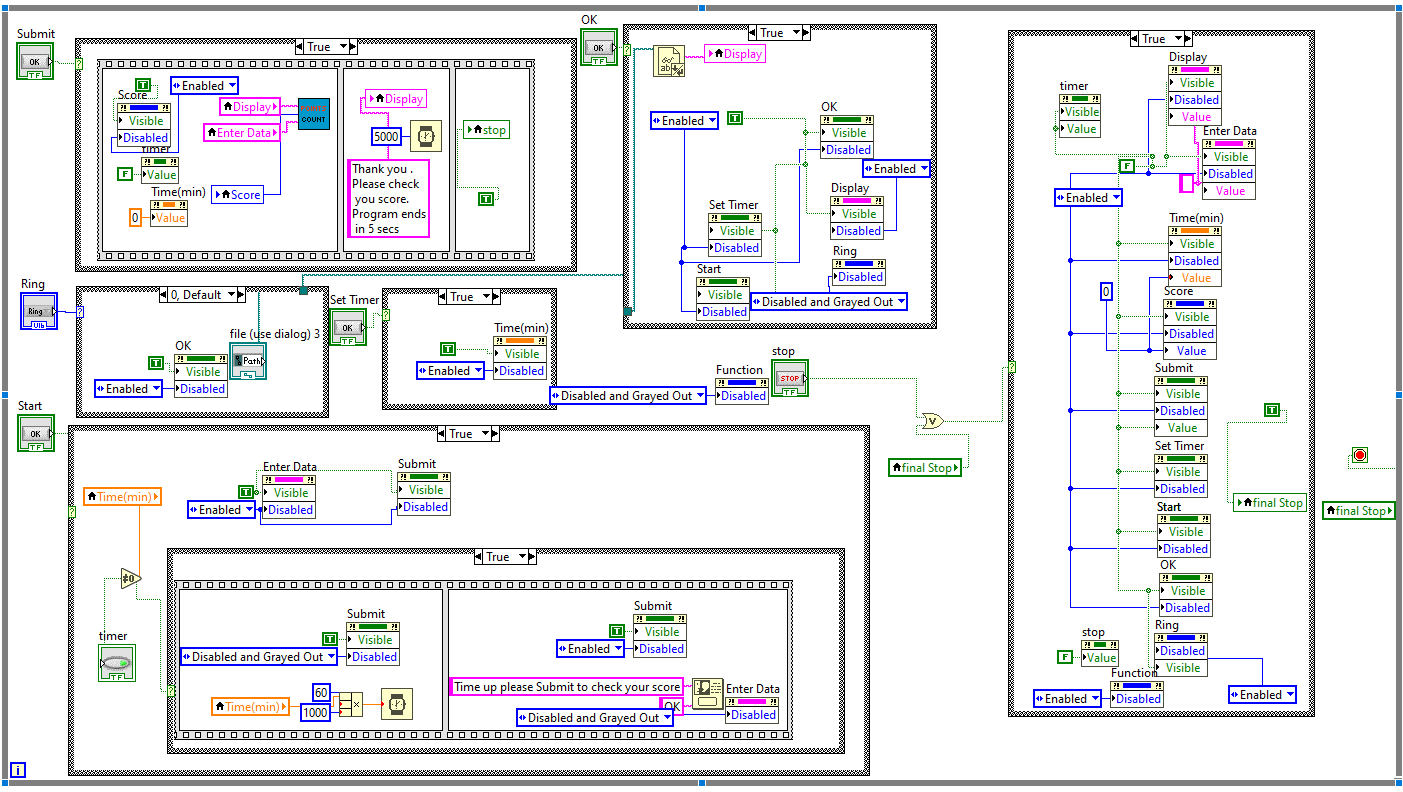


Figure 4.1

(The entire block of the typing mode present in the while loop on the up)

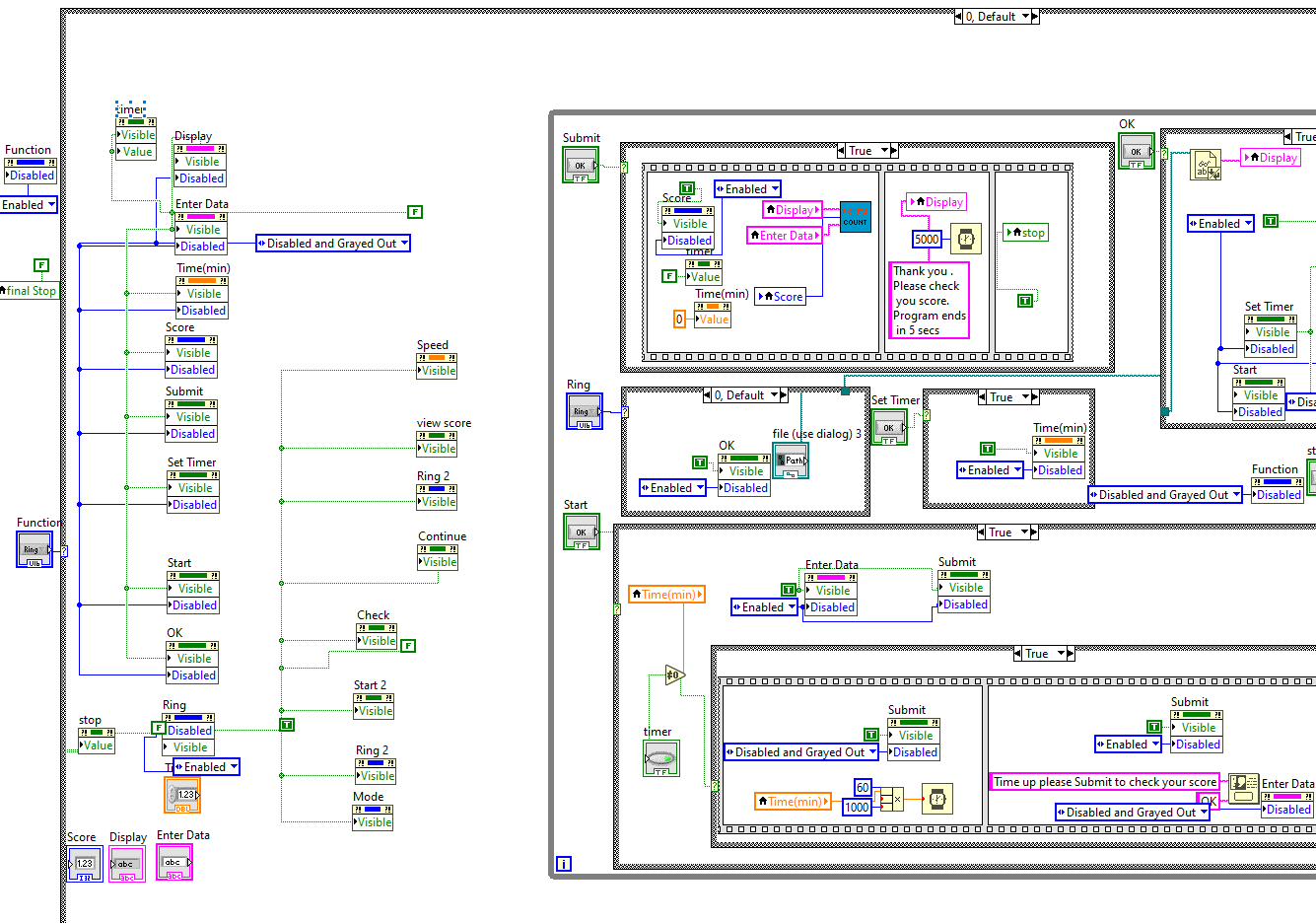


Figure 4.2

(The block panel of starting side of typing mode which makes the controls and indicators to visible and disable at start on left side)

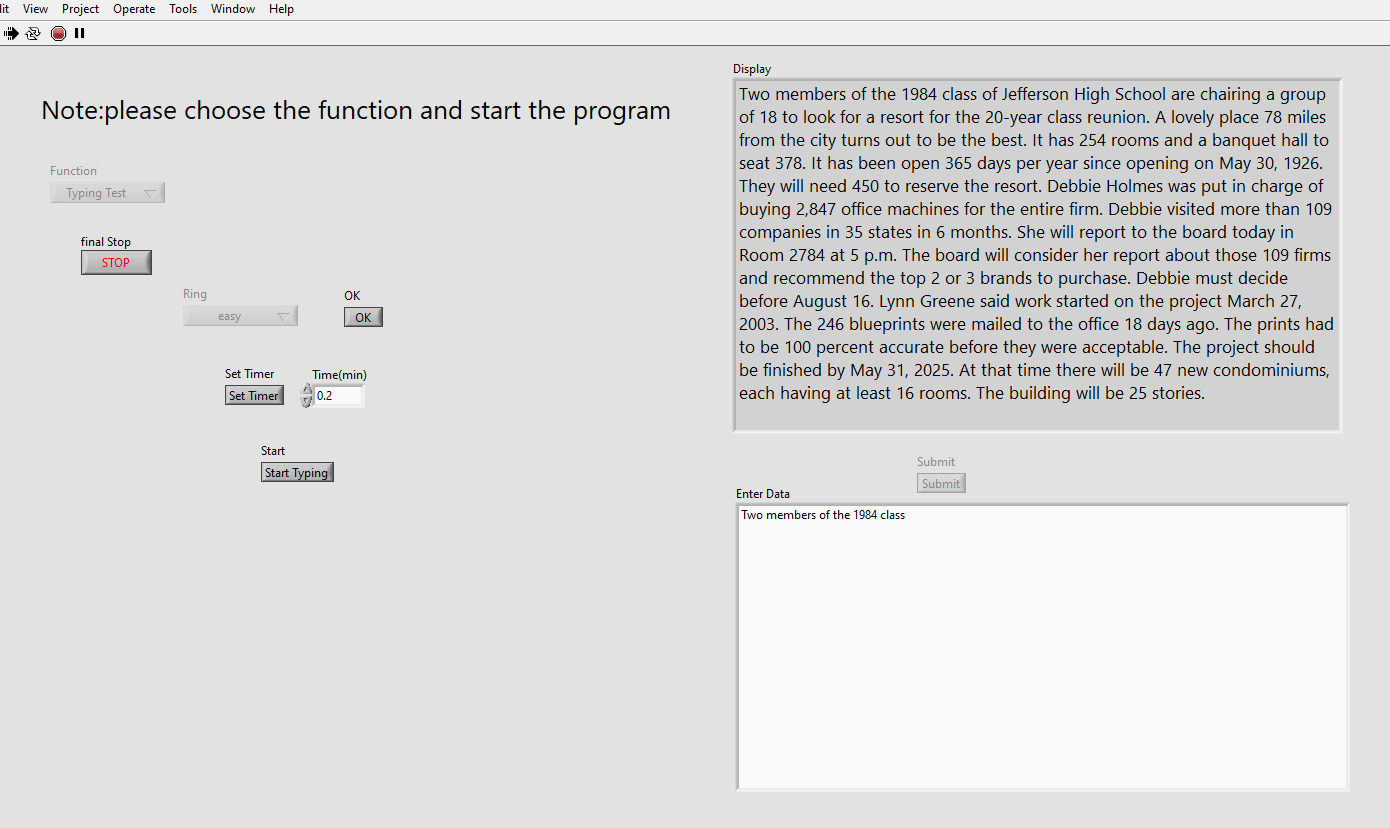


Figure 4.3

(The entire Front panel of the typing mode when user is entering the data)

**5. Results and Conclusion**

**5.1. Results**

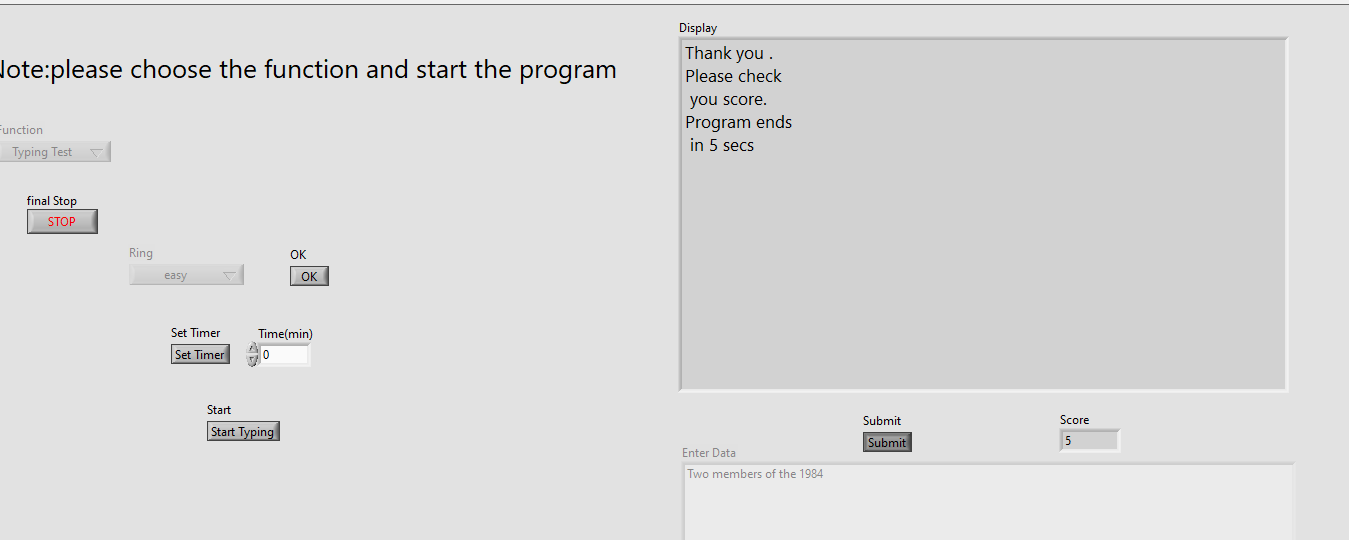
****

Figure 5.1.1

(The output Screen of the front panel showing the score of typing test and resetting the program)

We will get the output screen in same manner even for listening test also as the difference is the audio would be played in the background.

**5.2. Conclusion**

So finally we can say that with the help of this application we can practice and improve our skills in easy manner. Here we were limited only to a few models but we can even add more data files or we can connect it with a database or spreadsheet with the new paths so the new data can be updated.

The problem we will face is for opening an audio file it need to be in wav format only with specific format only otherwise the labview can’t read them

**5.3. Reference**

[**Typing and listening test using LABVIEW - YouTube**](https://www.youtube.com/watch?v=NmVKXCAptuo)