**SCALES OF MUSICAL COMPOSITION**

PROJECT REPORT

***Submitted by***

**AHILYA KALE** – 19MIM10011

**FRENY REJI** – 19MIM10028

**GOVIT KHASARE** – 19MIM10094

**ARYAN SUNIL AHUJA** – 19MIM10096

*in partial fulfillment for the award of the degree*

*of*

**INTEGRATED MTECH**

*In*

# COMPUTER SCIENCE AND ENGINEERING

*Specialization in*

***Artificial Intelligence and Machine Learning***



**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

**October, 2020**

**VIT BHOPAL UNIVERSITY,KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“GUITAREX- PLAYING MADE EASIER”** is the bonafide work of **- Ahilya Kale**-19MIM10011**,**

**Freny Reji** – 19MIM10028, **Govit Khasare** – 19MIM10094,

**Aryan Sunil Ahuja** – 19MIM10096;

who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

Dr.V.Pandimurugan, Assistant Professor

Dr.M.Ashwin, Assistant Professor (Senior)

School of AI &ML division School of AI &ML division

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

First and foremost, we would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

We wish to express our heartfelt gratitude to Dr. Manas Kumar Mishra, Head of the Department, School of Computing Science and Engineering for much of his valuable support encouragement in carrying out this work.

We would like to thank our internal guide Dr.M.Ashwin, for continually guiding and actively participating in our project, giving valuable suggestions to complete the project work.

We would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not the least, we are deeply indebted to our parents who have been the greatest support while we worked day and night for the project to make it a success.

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **SR. NO** | **ABOUT** |
| FIGURE 1 | ORGANISATION OF THE THESIS |
| FIGURE 2 | FRETBOARD |
| FIGURE 3 | ARCHITECTURE FLOW |
| FIGURE 4 | EXISITNG WORK |
| FIGURE 5 | SCALES AND FREQUENCIES |
| FIGURE 6 | AUTOMATED ML COMMAND |
| FIGURE 7 | COMMAND WINDOW |
| FIGURE 8 | MAIN GUI WINDOW |
| FIGURE 9 | PERFORMANCE ANALYSIS |

**ABSTRACT**

Music theory is a complex subject made up of a lot of components. For the purpose of this project, we will be simplifying things here quite a bit, so if you are a musician don’t take it too seriously. Pretend that you can imagine the spectrum of audible sound, and try to divide it into a musical system that makes sense for you and everyone around you.

Beginners in the field of music learning, especially guitar learners face a lot of dilemma on where to begin with and many at times they lose a track of their learning which makes learning a complicated task. Therefore, to cut this slack off, we come with an easy, yet an efficient interface that would facilitate better learning.

The GUI will provide the user with a virtual form of a guitar fretboard. The user will be then able to enter the scale they want to learn and the GUI will highlight the tabs accordingly that should be played. This project is made by implementing Tkinter that can be used for building a creative application that visually represents different scales, notes, modes, and keys. Such tools are particularly common for string instruments like the guitar.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | List of Figures  Abstract | 6  7 |
| 1 | **INTRODUCTION** Introduction  * 1. Motivation for the work   2. Introduction to Guitarex   3. Problem Statement   4. Objective of the work   5. Organization of the thesis | 8  9  10  11  11  12  12 |
| 2 | **LITERATURE SURVEY**   * 1. Introduction   2. The Main Built of the GUI   3. Existing Work and Algorithms   4. Observations Made   5. Summary | 13  14  15  16  16 |
| 3 | **SYSTEM ANALYSIS**  3.1 Introduction  3.2 Limitations in the GUI  3.3 Proposed System  3.4 Summary | 17  17  18  19 |
| 4 | **SYSTEM DESIGN AND IMPLEMENTATION**  4.1 Introduction  4.2 Module 1- Tkinter  4.3 Module 2- Automated ML  4.4 Summary | 20  21  27  28 |
| 5 | **PERFORMANCE ANALYSIS**  5.1Introduction  5.2 Performance Measures  5.3 Performance Analysis  5.4 Summary | 29  29  30  30 |
| 6 | **FUTURE ENHANCEMENT AND CONCLUSION**  6.1 Limitation/Constraints of the System  6.2 Future Enhancements  6.3 Conclusion | 31  31  31 |

1. **INTRODUCTION**
   1. **FOUNDATION**

* Pretend that you can imagine the spectrum of audible sound, and try to divide it into a musical system that makes sense for you and everyone around you. This is not easy! After centuries of struggle, humans have come up with a system but there are still disputes going on. Using the intervals of a given scale as index, we can retrieve the notes that make up the scale. This is all you need to understand. We will be applying this principle over and over again to extract the notes that compose a given scale. Using this information, we will then plot where these notes are located in the guitar. These plots all together make a SCALE CHART.
* Guitarex is a GUI based application that provides the user, especially the beginners in the field of guitar learning, a hassle-free path towards learning the instrument in a simpler yet efficient method.
  1. **MOTIVATION BEHIND COMING UP WITH THIS IDEA:**
* We did not just want our project to be limited to the boundaries of Computer Science and Technology. Our aim was two combine the facets of multiple industries, i.e the music industry and the technical industry. We wanted to come up with something that aids a better a learning experience for music learners all across the world.
* Being a music enthusiast, we understand how much crucial it is to understand the theory parts of music to their roots in order to produce quality music. Guitar, our targeted instrument requires great precision and in-depth knowledge of how the notes are distributed on the fretboard.
* Thus, having bits and parts of both, computer science and music, we worked with a vision to unite both the fields with something that helps both them in the long run.
  1. **INTRODUCTION TO GUITAREX**
* The stigma associated with pursuing an education in the musical arts affects the decision of many musicians nationwide. The appeal of guaranteed financial and career stability of STEM and other paths of high demand jobs is very difficult to pass up, even by the most dedicated musicians. Then, the input and advice of outsiders come into play. When a child particularly at a young age learns how to play an instrument preferably in a social setting is provides the brain of that child with extra dopamine, new neurological connections, better behaviour, and higher test scores. Schools are turning to new programs to aid music education and there are many reasons to support their decision. Music has been proven over many studies to show a connection to education. The study will be conducted through the use of Python 3 and Tk for building a creative application that visually represents different scales, notes, modes, and keys. The purpose of this project is to create a program with which you can find the position of the notes of a given scale in the guitar.

* 1. **PROBLEM STATEMENT**
* STATEMENT 1:

Beginners in the filed of guitar playing easily get confused between the notes and their corresponding positions on the fretboard.

* STATEMENT 2:

A GUI can be built up that virtually represents a guitar fretboard. It would enable the user to enter the scale on their own will and hence, display the notes that are to be played.

* STATEMENT 3:

The second STATEMENT provides an optimum solution facilitating the music learners with a confident, easy yet an understandable path to learn guitar in the most correct manner possible.

* 1. **OBJECTIVE OF OUR WORK**

The main objective of this project is unification of technology with art. Not only this, but keeping in mind the user expectations, we wanted to come up with something that solves the issue of the representation of notes on the guitar. We very well know that virtual representation of anything makes learning faster and enhanced. Thus, we wanted to give a virtual representation of the fretboard, enabling the users to be their own master.

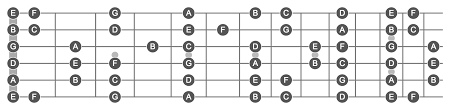
* 1. **ORGANISATION OF THE THESIS**

**FIGURE 1**

* 1. **SUMMARY**
* Thus, we aim to create an easy and efficient way to represent chromatic scales. This GUI can be used by people of any age and with any or minimum knowledge about the music theory of guitar.
* We first researched on the existing models, made conclusive results and then came with our own application stating its novelty and uniqueness.
* We used the concept of automated ML to built an .exe setup with a testing process that is done in the background.
* Thus, this project covers two important backgrounds of Computer Science; Machine Learning and Programming using Python.

1. **LITERATURE SURVEY**
   1. **INTRODUCTION**

* Python 3 and Tk can be used for building a creative application that visually represents different scales, notes, modes, and keys. Such tools are particularly common for string instruments like the guitar. Here, the users can navigate different scales (major, natural minor, harmonic minor, pentatonic, blues, etc.) and chords (5 chords, major, minor, diminished, augmented, and so on) on a 24- fret chart. The purpose of this project is to create a program with which you can find the position of the notes of a given scale in the guitar fretboard.
* Here we will treat the fretboard as a UX Component and treat it as an image.  Based on the design guidelines and the test variations, we introduce potential flaws in direct correlation to the design input. These flawed design mockups are manifested as images. Proper labeling of these images ensures proper organization of test data. Once we have a minimal set of images in our arsenal, we are ready to train our model.



**FIGURE 2**

* 1. **MAIN BUILT OF THE GUI**
* The purpose of this project is to create a program with which you can find the position of the notes of a given scale in the guitar.

For that, we will have to figure out methods to:

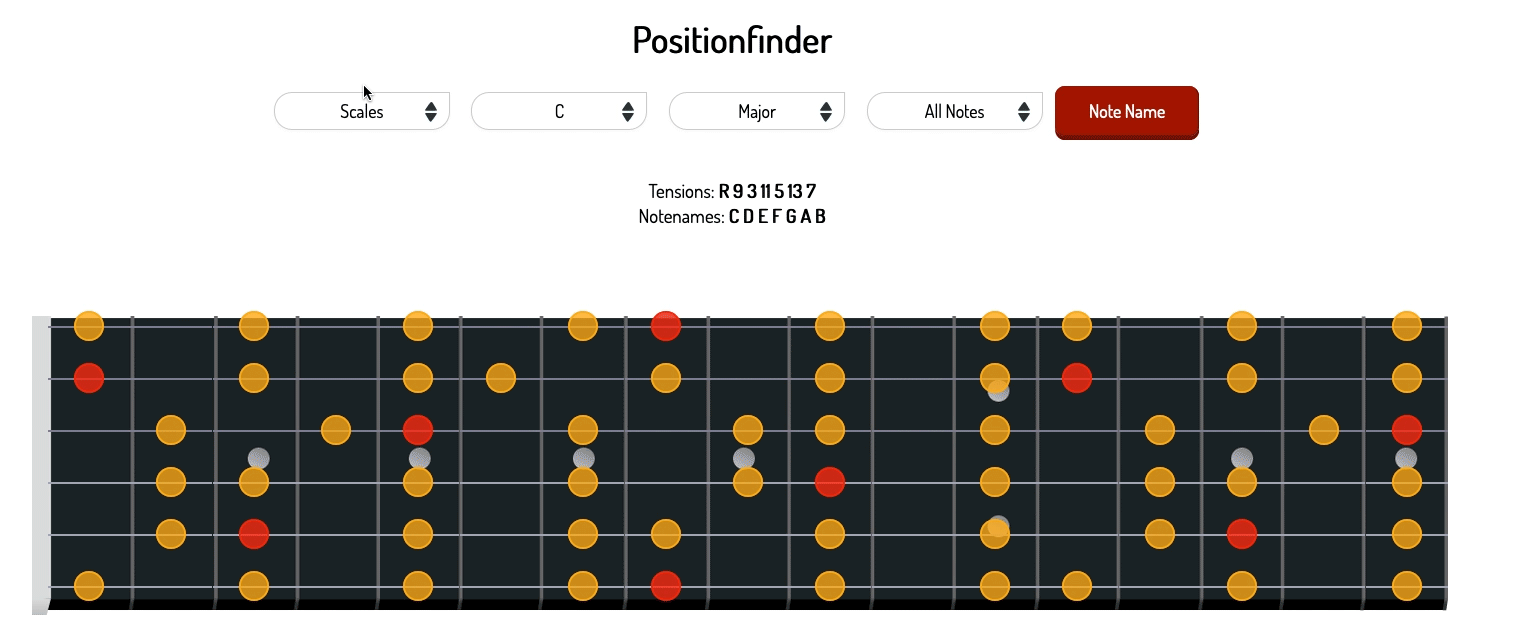
• Extract the notes composing a given scale. An easy way to create chromatic scales (without typing them explicitly) is to concatenate a list with all the notes within itself and slice the desired elements.

• Locate where they are positioned in the strings of the guitar.

* As our project is based on GUI i.e. Graphical User Interface which allows us to interact with a computer program using a pointing device that manipulates small pictures on a computer screen. The small pictures are called icons or widgets. We refer to programs that use a graphical user interface as “GUI programs.” A GUI program is very different from a program that uses a command line interface which receives user input from typed characters on a keyboard.
* The project being a GUI based application is made using Tkinter, a Python binding to the GUI toolkit. We started off from scratch and saw how we can use lists, dictionaries to build a tool that can help us play the guitar. We all know Music theory is a complex subject made up of a lot of components. For the purposes of this post, we will be simplifying things here quite a bit.
* The uses of a pointer that serves as navigation to interact with different visually appealing Graphical icons. Abstraction is a major concept that has been used in a GUI operating system. Users can use the pointer to click on the icon which initiates a series of actions. Normally an application or functionality will get started. Then the user will have to provide input to generate the desired action from the machine. The GUI actually translates user language which comprises simple one-line commands, single click and double clicks to machine language or assembly language. Machine language is understood by the machine and hence the machine responds to the task initiated which is translated to use language and communicated to the user via GUI.

**FIGURE 3**

* 1. **EXISTING WORK & ALGORITHMS**
* There are various interfaces available in the market that claim to plot and exhibit all the notes on the fretboard.
* Their framework is usually based on Django and the JavaScript makes the data visible on a HTML fretboard.
* Some of these available projects use complex renders like Angular Universal and Netlify.



**FIGURE 4**

* 1. **OBSERVATIONS MADE**

The projects we found on prestigious platforms like Github.com had wrong markings. As the FIGURE 4 depicts, there are no string or fret markings on the fretboard. This might lead to even more confusion among the people who are newbies in the field of guitar playing.

* 1. **SUMMARY**

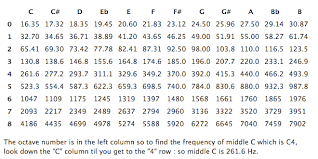
Our GUI will be based on a simple framework made using Tkinter and hence navigation through the application will be way easier and the representation will be highly precise and easy to adapt.

1. **SYSTEM ANALYSIS**
   1. **INTRODUCTION**

* Now, we’ll have a brief analysis of our GUI. This includes the system constraints, our proposed systems and the functionalities of the GUI.
* What makes our interface better than the currently available projects is the fact it will not only depict the notes that are required to be played, but will also implicate the corresponding scales so that the user will not have to google them again and again.
* We will perform a GUI testing using the idea of Deep Learning to validate the accuracy of the scales and notes displayed.
* Our project will work on a simpler yet efficient framework to have scope of development in the future and also make the interface user friendly.
  1. **LIMITATIONS OF THE SYSTEM**
* The GUI will be a single window application. Thus, work is required on improving the functionality and aesthetics of the application.
* The code might attract viruses due to its simple built.
* The run time of the code should be kept in mind to avoid lag and disruptions.
  1. **PROPOSED SYSTEM**
* We will have to perform the following tasks for the project:

1. Import Tkinter package to create the GUI application
2. Create lists and dictionaries to store the scale information and the notations.
3. Use API packages to display the musical notation.
4. Performing a GUI testing process using Deep Learning.

* The first note of a scale is called the root and gives the scale its name. It is also the place where you rest your fingers on when you are playing (as it feels like the base sound that holds everything together). Let’s see another chromatic scale starting with a root in C (the C chromatic scale):
* C = ['C','C#','D','D#','E','F','F#' ,'G','G#','A','A#', 'B'] Interestingly, even though these two scales have exactly the same notes, because they start with a different root the intervals between the individual notes and the root is completely different, which makes these scales feel nothing like each other.
* To check whether the depicted notes are correct, we will match the frequency of the depicted notes to that of the key notes. This will be our test data.
* Here we will treat the fretboard as a UX Component and treat it as an image.  Based on the design guidelines and the test variations, we introduce potential flaws in direct correlation to the design input. These flawed design mockups are manifested as images. Proper labeling of these images ensures proper organization of test data. Once we have a minimal set of images in our arsenal, we are ready to train our model.



**FIGURE 5**

* 1. **SUMMARY**

Our project will work on a simpler yet efficient framework to have scope of development in the future and also make the interface user friendly. We will perform a GUI testing using the idea of Deep Learning to validate the accuracy of the scales and notes displayed. What makes our interface better than the currently available projects is the fact it will depict the notes that are required to be played.

1. **SYSTEM DESIGN AND IMPLEMENTATION**
   1. **INTRODUCTION**

The flow of the modules will be as follows:

* 1. **MODULE 1- TKINTER**

Tkinter Programming: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps−

• Import the Tkinter module.

• Create the GUI application main window.

• Add one or more of the above-mentioned widgets to the GUI application.

• Enter the main event loop to take action against each event triggered by the use.

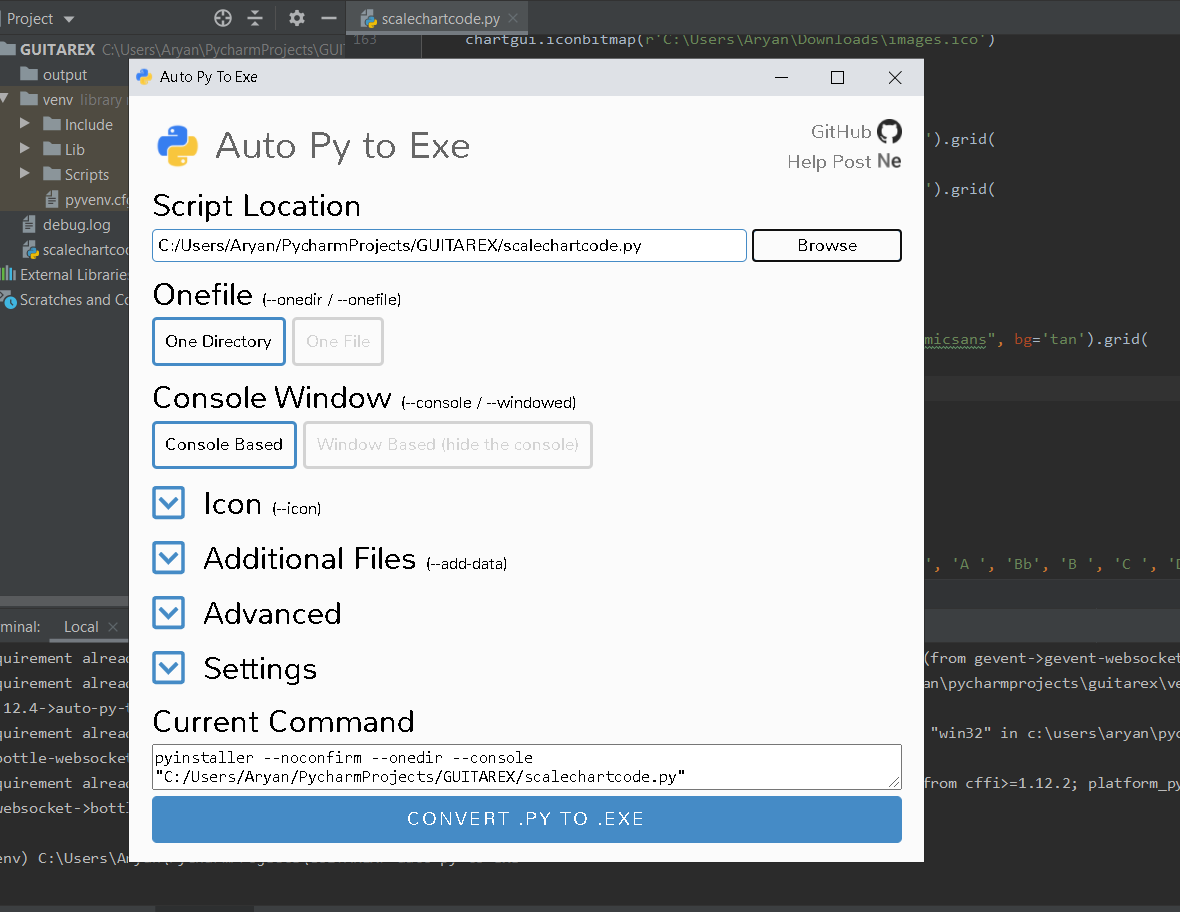
**MAIN PROGRAM USING TKINTER:**

#python3  
import sys  
from tkinter import \*  
from collections import OrderedDict  
  
  
# This program creates a guitar scale gui from grid elements, and fills them in color-coded appropriately.  
  
# Get Note name from a 0-11 INT  
def getnotename(tonename):  
 notedict = ['E ', 'F ', 'F#', 'G ', 'Ab',  
 'A ', 'Bb', 'B ', 'C ', 'Db', 'D ', 'Eb']  
 return notedict[tonename % 12]  
  
  
# Contains int offsets, based on note string, added for convenience, which  
# is simply offset relative to C.  
def getoffset\_tonename(tonename):  
 scaleref = {  
 'E ': 0, 'F ': 1, 'F#': 2, 'G ': 3, 'Ab': 4, 'A ': 5, 'Bb': 6, 'B ': 7, 'C ': 8, 'Db': 9, 'D ': 10,  
 'Eb': 11}  
  
  
 return scaleref[tonename]  
  
  
# Return array that is rotated circular  
def rotate(l, n):  
 return l[-n:] + l[:-n]  
  
  
scales = OrderedDict([  
 ('Major', [0, 2, 2, 1, 2, 2, 2, 1]),  
 ('Natural minor', [0, 2, 1, 2, 2, 1, 2, 2]),  
 ('Harmonic minor', [0, 2, 1, 2, 2, 1, 3, 1]),  
 ('Melodic minor', [0, 2, 1, 2, 2, 2, 2, 2]),  
 ('Dorian mode', [0, 2, 1, 2, 2, 2, 1, 2]),  
 ('Phrygian mode', [0, 1, 2, 2, 2, 1, 2, 2]),  
 ('Lydian mode', [0, 2, 2, 2, 1, 2, 2, 1]),  
 ('Mixolydian mode', [0, 2, 2, 1, 2, 2, 1, 2]),  
 ('Locrian mode', [0, 1, 2, 2, 1, 2, 2, 2]),  
 ('Ahava raba mode', [0, 1, 3, 1, 2, 1, 2, 2]),  
 ('Minor pentatonic', [0, 3, 2, 2, 3, 2]),  
 ('Pentatonic', [0, 2, 2, 3, 2, 3]),  
 ('Blues', [0, 3, 2, 1, 1, 3]),  
 ('5 chord', [0, 7]),  
 ('Major chord', [0, 4, 3]),  
 ('Minor chord', [0, 3, 4]),  
 ('Diminished chord', [0, 3, 3]),  
 ('Augmented chord', [0, 4, 4]),  
 ('Sus2 chord', [0, 2, 5]),  
 ('Sus4 chord', [0, 5, 2]),  
 ('Maj7 chord', [0, 4, 3, 4]),  
 ('min7 chord', [0, 3, 4, 3]),  
 ('7 chord', [0, 4, 3, 3]),  
 ('min7b5 chord', [0, 3, 3, 4]),  
 ('dim7 chord', [0, 3, 3, 3]),  
 ('9 chord', [0, 4, 3, 3, 4]),  
 ('Maj9 chord', [0, 4, 3, 4, 3]),  
 ('m9 chord', [0, 3, 4, 3, 4]),  
 ('11 chord', [0, 4, 3, 3, 4, 3]),  
 ('Maj11 chord', [0, 4, 3, 4, 3, 3]),  
 ('min11 chord', [0, 3, 4, 3, 4, 3]),  
])  
  
# returns a scale of 16 notes, from the key tonic + 24  
  
  
def makescale(keyroot, keyopt):  
 keywheel = []  
 keywheel.extend(scales[keyopt])  
 filler = 0  
 # fill array with 16 notes relevant to key and option.  
 ourscale = []  
 lenvar = len(keywheel) # of notes we use (2 octaves of key notes)  
 for inte in range(lenvar):  
 filler += keywheel[inte % len(keywheel)]  
 ourscale.append(int(filler + getoffset\_tonename(keyroot)))  
 return ourscale  
  
  
# fetches a default scale  
ourscale = makescale('E ', 'Major')  
  
# Used for note offsets  
e = 0  
  
# high e = (e+4), b = (e-1), g = (e+7), d = (e+2),a = (e+9), low e = (e+4)  
# added to each string to offset and identify the notes.  
offsetArray = [e, e + 7, e + 3, e + 10, e + 5, e]  
  
# default e Major  
chartgui = Tk()  
  
# our callback variables that change when menu options are selected  
variable = StringVar(chartgui)  
variable.set('E ')  
variable2 = StringVar(chartgui)  
variable2.set('Major')  
  
  
# variable3 = StringVar(chartgui)  
# variable3.set('View 1')  
  
  
# for clearing all our values, used for the "Reset" button.  
def resettable():  
 print("Tried to reset!")  
 for i in range(0, 25):  
 for gss in range(0, 6):  
 Label(chartgui, text=getnotename(i + offsetArray[gss]), bg='tan').grid(  
 row=gss + 2, column=i + 1, padx=0, pady=0)  
  
  
# redraw our whole scale, the action for the "Apply" button  
def applyit(val):  
 print("Trying to apply!")  
 # print("var1 = %s"%variable.get())  
 # print("var2 = %s"%variable2.get())  
 # print("var3 = %s"%variable3.get())  
 ourtonic = str(variable.get())  
 ourkey = str(variable2.get())  
 print(ourtonic)  
 print(ourkey)  
  
 ourscale = makescale(ourtonic, ourkey)  
 print(ourscale)  
 ournotes = []  
  
 for notes in ourscale:  
 ournotes.append(getnotename(notes))  
  
 print (ournotes)  
  
 # draw our whole scale  
 for i in range(0, 25):  
 for gss in range(0, 6):  
 start = offsetArray[gss]  
  
 # draw lawngreen for roots  
 if ourtonic == getnotename(i + start % 12):  
 Label(  
 chartgui, text=getnotename(i + start), bg='lawngreen').grid(row=gss + 2, column=i + 1,  
 padx=0, pady=0)  
 # draw cyan for notes in the scale  
 elif getnotename(i + start) in ournotes:  
 Label(  
 chartgui, text=getnotename(i + start), bg='cyan').grid(row=gss + 2, column=i + 1,  
 padx=0, pady=0)  
 # only write notename  
 else:  
 Label(chartgui, text=getnotename(i + start), bg='tan').grid(  
 row=gss + 2, column=i + 1, padx=0, pady=0)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 chartgui.geometry('700x300+400+300')  
 chartgui.title('Guitarex- Playing Made Easier ')  
 chartgui.configure(bg= 'tan')  
 ourx = 40  
 oury = 20  
 chartgui.iconbitmap(r'C:\Users\Aryan\Downloads\images.ico')  
  
 # For our fret (column) labels on top and bottom  
 for i in range(0, 25):  
 Label(chartgui, text=i, font='boulder', bg='tan').grid(  
 row=0, column=i + 1, padx=0, pady=10)  
 Label(chartgui, text=i, font='boulder', bg='tan').grid(  
 row=9, column=i + 1, padx=0, pady=10)  
  
 # For our string (row) labels  
 stringarray = ['E', 'B', 'G', 'D', 'A', 'E']  
 for gss in range(0, 6):  
 Label(chartgui, text=stringarray[gss], font="comicsans", bg='tan').grid(  
 row=gss + 2, column=0, padx=10, pady=0)  
  
 print(ourscale)  
  
 # draw our whole scale  
 applyit("")  
  
 keymenu = OptionMenu(  
 chartgui, variable, 'E ', 'F ', 'F#', 'G ', 'Ab', 'A ', 'Bb', 'B ', 'C ', 'Db', 'D ',  
 'Eb', command=applyit).place(x=ourx \* 4, y=oury \* 13)  
 scalemenu = OptionMenu(chartgui, variable2, \*scales.keys(), command=applyit).place(  
 x=ourx \* 6, y=oury \* 13)  
 resetbutton = Button(chartgui, text=' Reset ', command=resettable).place(  
 x=ourx \* 10, y=oury \* 13)  
  
 chartgui.mainloop()

* 1. **MODULE 2- AUTOMATED ML**
* ML simulates the human way of finding errors or anomalies. Humans are driven by past experience and conditioning to make decisions. Machines with the proper application of training or conditioning can detect errors that surpass human precision.
* We begin our understanding of ML as the subset of a broader class called as the supervised machine learning algorithm. The supervised learning algorithms take a set of training examples called as the training data. The learning algorithm is provided with the training data to learn a desired function. Further, we also validate our learning algorithm by a set of test data. This process of learning from training data and validating against test data is called modeling.
* We used a readily available ML package from github.com, that tests the GUI and converts into an .exe setup, hence relieving us from the manual testing process.

****

**FIGURE 6**



**FIGURE 7**

* 1. **SUMMARY**

We have the following output GUI:



**FIGURE 8**

1. **PERFORMANCE ANALYSIS**
   1. **INTRODUCTION**

Code performance is important because it limits the amount of data a program can handle. Imagine you are working on an application that lets users access web forums organized by topic, with the ability to read and post from a smartphone. If you let people use the app without an internet connection, the app may need to store some data on the smartphone. It’s difficult to estimate the amount of data that will be stored, but you can predict that if the user loves the app, you will need to store a lot of forum posts.

* 1. **PERFORMANCE MEASURES**

To get a precise performance analysis, we performed the following steps-

* Bifurcated the functions.
* Calculated the NLOC and time complexity for each.
* We used an online analyzer lizard.ws for the same.
  1. **PERFORMANCE ANALYSIS**

Following was the analysis we made for the GUI:



**FIGURE 9**

* 1. **SUMMARY**
* What makes our interface better than the currently available projects is the fact it will not only depict the notes that are required to be played, but will also implicate the corresponding scales so that the user will not have to google them again and again.
* We performed a GUI testing using the idea of Deep Learning to validate the accuracy of the scales and notes displayed.
* Our project will work on a simpler yet efficient framework to have scope of development in the future and also make the interface user friendly.
* The GUI exhibits a good performance in terms of time as well as space complexity.

1. **FUTURE ENHANCEMENT**
   1. **CONSTRAINTS FOUND**

* There are no disadvantages as such. The only constraint is that for now, the scale representation is limited only to a single instrument.
* Also, work is needed on improving the aesthetics and functionality of the GUI as a whole.
  1. **FUTURE DEVELOPMENT**
* We aim at extending the project for several other instruments like Piano and Wind instruments
* We'd work to include a sound assistant that'd exclaim how a particular scale would sound when played correctly
* We also want to improve the aesthetics of the GUI to make it more appealing**.**
  1. **CONCLUSION**
* Thus, by combining high end programming with music, we came up with an easy and efficient interface that would facilitate better learning.
* Now beginners can play any scale they want without having to google it every time.