



A MINI PROJECT REPORT
ON

“Conference Table Arrangement”

Submitted By:

KASUKURTHI NEELESH VIJAY BABU
SARAH ARIF KHAN
DHRUVI MOHAN GORI

VU1F1819057
VU1F1819066
VU1F1819065

Under the guidance of
Mr. SRIKANT BAGEWADI

Conference Table Arrangement

**Padmabhushan Vasantdada Patil Pratishthan's
College of Engineering
Sion, Mumbai-400 022
Mumbai University**



**A MINI PROJECT REPORT
ON**

"Conference Table Arrangement "

**Submitted to the
Padmabhushan Vasantdada Patil Pratishthan's
College of Engineering
Bachelor of Engineering
In
Computer Engineering**

Submitted By:

KASUKURTHI NEELESH VIJAY BABU	VU1F1819057
SARAH ARIF KHAN	VU1F1819066
DHRUVI MOHAN GORI	VU1F1819065

Under the guidance of
Mr. SRIKANT BAGEWADI
Department of Computer Engineering

Conference Table Arrangement

**Padmabhushan Vasantdada Patil Pratishthan's
College of Engineering Sion, Mumbai-400 022**

**Mumbai University
2019-2020**

INDEX

	Page
• AIM	3
• ABSTRACT.	3
• INTRODUCTION TO PYTHON	4
• EXPLANATION ABOUT MINI PROJECT	11
• LIST OF CONCEPTS USED IN MINI PROJECT	12
• EXPLANATION ABOUT THE CONCEPTS IN MINI PROJECT	12
• CODING / IMPLEMENTATION	13
• OUTPUT	35
• CONCLUSION	37
• REFERENCE	37

Conference Table Arrangement

AIM: To Design and Implement a Conference Table Arrangement using concepts in Python.

ABSTRACT

This is an error free mini project about a Conference Table Arrangement that is being used by the user to arrange the position of the person with the priority order that particular person. Its an easy to use interface wherein the user only needs to get only priority of the persons going to be present for the Conference . It's completely user friendly and lets user use it without any hassle because there are no complicated systems involved in it.

INTRODUCTION

• INTRODUCTION TO PYTHON:

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. Python is managed by the non-profit Python Software Foundation.

History :

Python was conceived in the late 1980s and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum remains Python's principal author. His continuing central role in Python's development is reflected in the title given to him by the Python community: *Benevolent Dictator For Life* (BDFL). On the origins of Python, Van Rossum wrote in 1996. Python 2.0 was released on 16 October 2000 and had many

Conference Table Arrangement

major new features, including a cycle-detecting garbage collector and support for Unicode. With this release, the development process became more transparent and community-backed. Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3. Python 2.7's end-of-life date was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. Python 3.6 had changes regarding UTF-8 (in Windows, PEP 528 and PEP 529) and Python 3.7.0b1 adds a new "UTF-8 Mode". In January 2017, Google announced work on a Python 2.7 to Go trans compiler to improve performance under concurrent workloads.

Features of Python

1. Simple -

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

2. Easy to Learn -

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

Conference Table Arrangement

3. Free and Open Source -

Python is an example of a FLOSS (Free/LibrÃ© and Open Source Software). In simple terms, you can freely distribute copies of this software, read it's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

4. High-level Language -

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

5. Portable -

Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and even PocketPC !

6. Interpreted -

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a

Conference Table Arrangement

language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

7. Object Oriented -

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

8. Extensible -

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can

Conference Table Arrangement

code that part of your program in C or C++ and then use them from your Python program.

9. Embeddable -

You can embed Python within your C/C++ programs to give 'scripting' capabilities for your program's users.

10.Extensive Libraries -

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), Tk, and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the 'Batteries Included' philosophy of Python.

Advantages of Python -

- **Extensive Support Libraries:-**

It provides large standard libraries that include the areas like string operations, Internet, web service tools, operating system interfaces and protocols. Most of the highly used programming tasks are already scripted into it that limits the length of the codes to be written in Python.

- **Integration Feature:-**

The language has extensive support libraries and clean object-oriented designs that increase two to ten fold of programmer's productivity while using the languages like Java, VB, Perl, C, C++ and C#.

- **Productivity:-**

Conference Table Arrangement

With its strong process integration features, unit testing framework and enhanced control capabilities contribute towards the increased speed for most applications and productivity of applications. It is a great option for building scalable multi-protocol network applications.

Disadvantages of Python -

- **Difficulty in Using Other Languages**

The Python lovers become so accustomed to its features and its extensive libraries, so they face problem in learning or working on other programming languages. Python experts may see the declaring of cast "values" or variable "types", syntactic requirements of adding curly braces or semi colons as an onerous task.

- **Weak in Mobile Computing**

Python has made its presence on many desktop and server platforms, but it is seen as a weak language for mobile computing. This is the reason very few mobile applications are built in it like Carbonnelle.

- **Gets Slow in Speed**

Python executes with the help of an interpreter instead of the compiler, which causes it to slow down because compilation and execution help it to work normally. On the other hand, it can be seen that it is fast for many web applications too.

- **Run-time Errors**

Conference Table Arrangement

The Python language is dynamically typed so it has many design restrictions that are reported by some [Python developers](#). It is even seen that it requires more testing time, and the errors show up when the applications are finally run.

- **Underdeveloped Database Access Layers**

As compared to the popular technologies like JDBC and ODBC, the Python's database access layer is found to be bit underdeveloped and primitive. However, it cannot be applied in the enterprises that need smooth interaction of complex legacy data.

- EXPLANATION ABOUT MINI PROJECT:

This is an error free mini project about a Conference Table Arrangement that is being used by the user to arrange the position of the person with the priority order that particular person. Its an easy to use interface wherein the user only needs to get only priority of the persons going to be present for the Conference . It's completely user friendly and lets user use it without any hassle because their are no complicated system involved in it.

- **LIST OF CONCEPTS USED IN MINI PROJECT:**

1. from tkinter import *
2. import tkinter as tk
3. If elif else

- **EXPLANATION ABOUT THE CONCEPTS IN MINI PROJECT**

1. Tkinter:

Most of the time, tkinter is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, tkinter includes a number of Python modules

2. if elif else:

The if-elif-else statement is used to conditionally execute a statement or a block of statements. Conditions can be true or false, execute one thing when the condition is true, something else when the condition is false.

CODING / IMPLEMENTATION

```
from tkinter import *
import tkinter as tk

root=Tk()

root.title('Confarence table')

root.geometry('900x600')


sp1=IntVar()
sp2=IntVar()
sp3=IntVar()
sp4=IntVar()
sp5=IntVar()
sp6=IntVar()
sp7=IntVar()
sp8=IntVar()
sp9 =IntVar()
sp10 =IntVar()

source = IntVar()

destination = IntVar()

position =" "

photo = PhotoImage(file=r"/Users/neeleshkasukurthi/
Desktop/os/Office-coAnference-table-top-view-details-Tue-
Jul-2018-07-05-56.png")

photoimage1 = photo.subsample(3,3)

Label(root, text='Confrence Table
Arrangement').grid(row=0, column=3)
```

Conference Table Arrangement

```
Label(root, image=photoimage1).grid(row=0, column=5)
```

```
Label(root, text='Person').grid(row=2, column=1)  
Label(root, text='Priority').grid(row=2, column=3)
```

```
def arrange_seat_1():  
    a= sp1.get()  
    val=int(a)  
  
    if (val == 0 or val == 1):  
        p1 = "          Person 1 Seat"  
        position = Text(root, height=2, width=25,  
fg='red')  
        position.grid(row=4, column=7)  
        position.insert(0.0, p1)  
  
    elif (val == 2):  
        p2 = "Person 1 Seat"  
        position = Text(root, height=2, width=15,  
fg='red')  
        position.grid(row=6, column=7)  
        position.insert(0.0, p2)  
  
    elif (val == 3):  
        p3 = "Person 1 Seat"  
        position = Text(root, height=2, width=15,  
fg='red')  
        position.grid(row=6, column=8)  
        position.insert(0.0, p3)  
  
    elif (val == 4):  
        p1 = "Person 1 Seat"  
        position = Text(root, height=2, width=15,  
fg='red')  
        position.grid(row=8, column=7)  
        position.insert(0.0, p1)  
  
    elif (val == 5):  
        p1 = "Person 1 Seat"  
        position = Text(root, height=2, width=15,  
fg='red')
```

Conference Table Arrangement

```
position.grid(row=8, column=8)
position.insert(0.0, p1)

elif (val == 6):

    p1 = "Person 1 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (val == 7):

    p1 = "Person 1 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
    position.insert(0.0, p1)

elif (val == 8):

    p1 = "Person 1 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=7)
    position.insert(0.0, p1)

elif (val == 9):

    p1 = "Person 1 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=8)
    position.insert(0.0, p1)

else:

    p1 = "          Person 1 Seat"
    position = Text(root, height=2, width=25,
fg='red')
    position.grid(row=17, column=7)
    position.insert(0.0, p1)

Label(root, text="Person 1").grid(row=3, column=1)
sp1=Entry(root)
sp1.grid(row=3, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_1).grid(row=3, column= 4)
```


Conference Table Arrangement

```
def arrange_seat_2():
    b = sp2.get()
    b=int(b)

    if (b == 0 or b == 1):

        p1= "                Person 2 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (b == 2):

        p2 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=6, column=7)
        position.insert(0.0, p2)

    elif (b == 3):

        p3 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=6, column=8)
        position.insert(0.0, p3)

    elif (b == 4):

        p1 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=8, column=7)
        position.insert(0.0, p1)

    elif (b == 5):

        p1 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=8, column=8)
        position.insert(0.0, p1)

    elif (b == 6):

        p1 = "Person 2 Seat"
```

Conference Table Arrangement

```
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=7)
        position.insert(0.0, p1)

    elif (b == 7):

        p1 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=8)
        position.insert(0.0, p1)

    elif (b == 8):

        p1 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=7)
        position.insert(0.0, p1)

    elif (b == 9):

        p1 = "Person 2 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=8)
        position.insert(0.0, p1)

    else:

        p1 = "                Person 2 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=17, column=7)
        position.insert(0.0, p1)

Label(root, text="Person 2").grid(row=4, column=1)
sp2=Entry(root)
sp2.grid(row=4, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_2).grid(row=4, column= 4)

def arrange_seat_3():

    c = sp3.get()
    c=int(c)
```

Conference Table Arrangement

```
if(c == 0 or c == 1):
    p1= "                Person 3 Seat"
    position = Text(root, height=2, width=25,
fg='red')
    position.grid(row=4, column=7)
    position.insert(0.0, p1)
elif (c == 2):
    p2 = "Person 3 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=7)
    position.insert(0.0, p2)

elif (c == 3):
    p3 = "Person 3 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=8)
    position.insert(0.0, p3)

elif (c == 4):
    p1 = "Person 3 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (c == 5):
    p1 = "Person 3 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (c == 6):
    p1 = "Person 3 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (c == 7):
```

Conference Table Arrangement

```
        p1 = "Person 3 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=8)
        position.insert(0.0, p1)

    elif (c == 8):

        p1 = "Person 3 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=7)
        position.insert(0.0, p1)

    elif (c == 9):

        p1 = "Person 3 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=8)
        position.insert(0.0, p1)

    else:

        p1 = "                Person 3 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=17, column=7)
        position.insert(0.0, p1)

Label(root, text="Person 3").grid(row=5, column=1)
sp3=Entry(root)
sp3.grid(row=5, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_3).grid(row=5, column= 4)

def arrange_seat_4():
    d = sp4.get()
    d=int(d)

    if (d == 0 or d == 1):
```

Conference Table Arrangement

```
p1= "                Person 4 Seat"
position = Text(root, height=2, width=25,
fg='red')
position.grid(row=4, column=7)
position.insert(0.0, p1)

elif (d == 2):

    p2 = "Person 4 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=7)
    position.insert(0.0, p2)

elif (d == 3):

    p3 = "Person 4 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=8)
    position.insert(0.0, p3)

elif (d == 4):

    p1 = "Person 4 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (d == 5):

    p1 = "Person 4 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (d == 6):

    p1 = "Person 4 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (d == 7):

    p1 = "Person 4 Seat"
```

Conference Table Arrangement

```
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=8)
        position.insert(0.0, p1)

    elif (d == 8):

        p1 = "Person 4 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=7)
        position.insert(0.0, p1)

    elif (d == 9):

        p1 = "Person 4 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=8)
        position.insert(0.0, p1)

    else :

        p1 = "                Person 4 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=17, column=7)
        position.insert(0.0, p1)

Label(root, text="Person 4").grid(row=6, column=1)
sp4=Entry(root)
sp4.grid(row=6, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_4).grid(row=6, column= 4)

def arrange_seat_5():

    e=sp5.get()
    e=int(e)

    if (e == 0 or e == 1):

        p1= "                Person 5 Seat"
        position = Text(root, height=2, width=15,
fg='red')
```

Conference Table Arrangement

```
position.grid(row=4, column=7)
position.insert(0.0, p1)

elif (e == 2):

    p2 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=7)
    position.insert(0.0, p2)

elif (e == 3):

    p3 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=8)
    position.insert(0.0, p3)

elif (e == 4):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (e == 5):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (e == 6):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (e == 7):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
```

Conference Table Arrangement

```
position.insert(0.0, p1)

elif (e == 8):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=7)
    position.insert(0.0, p1)

elif (e == 9):

    p1 = "Person 5 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=8)
    position.insert(0.0, p1)

else:

    p1 = "                Person 5 Seat"
    position = Text(root, height=2, width=25,
fg='red')
    position.grid(row=17, column=7)
    position.insert(0.0, p1)

Label(root, text="Person 5").grid(row=7, column=1)
sp5=Entry(root)
sp5.grid(row=7, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_5).grid(row=7, column= 4)

def arrange_seat_6():
    f=sp6.get()
    f=int(f)

    if (f == 0 or f == 1):

        p1= "                Person 6 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (f == 2):
```


Conference Table Arrangement

```
p2 = "Person 6 Seat"
position = Text(root, height=2, width=15,
fg='red')
position.grid(row=6, column=7)
position.insert(0.0, p2)

elif (f == 3):

    p3 = "Person 6 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=8)
    position.insert(0.0, p3)

elif (f == 4):

    p1 = "Person 6 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (f == 5):

    p1 = "Person 6 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (f == 6):

    p1 = "Person 6 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (f == 7):

    p1 = "Person 6 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
    position.insert(0.0, p1)

elif (f == 8):

    p1 = "Person 6 Seat"
```

Conference Table Arrangement

```
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=7)
        position.insert(0.0, p1)

    elif (f == 9):

        p1 = "Person 6 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=8)
        position.insert(0.0, p1)

    else:

        p1 = "                Person 6 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=17, column=7)
        position.insert(0.0, p1)

Label(root, text="Person 6").grid(row=8, column=1)
sp6=Entry(root)
sp6.grid(row=8, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_6).grid(row=8, column= 4)

def arrange_seat_7():
    g=sp7.get()
    g=int(g)

    if (g == 0 or g == 1):

        p1= "                Person 7 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (g == 2):

        p2 = "Person 7 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=6, column=7)
```

Conference Table Arrangement

```
position.insert(0.0, p2)

elif (g == 3):

    p3 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=6, column=8)
    position.insert(0.0, p3)

elif (g == 4):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (g == 5):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (g == 6):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (g == 7):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
    position.insert(0.0, p1)

elif (g == 8):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=7)
    position.insert(0.0, p1)
```

Conference Table Arrangement

```
elif (g == 9):

    p1 = "Person 7 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=8)
    position.insert(0.0, p1)

else:

    p1 = "          Person 7 Seat"
    position = Text(root, height=2, width=25,
fg='red')
    position.grid(row=17, column=7)
    position.insert(0.0, p1)

Label(root, text="Person 7").grid(row=9, column=1)
sp7=Entry(root)
sp7.grid(row=9, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_7).grid(row=9, column= 4)

def arrange_seat_8():

    h=sp8.get()
    h=int(h)

    if (h == 0 or h == 1):

        p1= "          Person 8 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (h == 2):

        p2 = "Person 8 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=5, column=7)
        position.insert(0.0, p2)

    elif (h == 3):
```

Conference Table Arrangement

```
p3 = "Person 8 Seat"
position = Text(root, height=2, width=15,
fg='red')
position.grid(row=6, column=8)
position.insert(0.0, p3)

elif (h == 4):

    p1 = "Person 8 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (h == 5):

    p1 = "Person 8 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (h == 6):

    p1 = "Person 8 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (h == 7):

    p1 = "Person 8 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
    position.insert(0.0, p1)

elif (h == 8):

    p1 = "Person 8 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=7)
    position.insert(0.0, p1)

elif (h == 9):
```

Conference Table Arrangement

```
p1 = "Person 8 Seat"
position = Text(root, height=2, width=15,
fg='red')
position.grid(row=12, column=8)
position.insert(0.0, p1)

else :

    p1 = "                Person 8 Seat"
    position = Text(root, height=2, width=25,
fg='red')
    position.grid(row=17, column=7)
    position.insert(0.0, p1)

Label(root, text="Person 8").grid(row=10, column=1)
sp8=Entry(root)
sp8.grid(row=10, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_8).grid(row=10, column= 4)

def arrange_seat_9():

    i=sp9.get()
    i=int(i)

    if (i == 0 or i == 1):

        p1= "                Person 9 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (i == 2):

        p2 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=5, column=7)
        position.insert(0.0, p2)

    elif (i == 3):

        p3 = "Person 9 Seat"
```

Conference Table Arrangement

```
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=6, column=8)
        position.insert(0.0, p3)

    elif (i == 4):

        p1 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=8, column=7)
        position.insert(0.0, p1)

    elif (i == 5):

        p1 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=8, column=8)
        position.insert(0.0, p1)

    elif (i == 6):

        p1 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=7)
        position.insert(0.0, p1)

    elif (i == 7):

        p1 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=10, column=8)
        position.insert(0.0, p1)

    elif (i == 8):

        p1 = "Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=7)
        position.insert(0.0, p1)

    elif (i == 9):

        p1 = "Person 9 Seat"
```

Conference Table Arrangement

```
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=12, column=8)
        position.insert(0.0, p1)

    else:
        p1 = "                Person 9 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=17, column=7)
        position.insert(0.0, p1)

Label(root, text="Person 9").grid(row=11, column=1)
sp9=Entry(root)
sp9.grid(row=11, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_9).grid(row=11, column= 4)

def arrange_seat_10():
    j=sp10.get()
    j=int(j)

    if (j == 0 or j == 1):

        p1= "                Person 10 Seat"
        position = Text(root, height=2, width=25,
fg='red')
        position.grid(row=4, column=7)
        position.insert(0.0, p1)

    elif (j == 2):

        p2 = "Person 10 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=5, column=7)
        position.insert(0.0, p2)

    elif (j == 3):

        p3 = "Person 10 Seat"
        position = Text(root, height=2, width=15,
fg='red')
        position.grid(row=6, column=8)
        position.insert(0.0, p3)
```


Conference Table Arrangement

```
elif (j == 4):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=7)
    position.insert(0.0, p1)

elif (j == 5):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=8, column=8)
    position.insert(0.0, p1)

elif (j == 6):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=7)
    position.insert(0.0, p1)

elif (j == 7):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=10, column=8)
    position.insert(0.0, p1)

elif (j == 8):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=7)
    position.insert(0.0, p1)

elif (j == 9):

    p1 = "Person 10 Seat"
    position = Text(root, height=2, width=15,
fg='red')
    position.grid(row=12, column=8)
    position.insert(0.0, p1)

else:
```

Conference Table Arrangement

```
p1 = "                Person 10 Seat"
position = Text(root, height=2, width=25,
fg='red')
position.grid(row=17, column=7)
position.insert(0.0, p1)

Label(root, text="Person 10").grid(row=12, column=1)
sp10=Entry(root)
sp10.grid(row=12, column=3)
Button(root, text='Arrange
seat',command=arrange_seat_10).grid(row=12, column= 4)

Button(root,text='                Exit
',command=root.destroy).grid(row=18,column=5)

Label(root, text="                TABLE").grid(row=3,
column= 7)

position = Text(root, height=2, width=17, fg='green')
position.grid(row=4, column=7)
position.insert(END, '                Person')

position = Text(root, height=2, width=15, fg='green')
position.grid(row=6, column=7)
position.insert(END, 'Person')

position = Text(root, height=2, width=15, fg='green')
position.grid(row=6, column=8)
position.insert(END, 'Person')

position = Text(root, height=2, width=15, fg='green')
position.grid(row=8, column=7)
position.insert(END, 'Person')

position = Text(root, height=2, width=15, fg='green')
position.grid(row=8, column=8)
position.insert(END, 'Person')

position = Text(root, height=2, width=15, fg='green')
position.grid(row=10, column=7)
```

Conference Table Arrangement

```
position.insert(END, 'Person')
```

```
position = Text(root, height=2, width=15, fg='green')  
position.grid(row=10, column=8)  
position.insert(END, 'Person')
```

```
position = Text(root, height=2, width=15, fg='green')  
position.grid(row=12, column=7)  
position.insert(END, 'Person')
```

```
position = Text(root, height=2, width=15, fg='green')  
position.grid(row=12, column=8)  
position.insert(END, 'Person')
```

```
position = Text(root, height=2, width=17, fg='green')  
position.grid(row=17, column=7)  
position.insert(END, 'Person')
```

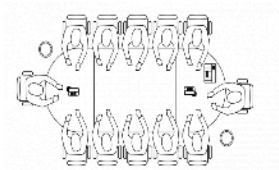
```
root.mainloop()
```

Conference Table Arrangement

OUTPUT

Conference table

Conference Table Arrangement



Person Priority

Person 1 Arrange seat

Person 2 Arrange seat

Person 3 Arrange seat

Person 4 Arrange seat

Person 5 Arrange seat

Person 6 Arrange seat

Person 7 Arrange seat

Person 8 Arrange seat

Person 9 Arrange seat

Person 10 Arrange seat

Exit

TABLE

Person

Person Person

Person Person

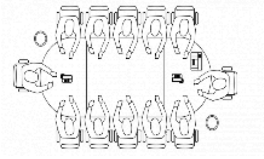
Person Person

Person Person

Person

Conference table

Conference Table Arrangement



Person Priority

Person 1 Arrange seat

Person 2 Arrange seat

Person 3 Arrange seat

Person 4 Arrange seat

Person 5 Arrange seat

Person 6 Arrange seat

Person 7 Arrange seat

Person 8 Arrange seat

Person 9 Arrange seat

Person 10 Arrange seat

Exit

TABLE

Person

Person Person

Person Person

Person Person


Person Person

Person Person

Person 1 Seat

Conference Table Arrangement

Conference Table Arrangement



Person	Priority	
Person 1	<input type="text" value="10"/>	<button>Arrange seat</button>
Person 2	<input type="text" value="3"/>	<button>Arrange seat</button>
Person 3	<input type="text" value="1"/>	<button>Arrange seat</button>
Person 4	<input type="text" value="5"/>	<button>Arrange seat</button>
Person 5	<input type="text" value="9"/>	<button>Arrange seat</button>
Person 6	<input style="border: 2px solid blue;" type="text" value="6"/>	<button>Arrange seat</button>
Person 7	<input type="text"/>	<button>Arrange seat</button>
Person 8	<input type="text"/>	<button>Arrange seat</button>
Person 9	<input type="text"/>	<button>Arrange seat</button>
Person 10	<input type="text"/>	<button>Arrange seat</button>

TABLE

Person 3 Seat
Person 2 Seat

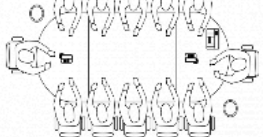
Person
Person 4 Seat

Person 6 Seat
Person

Person
Person 5 Seat

Person 1 Seat

Conference table



Conference Table Arrangement

Person	Priority	
Person 1	<input type="text" value="10"/>	<button>Arrange seat</button>
Person 2	<input type="text" value="3"/>	<button>Arrange seat</button>
Person 3	<input type="text" value="1"/>	<button>Arrange seat</button>
Person 4	<input type="text" value="5"/>	<button>Arrange seat</button>
Person 5	<input type="text" value="9"/>	<button>Arrange seat</button>
Person 6	<input type="text" value="6"/>	<button>Arrange seat</button>
Person 7	<input type="text" value="2"/>	<button>Arrange seat</button>
Person 8	<input type="text" value="4"/>	<button>Arrange seat</button>
Person 9	<input type="text" value="7"/>	<button>Arrange seat</button>
Person 10	<input type="text" value="8"/>	<button>Arrange seat</button>

TABLE

Person 3 Seat

Person 7 Seat Person 2 Seat

Person 8 Seat Person 4 Seat

Person 6 Seat Person 9 Seat

Person 10 Seat Person 5 Seat

Person 1 Seat

Exit

CONCLUSION

The Conference Table Arrangement System is developed and it successfully meets the objectives of the system for which it has been developed. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency and all the teachers and user associated with the system understands its advantage. The system solves the problem. It was intended to solve as requirement specification.

REFERENCE

1. GOOGLE
2. GEEKS GEEKS
3. QUORA
4. STACKOVERFLOW

Conference Table Arrangement

Marks:

Signature: