**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOVERNMENT COLLEGE OF ENGINEERING, AMRAVATI**

**B.TECH-THIRD SEMESTER**

**DATA STRUCTURE and ALGORITHMS**

**PROJECT: FILE EXPLORER MENU USING GUI IN PYTHON**

**GUIDED BY : Prof. A. W. BHADE**

**Submitted by**

**Name of Student:-**

**1.VAIBHAV NIRMAL 2.ANUSHA SAHI 3.ADITI NIMKAR Registration ID:-**

**1. 19007058 2. 19007053 3. 190070162**



**Submitted to:**

**GOVERNMENT COLLEGE OF ENGINEERING,**

**AMRAVATI**

**SUMMER-2020**

**PROJECT: FILE EXPLORER MENU USING GUI IN PYTHON**

**INTRODUCTION:-**

**Managing files is an easy task. So, today let’s head over building a file manager. This is a pretty easy application. To make it more interactive and interesting, we will use Tkinter along with Python.**

**You can use File Explorer for a variety of tasks. In addition to management and organization of files and folders, it's also used to view and manage the resources of your computer such as internal storage, attached storage, and optical drives, File management is the process of administering a system that correctly handles digital data. Therefore, an effective file management system improves the overall function of a business workflow. It also organizes important data and provides a searchable database for quick retrieval.**

**PROCEDURE:-**

**Our aim in this article is to come up with our own file manager or file explorer, which can perform operations like opening a file, copying it, renaming, moving, and deleting. Also, we will include options for making and deleting a folder. Along with listing all the files present in the folder.**

**Step 1: Importing required modules**

**We will import the following modules:**

* **Tkinter: It provides wide APIs for GUI functionalities.**
* **Shutil: This module offers a number of high-level operations on files and collections of files.**
* **Os: utility module for OS interaction in programs.**
* **easygui: It allows us to select any file from the system**

**Step 2: Defining functionalities**

**In this step, we will define all the functions which will be linked afterward to different buttons**

**Step 2.1: Opening a file box**

**# open a file box window**

**# when we want to select a file**

def open\_window():

read=easygui.fileopenbox()

return read

**Explanation:**

**The above code opens the file box, i.e the pop-up box to choose the file from the device, which opens every time you run the code. fileopenbox() is the method in easyGUI module which returns the path of the chosen file as a string.**

**We will call this function every time we want to select any file.**

**Step 2.2: Opening a file**

**Opening a file when Open a file button is clicked**

**Code:**

**# open file function**

def open\_file():

string = open\_window()

try:

os.startfile(string)

except:

mb.showinfo('confirmation', "File not found!")

**Explanation:**

**The above code calls open\_window() function and stores the returned path of file chosen in variable “string”. If a file is chosen os.startfile(string) function opens the file. Otherwise it opens a popup message box using mb.showinfo(‘confirmation’, “File not found!”).**

**Step 2.3: Copying a file**

**Copying a file when copy a file button is clicked**

**Code:**

**# copy file function**

def copy\_file():

source1 = open\_window()

destination1=filedialog.askdirectory()

shutil.copy(source1,destination1)

mb.showinfo('confirmation', "File Copied !")

**Explanation:**

**The above code calls open\_window() function and stores the returned path of file chosen in variable “source1”. Variable “destination 1” stores the address of the folder returned by askdirectory() function in filedialog module.**

**shutil.copy(source1, destination1) copies a file present at source1 to destination1 address. As a result, it opens a popup message box using mb.showinfo(‘confirmation’, “File Copied !”).**

**Step 2.4: Deleting a file**

**Deleting a file when Delete a file button is clicked**

**Code:**

**# delete file function**

def delete\_file():

del\_file = open\_window()

if os.path.exists(del\_file):

os.remove(del\_file)

else:

mb.showinfo('confirmation', "File not found !")

**Explanation:**

**The above code calls open\_window() function and stores the returned path of file chosen in variable “string”. If a file is chosen os.remove(string) function deletes the file. Otherwise, it opens a popup message box using mb.showinfo(‘confirmation’, “File not found!”).**

**Step 2.5: Renaming a file**

**Renaming a file when Rename a file button is clicked**

**Code:**

**# rename file function**

def rename\_file():

chosenFile = open\_window()

path1 = os.path.dirname(chosenFile)

extension=os.path.splitext(chosenFile)[1]

print("Enter new name for the chosen file")

newName=input()

path = os.path.join(path1, newName+extension)

print(path)

os.rename(chosenFile,path)

mb.showinfo('confirmation', "File Renamed !")

**Explanation:**

**The above code calls open\_window() function and stores the returned path of file chosen in variable “chosenFile”. Then, variable path1 is used to store the path of directory of the file chosen. os.path.splitext() splits the path into root and extensions. As we rename the file, we keep the same extension but need to manipulate only the name of the file mentioned in the root part.**

**Now, program asks the new name of the file. Once entered, the new name of file is stored in “newName” variable. Finally, we join newname to root and join it to extension, forming the path of the file with a new name.**

**os.rename(chosenFile,newName) function renames the chosen file. And, there appears a popup message box using mb.showinfo(‘confirmation’, “File Renamed !”) for confirmation.**

**Step 2.6: Moving a file**

**Moving a file when Move a file button is clicked**

**Code:**

**# move file function**

def move\_file():

source = open\_window()

destination =filedialog.askdirectory()

if(source==destination):

mb.showinfo('confirmation', "Source and destination are same")

else:

shutil.move(source, destination)

mb.showinfo('confirmation', "File Moved !")

**Explanation:**

**The above code calls open\_window() function and stores the returned path of file chosen in variable “source”. And, the variable “destination” stores the address of destination folder address using filedialog.askdirectory().**

**If source and destination are the same, it opens a pop up box saying “Source and destination is same”. Else, the file is moved and opens a popup message box using mb.showinfo(‘confirmation’, “File Moved !”).**

**Step 2.7: Making a folder**

**Making a file when Make a folder button is clicked**

**Code:**

**# function to make a new folder**

def make\_folder():

newFolderPath = filedialog.askdirectory()

print("Enter name of new folder")

newFolder=input()

path = os.path.join(newFolderPath, newFolder)

os.mkdir(path)

mb.showinfo('confirmation', "Folder created !")

**Explanation:**

**Above code uses askdirectory function to open pop up box to select the folder and stores the returned path of the folder chosen in the variable “newFolderPath”.**

**Now, the program asks for the name of the new folder and stores it in the “new/folder” variable.**

**os.path.join() method in Python is used to join one or more path components intelligently. This method concatenates various newFolderPath and nerFolder with exactly one directory separator (‘/’) following non-empty parts except the last path component. Once the appended path is stored in “path” variable, mkdir() method creates a folder at the chosen folder. Finally, a popup message box is displayed using mb.showinfo(‘confirmation’, “Folder Created !”).**

**Step 2.8: Deleting a folder**

**Deleting when Delete a folder button is clicked**

**Code:**

**# function to remove a folder**

def remove\_folder():

delFolder = filedialog.askdirectory()

os.rmdir(delFolder)

mb.showinfo('confirmation', "Folder Deleted !")

**Explanation:**

**The above code uses askdirectory function to open pop up box to select the folder and stores the returned path of folder chosen in the variable “delFolder”.**

**Now, rmdir() method deletes the folder selected. Finally, a popup message box is displayed using mb.showinfo(‘confirmation’, “Folder Created !”).**

**Step 2.9: Listing files in a folder**

**Listing files in a folder when List all files in a directory button is clicked.**

**Code:**

**# function to list all the files in folder**

def list\_files():

folderList = filedialog.askdirectory()

sortlist=sorted(os.listdir(folderList))

i=0

print("Files in ", folderList, "folder are:")

while(i<len(sortlist)):

print(sortlist[i]+'\n')

i+=1

**Explanation:**

**The above code uses askdirectory function to open pop up box to select the folder and stores the returned path of the folder chosen in the variable “folderList”.**

**Now, lstdir() method returns the list of all the files present in the folder selected. The list is sorted, and displayed using a while loop.**

**Step 3: Building File Manager UI using Tkinter**

**Code:**

**root = Tk()  
root.geometry("495x700")  
root.maxsize(495,700);root.minsize(400,600);  
root.title("My Group Project using GUI")  
root.configure(background="darkorchid3")  
Label(root, text="Created by Vaibhav, Anusha ,Aditi", font=("Helvetica 10 bold"), fg="black", padx="100", pady="15", bg="gray39"  
 , relief=RAISED  
 , borderwidth="5").grid(row=95, column=2)  
  
Label(root, text=" AVA File Explorer Menu ", font=("Helvetica 16 bold"), fg="black", padx="100", pady="30", bg="gray"  
 , relief=RAISED, borderwidth="10").grid(row=5, column=2)  
  
Button(root, text="Open a File", command=open\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=15, column=2)  
  
Button(root, text="Copy a File", command=copy\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=25, column=2)  
  
Button(root, text="Delete a File", command=delete\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=35, column=2)  
  
Button(root, text="Rename a File", command=rename\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=45, column=2)  
  
Button(root, text="Move a File", command=move\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=55, column=2)  
  
Button(root, text="Make a Folder", command=make\_folder, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=75, column=2)  
  
Button(root, text="Remove a Folder", command=remove\_folder, padx="20", pady="5", borderwidth="10", bg="black",  
 fg="white", font=("Helvetica 16 bold")).grid(row=65, column=2)  
  
Button(root, text="List all Files in Directory", command=list\_files, padx="20", pady="5", borderwidth="10", bg="black",  
 fg="white", font=("Helvetica 16 bold")).grid(row=85, column=2)  
root.mainloop()**

**Explanation:**

**The above code is used to create a tkinter UI.**

**root=TK() is used to make a window. We use canvas() to create a canvas, and an image is applied in the background of our window using ImageTk.**

**Tkinter has two types of layout managers, grid and pack. For layout simplicity, we use a grid layout manager. Simply, using grid, we place every button in a single column.**

**Label() is used to place text label in the root window, with defined font, font size, and color. Row and column are also specified at the end.**

**Button() function placed in root window with text specified as text=”..” . The next parameter, “command” specifies the function which gets executed whenever a button is clicked. Please note, the function is specified without brackets in the statement. Similar to label, in every grid widget, we can specify row and column. To place the buttons one below the other, we specify the same column number, but change the row according to our need.**

**Implementing Queues**

Queue Methods

•queue.Enqueue()

The queue.Enqueue() method adds an element at the rear of the queue.

Time Complexity -> O(1)

•queue.Dequeue()

The queue.Dequeue() method removes an element from the front of the queue.

Time Complexity -> O(1)

•queue.Front()

The queue.Front() method returns the front item from the queue.

Time Complexity -> O(1)

•queue.Rear()

The queue.Rear() method returns the rear item from the queue.

Time Complexity -> O(1)

•queue.isEmpty()

The queue.isEmpty() method returns True if the queue is empty, else returns False.

Time Complexity -> O(1)

**Queue in Python can be implemented by the following ways:**

**•list**

**•collections.deque**

**•queue.Queue**

Implementation using list :

List is a Python’s built-in data structure that can be used as a queue. Instead of enqueue() and dequeue(), append() and pop() function is used. However, lists are quite slow for this purpose because inserting or deleting an element at the beginning requires shifting all of the other elements by one, requiring O(n) time.

Code:

**print("\n \t<<\tWELCOME TO AVA FILE EXPLORER MENU\t>>\n \t>>>This mini project is created by :")  
  
class Queue:  
 def \_\_init\_\_(self):  
 self.s1 = []  
 self.s2 = []  
  
 def enQueue(self, x):  
 self.s1.append(x)  
  
 def deQueue(self):  
  
 if len(self.s1) == 0 and len(self.s2) == 0:  
 print("Q is Empty")  
 return  
  
 elif len(self.s2) == 0 and len(self.s1) > 0:  
 while len(self.s1):  
 temp = self.s1.pop()  
 self.s2.append(temp)  
 return self.s2.pop()  
  
 else:  
 return self.s2.pop()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 q = Queue()  
 q.enQueue('\t>>> Vaibhav Nirmal \tID:(19007058)')  
 q.enQueue('\t>>> Aditi Nimkar \tID:(19007062)')  
 q.enQueue('\t>>> Anusha Sahi \tID:(19007053)')  
  
 print(q.deQueue())  
 print(q.deQueue())  
 print(q.deQueue())**

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

**SOURCE CODE OF FILE EXPLORER :-**

**from tkinter import \*  
from PIL import ImageTk, Image  
import shutil  
import os  
import easygui  
import pygame  
  
  
from tkinter import filedialog  
from tkinter import messagebox as mb  
  
  
def open\_window():  
 read = easygui.fileopenbox()  
 return read  
  
def open\_file():  
 string = open\_window()  
 try:  
 os.startfile(string)  
 except:  
 mb.showinfo('confirmation', "File not found!")  
  
def copy\_file():  
 source1 = open\_window()  
 destination1 = filedialog.askdirectory()  
 shutil.copy(source1, destination1)  
 mb.showinfo('confirmation', "File Copied !")  
  
  
def delete\_file():  
 del\_file = open\_window()  
 if os.path.exists(del\_file):  
 os.remove(del\_file)  
 else:  
 mb.showinfo('confirmation', "File not found !")  
  
  
def rename\_file():  
 chosenFile = open\_window()  
 path1 = os.path.dirname(chosenFile)  
 extension = os.path.splitext(chosenFile)[1]  
 print("Enter new name for the chosen file")  
 newName = input()  
 path = os.path.join(path1, newName + extension)  
 print(path)  
 os.rename(chosenFile, path)  
 mb.showinfo('confirmation', "File Renamed !")  
  
  
def move\_file():  
 source = open\_window()  
 destination = filedialog.askdirectory()  
 if (source == destination):  
 mb.showinfo('confirmation', "Source and destination are same")  
 else:  
 shutil.move(source, destination)  
 mb.showinfo('confirmation', "File Moved !")  
  
  
def make\_folder():  
 newFolderPath = filedialog.askdirectory()  
 print("Enter name of new folder")  
 newFolder = input()  
 path = os.path.join(newFolderPath, newFolder)  
 os.mkdir(path)  
 mb.showinfo('confirmation', "Folder created !")  
  
  
def remove\_folder():  
 delFolder = filedialog.askdirectory()  
 os.rmdir(delFolder)  
  
 mb.showinfo('confirmation', "Folder Deleted !")  
  
  
def list\_files():  
 folderList = filedialog.askdirectory()  
 sortlist = sorted(os.listdir(folderList))  
 i = 0  
 print("Files in ", folderList, "folder are:")  
 while (i < len(sortlist)):  
 print(sortlist[i] + '\n')  
 i += 1  
  
print("\n \t<<\tWELCOME TO AVA FILE EXPLORER MENU\t>>\n \t>>>This mini project is created by :")  
  
class Queue:  
 def \_\_init\_\_(self):  
 self.s1 = []  
 self.s2 = []  
  
 def enQueue(self, x):  
 self.s1.append(x)  
  
 def deQueue(self):  
  
 if len(self.s1) == 0 and len(self.s2) == 0:  
 print("Q is Empty")  
 return  
  
 elif len(self.s2) == 0 and len(self.s1) > 0:  
 while len(self.s1):  
 temp = self.s1.pop()  
 self.s2.append(temp)  
 return self.s2.pop()  
  
 else:  
 return self.s2.pop()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 q = Queue()  
 q.enQueue('\t>>> Vaibhav Nirmal \tID:(19007058)')  
 q.enQueue('\t>>> Aditi Nimkar \tID:(19007062)')  
 q.enQueue('\t>>> Anusha Sahi \tID:(19007053)')  
  
 print(q.deQueue())  
 print(q.deQueue())  
 print(q.deQueue())  
  
root = Tk()  
root.geometry("495x700")  
root.maxsize(495,700);root.minsize(400,600);  
root.title("My Group Project using GUI")  
root.configure(background="darkorchid3")  
Label(root, text="Created by Vaibhav, Anusha ,Aditi", font=("Helvetica 10 bold"), fg="black", padx="100", pady="15", bg="gray39"  
 , relief=RAISED  
 , borderwidth="5").grid(row=95, column=2)  
  
Label(root, text=" AVA File Explorer Menu ", font=("Helvetica 16 bold"), fg="black", padx="100", pady="30", bg="gray"  
 , relief=RAISED, borderwidth="10").grid(row=5, column=2)  
  
Button(root, text="Open a File", command=open\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=15, column=2)  
  
Button(root, text="Copy a File", command=copy\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=25, column=2)  
  
Button(root, text="Delete a File", command=delete\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=35, column=2)  
  
Button(root, text="Rename a File", command=rename\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=45, column=2)  
  
Button(root, text="Move a File", command=move\_file, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=55, column=2)  
  
Button(root, text="Make a Folder", command=make\_folder, padx="20", pady="5", borderwidth="10", bg="black", fg="white",  
 font=("Helvetica 16 bold")).grid(row=75, column=2)  
  
Button(root, text="Remove a Folder", command=remove\_folder, padx="20", pady="5", borderwidth="10", bg="black",  
 fg="white", font=("Helvetica 16 bold")).grid(row=65, column=2)  
  
Button(root, text="List all Files in Directory", command=list\_files, padx="20", pady="5", borderwidth="10", bg="black",  
 fg="white", font=("Helvetica 16 bold")).grid(row=85, column=2)  
root.mainloop()**

**OUTPUT:-**



