OS Assignment - 10
Name: Veeransh Shah
Reg Id: 221070063

Aim:
The aim of this project is to develop a File Explorer application using Python's Tkinter library that provides graphical user interface (GUI) functionality. The application allows users to navigate the file system, create and delete directories and files, edit files using an external text editor, and navigate through directory history.

Theory:

The File Explorer application utilizes Python's Tkinter library to create a graphical user interface that allows users to navigate and manage the filesystem directly from the application. Tkinter provides a robust framework for GUI development, enabling the creation of intuitive layouts and interactive elements such as buttons, list boxes, and dialog boxes. This project leverages the os module for handling directory and file operations like listing contents, changing directories, creating, and deleting files or directories. The subprocess module is employed to integrate external applications, allowing the editing of files in an external text editor like gedit, demonstrating Python's capability to interact with other programs.

For file and directory management, Python's built-in functions from the os module are crucial, providing a platform-independent way of handling file system operations. The event-driven architecture of Tkinter ensures that the application remains responsive, with user actions triggering specific functions coded in the application. This approach not only simplifies the complexity involved in file management but also enhances user experience by providing a graphical rather than command-line interface.

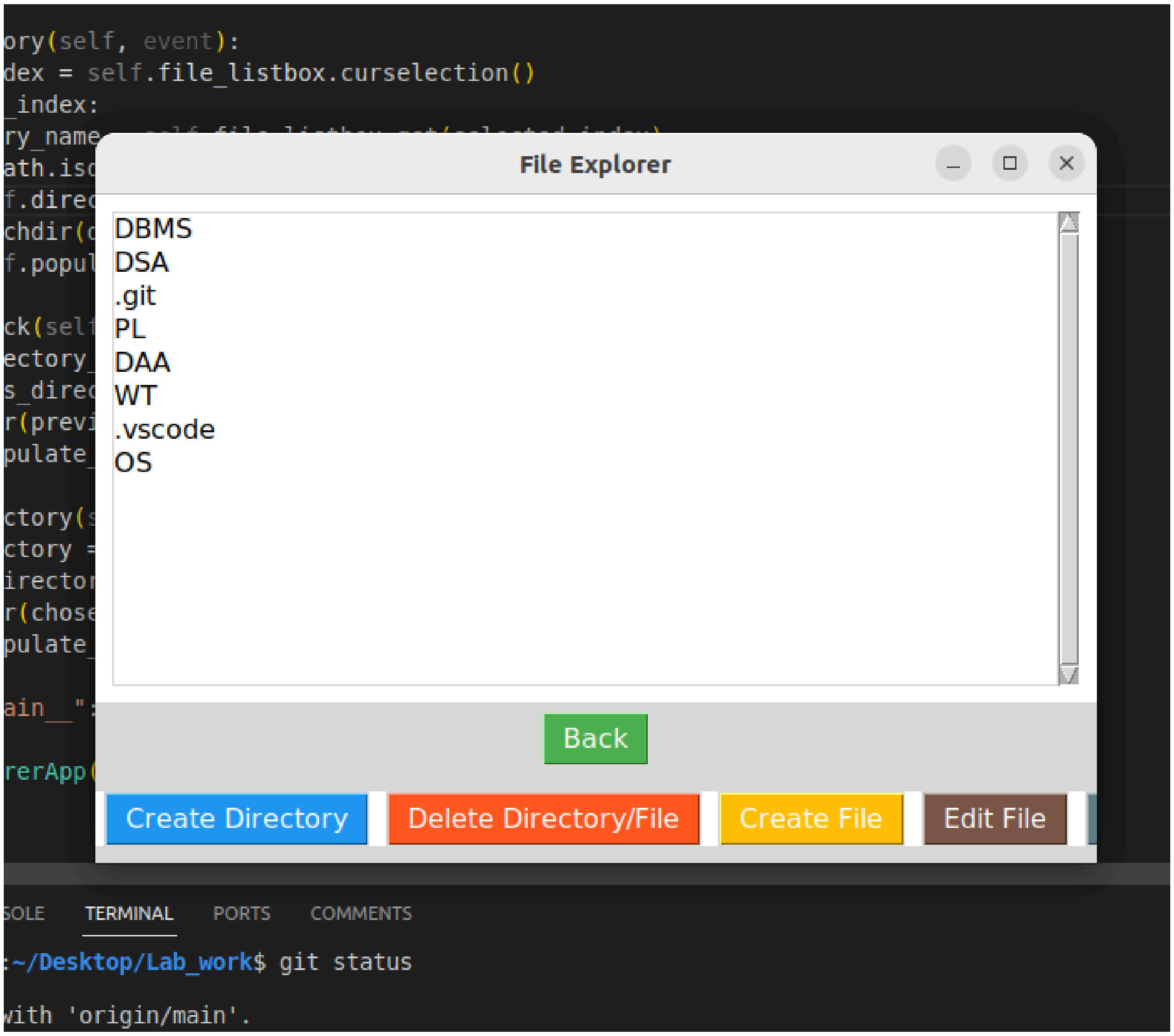
Code:

```python
import subprocess
import os
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog

class FileExplorerApp:
   def __init__(self, root):
       self.root = root
       self.root.title("File Explorer")

       # Set window icon
       # Uncomment the next line after placing the correct path to your
icon file
       # self.root.iconbitmap("path_to_icon_file")

       # Set window size and position
       window_width = 600
       window_height = 400
       screen_width = root.winfo_screenwidth()
       screen_height = root.winfo_screenheight()
       x = (screen_width - window_width) // 2
       y = (screen_height - window_height) // 2
       root.geometry(f"{window_width}x{window_height}+{x}+{y}")

       # Main frame
       self.frame = tk.Frame(self.root, bg="white")
       self.frame.pack(fill=tk.BOTH, expand=True)

       # File listbox
       self.file_listbox = tk.Listbox(self.frame, selectmode=tk.SINGLE,
bg="white", bd=0, font=("Algerian", 12))
       self.file_listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True,
padx=(10, 0), pady=10)
       self.file_listbox.bind("<Double-Button-1>", self.open_directory)

       # Scrollbar
       self.scrollbar = tk.Scrollbar(self.frame, orient=tk.VERTICAL,
command=self.file_listbox.yview)
```

```
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y, padx=(0, 10),
pady=10)
        self.file_listbox.config(yscrollcommand=self.scrollbar.set)

        # Back button
        self.back_button = tk.Button(self.root, text="Back",
command=self.navigate_back, bg="#4CAF50", fg="white", font=("Algerian",
12), padx=10)
        self.back_button.pack(pady=5)

        # Buttons frame
        self.button_frame = tk.Frame(self.root, bg="white")
        self.button_frame.pack(pady=10)

        # Create Directory button
        self.create_dir_button = tk.Button(self.button_frame, text="Create
Directory", command=self.create_directory, bg="#2196F3", fg="white",
font=("Algerian", 12), padx=10)
        self.create_dir_button.grid(row=0, column=0, padx=5)

        # Delete Directory/File button
        self.delete_dir_button = tk.Button(self.button_frame, text="Delete
Directory/File", command=self.delete_item, bg="#FF5722", fg="white",
font=("Algerian", 12), padx=10)
        self.delete_dir_button.grid(row=0, column=1, padx=5)

        # Create File button
        self.create_file_button = tk.Button(self.button_frame, text="Create
File", command=self.create_file, bg="#FFC107", fg="white",
font=("Algerian", 12), padx=10)
        self.create_file_button.grid(row=0, column=2, padx=5)

        # Edit File button
        self.edit_file_button = tk.Button(self.button_frame, text="Edit
File", command=self.edit_file, bg="#795548", fg="white", font=("Algerian",
12), padx=10)
        self.edit_file_button.grid(row=0, column=3, padx=5)

        # Change Directory button
        self.change_dir_button = tk.Button(self.button_frame, text="Change
Directory", command=self.change_directory, bg="#607D8B", fg="white",
font=("Algerian", 12), padx=10)
        self.change_dir_button.grid(row=0, column=4, padx=5)

        # Stack to store directory history
        self.directory_stack = []
        self.populate_listbox()

    def populate_listbox(self):
        self.file_listbox.delete(0, tk.END)
        current_dir = os.getcwd()
        files = os.listdir(current_dir)
        for file in files:
            self.file_listbox.insert(tk.END, file)

    def create_directory(self):
        directory_name = simpledialog.askstring("Create Directory", "Enter
directory name:")
        if directory_name:
            try:
                os.mkdir(directory_name)
                messagebox.showinfo("Success", f"Directory
'{directory_name}' created successfully.")
                self.populate_listbox()
            except OSError as e:
                messagebox.showerror("Error", f"Failed to create directory:
{e}")

    def delete_item(self):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            item_name = self.file_listbox.get(selected_index)
            item_path = os.path.join(os.getcwd(), item_name)
            confirm = messagebox.askyesno("Confirm Deletion", f"Are you
sure you want to delete '{item_name}'?")
            if confirm:
                try:
                    if os.path.isdir(item_path):
                        os.rmdir(item_path)
```

```
                   else:
                       os.remove(item_path)
                   messagebox.showinfo("Success", f"'{item_name}' deleted
successfully.")
                   self.populate_listbox()
               except OSError as e:
                   messagebox.showerror("Error", f"Failed to delete: {e}")
       else:
           messagebox.showwarning("Warning", "Please select an item to
delete.")

   def create_file(self):
       file_name = simpledialog.askstring("Create File", "Enter file
name:")
       if file_name:
           try:
               with open(file_name, 'w') as file:
                   pass
               messagebox.showinfo("Success", f"File '{file_name}' created
successfully.")
               self.populate_listbox()
           except OSError as e:
               messagebox.showerror("Error", f"Failed to create file:
{e}")

   def edit_file(self):
       selected_index = self.file_listbox.curselection()
       if selected_index:
           file_name = self.file_listbox.get(selected_index)
           try:
               subprocess.Popen(["gedit", file_name])
           except OSError as e:
               messagebox.showerror("Error", f"Failed to edit file: {e}")
       else:
           messagebox.showwarning("Warning", "Please select a file to
edit.")

   def open_directory(self, event):
       selected_index = self.file_listbox.curselection()
       if selected_index:
           directory_name = self.file_listbox.get(selected_index)
           if os.path.isdir(directory_name):
               self.directory_stack.append(os.getcwd())
               os.chdir(directory_name)
               self.populate_listbox()

   def navigate_back(self):
       if self.directory_stack:
           previous_directory = self.directory_stack.pop()
           os.chdir(previous_directory)
           self.populate_listbox()

   def change_directory(self):
       chosen_directory = filedialog.askdirectory()
       if chosen_directory:
           os.chdir(chosen_directory)
           self.populate_listbox()

if __name__ == "__main__":
   root = tk.Tk()
   app = FileExplorerApp(root)
   root.mainloop()
```

Output:



```
ory(self, event):
dex = self.file_listbox.curselection()
_index:
ry_name
ath.isd
f.direc
chdir(c
f.popul

ck(self
ectory_
s_direc
r(previ
pulate_

ctory(s
ctory =
irector
r(chose
pulate_

ain__":

rerApp(
```

```
DBMS
DSA
.git
PL
DAA
WT
.vscode
OS
```

```
:~/Desktop/Lab_work$ git status

with 'origin/main'.
```

Conclusion:

This File Explorer application effectively showcases the practical use of Python's Tkinter and os modules for creating a fully functional desktop tool for file management. It serves as an exemplary project for understanding how to build interactive applications in Python that require direct interaction with the operating system's filesystem.