DIVYANSHU MAKHARIA
221070036

ASSIGNMENT 10

AIM: Implement file system

THEORY:

File handling is an important activity in every web app. The types of activities that you can perform on the opened file are controlled by Access Modes. These describe how the file will be used after it has been opened.

In Python, there are six methods or access modes, which are:

1. Read Only ('r'): This mode opens the text files for reading only. The start of the file is where the handle is located. It raises the I/O error if the file does not exist. This is the default mode for opening files as well.
2. Read and Write ('r+'): This method opens the file for both reading and writing. The start of the file is where the handle is located. If the file does not exist, an I/O error gets raised.
3. Write Only ('w'): This mode opens the file for writing only. The data in existing files are modified and overwritten. The start of the file is where the handle is located. If the file does not already exist in the folder, a new one gets created.
4. Write and Read ('w+'): This mode opens the file for both reading and writing. The text is overwritten and deleted from an existing file. The start of the file is where the handle is located.
5. Append Only ('a'): This mode allows the file to be opened for writing. If the file doesn't yet exist, a new one gets created. The handle is set at the end of the file. The newly written data will be added at the end, following the previously written data.
6. Append and Read ('a+'): Using this method, you can read and write in the file. If the file doesn't already exist, one gets created. The handle is set at the end of the file. The newly

written text will be added at the end, following the previously written data.

CODE:

```
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")

        self.frame = tk.Frame(self.root)
        self.frame.pack(fill=tk.BOTH, expand=True)

        self.file_listbox = tk.Listbox(self.frame, selectmode=tk.SINGLE)
        self.file_listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.file_listbox.bind("<Double-Button-1>", self.open_directory)

        self.scrollbar = tk.Scrollbar(self.frame, orient=tk.VERTICAL)
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        self.file_listbox.config(yscrollcommand=self.scrollbar.set)
        self.scrollbar.config(command=self.file_listbox.yview)

        self.back_button = tk.Button(self.root, text="Back",
command=self.navigate_back)
        self.back_button.pack(pady=5)

        self.directory_stack = []  # Stack to store directory history
        # self.current_directory = os.path.expanduser('~')  # Initial
directory set to home directory
        self.populate_listbox()
        # Buttons
        self.create_dir_button = tk.Button(self.root, text="Create
Directory", command=self.create_directory)
        self.create_dir_button.pack(pady=5)
```

```
        self.delete_dir_button = tk.Button(self.root, text="Delete
Directory", command=self.delete_directory)
        self.delete_dir_button.pack(pady=5)

        self.create_file_button = tk.Button(self.root, text="Create File",
command=self.create_file)
        self.create_file_button.pack(pady=5)

        self.delete_file_button = tk.Button(self.root, text="Delete File",
command=self.delete_file)
        self.delete_file_button.pack(pady=5)

        self.edit_file_button = tk.Button(self.root, text="Edit File",
command=self.edit_file)
        self.edit_file_button.pack(pady=5)

    def populate_listbox(self):
        # Clear existing items
        self.file_listbox.delete(0, tk.END)

        # Get list of files in current directory
        # directory_name="C:\Users\Divyanshu"

        current_dir = os.getcwd()
        files = os.listdir(current_dir)

        # Add files to listbox
        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()  # Refresh list after creation
            except OSError as e:
```

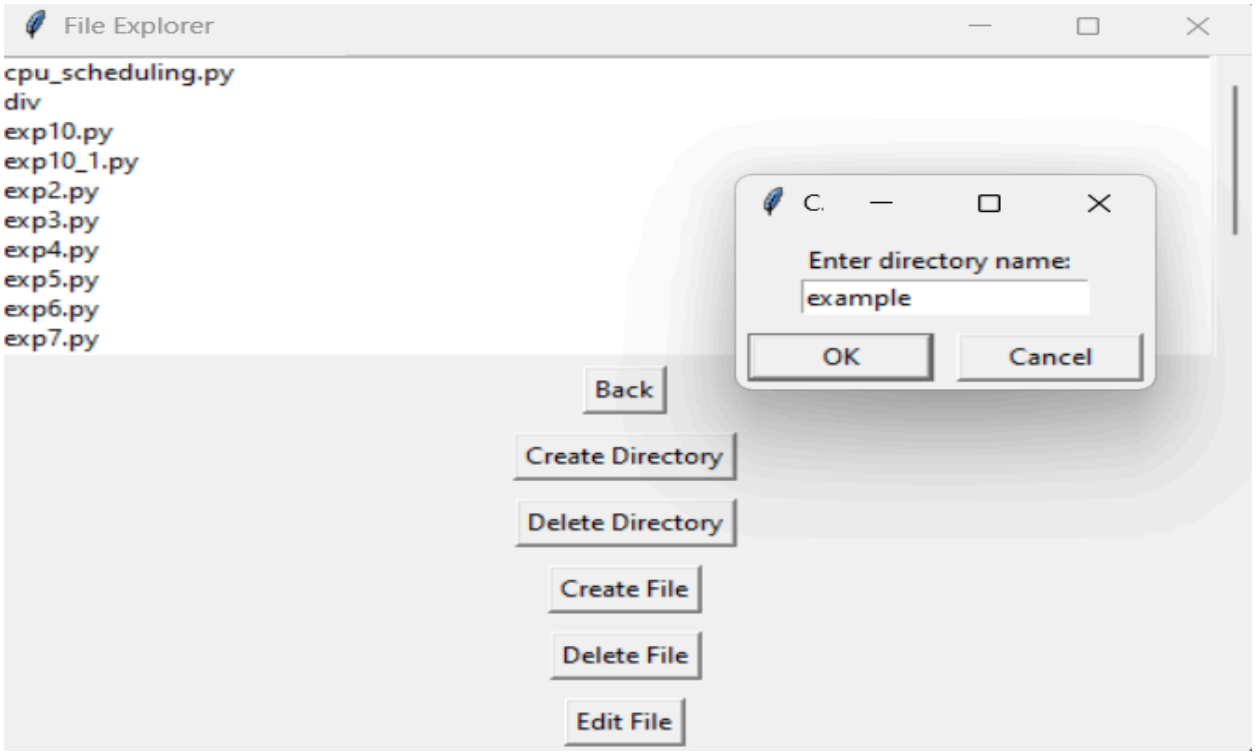
```
                messagebox.showerror("Error", f"Failed to create
directory: {e}")

    def delete_directory(self):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            directory_name = self.file_listbox.get(selected_index)
            confirm = messagebox.askyesno("Confirm Deletion", f"Are you
sure you want to delete '{directory_name}'?")
            if confirm:
                try:
                    os.rmdir(directory_name)
                    messagebox.showinfo("Success", f"Directory
'{directory_name}' deleted successfully.")
                    self.populate_listbox()  # Refresh list after deletion
                except OSError as e:
                    messagebox.showerror("Error", f"Failed to delete
directory: {e}")
        else:
            messagebox.showwarning("Warning", "Please select a directory
to delete.")

    def create_file(self):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            directory_name = self.file_listbox.get(selected_index)
        else:
            directory_name=os.getcwd()
        file_name = simpledialog.askstring("Create File", f"Enter file
name in '{directory_name}':")
        if file_name:
            try:
                    if directory_name:
                     with open(os.path.join(directory_name, file_name),
'w'):
                        pass  # Create an empty file
                    else:
                        directory_name=os.getcwd()
                        with open(os.path.join(directory_name, file_name),
'w'):
```

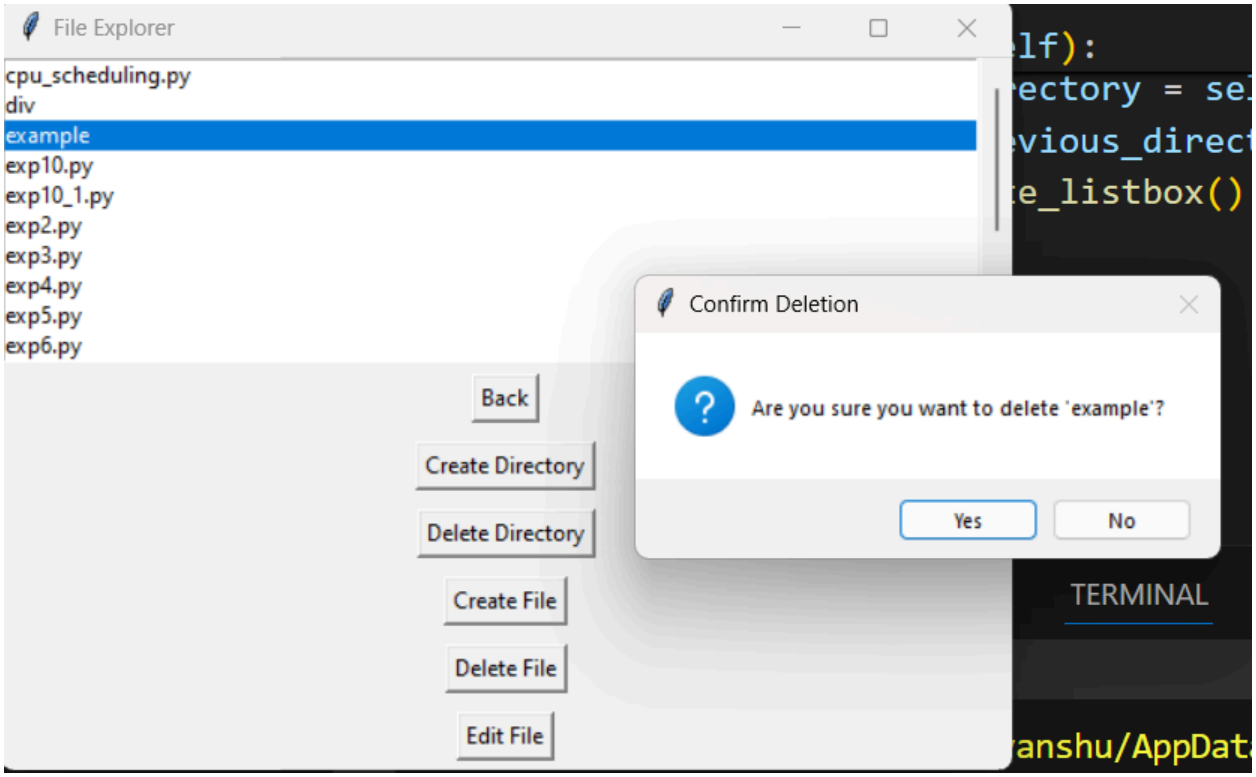
```
                        pass
                    messagebox.showinfo("Success", f"File '{file_name}'
created successfully in '{directory_name}'.")
                    self.populate_listbox()  # Refresh list after creation
            except OSError as e:
                    messagebox.showerror("Error", f"Failed to create file:
{e}")
        # else:
        #     messagebox.showwarning("Warning", "Please select a directory
to create a file inside.")

    def delete_file(self):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            file_name = self.file_listbox.get(selected_index)
            confirm = messagebox.askyesno("Confirm Deletion", f"Are you
sure you want to delete '{file_name}'?")
            if confirm:
                try:
                    os.remove(file_name)
                    messagebox.showinfo("Success", f"File '{file_name}'
deleted successfully.")
                    self.populate_listbox()  # Refresh list after deletion
                except OSError as e:
                    messagebox.showerror("Error", f"Failed to delete file:
{e}")
        else:
            messagebox.showwarning("Warning", "Please select a file to
delete.")

    def edit_file(self):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            file_name = self.file_listbox.get(selected_index)
            try:
                os.system(f'notepad "{file_name}"')  # Open file in
Notepad for editing
            except OSError as e:
                messagebox.showerror("Error", f"Failed to edit file: {e}")
        else:
```

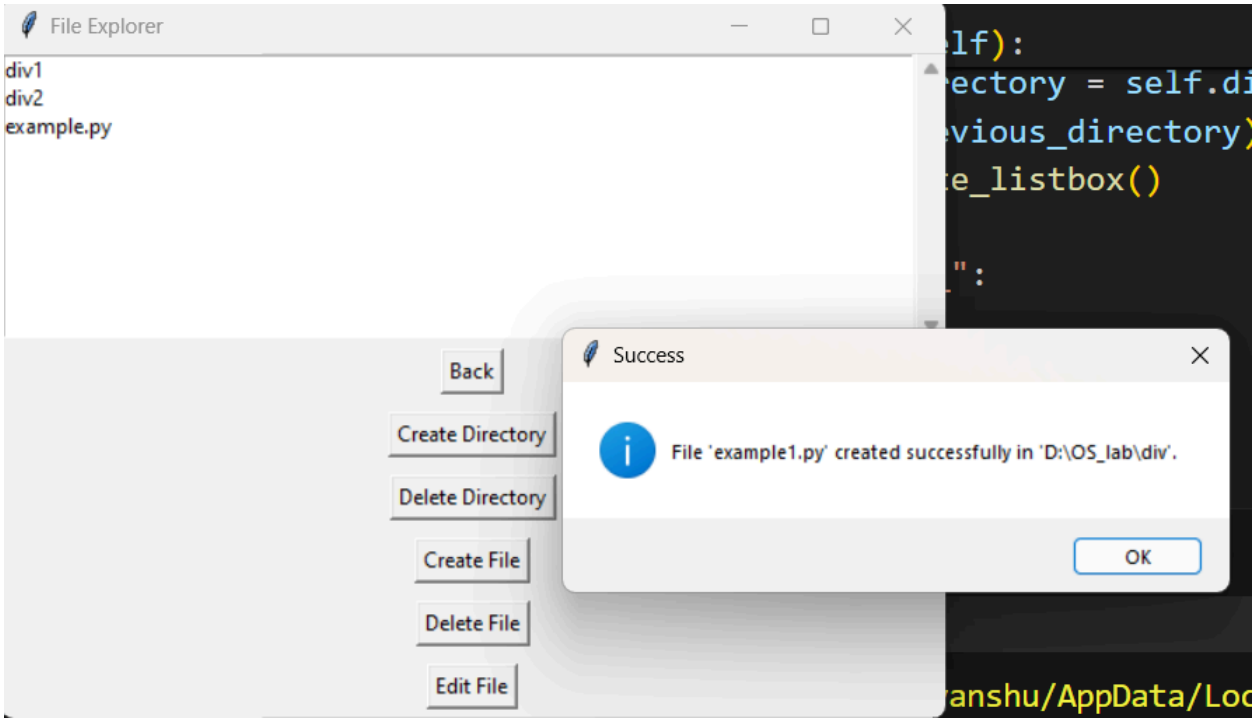
```
            messagebox.showwarning("Warning", "Please select a file to
edit.")

    def open_directory(self, event):
        selected_index = self.file_listbox.curselection()
        if selected_index:
            directory_name = self.file_listbox.get(selected_index)
            self.directory_stack.append(os.getcwd())  # Push current
directory to stack
            os.chdir(directory_name)
            self.populate_listbox()

    def navigate_back(self):
        if self.directory_stack:
            previous_directory = self.directory_stack.pop()  # Pop
previous directory from stack
            os.chdir(previous_directory)
            self.populate_listbox()

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

DIVYANSHU MAKHARIA
221070036

OUTPUT:







CONCLUSION:

Thus,we have built a GUI using Tkinter in python to implement file systems.