FILE SHARING APP USING PYTHON

PROJECT PRESENTATION

PROJECT INTRODUCTION

In the ever-evolving landscape of digital collaboration, File sharing app emerges as a versatile and user-friendly file-sharing application built on the robust foundation of Python. File sharing app is designed to streamline the process of sharing files securely and efficiently, catering to the diverse needs of users in various collaborative environments

server.py - C:/Users/Admin/AppData/Local/Programs/Python/Python312/server.py (3.12.1)

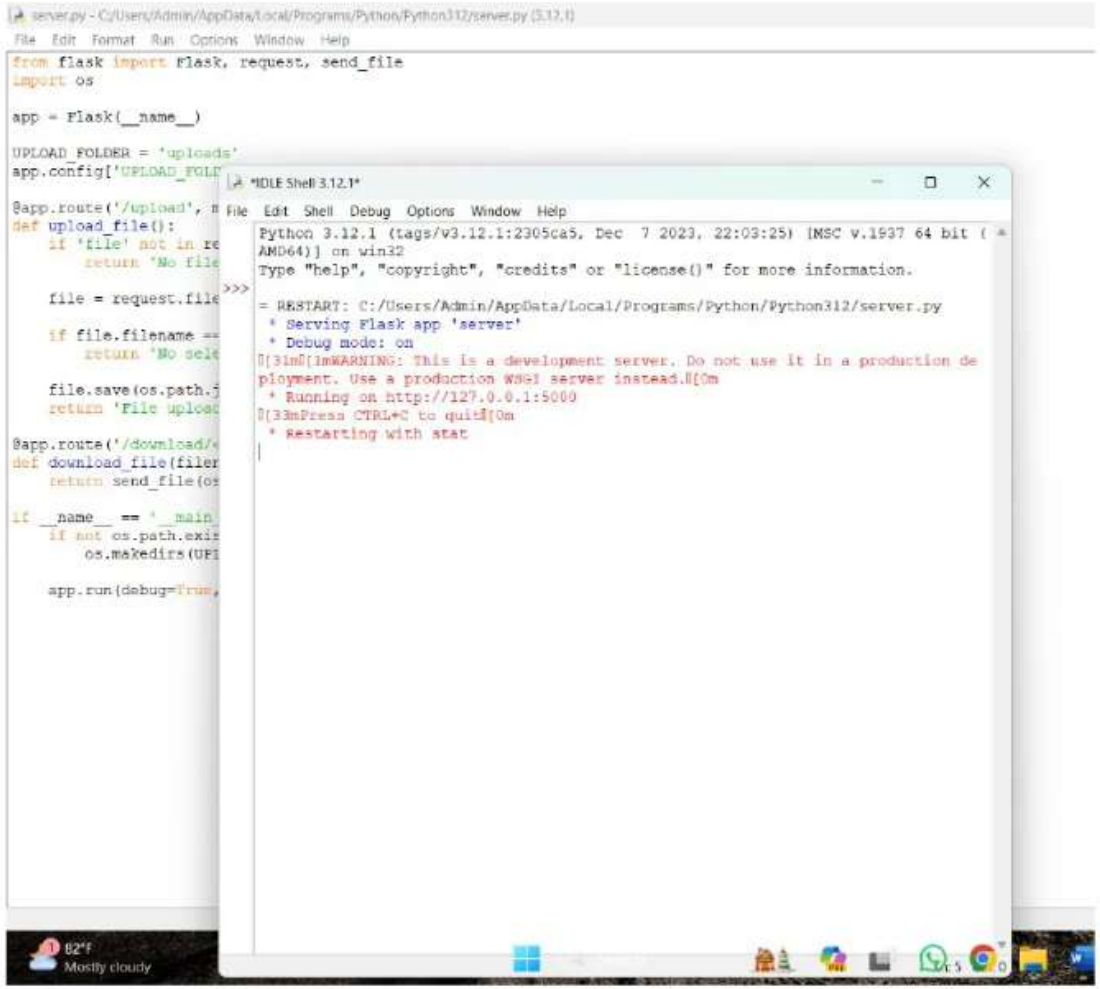
File Edit Format Run Options Window Help

```python
from flask import Flask, request, send_file
import os

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part'

    file = request.files['file']

    if file.filename == '':
        return 'No selected file'

    file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
    return 'File uploaded successfully'

@app.route('/download/<filename>', methods=['GET'])
def download_file(filename):
    return send_file(os.path.join(app.config['UPLOAD_FOLDER'], filename), as_attachment=True)

if __name__ == '__main__':
    if not os.path.exists(UPLOAD_FOLDER):
        os.makedirs(UPLOAD_FOLDER)

    app.run(debug=True, threaded=True)
```

82°F
Mostly cloudy

Search

it is the server code



after running a server code
we will get output like this

client.py - C:/Users/Admin/AppData/Local/Programs/Python/Python312/client.py (3.12.1)

File Edit Format Run Options Window Help

```python
import tkinter as tk
from tkinter import filedialog
import requests

class FileShareApp:
    def __init__(self, root):
        self.root = root
        self.root.title("File Sharing App")
        self.root.geometry("300x150")

        self.label = tk.Label(root, text="Select a file to upload:")
        self.label.pack(pady=10)

        self.upload_button = tk.Button(root, text="Upload File", command=self.upload_file)
        self.upload_button.pack(pady=10)

        self.download_button = tk.Button(root, text="Download File", command=self.download_file)
        self.download_button.pack(pady=10)

    def upload_file(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            files = {'file': open(file_path, 'rb')}
            response = requests.post('http://localhost:5000/upload', files=files)
            print(response.text)

    def download_file(self):
        filename = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("All Files", "*.*")])
        if filename:
            filename = filename.split("/")[-1]  # Extract only the filename
            response = requests.get(f'http://localhost:5000/download/{filename}')
            with open(filename, 'wb') as file:
                file.write(response.content)
            print(f'File downloaded as {filename}')

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

This is the client code

client.py - C:/Users/Admin/AppData/Local/Programs/Python/Python312/client.py (3.12.1)

File Edit Format Run Options Window Help

```
import tkinter as tk
from tkinter import filedialog
import requests

class FileShareApp:
    def __init__(self, root):
        self.root = roo
        self.root.title
        self.root.geome

        self.label = tk
        self.label.pack

        self.upload_but
        self.upload_but

        self.download_b
        self.download_b

    def upload_file(sel
        file_path = fil
        if file_path:
            files = {'i
            response =
            print(respo

    def download_file(s
        filename = file
        if filename:
            filename =
            response =
            with open(f
                file.wr
            print(f'fil

if __name__ == '__main_
    root = tk.Tk()
    app = FileShareApp(
    root.mainloop()
```

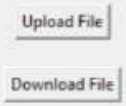
IDLE Shell 3.12.1

File Edit Shell Debug Options Window Help

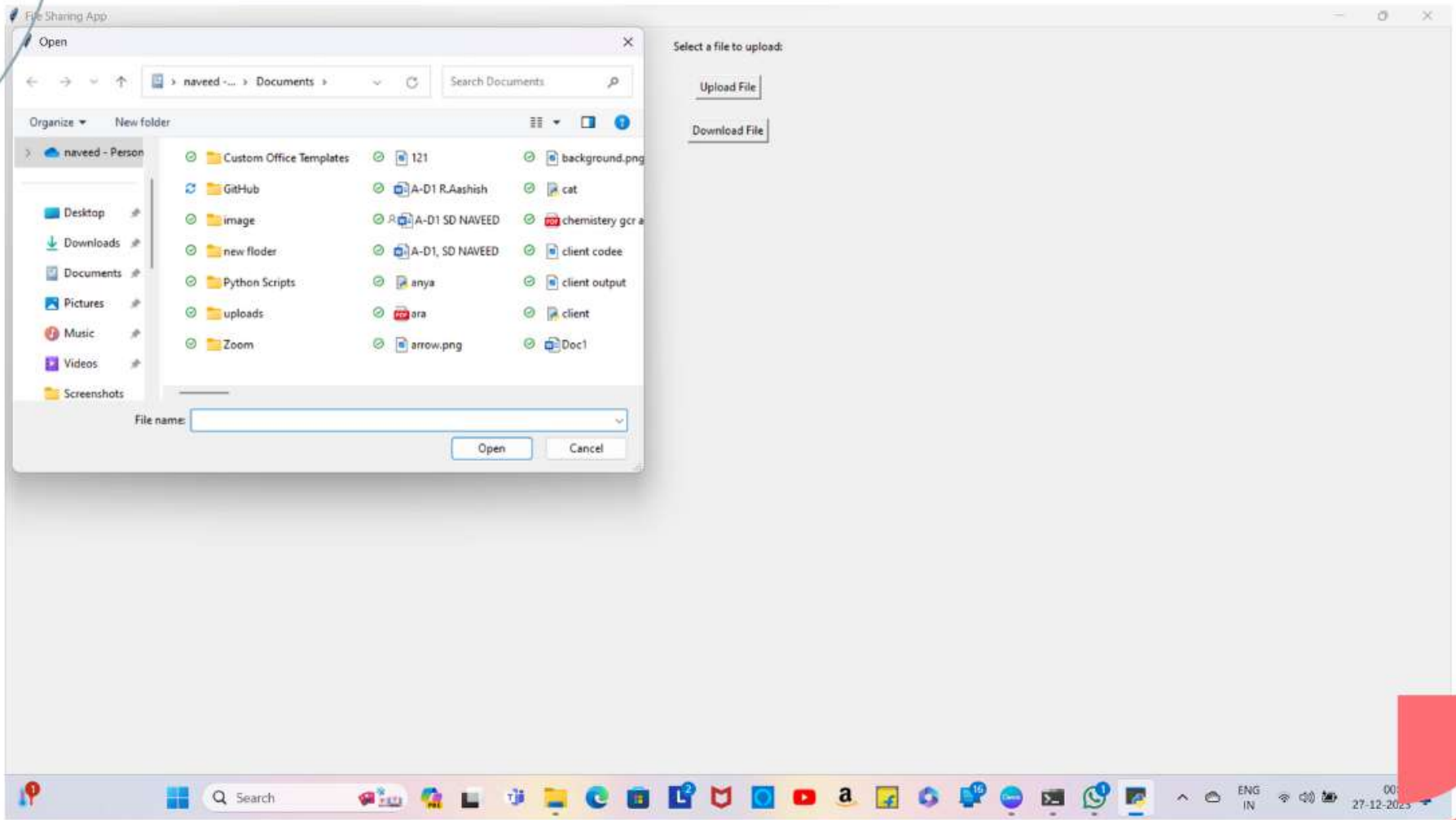
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/server.py
 * Serving Flask app 'server'
 * Debug mode: on
[31m[1mWARNING: This is a development server. Do not use it in a production de
ployment. Use a production WSGI server instead.[0m
 * Running on http://127.0.0.1:5000
[33mPress CTRL+C to quit[0m
 * Restarting with stat

=== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python312/client.py ==
```

82°F
Mostly cloudy

for client code we will get the output like this



After that we will get a notepad in the name of file sharing app
in that one we will have two buttons Upload file&
Download file



After selecting upload file we can see the all files and we can upload them.

modules used for server code

1. **Flask (flask):**
- *Explanation:* Flask is a web framework for Python. In this code, it is used to create a simple web server that handles file uploads and downloads.

2. **request from Flask:**
- *Explanation:* The request object from Flask is used to access information about the incoming HTTP request, such as form data in the case of file uploads.

3. **send_file from Flask:**
- *Explanation:* The send_file function is used to send a file in the HTTP response. In this code, it's used to send the uploaded file as a downloadable attachment.

4. **os:**
- *Explanation:* The os module provides a way to interact with the operating system. It is used here to manage the upload folder and create directories if they don't exist.

modules used for Client code

1. **tkinter (tk):**
- *Explanation:* Tkinter is the standard GUI (Graphical User Interface) toolkit that comes with Python. It is used to create the graphical interface for the file-sharing client.

2. **filedialog from Tkinter:**
- *Explanation:* The filedialog module provides dialogs for opening and saving files. In this code, it is used to prompt the user to select files for upload and choose a location for saving downloaded files.

3. **requests:**
- *Explanation:* The requests module allows sending HTTP requests. In this code, it is used to send POST requests for uploading files and GET requests for downloading files from the Flask server.

These modules play key roles in the functionality of the file-sharing application, with Flask handling the server-side logic and Tkinter providing the client-side GUI. The other modules support these primary functionalities, enabling file upload, download, and interaction between the server and client.

THANK YOU