Schritt-für-Schritt-Anleitung: Server einrichten

1. Vorbereitung

1. **Python installieren**:
   - Lade Python von der offiziellen Webseite herunter: https://www.python.org/.
   - Installiere Python und achte darauf, die Option **"Add Python to PATH"** während der Installation zu aktivieren.
   - Überprüfe die Installation, indem du in der Kommandozeile folgendes eingibst:

     ```
     python --version
     ```

2. **Virtuelle Umgebung erstellen**:
   - Erstelle ein Projektverzeichnis, z. B. `bilder-collage-server`.
   - Wechsle in dieses Verzeichnis:

     ```
     cd bilder-collage-server
     ```

   - Erstelle eine virtuelle Umgebung:

     ```
     python -m venv venv
     ```

   - Aktiviere die virtuelle Umgebung:
     - Windows:

       ```
       venv\Scripts\activate
       ```

     - macOS/Linux:

       ```
       source venv/bin/activate
       ```

3. **Erforderliche Python-Bibliotheken installieren**:
   - Installiere die benötigten Bibliotheken:

```
pip install Flask Flask-SocketIO Pillow
```

2. Server-Code schreiben

1. **Server-Datei erstellen**:
   - Erstelle im Projektverzeichnis eine Datei namens `server.py`.
   - Kopiere folgenden Code in die Datei:

```
from flask import Flask, request, jsonify, send_from_directory
from flask_socketio import SocketIO, emit
from PIL import Image
import os
from datetime import datetime

app = Flask(__name__)
socketio = SocketIO(app)

# Verzeichnis zum Speichern der Bilder
UPLOAD_FOLDER = 'uploaded_images'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/')
def index():
    return "Server läuft. Web-Frontend ist bereit."

@app.route('/upload', methods=['POST'])
def upload_image():
    username = request.form.get('username')
    file = request.files['image']
    if not username or not file:
        return jsonify({'error': 'Benutzername oder Datei fehlt'}), 400

    # Dateiname mit Zeitstempel
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    filename = f"{username}_{timestamp}.jpg"
    filepath = os.path.join(UPLOAD_FOLDER, filename)

    # Bild speichern
    image = Image.open(file)
    image.save(filepath)

    # Bild an alle Clients senden
    socketio.emit('new_image', {'username': username, 'filepath':
filename})

    return jsonify({'message': 'Bild erfolgreich hochgeladen'}), 200

@app.route('/images/<filename>')
```

```
def get_image(filename):
    return send_from_directory(UPLOAD_FOLDER, filename)

if __name__ == '__main__':
    socketio.run(app, debug=True)
```

2. **Verzeichnisstruktur erstellen**:
    - Stelle sicher, dass das Verzeichnis `uploaded_images` vorhanden ist. Dieses wird automatisch erstellt, wenn der Code ausgeführt wird.

3. Server starten

1. **Server ausführen**:
    - Aktiviere die virtuelle Umgebung (falls noch nicht aktiviert) und starte den Server:

    ```
    python server.py
    ```

    - Der Server sollte nun laufen. Standardmäßig wird er unter `http://127.0.0.1:5000` erreichbar sein.

2. **Teste die Verbindung**:
    - Öffne einen Browser und rufe die Adresse `http://127.0.0.1:5000` auf. Du solltest eine einfache Bestätigung sehen, dass der Server läuft.

4. Kommunikation einrichten

1. **SocketIO-Events überwachen**:
    - Der Server ist so eingerichtet, dass er neue Bilder empfängt und an alle verbundenen Clients sendet.
    - Teste dies später mit dem Frontend.

5. Debugging

- **Fehlerbehebung**:
    - Falls es zu Fehlern kommt, überprüfe die Konsolenausgabe des Servers.
    - Stelle sicher, dass die Python-Bibliotheken korrekt installiert wurden und der richtige Pfad zum Bildverzeichnis genutzt wird.