**Project Report: Image Enhancement Flask Web Application**

**1. Project Overview**

The project is a simple **Flask web application** that allows users to upload an image, enhance it using **OpenCV**’s cv2.detailEnhance() method, and then download the enhanced version of the image. The application involves basic **image processing**, file handling, and **web development**.

**2. Objectives**

* **Image Upload**: The application allows users to upload an image via a web interface.
* **Image Enhancement**: The uploaded image is enhanced using **OpenCV** to improve the image details.
* **Image Download**: After enhancement, the user can download the processed image.

**3. Technologies Used**

* **Python**: The core programming language for the backend logic.
* **Flask**: A Python-based web framework used for creating the web application.
* **OpenCV (cv2)**: A powerful computer vision library used for image enhancement.
* **HTML**: For creating the web interface.
* **Jinja2**: Template engine used by Flask to render HTML templates dynamically.

**4. Project Structure**

The project is structured as follows:

project\_folder/

│

├── app.py # Flask app with image upload and enhancement

├── requirements.txt # Lists the required dependencies for the project

├── uploads/ # Directory for storing uploaded images

├── enhanced/ # Directory for saving enhanced images

├── templates/ # Folder for HTML templates (Flask renders these)

│ └── index.html # The HTML file for uploading images

**5. Detailed Explanation of Code Components**

1. **app.py**:  
   This file contains the main Flask application. The code structure is as follows:
   * **Imports**:
     + Flask: The web framework used to build the app.
     + request: Used to handle HTTP requests (like file uploads).
     + render\_template: Used to render the HTML template for the web interface.
     + send\_file: Used to send the processed image back to the user.
     + os: Used to manage file and directory paths.
     + cv2: OpenCV library used for image processing.
   * **Flask App Setup**:
     + UPLOAD\_FOLDER and ENHANCED\_FOLDER are directories where the app will save uploaded images and enhanced images, respectively.
     + os.makedirs() ensures that these directories are created if they don’t already exist.
   * **Image Enhancement Function**:
     + The function enhance\_image(input\_path, output\_path) is responsible for reading an image using OpenCV’s cv2.imread(), enhancing it with the cv2.detailEnhance() function, and then saving the enhanced image.
   * **Flask Routes**:
     + **/ (Root Route)**: Handles both GET and POST requests.
       - For GET, it renders the **index.html** page where the user can upload an image.
       - For POST, it processes the uploaded image, enhances it, and sends the enhanced image back to the user for download.
2. **opencv\_tools.py**:  
   This file contains the actual logic for enhancing an image using OpenCV.
   * **enhance\_image(filepath, output\_dir)**:
     + This function loads an image from the file path, applies the detail enhancement, and saves the enhanced version to the specified output directory.
3. **genai\_tools.py**:
   * This file contains a placeholder function for future **AI-based image enhancement**. Currently, the function does nothing but can be updated later to implement AI-based enhancements.
4. **requirements.txt**:
   * Lists the necessary Python dependencies:
     + **Flask**: Web framework.
     + **gunicorn**: WSGI HTTP server to run Flask in production.
     + **opencv-python**: For handling image processing and enhancement.
5. **index.html** (inside the templates/ folder):
   * This is the HTML page where the user can upload an image. It contains:
     + A simple form that allows the user to choose a file and submit it.
     + The form sends a POST request to the Flask app to process the uploaded image.

**6. How It Works**

1. **User uploads an image**:
   * The user navigates to the web interface and selects an image file to upload.
   * The POST request is sent to the server, and the uploaded image is saved in the uploads/ folder.
2. **Image Enhancement**:
   * Once the image is uploaded, it is processed by the enhance\_image function in **app.py**.
   * OpenCV’s cv2.detailEnhance() function is used to improve the details of the image (based on parameters like sigma\_s and sigma\_r).
3. **Sending back the enhanced image**:
   * After enhancement, the processed image is saved in the enhanced/ folder.
   * The enhanced image is then sent back to the user as a downloadable file using the send\_file() function in Flask.

**7. Steps to Run the Project**

1. **Set Up Python Environment**:
   * Make sure Python is installed on your machine.
   * It is recommended to use a virtual environment to manage dependencies. Run the following commands:

bash

CopyEdit

python -m venv venv # Create a virtual environment

.\venv\Scripts\activate # Activate the virtual environment (Windows)

1. **Install Required Dependencies**:
   * Install the required dependencies by running:

bash

CopyEdit

pip install -r requirements.txt

pip install opencv-python # If OpenCV is not already installed

1. **Run the Flask App**:
   * Run the Flask application:

bash

CopyEdit

python app.py

* + Open your browser and navigate to:

cpp

CopyEdit

http://127.0.0.1:5000/

* + You should see the upload form. Upload an image, and it will be enhanced and sent back for download.

**8. Future Improvements**

1. **AI-based Image Enhancement**:
   * Integrate **AI-driven** image enhancement in genai\_tools.py. This could involve using deep learning models or pre-trained models for image upscaling, color correction, or noise reduction.
2. **Error Handling**:
   * Improve error handling for file uploads, invalid file types, and other edge cases.
3. **UI Improvements**:
   * Enhance the user interface to display more information about the uploaded image (e.g., original vs. enhanced image preview).
4. **Support for Multiple File Types**:
   * Extend the app to support multiple image formats (JPEG, PNG, BMP, etc.) and provide feedback if an unsupported file type is uploaded.
5. **Cloud Integration**:
   * Implement cloud storage to store images and enhanced files, allowing users to download files from a URL instead of serving them from the local server.

**Conclusion**

This project demonstrates a basic **image enhancement web app** built with **Flask** and **OpenCV**. It allows users to upload images, enhances them with OpenCV, and returns the enhanced images for download. The app can be further expanded with **AI-based** enhancements and more features for a robust, production-ready application.

**Output:**

****

****