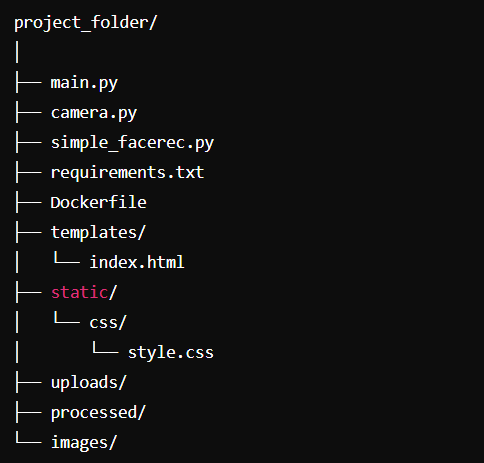
**Directory Structure**

****

* **main.py**: Flask application handling video input selection, streaming, and processing.
* **camera.py**: Defines VideoCamera class for capturing and processing video frames.
* **simple\_facerec.py**: Implements facial recognition using the face\_recognition and MTCNN libraries.
* **templates/**: Directory for Flask templates, containing index.html.
* **static/**: Directory for static files, including style.css.
* **uploads/**: Directory to store uploaded video files.
* **processed/**: Directory to store processed video files.

**Packages Used and Their Purposes**

1. **Flask**
   * **Purpose**: Web framework for building the user interface and handling HTTP requests.
   * **Usage**: Flask, render\_template, Response, request, redirect, url\_for, send\_file.
2. **OpenCV (cv2)**
   * **Purpose**: Used for capturing video frames, processing images, and video encoding.
   * **Usage**: cv2.VideoCapture, cv2.VideoWriter, cv2.imencode, frame processing.
3. **face\_recognition**
   * **Purpose**: Provides facial recognition capabilities, including face detection and encoding.
   * **Usage**: face\_recognition.face\_encodings, face\_recognition.compare\_faces, face\_recognition.face\_distance.
4. **MTCNN**
   * **Purpose**: Pre-trained model for detecting faces and their keypoints (eyes, nose, mouth).
   * **Usage**: MTCNN for face detection and keypoints extraction.

**Prerequisites for Windows**

1. **CMake**:
   * **Install CMake**: Download and install CMake from the official website.
   * Ensure CMake is added to the system path during installation.
2. **dlib**:
   * **Download the appropriate dlib .whl file**: You can find the .whl file for your Python version at <https://pypi.org/simple/dlib/>.
   * For example, if you are using Python 3.8, download dlib-19.22.1-cp38-cp38-win\_amd64.whl.

**Running the Code**

**Step-by-Step Procedure**

1. **Clone the Repository**:

cd project\_folder

1. **Install Dependencies**: Ensure you have Python and pip installed.
   * **Install CMake**:

pip install cmake

* + **Install dlib**: Navigate to the directory where you downloaded the .whl file and install it:

pip install dlib-19.22.1-cp38-cp38-win\_amd64.whl

* + **Install other required packages**:

pip install flask opencv-python numpy face-recognition mtcnn

1. **Start the Flask Application**: Run the Flask application (main.py):

python main.py

The server will start running at http://localhost:5000.

**Steps to Interact with the Application**

1. **Access the Web Interface**: Open a web browser and go to http://localhost:5000.
2. **Select Video Source**:
   * Click on "Use Webcam" to stream from your webcam.
   * Enter a direct video link and click "Submit Link".
   * Upload a video file by choosing a file and clicking "Submit File".
3. **View Video Feeds**:
   * The "Video Feed" shows the live video stream.
   * The "Processed Video Feed" displays the stream with recognized faces and keypoints.
4. **Download Processed Video**:
   * After selecting and processing a video file, you can download the processed video by clicking the download link.

**Viewing the Output**

* **Live Video Feed**: Displays real-time video from the selected source.
* **Processed Video Feed**: Shows the live video with recognized faces marked with names and facial keypoints.
* **Downloaded Video**: After processing a video file, download the processed video from the browser.

**Dockerization**

**Dockerfile**

**Step-by-Step Procedure to Build and Run the Docker Container**

1. **Create a requirements.txt File**: Ensure you have a requirements.txt file listing all the Python dependencies:

Flask

opencv-python

numpy

tensorflow

face-recognition

mtcnn

1. **Build the Docker Image**: Navigate to the project directory containing the Dockerfile and run:

docker build -t face\_recognition\_app .

1. **Run the Docker Container**: Once the image is built, run the container:

docker run -p 5000:5000 face\_recognition\_app

1. **Access the Application**: Open a web browser and go to http://localhost:5000 to access the application.

**Viewing the Output**

* **Live Video Feed**: Displays real-time video from the selected source.
* **Processed Video Feed**: Shows the live video with recognized faces marked with names and facial keypoints.
* **Downloaded Video**: After processing a video file, download the processed video from the browser.

### System Dependencies and Development Platform

#### Development Platform

* **Platform**: Kaggle Notebooks
  + Kaggle Notebooks provide a cloud-based environment for data science and machine learning tasks with pre-configured settings, enabling easy access to GPU resources and various datasets.
  + **GPU Acceleration**: Ensure GPU acceleration is enabled in Kaggle Notebooks for optimal performance, especially for tasks involving video processing and facial recognition.

#### System Used for Development

* **System**: Lenovo Ideapad Gaming 3
  + **Operating System**: Windows 11
  + **Storage**: 512 GB SSD

#### System Hardware Dependencies

1. **Processor (CPU)**:
   * Intel i5 or AMD Ryzen 5 or better.
2. **Memory (RAM)**:
   * Minimum 8 GB RAM (16 GB recommended for better performance).
3. **Storage**:
   * 512 GB SSD (minimum).
4. **Graphics Card (GPU)**:
   * NVIDIA GeForce GTX 1650 or better (recommended for video processing tasks).
5. **Operating System**:
   * Windows 11 (compatible with Windows 10, macOS, and Linux with necessary adjustments).