# FLOWER FINDER

This project is designed to be a "Flowers Recognition" program that analyzes images and videos to identify flowers. Its feature is the ability to then suggest a flower to the user based on a specific "desired effect" they are looking for, by cross-referencing the flower's known natural active principles. The system takes **videos** and **photos** as input, along with the user's **desired effect**, and outputs a photo and description of a matching flower found in the inputs.

**System Architecture and Workflow**

The project's workflow is structured into three main, sequential tasks, forming a Directed Acyclic Graph (DAG) structure. The process begins with user-provided data and concludes with a specific output tailored to the user's request.

**Inputs and Outputs:**

- **Primary Inputs:** The system requires three types of data from the user:
    - Videos
    - Photographs
    - A "desired effect" specified by the user
- **Core Data:** A "Flower Benefits Dataset" is a given, crucial component that links flowers to their properties.

  **Structure of the Dataset:** Simple JSON file or a CSV. This file should map a flower's name to its benefits, active principles, and a short description.
- **Final Output:** The system will provide the user with the name of a flower that matches the desired effect, along with its description and a corresponding photo.

**Granular Task Breakdown**

The overall process is divided into the following computational tasks:

1. **Video Processing (python coded):**

   The first task is **video processing**, which extracts important images (keyframes) from the input videos. This is crucial for efficiency, as it avoids analyzing every single frame of a video.

    - **Input:** Video files provided by the user.
    - **Process:** This initial stage focuses on keyframe extraction from the input videos. The extraction can be triggered based on scene changes or at set time intervals. We are using a hybrid approach:

      **Scene-Change Based:** This method is more effective. It involves comparing consecutive frames and saving a new keyframe only when the difference between them is significant. This ensures capture new scenes without creating duplicate images.

      **Time-Based:** This prevent too few keyframes axtracted in static or one scene videos

- **Output:** A set of keyframes (images) that represent the significant content of the videos.

2. **Flower Recognition (python coded):**

   It takes the keyframes from the previous step and any extra photos the user uploaded, then identifies the flowers in them. The output is a **list of recognized flowers**.

   - **Input:** The keyframes generated from the video processing task, along with any additional photos uploaded by the user.
   - **Process:** The system processes these images to identify and recognize the flowers present. The process should handle multiple inputs so it should be able to process a directory of images (the keyframes and uploaded photos) in a loop, running the recognition on each one and compiling a unique list of all flowers found.
   - **Output:** A list containing the names of the recognized flowers.

3. **Dataset Matching (python coded):**

   The final task connects the recognized flowers with the user's needs. It takes the list of recognized flowers and the **desired effect** text from the user, matches them against a **flower benefits dataset**, and returns a final description of the most suitable flower.

   - **Input:** The list of recognized flowers and the desired effect as specified by the user.
   - **Process:** This task involves performing a textual match. It compares the recognized flowers and the user's desired effect against the information stored in the "Flower Benefits Dataset" to find a correlation.
   - **Output:** The description of the flower that is suitable for the requested effect. This output is then presented to the user, fulfilling the project's main goal.