**Software Requirement Specification Document**

**Enchanted Wings: Marvels of Butterfly Species**

**Prepared by:**

#### Project Team (ID: LTVIP2025TMID32435)

Palepu Kavya (24P35A0539)

Attili Nithin Datta(23MH1A05L8)

A Basava Manikanta (23MH1A0573)

Mummidi Vijendra Kumar (24P35A0531)

#### College Affiliation

Department of Computer Science and Engineering at Aditya College of Engineering and Technology

**Introduction:**

This project focuses on creating a robust butterfly image classification model using transfer learning techniques. Leveraging a dataset comprising diverse butterfly species, including 75 classes with a total of 6499 images, the dataset is partitioned into training, validation, and test sets. Transfer learning utilizes pre-trained convolutional neural networks (CNNs) to accelerate model training by extracting relevant features from butterfly images. This method enhances classification accuracy while reducing computational resources and training time, ensuring efficient and effective species identification.

**Problem Scenario:**

**Scenario 1:**

 Biodiversity Monitoring In the context of biodiversity monitoring, a butterfly image classification system based on transfer learning can contribute significantly. Field researchers and conservationists can use this system to quickly identify butterfly species in diverse habitats. By capturing images in the field, the system identifies butterflies in real-time, aiding in species inventory, population studies, and habitat management efforts. This facilitates data-driven conservation strategies and promotes ecosystem health monitoring.

**Scenario 2:**

Ecological Research For ecological research, especially studies on butterfly behavior and distribution patterns, automated image classification systems are invaluable. Researchers can deploy cameras equipped with the classification system to monitor butterfly activities over extended periods. This enables tracking of migratory patterns, habitat preferences, and responses to environmental changes. The system's ability to accurately classify butterflies supports scientific discoveries and informs conservation practices aimed at preserving vulnerable species.

**Scenario 3:**

 Citizen Science and Educational initiatives and citizen science projects benefit from interactive butterfly classification tools. These tools engage enthusiasts and students in butterfly identification and data collection. Users can capture butterfly images using mobile devices, and the classification system instantly provides species identification and relevant educational information. Such tools promote environmental awareness, citizen participation in scientific research, and foster a deeper understanding of butterfly ecology and conservation.

By applying transfer learning to butterfly image classification, this project not only advances scientific research and conservation efforts but also enhances public engagement and educational outreach in the field of biodiversity conservation.

**Architecture:**

A diagram of a software algorithm

AI-generated content may be incorrect.

**Project Flow**

### **1. Data Collection**

* **Images** are gathered as raw data for training and testing.
* These images are then processed through the next stage.

### **2. Image Preprocessing**

* Techniques like resizing, normalization, augmentation, grayscale conversion, etc., are applied.
* The preprocessed data is split into:
  + **Train Data** – used to train the model.
  + **Test Data** – used to evaluate model performance.

### **3. Model Development**

* A **DL Algorithm** (e.g., CNN, ResNet, etc.) is used.
* **Train Data** is fed into this model to help it learn features and patterns.

### **4. Model Evaluation**

* After training, the model is evaluated using **Test Data**.
* Performance metrics like accuracy, precision, recall, etc., are calculated.

### **5. Prediction**

* Once evaluated, the model can be used for **prediction/inference**.
* It takes new **Inputs** from the user and returns results.

### **6. User Interface (UI)**

* Users interact with the system through a **UI**.
* They can upload images or give input and get predictions from the model.

**Prior Knowledge**

You must have prior knowledge of the following topics to complete this project.

* DL Concepts
  + Neural Networks:: [https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/](https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/%20)
* Deep Learning Frameworks:: [https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow](about:blank)
* Transfer Learning: [https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a](https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a%20)
* VGG16: https://www.geeksforgeeks.org/vgg-16-cnn-model/
* Convolutional Neural Networks (CNNs): [https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning](https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/%20s:/www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)
* Overfitting and Regularization: [https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/](https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/%20)
* Optimizers: [https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/](https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/%20ttps:/www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/)
* Flask Basics: <https://www.youtube.com/watch?v=lj4I_CvBnt0>

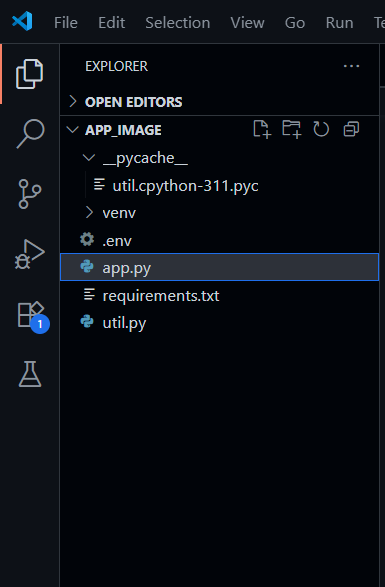
**Project Objectives**

By the end of this project, you will:

* Know fundamental concepts and techniques used for Deep Learning.
* Gain a broad understanding of data.
* I have knowledge of pre-processing data/transformation techniques on outliers and some visualization concepts.

**Project Structure**

Create the Project folder which contains files as shown below:



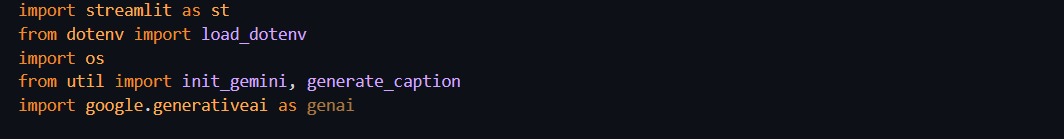
**Requirements Specification**

The required libraries can be installed directly into Colab for use.

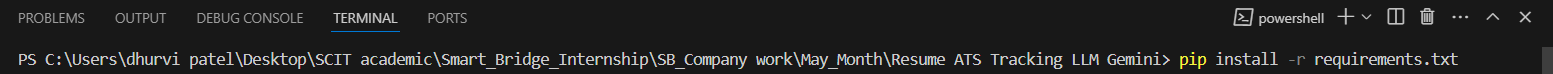
**Create a requirements.txt file to list the required libraries**

* streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
* streamlit\_extras: Additional utilities and enhancements for Streamlit applications.
* google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
* python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
* PyPDF2: It is a Python library for extracting text and manipulating PDF documents.

Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats



**Install the required libraries:**



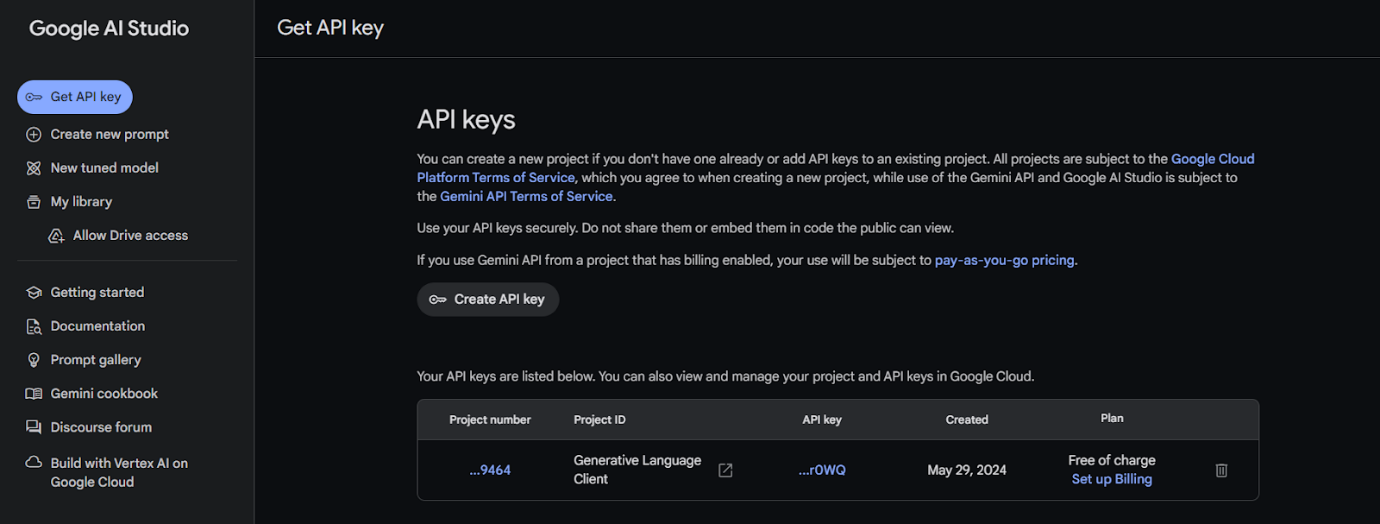
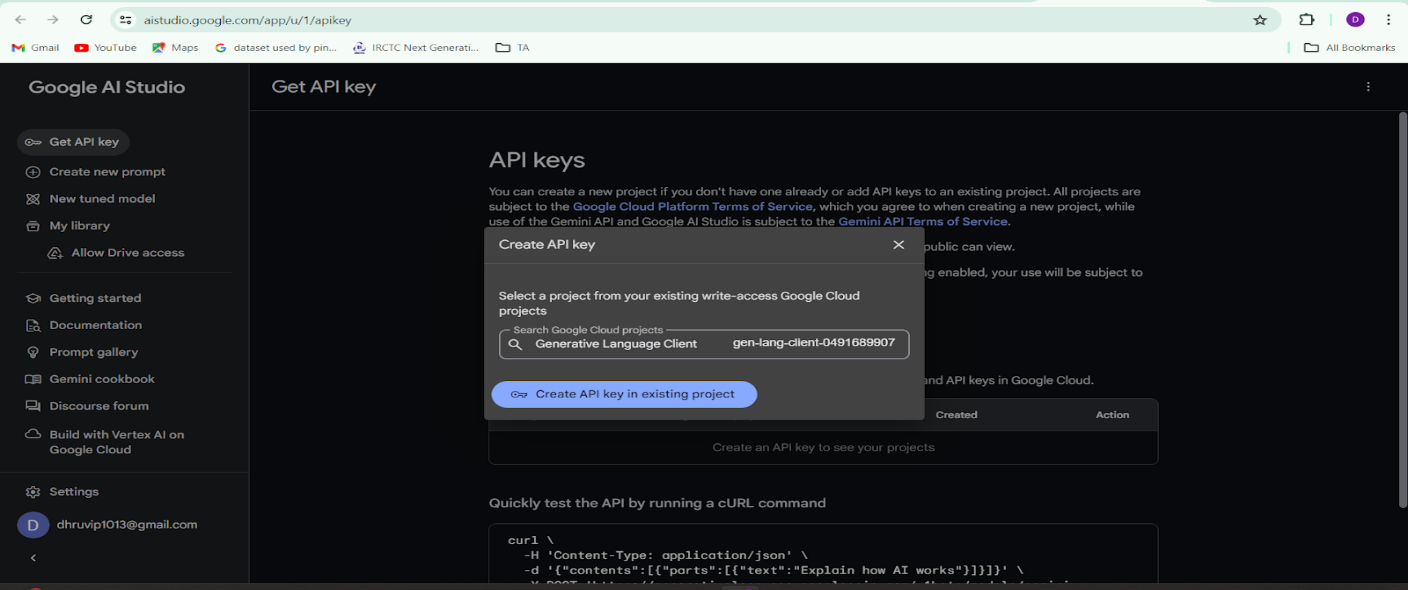
* Open the terminal.
* Run the command: pip install -r requirements.txt
* This command installs all the libraries listed in the requirements.txt file.
* **Initialization of Google API Key**
* The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing APIrequests, ensuring that only authorized users can access and utilize Google's services.

### Generate Google API Key

Click the provided link to access the following webpage.

Link: <https://ai.google.dev/gemini-api/docs/api-key>

After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below.

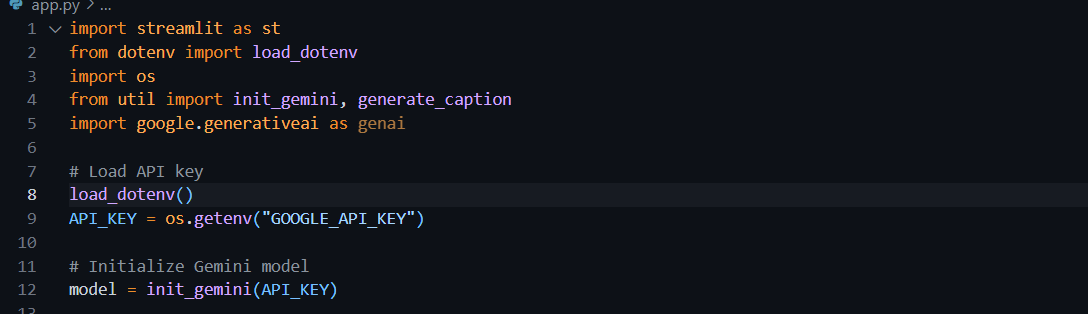
Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'. Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model. **Initialize Google API Key**

* Create a .env file and define a variable named GOOGLE\_API\_KEY.
* Assign the copied Google API key to this variable.
* Paste the API key obtained from the previous steps here.

### Interfacing with Pre-trained Model

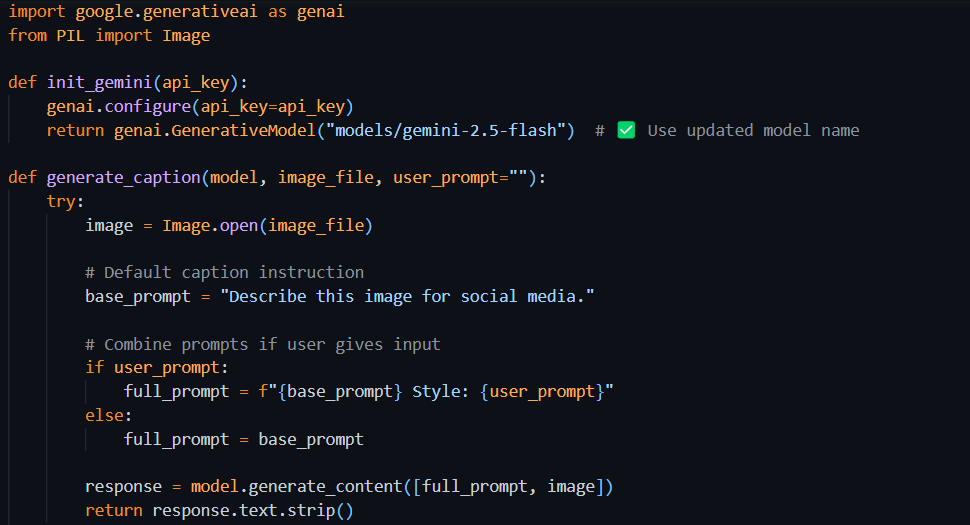
To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

**Load the Gemini Pro API**

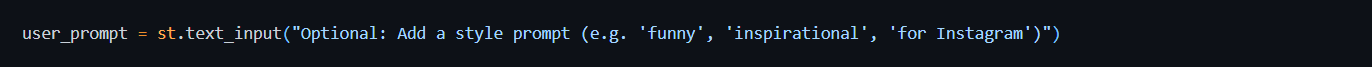
This code snippet is for initializing a health management application using Streamlit, an open-source app framework, and Google Generative AI services. The script starts by loading environment variables from a .env file using the load\_dotenv() function from the dotenv package. It then imports necessary libraries: streamlit for creating the web app interface, os for accessing environment variables, google.generativeai for utilizing Google's Generative AI capabilities, and PIL.Image for image processing. The genai.configure() function is called to set up the Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services. **Implement a function to get gemini response**

* The function get\_gemini\_response takes an input text as a parameter.
* It calls the generate\_content method of the model object to generate a response.
* The generated response is returned as text.

### Implement a function to read the Image and set the image format for Gemini Pro model Input using util.py

Implement a function to setup image and processes an uploaded image file for a health management application in util.py . It first checks if a file has been uploaded. If a file is present, it reads the file's content into bytes and creates a dictionary containing the file's MIME type and its byte data. This dictionary is then stored in a list named image\_parts, which is returned by the function. If no file is uploaded, the function raises a FileNotFoundError, indicating that an image file is required but not provided. This setup ensures that the uploaded image is correctly formatted and ready for further processing or analysis in the application. 

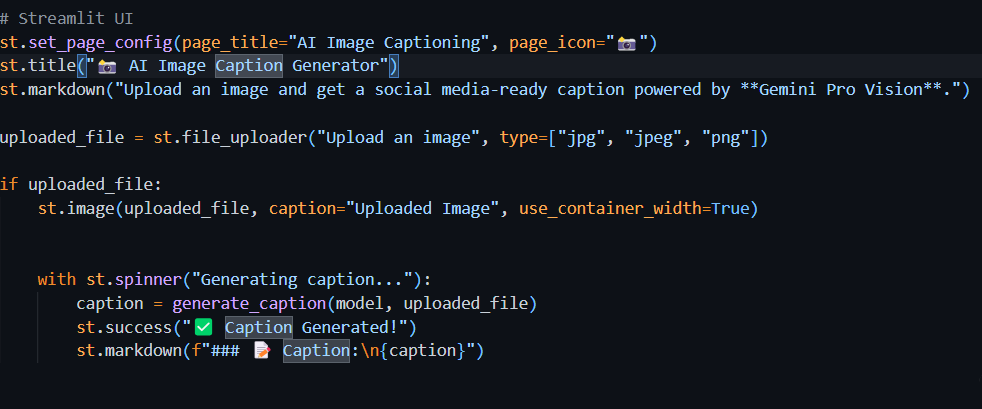
**Write a prompt for gemini model**

The variable input\_prompt is a single-line string designed as a prompt for a captioning AI model. It instructs the model to analyze an image .Additionally, the model is to provide a detailed breakdown of each item with its respective count

**Model Deployment**

We deploy our model using the Streamlit framework, a powerful tool for building and sharing data applications quickly and easily. With Streamlit, we can create interactive web applications that allow users to interact with our models in real-time, providing an intuitive and seamless experience.

**Integrate with Web Framework**



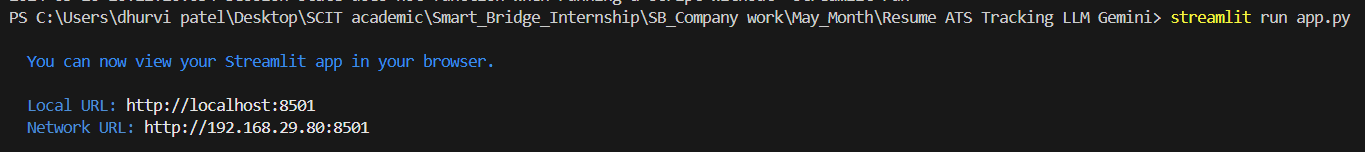
This code initializes a Streamlit application titled "IMAGE TO CAPTION" by setting the page title and creating the app's header. It includes a text input field for users to enter a custom prompt and a file uploader for users to upload an image in JPG, JPEG, or PNG format. If an image is uploaded, it is opened using the PIL library and displayed within the app with a caption. A button labeled "Give me caption in one phrase" is also provided, which users can click to trigger the application's functionality for analyzing the uploaded image to calculate and display the Caption

**Host the Application**

Launching the Application:

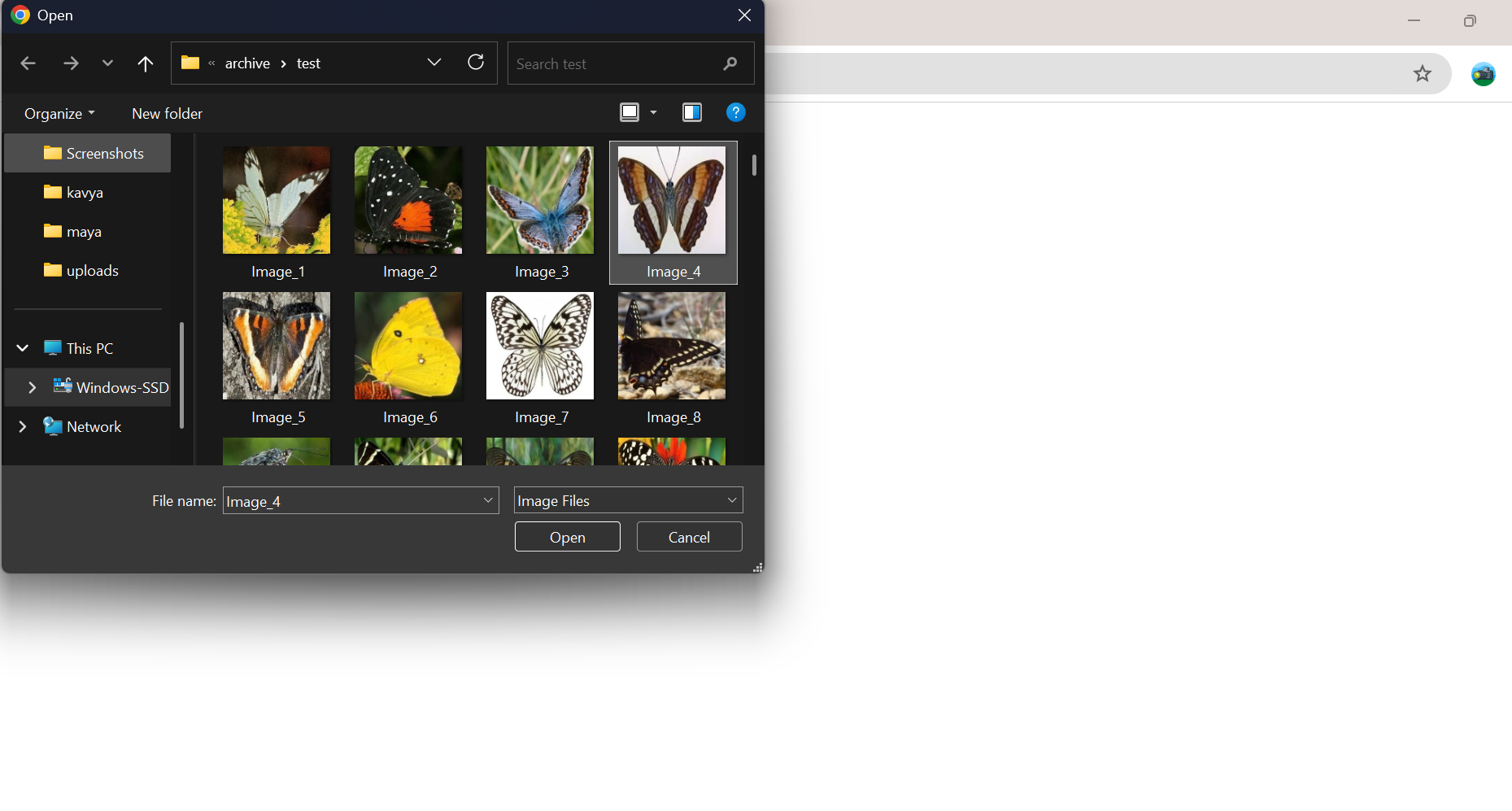
* To host the application,  go to the terminal, type - streamlit run app.py
* Here app.py refers to a python script.

Run the command to get the below results



USER INTERFACE : 

Input :





Output :  