**Pollen grains Classification Project Documentation**

**Introduction**

**Project Title:**

Pollen's Profiling: Automated Classification of PollenGrain

**Team ID:** LTVIP2025TMID40923

**Team Members:**

* Sowjanya Pabbineedi
* Ungarala Jaya Sravanthi
* Singipilli Mahalakshmi
* Sailada Aswini

**Purpose:**

The purpose of this project is to build a deep learning-based image classification model to identify Pollen Grains from flowers. This project addresses real-world challenges in biodiversity conservation and species recognition by providing a tool for automated species detection using AI/ML & deeplearning.

**Goals:**

**Automate:**  the classification of pollen grain images into respective plant species.

**Assist :** botanists, researchers, and agriculture experts in accurate pollen identification.

**Develop:** a scalable and generalizable deep learning model that performs well across diverse datasets of microscopic pollen images

**Key Features:**

* High-accuracy pollen grain classification using CNN-based deep learning models..
* Robust image preprocessing (resizing, normalization, augmentation)
* . User-friendly UI built with smooth ui for easy image upload and real-time prediction.
* Model performance evaluation and confusion matrix visualization. **Architecture**

**Frontend:**

* **Framework Used:** (HTML) & CSS
* **Functionality:**
  + Upload pollen grain images..
  + Optional: View prediction history, model version used, and sample images.
  + Intuitive UI suitable for both technical and non-technical users.

**Backend:**

**Technology Used:** Python with Flask

**Responsibilities:**

 Receives images from frontend, processes them, and sends to trained model.

 Loads a trained CNN model (e.g keras, tensorFlow, custom CNN).

 Returns prediction label and confidence score.

 Handles error messages, input validation, and request logging.

**Database:**

**Type:** MongoDB or SQLite (based on project scale)

**Stored Data:**

 Prediction logs (image name, predicted species, confidence, timestamp).

 User session metadata (if login implemented).

 Model version history for auditability.

Metadata about images and model version used.

**Python Lib**

Python 3.x

Libraries: TensorFlow/Keras, NumPy, Pandas, OpenCV, Matplotlib, Streamlit

GPU or Google Colab (recommended for training) **Installation Steps:**

**STEP 1: Install Anaconda (if not already installed)**

Download the Anaconda distribution from the official website: <https://www.anaconda.com/products/distribution>

Follow the installation instructions based on your operating system.

**STEP 2: Create a Virtual Environment**

Open **Anaconda Prompt** and run the following commands:

conda create -n pollen-grain python=3.9

conda activate pollen-grain

**STEP 3: Install Required Packages**

While inside the virtual environment, install all the necessary dependencies:

pip install tensorflow keras flask matplotlib numpy opencv-python pillow

**STEP 4: Run the Web Application**

1. Navigate to your project directory. Example: cd /d D:\pollen-grain\data\archive
2. Then, start the Flask web app by running: python app.py
3. You should see an output like:

\* Running on http://127.0.0.1:5000/

4. Open your browser and go to that address to use the application.

**How to Use:**

Upon launching the butterfly classifier app:

1. **Welcome Screen** o Displays an animated translation of images & background with a honeybees on flowers
2. **Image Upload** o Click “Upload Image” to proceed to the input screen.
3. **Prediction Step** o Select a pollen grain image and hit “Predict”.
4. **Results Displayed** o Model outputs:

* **Predicted grain**
* **Uploaded image preview**
* **Visually pleasing animated layout**

# Offline Usage

This app is fully functional **offline**, provided:

* You have the following in your local folder:
  + vgg16\_model.h5 (Trained model file)
  + static/ (All backgrounds, styles, and uploaded images)
  + templates/ (HTML layout files)
* All required packages are pre-installed in the **Anaconda environment**.
* No external fonts or image links are fetched from the internet.

# Folder Structure

Project/

└── \_Pollen-grain /

├── app.py ← Flask web app script

├── model.h5 ← Trained classification model

├── cnn.h5 ← Training dataset (class-wise folders)

├── model.keras ← Optional testing images

├── templates

| |-- data/ ← Static assets (CSS, backgrounds, uploads)

│ ├── archive

│ └── All images of dataset ← Stores user-uploaded images

├── templates/ ← HTML templates for web pages

│ ├─-prediticed.html

│ ├── input.html ← Static assets (CSS, backgrounds, uploads)

│

├── Training\_set.csv ← Training dataset metadata

├── Testing\_set.csv ← Test dataset metadata

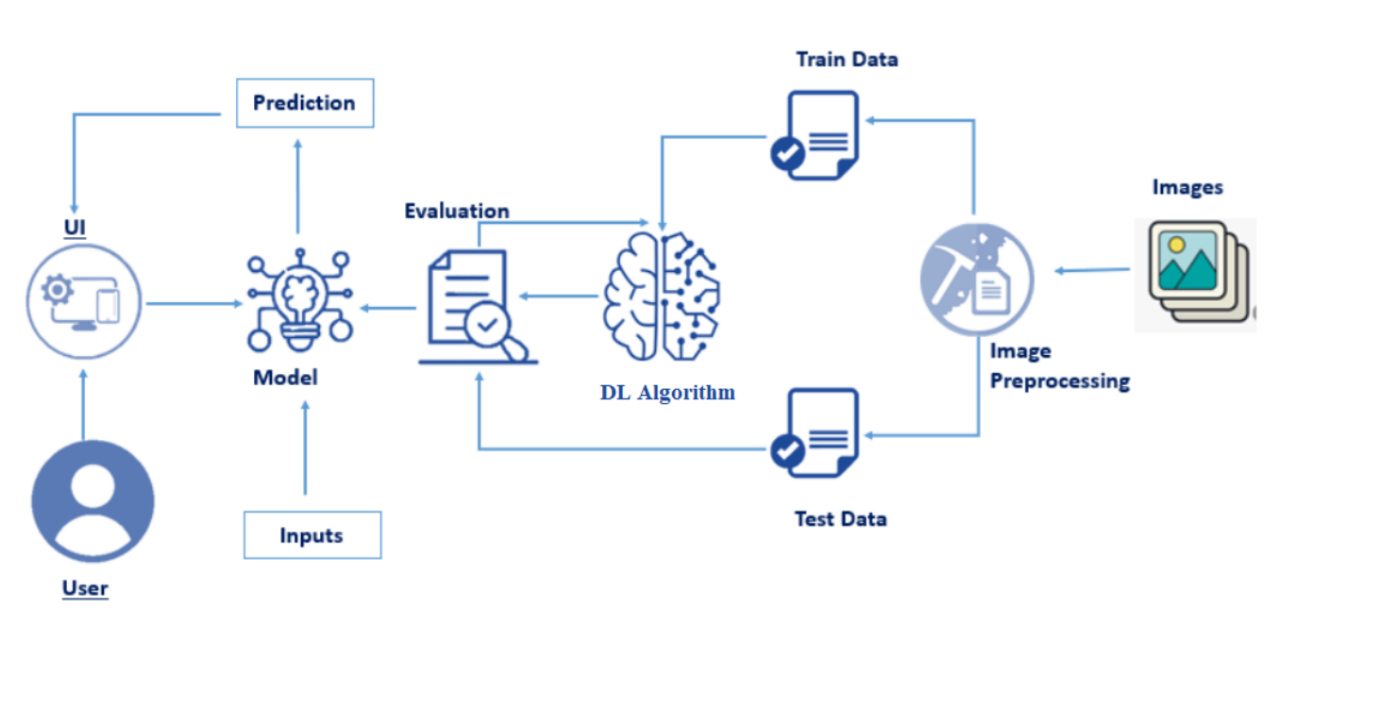
└── README.txt ← Basic setup and usage

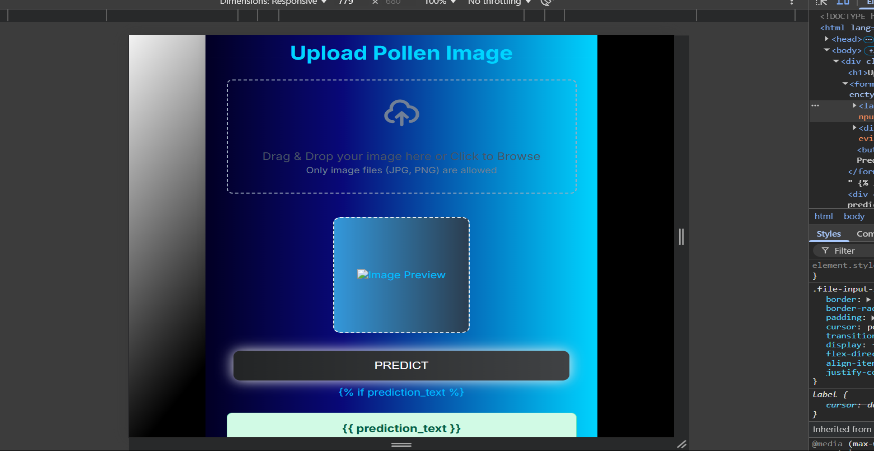
**User Interface**

* Simple image upload panel (via tailwind css or Flask).
* Displays predicted pollengrain with confidence score. 
* Option to upload multiple images.

**Testing**

* Unit testing with PyTest or Unittest for core modules.
* Manual testing of prediction outputs on unseen data.
* Validation accuracy and confusion matrix plotted for model performance insights. **Screenshots or Demo**

****

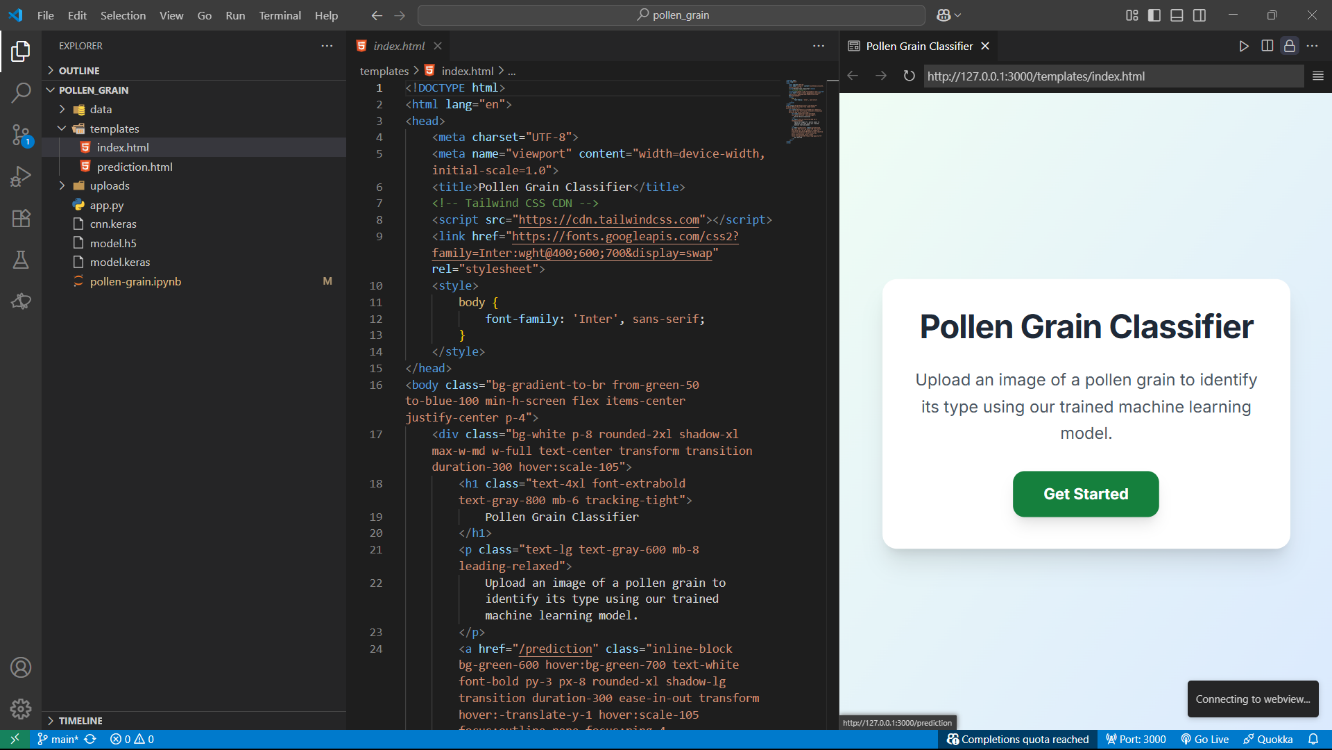
****

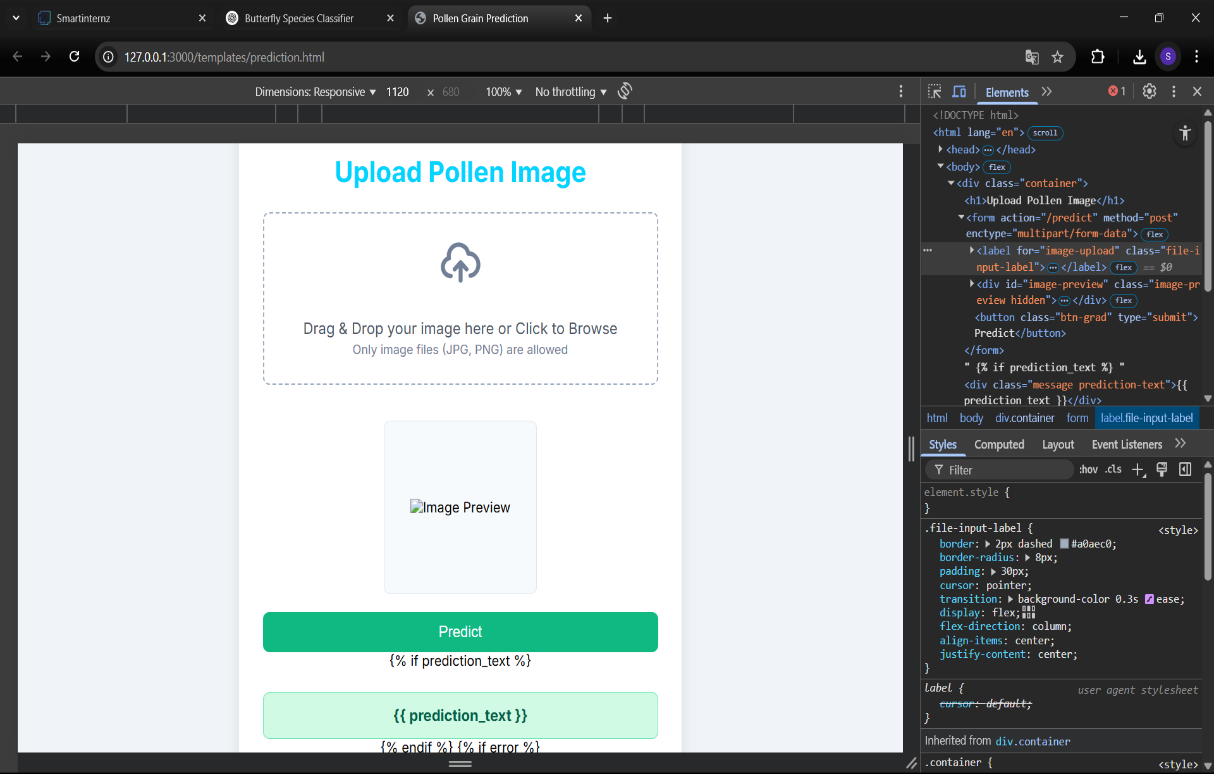
**Known Issues**

* Misclassification between visually similar species.
* Lower confidence for blurred or low-resolution images.
* Limited generalization for unseen environments or backgrounds.

**Future Enhancements**

* Expand dataset with regional and rare butterfly species.
* Deploy on cloud with scalable APIs.
* Add user login and history tracking.
* Integrate map-based geolocation for species prediction.

****



**Trainig the Model at the JupiterNotebook server with vs code**

