Artificial Intelligence And Machine Learning Project Documentation

Introduction:

• Project Title: Enchanted Wings: Marvels of Butterfly Species

Enchanted Wings is a deep learning-based image classification system that identifies and classifies butterfly species from photographs using transfer learning. The system is designed to support biodiversity monitoring, ecological research, and citizen science education by accurately recognizing 75 species of butterflies. It leverages a pre-trained VGG16 model and custom image preprocessing techniques to provide efficient, scalable, and user-friendly identification of butterfly species in real-time.

The project integrates a web-based interface built using Flask and HTML, allowing users to upload butterfly images and receive instant predictions, enhancing both educational outreach and scientific data collection.

Team Members

• Siva Rama Krishana Veepu [Member 1] – Project Lead & ML Developer
(Dataset preparation, model selection, transfer learning implementation, evaluation)

• Anjali Yanamadala [Member 2] – Frontend Developer
(Design and development of HTML pages, user interface for image upload and result display)

• Anuj Kumar [Member 3] – Flask Backend Developer
(API integration, Flask app development, model deployment on local server)

• Dinesh Mortha [Member 4] – Documentation & Testing
(Project documentation, user testing, output validation, report writing)

Project Overview:

Enchanted Wings: Marvels of Butterfly Species is an AI-powered image classification system that identifies 75 butterfly species using deep learning and transfer learning (VGG16). Designed for biodiversity monitoring and citizen science, it helps researchers, educators, and enthusiasts accurately recognize butterfly species from images.

The system features a user-friendly web interface built with Flask and HTML, allowing easy image uploads and real-time predictions. By combining labeled datasets, image augmentation, and a fine-tuned neural network, Enchanted Wings transforms simple photos into valuable ecological insights.

ARCHITECTURE

Frontend Architecture – HTML

This is an HTML file for the user interface of your Enchanted Wings butterfly classification project, allowing users to upload butterfly images. It sends the image to a Flask backend for species prediction and displays the result.

Structure Breakdown

<!DOCTYPE html> & <html>:
Defines the document type and begins the HTML page structure.

<head>:
Contains metadata:
• Character encoding is set to UTF-8.
• The page title is "Butterfly Species Classification".

<body>:
• Sets a nature-themed background image (e.g., butterflies or greenery) using a URL.
• Text color is set to black (text="black").
• A or tag wraps the main form content.

Form Purpose

The <form> is designed to collect a butterfly image from the user and send it to the Flask backend:
• action="{{ url_for('predict') }}" – sends the image to the Flask route named predict.
• method="post" – uses POST to securely transfer the image file.
• enctype="multipart/form-data" – ensures file upload capability.

Input Field

1. Image Upload –
   • <input type="file" name="file" required>
   • Allows user to upload a butterfly image for prediction.

Submit Button

• A Bootstrap-styled button:
"Predict" – triggers the form submission to the Flask backend.

Prediction Output Display

• Below the form, a <h2> tag displays the result using Jinja2 syntax:
• {{ prediction_text }} – shows the predicted butterfly species.
• Styled with inline CSS (semi-transparent background, white text) for visibility.

Styling Notes

• The form is visually centered using or flexbox.
• Simple styling includes:

- Background image
- Bootstrap button (btn btn-primary)
- Inline CSS for the output display
  • Optionally add CSS in a <style> block or external stylesheet.

Functionality Summary

Feature	Description
Image Upload	User selects a butterfly image to classify
Data submission	Sends image to Flask backend for processing
Result display	Shows predicted species below the form
Template integration	Uses Jinja2 ({{ }}) for dynamic Flask output

Backend Architecture – Flask

This Python-based backend powers the butterfly species classification system by integrating a deep learning model with a Flask-based web interface. It acts as the communication bridge between the user interface (HTML) and the image classification model.

Purpose

The backend:

- Receives image input from the HTML form.
- Processes and prepares the image for the deep learning model.
- Loads a pre-trained transfer learning model (VGG16-based) to predict butterfly species.
- Sends the prediction result back to the frontend for display.

Key Functionalities

1. **Homepage Display**
    - When users visit the root URL (/), the index.html page is rendered.
    - This page provides an image upload interface.
2. **Image Upload and Handling**
    - On submitting the form, the image is sent to the /predict route via POST method.
    - The backend reads the uploaded image, resizes it to the required input size (e.g., 128x128), and converts it into a NumPy array suitable for model input.
3. **Model Loading and Prediction**
    - A fine-tuned VGG16 model is loaded using TensorFlow/Keras.
    - The preprocessed image is passed into the model for classification.
    - The model returns the predicted class (e.g., butterfly species name or label).
4. **Displaying the Result**
    - The prediction result is passed back to the HTML page using Jinja2 ({{ prediction_text }}).
    - If an error occurs (e.g., file not uploaded, model issue), a relevant error message is displayed.

Files Involved

- model.h5: The trained Keras deep learning model file for butterfly classification.
- label_map.json (optional): A JSON file mapping class indices to butterfly species names.
- index.html: Frontend for image upload and displaying results.
- app.py: Main Flask application handling routing, model invocation, and response rendering.

How It Works in the Project

- The user uploads a butterfly image through the frontend.
- Flask receives the image, processes it, and sends it to the deep learning model.
- The model predicts the butterfly species.
- The prediction is shown in the browser in real-time.

Role in Enchanted Wings

This backend plays a central role in:

- Connecting the user interface to the deep learning model.
- Enabling real-time butterfly species prediction from uploaded images.
- Managing end-to-end data flow and ensuring interactive, instant results.

Machine Learning Model

- A transfer learning model based on VGG16 is used.
- It is fine-tuned on a dataset of 75 butterfly species.
- The model is implemented in Keras and saved as model.h5.
- Input images are resized to 128x128 pixels, normalized, and passed through the model to return a class prediction.
- The output is mapped to a human-readable species name using a label map.

Database

In the Enchanted Wings project, the database is used to store and manage information related to each butterfly species prediction made through the web interface.

Type of Database

The project uses MongoDB, a NoSQL document-oriented database, ideal for storing flexible and image-related metadata like filenames, predicted species, and timestamps.

How the Database is Used

When a user uploads a butterfly image through the frontend and the Flask backend generates a species prediction using the deep learning model:
• The uploaded image filename
• The predicted butterfly species
• The exact date and time of prediction (timestamp)
are stored together as a document in the MongoDB database.

Structure of Each Record

Each record (document) saved in MongoDB includes:
• **filename:** The name of the uploaded image file.
• **predictedSpecies:** The butterfly species predicted by the model.
• **timestamp:** The date and time when the prediction occurred.

This structure helps track user predictions, monitor classification history, and analyze species distribution trends over time.

Database Integration

• The Flask backend connects to MongoDB using a library like PyMongo.
• After every successful prediction, the corresponding data is inserted into the database automatically.
• The database can be queried to display a log of past predictions or build features like analytics dashboards and user history in future enhancements.

Setup Instructions

This section outlines the tools and steps required to install and run the Enchanted Wings: Marvels of Butterfly Species project on a local machine.

Prerequisites:

Before setting up the project, ensure the following dependencies and tools are installed:

System Requirements

- Operating System: Windows 10/11, macOS, or Linux
- Internet connection (for downloading libraries or connecting to cloud storage if needed)

Software Dependencies

1. Python (3.x)
    - Required to run the Flask backend and load the trained deep learning model (model.h5)
    - Used for image processing and prediction logic using libraries like TensorFlow and NumPy
2. Flask
    - A lightweight Python web framework used to build the backend server for handling user input and returning species predictions
3. TensorFlow / Keras
    - TensorFlow with Keras API is used to load and run the pre-trained VGG16 model
    - Supports transfer learning and deep learning operations
4. NumPy and Pillow
    - NumPy handles numerical image arrays
    - Pillow is used for image file processing and conversion
5. MongoDB (Optional)

    - A NoSQL database used to store image filenames, predicted species, and timestamps
    - Can be used locally or through MongoDB Atlas for remote access
6. HTML (Frontend)
    - HTML file provides the interface to upload butterfly images and view predictions
    - Bootstrap can be used for styling the form and buttons
7. Git (Optional)
    - Useful for version control and pushing the project to GitHub
8. Code Editor
    - Any text editor or IDE like VS Code, Jupyter Notebook, or PyCharm is recommended

Installation

This step-by-step guide helps you set up Enchanted Wings on your local machine:

Step 1: Organize Project Files

Create a main folder (e.g., EnchantedWings) and include:

- app.py – The Flask backend application
- model.h5 – The trained deep learning model (VGG16-based)
- templates/index.html – HTML file for image upload interface
- (Optional) static/ – Folder for CSS or image assets
- (Optional) label_map.json – Maps class numbers to species names

Step 2: Set Up Python and Dependencies

Ensure Python 3.x is installed. Then install required libraries using pip:

pip install flask tensorflow keras numpy pillow

These dependencies allow the system to receive images, preprocess them, and perform predictions using the trained model.

Step 3: Configure Flask Application

In app.py, make sure the routing logic:

- Loads the index.html page at the home route /
- Accepts uploaded images via the /predict POST route
- Preprocesses images and returns the predicted butterfly species

Ensure your templates/ folder is placed correctly so Flask can render the form.

Step 4: Set Up MongoDB (Optional)

If you want to store prediction history:

- Use MongoDB or MongoDB Atlas to create a collection like butterfly_logs
- Each document stores:
    - filename – name of uploaded image
    - predictedSpecies – result from the model
    - timestamp – date and time of prediction

This enables future features like viewing logs or species trend analysis.

Step 5: Run Enchanted Wings Locally

To launch the project:

1. Open your terminal or command prompt
2. Navigate to your project directory
3. Run the Flask app:

python app.py

4. Open a browser and go to: http://localhost:5000
   You'll see the Butterfly Species Prediction form. Upload an image and view the result instantly.

Folder Structure

The Enchanted Wings project is organized into two main parts: the client (frontend) and the server (backend), each responsible for specific tasks in the butterfly classification workflow.

Client – Frontend Structure (HTML-based UI)

The client side handles the user interface where users upload butterfly images and view the predicted species.

Folder: templates/

• This folder is used by Flask to serve HTML pages.
• Contains the main interface file:

- index.html – HTML form where users upload butterfly images. It also displays the predicted butterfly species returned by the backend.

Optional Folder: static/

• Used to hold CSS, JavaScript, or image files for UI enhancement.
• Example structure:

- static/css/style.css – For custom form and page styling
- static/images/ – For background images or UI icons

Server – Backend Structure (Flask Application)

The server side handles image processing, prediction using the deep learning model, and communication between the frontend and backend.

Root Files:

• app.py – Main Flask script

- Loads the trained butterfly classification model (model.h5)
- Defines two routes:
  ▪ / to render the homepage
  ▪ /predict to handle image uploads and return predictions

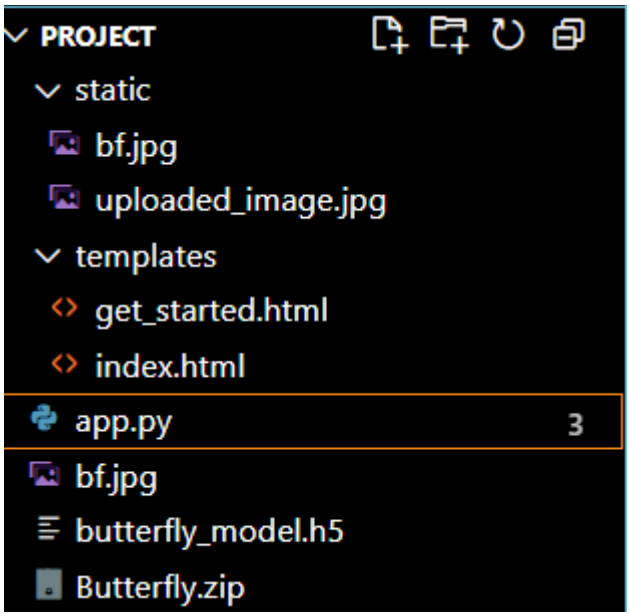
• model.h5 – Pre-trained VGG16 model fine-tuned on butterfly images

Optional Files:

• label_map.json – Maps model output (numeric class) to butterfly species names
• .env – For environment variables like MongoDB URI (if database is used)

• requirements.txt – List of all required Python packages (Flask, TensorFlow, Pillow, etc.)
• predict.py – (Optional) Separate script containing image preprocessing and model logic.

Structure:



Running the Application

To run the Enchanted Wings application locally, both the frontend and backend must be properly set up. The backend handles image classification using a deep learning model, while the frontend allows users to upload butterfly images and view results.

Frontend (User Interface)

The frontend is a static HTML page served through Flask. It provides an interface for users to upload images and view predictions.

- No need to run npm start or any frontend-specific command.
- The main HTML file (index.html) is located in the templates/ folder.
- Flask automatically renders the frontend when the backend is started and the user visits:
  http://localhost:5000

Frontend is served directly by Flask — no separate startup required.

Backend (Flask Server with Deep Learning Model)

The backend is the core of the system. It handles image uploads, runs the pre-trained model, and returns the predicted butterfly species.

To run the backend:

1. Open a terminal or command prompt
2. Navigate to the project directory where app.py is located
3. Start the Flask server using:

python app.py

4. Open your browser and go to:
   http://localhost:5000
5. You will see the *Butterfly Species Classification* interface
6. Upload a butterfly image and click "Predict" to see the result returned from the model

Summary:

Component	Startup Required	How to Run
Frontend	No	Served by Flask at localhost:5000
Backend	Yes	Run using python app.py

API Documentation

The Enchanted Wings backend exposes two main endpoints via the Flask server. These endpoints handle webpage rendering and butterfly species prediction based on user-uploaded images.

1. Home Endpoint

- Endpoint: /
- Method: GET
- Purpose: Renders the main HTML form where users can upload butterfly images.
- Parameters: None
- Function: Serves as the landing page of the application, providing the image upload interface.

2. Prediction Endpoint

- Endpoint: /predict
- Method: POST
- Purpose: Receives the uploaded image file, preprocesses it, and passes it to the pre-trained model for species classification.
- Function: Returns the predicted butterfly species and displays it on the same page.

Request Parameters

The form sends the following parameter to the backend:

Parameter	Type	Description
File	File (JPEG/PNG)	The uploaded butterfly image file

The file must be an image format (e.g., .jpg, .jpeg, or .png) and is processed using Pillow and NumPy to match the model's input format.

Responses

- On Success:
  The predicted butterfly species name is displayed below the upload form on the same page.
- On Failure:
  An appropriate error message is shown (e.g., no image uploaded, invalid file type, or internal model error).

Summary

- The API design is simple and file-based, suitable for single-page web applications.
- It allows real-time interaction between the user and the image classification model using just two endpoints.
- This clean structure makes it easy to scale the application in the future with features like prediction history, REST APIs, or mobile integration.

Authentication

Current State

At present, the Enchanted Wings project does not include any user authentication or authorization mechanisms. The application is designed as an open-access, single-page tool where any user can upload a butterfly image and receive a species prediction from the deep learning model — without logging in or creating an account.

Reason for No Authentication

- The main purpose of the project is to demonstrate butterfly species classification using transfer learning.
- It is designed as a prototype or academic project, making it easier to test and showcase without user access restrictions.
- Since the system doesn't involve user accounts, features like sessions, tokens, or login forms are not implemented.

Future Scope for Authentication

If the project is expanded into a multi-user educational platform or citizen science portal, the following authentication features could be added:

1. **User Login System**
    - Enable personal accounts for saving prediction history.
    - Store user credentials securely in MongoDB.
2. **Session Management**
    - Use Flask sessions or JWT to manage logged-in states across sessions.
    - Allow user-specific access to their own prediction logs.
3. **Role-Based Access Control (RBAC)**
    - Define roles like *Admin*, *Researcher*, or *General User*.
    - Restrict access to model management, history logs, or analytics pages based on roles.
4. **Token-Based Authentication (Advanced)**

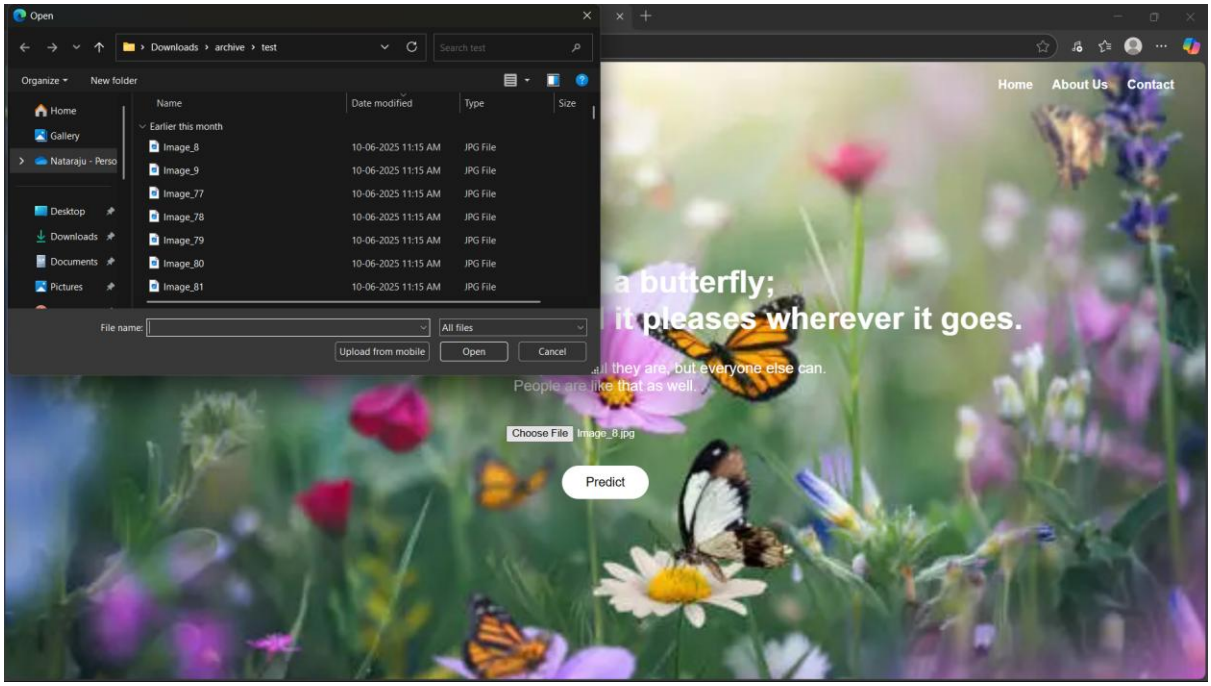
- Use JWT tokens for secure stateless login, suitable for API usage.
- Ideal for integrating with mobile apps or advanced frontend frameworks like React.

Summary

Feature	Current Status
User Login	Not Implemented
Session Handling	Not Implemented
Token Authentication	Not Implemented
Open Access	Enabled

The current system allows open access to simplify usage, but it is fully scalable to include secure user authentication features in the future.

User Interface



Testing

Testing in the *Enchanted Wings* project focuses on validating the accuracy of the deep learning model and the proper functionality of the Flask backend. Both model evaluation and end-to-end interaction were tested to ensure a smooth user experience.

1. Model Testing

- The VGG16-based model was tested using classification metrics:
    - **Accuracy**, **Precision**, **Recall**, and **F1-score**
- It was evaluated on unseen butterfly images to ensure generalization.
- The best-performing model was saved as model.h5 after comparing results from different tuning methods.

2. Backend Testing

- Manual testing was done by uploading various butterfly images through the HTML form.
- The prediction output was checked for correctness and proper display.
- Error cases like no file upload or invalid file type were tested to ensure user-friendly error handling.

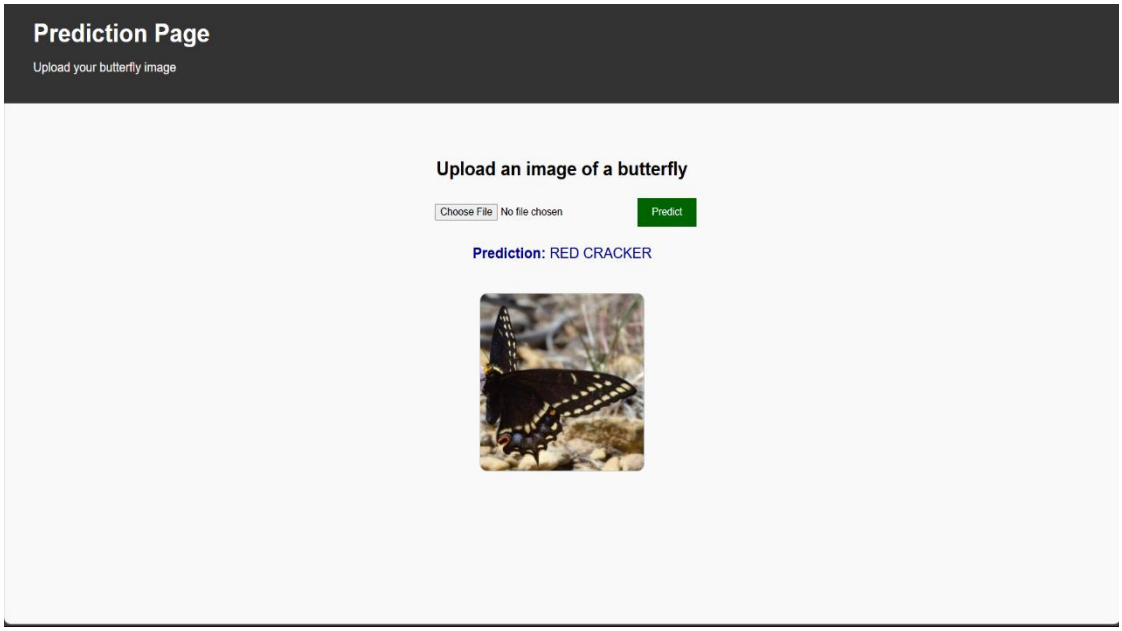
3. Integration Testing

- The full pipeline—from image upload to prediction display—was tested end-to-end.
- Ensured correct communication between frontend (HTML) and backend (Flask + TensorFlow).
- Verified that image preprocessing, model prediction, and output rendering worked seamlessly.

Testing Tools Used

- **Jupyter Notebook**: For model training and evaluation
- **Browser-based manual testing**: For frontend-backend functionality
- No formal unit tests (e.g., pytest) were used, as the focus was on prototype validation.

Screenshots or Demo



PROJECT DEMO LINK :

https://drive.google.com/file/d/1Ozv8DN2UTRxOFc8RrhPsS2dLkAuvY_tt/view?usp=sharing

Known Issues

The Enchanted Wings project is functional and effective as a prototype, but there are several known limitations that affect its scalability, robustness, and user experience. These are summarized below:

1. Lack of User Authentication

Currently, the system does not support user login or authentication. Any user can access the interface and use the prediction feature. This limits the ability to personalize the experience, store individual prediction history, or restrict access based on roles.

2. Minimal Input Validation

The form accepts image uploads, but there is no advanced validation to check for corrupted files, inappropriate sizes, or unsupported formats. This may lead to incorrect predictions or app crashes if invalid images are submitted.

3. No Automated Testing

All testing has been done manually using browsers and Jupyter Notebook. There are no automated unit or integration tests, making it harder to detect bugs or ensure consistency after updates.

4. Static Deep Learning Model

The VGG16-based model (model.h5) is fixed and trained only once. There is no mechanism to retrain the model with newly collected butterfly images, which could reduce the model's relevance or accuracy over time.

5. Limited Error Handling

While basic errors (like missing file uploads) are handled, internal errors such as failed model loading or incorrect file paths are not user-friendly and may confuse users or developers during debugging.

6. Basic UI and Limited Responsiveness

The HTML frontend is simple and may not render well on mobile devices. There's minimal styling, and it lacks responsive design features for different screen sizes.

7. No Log Viewer or Dashboard

Although predictions can be stored (e.g., in MongoDB), there is currently no interface to view or manage past predictions. This limits the application's usability for tracking or research purposes.

8. Not Deployed Online

The application runs only on a local server (localhost). It has not been deployed to platforms like Heroku,or AWS, which would make it accessible to wider audiences.

Future Enhancements

While Enchanted Wings demonstrates the effectiveness of transfer learning in classifying butterfly species, there are several potential improvements to enhance its usability, scalability, and real-world impact:

1. User Authentication System

- Add login and registration features.
- Allow users to track and view their prediction history.
- Introduce role-based access (e.g., Admin, Researcher, General User).

2. Modern, Responsive UI

- Upgrade the frontend using frameworks like **React.js**, **Bootstrap**, or **Tailwind CSS**.
- Improve mobile responsiveness and visual design.
- Integrate species info cards, animated feedback, and user-friendly layouts.

3. Prediction Logs and Analytics

- Store prediction data (image, result, timestamp) in a database.
- Display logs in a user dashboard.
- Add analytics like “Most Predicted Species”, “Monthly Upload Trends”, etc.

4. Real-Time Data Integration

- Integrate APIs to fetch real-time butterfly sighting data (e.g., iNaturalist, Butterfly Conservation).
- Enrich predictions with context like region and season.

5. Model Retraining Pipeline

- Enable periodic retraining using newly uploaded and verified images.
- Use tools like **AutoML**, **Keras Tuner**, or **MLFlow** for improved model accuracy.
- Apply **transfer learning** on newer models like EfficientNet or MobileNetV3.

6. Notifications or Educational Alerts

- Notify users when a rare or endangered species is detected.
- Provide conservation tips or fun facts post-prediction.

7. Cloud Deployment

- Deploy the system to **Render**, **Heroku**, or **AWS** for public access.
- Connect to a custom domain and enable HTTPS for security.

8. REST API Development

- Convert classification logic into a RESTful API.
- Allow mobile apps or third-party websites to use *Enchanted Wings* as a service.

9. Location-Based Classification

- Add optional geolocation input for more accurate predictions.
- Adapt predictions based on species' geographical distribution.

10. Multi-Language Support

- Translate UI content into regional languages.
- Increase accessibility and encourage global citizen science participation.

-------THANK YOU------