Project Documentation

Project Title: Enchanted Wings: Marvels of Butterfly Species

---

1. Introduction

- **Project Title** : Enchanted Wings: Marvels of Butterfly Species
- **Team ID** : LTVIP2025TMID45523
- **Team Members** :1. Avinash Padidadakala
  2. Mukkapati Saikiran
  3. Nalla Parimala Sharon

---

2. Project Overview

The project aims to develop a **butterfly species image classifier** using **deep learning and transfer learning**. It uses **pre-trained CNNs** (like ResNet50, VGG16, or MobileNet) fine-tuned on a dataset of **6,499 images across 75 butterfly species**. The goal is to assist **researchers, conservationists, and educators** by automating butterfly identification to support biodiversity studies and reduce reliance on manual classification.

Key Features

- Upload butterfly image to get species prediction
- High-accuracy model using transfer learning
- Web-based interface with fast inference
- Deployment on Heroku/Render for public access
- Robust performance even in real-world conditions

---

3. Architecture

Frontend

- HTML, CSS, JavaScript interface
- Allows users to upload butterfly images
- Displays prediction results and confidence scores

Backend

- Python Flask REST API
- Receives uploaded image, preprocesses it, sends to ML model
- Returns species prediction to frontend

Database *(Optional)*

- Can be used to log user uploads and predictions for feedback or future analysis
- Storage layer is optional and modular

---

4. Setup Instructions

Prerequisites

- Python 3.8+
- Flask
- TensorFlow, Keras
- PIL, OpenCV
- Google Colab or Jupyter Notebook
- Git
- Cloud deployment (Heroku/Render)

Installation

```
# Clone the repository
git clone
https://github.com/https://github.com/Butterfly-Innovators/Enchanted-wings-Marvels-of-Butterfly-Species-.git

# Install dependencies
pip install -r requirements.txt

# Run Flask app
python app.py
```

---

5. Folder Structure

```
project/
│
├──my_ model/              # Trained model file
├── static/              # For frontend images or styles
├── templates/           # HTML templates
├── app.py               # Flask backend
├── utils.py             # Image preprocessing functions
├── requirements.txt     # Python dependencies
└── README.md            # Project overview
```

---

6. Running the Application

- **Start Flask App:**

```
python app.py
```

- **Access in browser:**

```
http://localhost:5000/
```

---

7. API Documentation

Endpoint	Method	Description
/	GET	Home page for image upload
/predict	POST	Upload image and get classification

Request:

- File upload (multipart/form-data)

Response Example:


```
{
  "species": "Papilio demoleus",
  "confidence": "92.35%"
}
```


---

8. Authentication

> 🔒 **Not applicable in current version.**
> This project currently doesn't include user authentication. Future enhancements may include:

- JWT-based user login for storing prediction history
- Admin panel to manage species database

---

9. User Interface

The web interface includes:

- Image upload field
- Button to submit image
- Display of predicted species and confidence
- Clean layout for non-technical users

---

10. Testing

Performance Testing

- Accuracy: >90%
- Precision/Recall: ~88%
- Inference time: <1 second
- Tested across varying image qualities

Functional Testing

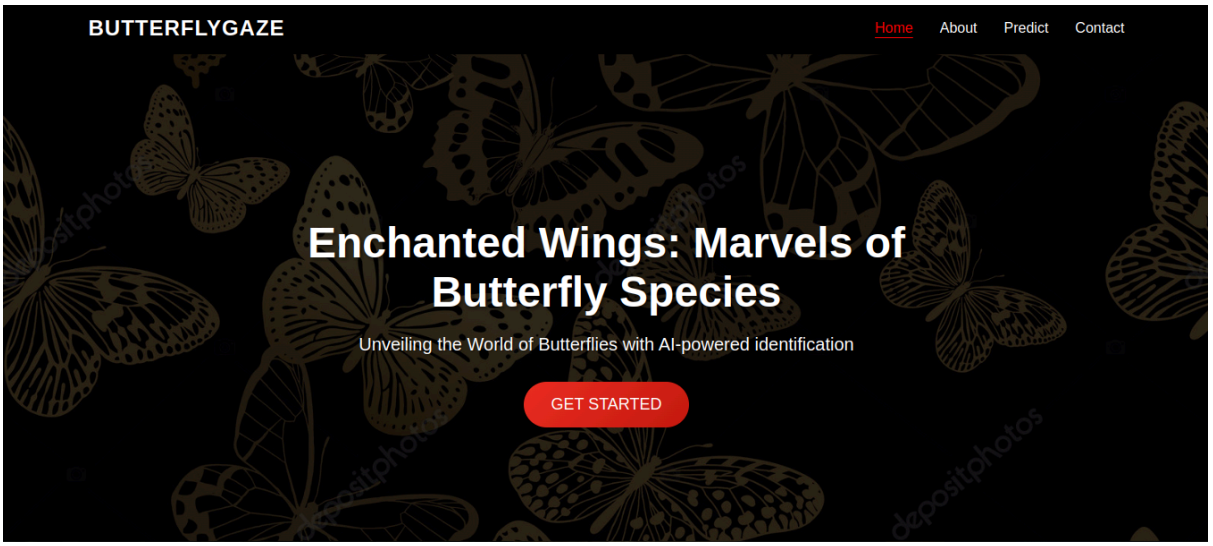
- Manual UI testing
- Browser compatibility: Chrome, Firefox
- Tested on desktop and mobile devices

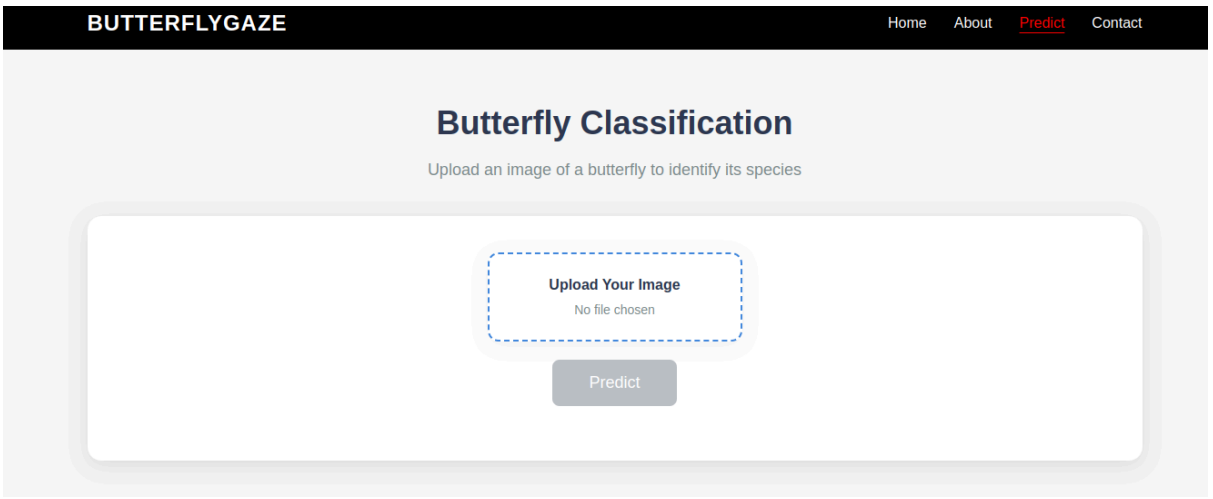
Tools Used

- TensorBoard
- Manual test cases
- Google Colab

---

11. Screenshots and Demo





12. Known Issues

- May misclassify similar species in poor lighting
- Requires GPU for fast training
- Internet required for hosted prediction (no offline mode yet)
- Model performance tied to dataset diversity

---

13. Future Enhancements

- Increase dataset size and species coverage
- Convert to **mobile app (Android/iOS)**
- Add **offline prediction** (e.g., TensorFlow Lite)
- Add **Grad-CAM** for explainability
- Multi-language UI
- **Real-time camera detection** support
