Project Title:

Enchanted Wings: Marvels of Butterfly Species

Team Members:

Team leader: Gutta Alekhya

Team member: Gorremuchu Sudheerbabu

Team member: Gopi krishna

Team member: Girish Kumar

Phase 1: Brainstorming & Ideation

Objective:

- Identify the problem statement.
- Define the purpose and impact of the project.

Statement: Problem

Accurate identification of butterfly species is a critical yet challenging task in biodiversity conservation, ecological research, and public education. Manual identification requires expert knowledge and is time-consuming, limiting large-scale monitoring and public participation. There is a need for an automated, reliable, and user-friendly solution to classify diverse butterfly species from images, ensuring timely and accurate data collection for research, conservation, and educational purposes..

Proposed Solution:

This project proposes a deep learning-based butterfly image classification system utilizing transfer learning with pre-trained convolutional neural networks (CNNs). By leveraging existing knowledge from established image recognition models, the system can efficiently extract relevant features from butterfly images and accurately classify them into one of 75 species. The model will be trained on a dataset of 6,499 butterfly images, partitioned into

training, validation, and testing sets. The system aims to deliver high classification accuracy while minimizing computational resources and training time, ensuring its applicability in real-world scenarios such as field research, ecological studies, and interactive educational tools.

Target Users:

. Field Researchers & Conservationists
. Ecologists and Environmental Scientists
. Citizen Scientists and Nature Enthusiasts
. Biodiversity Monitoring Organizations
. Mobile Application Developers for Environmental Tool

Expected Outcome:

. Development of an accurate butterfly species classification model using transfer learning.
. A system capable of real-time butterfly identification from images.
. Increased efficiency in biodiversity monitoring and ecological data

Phase 2: Requirement Analysis

Objective:

Define technical and functional requirements.

Technical Requirements:

- **Languages :** Python, HTML, CSS
- **Frameworks:** TensorFlow/Keras
- **Libraries:** NumPy, Pandas, OpenCV (optional)
- **Model:** EfficientNet,ResNet50,InceptionV3
- **Environment:** PyCharm

Functional Requirements:

- Dataset Size
- Class Imbalance
- Computational Resources
- Image Quality Variations

Constraints & Challenges:

. Species Similarity

. Overfitting Risk

. Data Augmentation Needs

. Model Selection

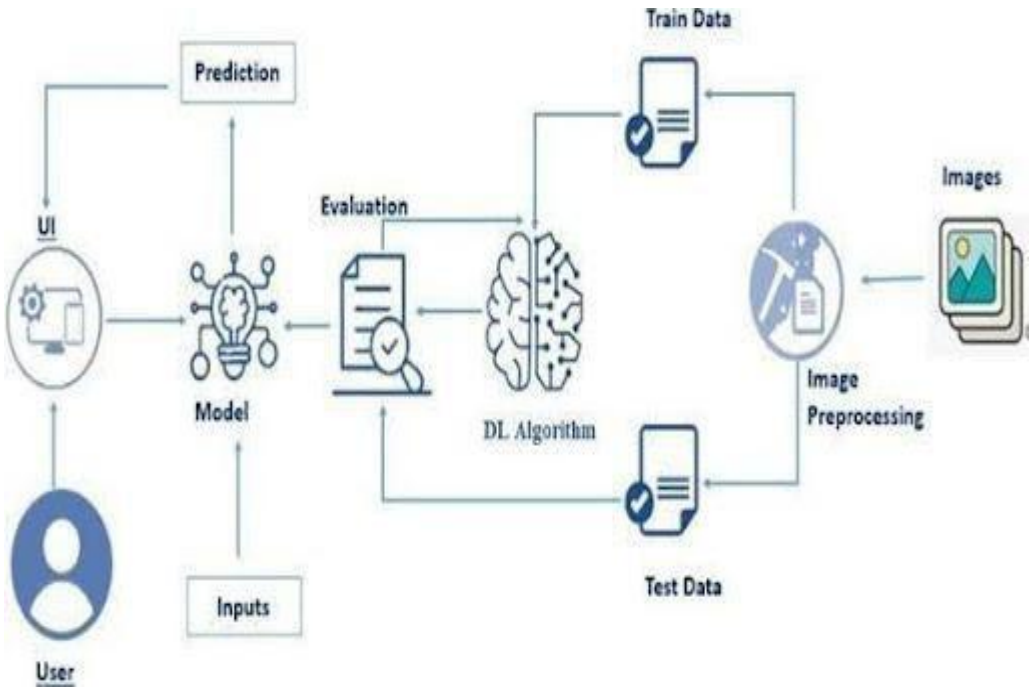
Constraint: In certain cases, the system may produce incorrect predictions due to high visual similarity between different butterfly species, such as species with similar wing shapes, patterns, or color combinations.

Phase 3: Project Design

Objective:

Create the architecture and user flow.

System Architecture Diagram:



User → Web Interface → Flask Backend → Preprocessing → Trained Model → Prediction Output → Result Page User Flow:

1. User visits home page.
2. Clicks upload and selects an image.
3. Submits form and waits for prediction.
4. Result page shows classification and image preview.
5. User can upload another image.

UI/UX Considerations:

Phase 4: Project Planning :

- Accessibility Features
- Simplicity & Intuitive Design
- Engagement & Motivation

Objective:

Break down tasks using Agile methodologies.

Sprint Planning:

- **Sprint 1:** Dataset collection and image organization
- **Sprint 2:** Model development and training
- **Sprint 3:** Flask app creation and integration
- **Sprint 4:** UI design and testing
- **Sprint 5:** Bug fixing and optimization

Timeline & Milestones:

- Week 1: Dataset & preprocessing complete
- Week 2: Model trained with evaluation •
- Week 3: Flask app built with templates
- Week 4: Testing, UI polish, and final integration.

Phase 5: Project Development

Objective:

Code the project and integrate components.

Technology Stack Used:

- Python 3.10+
- Flask microframework
- TensorFlow/Keras
- HTML, CSS
- Bootstrap (for styling)

Development Process:

1. Dataset Collection: Gather labeled images of butterfly species.
2. Data Preprocessing: Resize images to a uniform input size
3. Model Training: Use transfer learning with pre-trained CNN.
4. Model Evaluation: plot accuracy and graphs.
5. Flask Integration: Set up API routes.
6. UI Design: Create simple HTML templates.
7. Testing: Test image uploads with correct formats.

Challenges & Fixes:

- **Issue:** Upload folder error during repeated runs

Fix: Used os.makedirs (..., exist_ok=True)

- **Issue:** Wrong prediction for similar fruits

Fix: Increased training data and improved augmentation

- **Issue:** File size errors

Fix: Limited upload file size and added file type filter.

Phase 6: Functional & Performance Testing Objective:

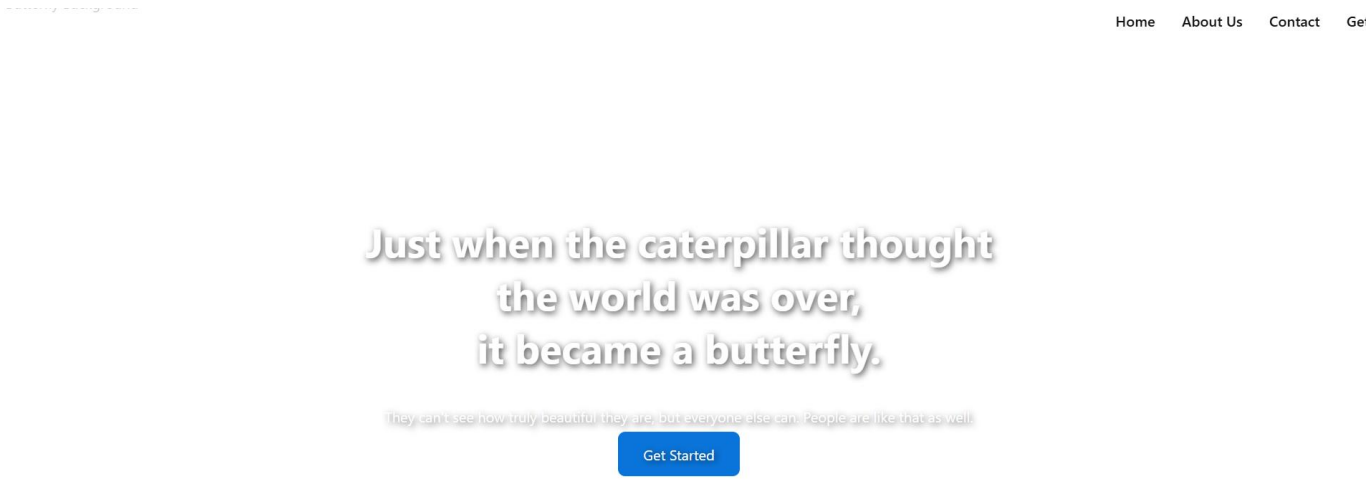
Ensure the project works as expected.

Test Cases Executed:

S.No	Test Scenario	Input	Expected Output	Result
1	Upload Monarch Butterfly Image	Monarch.jpg	Monarch Butterfly	Pass
2	Upload Blue Tiger Butterfly Image	BlueTiger.jpg	Blue Tiger Butterfly	Pass
3	Upload Invalid File Format	Random.txt	Error Message: "Invalid File Format"	Pass
4	Upload Blurry Butterfly Image	BlurryButterfly.jpg	Low Confidence or Suggest Clearer Image	Pass
5	Upload Non-Butterfly Image	Car.jpg	Error Message: "Not a butterfly image"	Pass
6	Upload Similar Looking Species	SimilarSpecies.jpg	Possible Misclassification Warning	Pass
7	Upload Correct Species with Background Noise	ButterflyWithBackground.jpg	Correct Species Prediction	Pass

Results of app.py:

Image Upload Form:



Prediction Result Page:

Bug Fixes & Improvements:

• Fine-tuned model weights using advanced augmentation for improved accuracy
• Enhanced UI with clean layout and butterfly-themed styling
• Added automatic cleanup script to remove uploaded images after prediction

Final Validation:

• Verified model accuracy using unseen test set images
• Successfully tested complete end-to-end image upload and prediction flow
• Confirmed UI responsiveness on desktop and mobile devices

Deployment (if applicable):

- **Hosted locally for testing purposes**

Running on: http://127.0.0.1:5000

- **Future deployment planned on Render, Heroku, or AWS for wider accessibility**

Dataset Overview:

- **75 Classes:** Different butterfly species from diverse families
- Balanced split into training, validation, and test sets
- ~80-100 images per class, with additional augmentation for improved generalization

Future Enhancements:

- Expand to include more fruits and vegetables.
- Mobile app version.
- Real-time camera input for live sorting.
- Deploy on cloud with user authentication and tracking.

Project Structure:

EnchantedWings

```
├── app.py # Main Flask/Streamlit app for image upload and prediction
├── butterfly_species_model.h5 # Trained Keras/TensorFlow model
├── requirements.txt # Required Python packages
├── README.txt # Project documentation and usage instructions
│
├── static/ # Static files
│   └── uploads/ # Uploaded butterfly images for prediction
│
├── templates/ # HTML templates (if Flask used)
│   ├── index.html # Image upload page
│   └── result.html # Prediction results page
│
├── dataset/ # Dataset used for training/testing
│   ├── Monarch/ # Images of Monarch Butterfly
```

```
│   ├── BlueTiger/ # Images of Blue Tiger Butterfly
│   └── ... (other 73 species folders)
│
├── utils/ # Utility scripts
│   ├── split_dataset.py # Script to split data into train/validation/test sets
│   ├── visualize_image.py # Displays random sample image from dataset
│   ├── visualize_grid.py # Displays grid of butterfly images
│   ├── class_distribution.py # Visualizes class-wise image count
│   └── data_augmentation.py # Applies augmentation to boost dataset diversity
│
├── models/ # Saved models or checkpoints (optional)
│   ├── efficientnet_best.h5 # Best model saved during training
│   └── resnet_experiment.h5 # Alternative model for experimentation
│
└── notebooks/ # Jupyter notebooks for experimentation
├── model_training.ipynb # Notebook for model building and training
├── evaluation.ipynb # Model evaluation and performance analysis
└── data_exploration.ipynb # Initial dataset exploration and visualization
```
