Artificial Intelligence and Machine learning

Project Documentation

1. Introduction

• **Project Title:** Enchanted Wings: Marvels Of Butterfly Species

• **Team Members:**

Team Leader: Alli Sai Aditya

Team Member: Atluri Mounavya

Team Member: Bandaru Durga Rao

Team Member: Abdul Ayesha Tarannum

2. Project Overview

• **Purpose:** The primary goal of this project is to develop an intelligent butterfly image classification system using deep learning and transfer learning techniques. The system helps in automated identification of butterfly species to assist in ecological research, biodiversity monitoring, and education.

• **Features:**

- Classification of 75 butterfly species
- Real-time prediction using mobile/web interface
- Dataset divided into training, validation, and test sets
- High accuracy through pre-trained CNN models
- User-friendly output with species information

3. Architecture

• **Frontend:** The frontend is built using React.js, providing a responsive and interactive user interface. Users can upload butterfly images, view classification results, and access species information. React components manage routing, state, and UI rendering efficiently using libraries like Axios for API communication.

• **Backend:** The backend is developed using Node.js with Express.js as the framework.
It handles:
Image upload and preprocessing
API endpoints for sending images to the classification model
Serving classification results to the frontend
User authentication and session management (if applicable)

• **Database:**
The system uses MongoDB as the database to store: User data (name, email, login info)
Uploaded image metadata (filename, upload time, user)
Classification results (image ID, predicted species, confidence score)
Data interactions are managed using Mongoose, which provides schema definitions and queries for efficient database access.

4. Setup Instructions

• **Prerequisites:** To run the project successfully, you must have Node.js (version 14 or higher) and MongoDB installed on your system. These are essential for running the backend server and managing the database.

• **Installation:** First, clone the project repository using Git. After cloning, navigate into the client folder and run npm install to install all frontend dependencies. Then move to the server folder and run npm install again to install backend dependencies. Create a env file in the server directory to define environment variables such as the MongoDB URI, server port, and any secret keys needed for authentication.

5. Folder Structure

• **Client:** The client folder contains the React frontend. It is organized into folders such as components for reusable UI elements, pages for different views/screens, assets for images and styles, and services for API calls. The structure promotes clean and modular code for scalability.

• **Server:** The server folder contains the Node.js + Express.js backend. It includes routes to define API endpoints, controllers to handle business logic, models to define MongoDB schemas using Mongoose, and middleware for tasks like authentication and error handling. This structured backend ensures separation of concerns and easy maintenance.

6. Running the Application

• Provide commands to start the frontend and backend servers locally.

○ **Frontend:** Navigate to the client folder and run npm start. This will launch the React development server, typically on http://localhost:3000.

○ **Backend:** Navigate to the server folder and run npm start. This will start the Express.js server, usually on http://localhost:5000 or as defined in the .env file.
Ensure MongoDB is running locally or connected via a cloud service like MongoDB Atlas.

7. API Documentation

• The backend provides RESTful APIs:

POST /api/classify: Accepts an image file and returns the predicted butterfly species with a confidence score.

GET /api/results: Retrieves the classification history for a user.

POST /api/auth/login: Authenticates a user and returns a JWT token.

POST /api/auth/register: Registers a new user account.

Each API expects standard headers (Content-Type: application/json or multipart/form-data) and returns JSON responses with proper status codes.

8. Authentication

• Authentication is implemented using JWT (JSON Web Tokens). When a user logs in or registers, the server generates a token, which is sent to the client and stored in local storage. The token is then included in the headers of future requests to authenticate the user. Protected routes verify this token using middleware before granting access, ensuring secure communication and data access.

9. User Interface

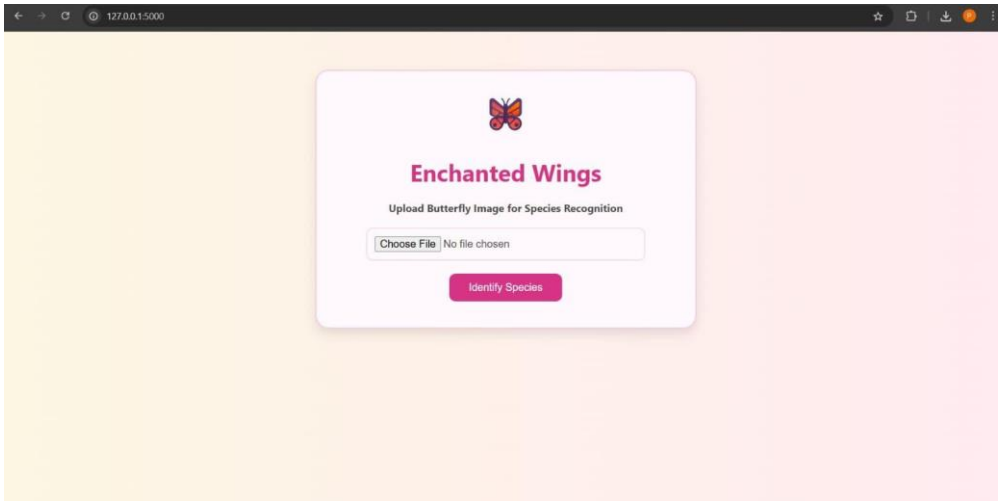
• The UI is clean, responsive, and user-friendly. It includes components for uploading butterfly images, viewing classification results, and browsing educational content. Navigation is handled by React Router. Visual feedback (like loading animations and modals) improves user interaction. The interface is also optimized for mobile devices to support citizen science participation.

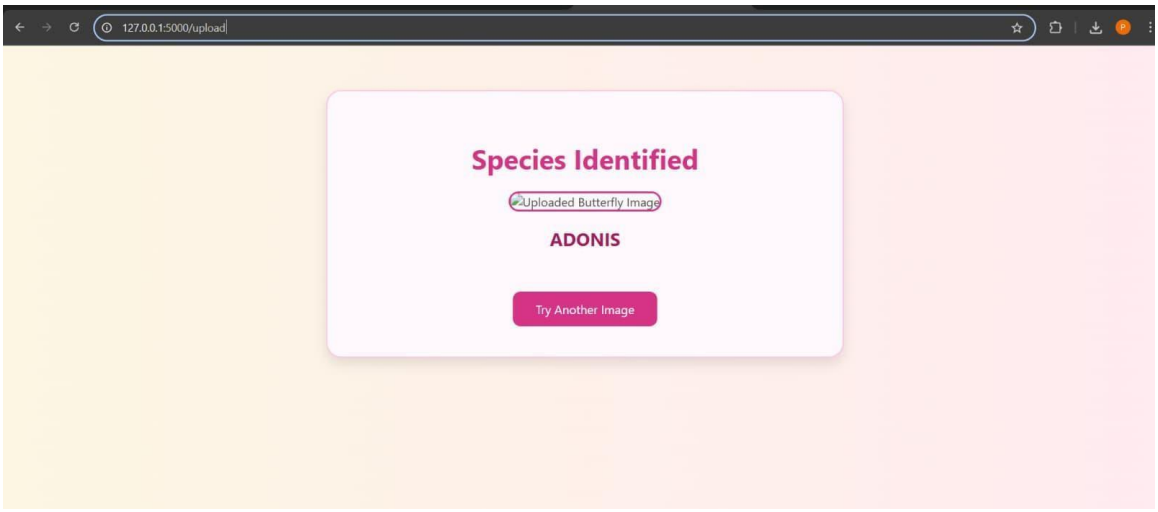
10. Testing

• Frontend testing is handled using Jest and React Testing Library, focusing on rendering components correctly and simulating user interactions.

Backend testing uses Mocha and Chai for unit and integration testing of API routes and logic. The testing setup ensures core functionality works correctly and remains stable with updates.

11. Screenshots or Demo





12. Known Issues

• Some known limitations include:

Lower accuracy on images with poor lighting or occlusions.

Processing delays on slower devices.

The model only supports the 75 trained species, limiting broader classification.

These issues are actively being monitored and improved in future updates.

13. Future Enhancements

• Planned future improvements include:

Adding camera integration for live image capture.

Expanding the species database beyond the current 75 classes.

Introducing a user feedback loop to improve classification accuracy.

Developing a mobile application version.

Deploying the application on the cloud for wider accessibility and scalability.