Solution Architecture

Date: 27 June 2025
Team ID: LTVIP2025TMID45523
Project Name: Enchanted Wings: Marvels of Butterfly Species
Maximum Marks: 4 Marks

Solution Architecture

Enchanted Wings leverages transfer learning to classify 6499 images across 75 butterfly species, enabling biodiversity monitoring, ecological research, and citizen science. The solution architecture bridges business problems (e.g., manual identification challenges) with technology solutions, defining the system's structure, behavior, features, and requirements.

System Overview

The solution comprises a mobile app and cloud-based API for real-time butterfly image classification, geolocation tagging, and data submission to a centralized database. It uses a fine-tuned ResNet-50 model for classification, optimized for low-resource devices, and integrates with a citizen science platform for user engagement and data aggregation.\

Architecture Components

1. **Client Layer**:
    a. **Mobile App**: iOS/Android app for image capture, classification, and geolocation tagging.
    b. **User Interface**: Displays species name, confidence score, and educational content.
    c. **Input**: Accepts .jpg/.png images from camera or gallery.
2. **Application Layer**:
    a. **API Gateway**: Handles image uploads and API requests (e.g., RESTful POST to https://enchantedwings.api/classify).
    b. **Authentication**: Validates user credentials and API keys for secure access.
    c. **Preprocessing Module**: Normalizes images (resize to 224x224, normalize pixel values) for model input.

3. **Model Layer**:
    a. **Classification Model**: Fine-tuned ResNet-50 (50 convolutional layers, softmax output for 75 classes).
    b. **Training Data**: 6499 images (70% training: 4549, 15% validation: 975, 15% test: 975).
    c. **Inference**: Outputs species name and confidence score (e.g., 90% for Monarch).
4. **Data Layer**:
    a. **Database**: Cloud-based relational database (e.g., PostgreSQL) stores classification results, geolocation, timestamps, and user contributions.
    b. **Schema**: Tables for Images (ID, file), Classifications (species, confidence), Metadata (geolocation, timestamp).
    c. **Backup**: Regular backups to ensure data integrity.
5. **Integration Layer**:
    a. **Citizen Science Platform**: API integration with global biodiversity databases (e.g., iNaturalist) for data sharing.
    b. **Analytics**: Aggregates data for species distribution and research reports.

Data Flow

1. User uploads an image via the mobile app or API.
2. Image is preprocessed (resized, normalized) and sent to the model.
3. ResNet-50 classifies the image and returns species name and confidence score.
4. Results, along with geolocation and timestamp, are stored in the database.
5. App displays results; data is shared with citizen science platforms if authorized.

Features

- Real-time classification with >85% accuracy.
- Geolocation tagging for species mapping.
- User-friendly app interface with educational content.
- API for third-party integration.
- Scalable database for growing contributions.

Development Phases

1. **Phase 1**: Model training and fine-tuning (ResNet-50 on 6499 images).
2. **Phase 2**: Mobile app development and API setup.

3. **Phase 3**: Database integration and citizen science platform connectivity.
4. **Phase 4**: Testing (functional, performance, UAT) and deployment.

Requirements

- **Hardware**: Cloud servers for API and database; mobile devices (iOS 13+, Android 10+).
- **Software**: TensorFlow for model, Flask/FastAPI for API, PostgreSQL for database.
- **Performance**: Classification <2 seconds, API response <5 seconds for 50 concurrent requests.
- **Security**: HTTPS, API key authentication, GDPR-compliant data handling.

Note

[Insert placeholder for architecture diagram, to be added in presentation tools like PowerPoint or Canva, showing client, application, model, data, and integration layers with data flow arrows.]