High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **PlantSnap - AI Leaf Identifier**. PlantSnap is a web application that enables users to upload a photo of a plant leaf and receive an AI-driven identification of the plant species. The project demonstrates image upload, basic image recognition, and introduces learners to computer vision in agriculture using open-source technologies.

---

1. System Architecture Overview

Architecture Summary:
PlantSnap is structured as a web-based client-server application. The frontend handles user interactions and image uploads, while the backend processes images using a machine learning model to identify plant species.

Component	Role/Responsibility
Web Frontend	User interface for image upload and result display
Backend API	Receives images, manages requests, returns predictions
ML Inference Engine	Processes images, runs plant identification model
Data Storage	Stores uploaded images and results (optional)

---

2. Component Interactions

Step	Interaction Description
1	User uploads a leaf image via the web frontend
2	Frontend sends the image to the backend API
3	Backend API forwards the image to the ML Inference Engine
4	ML Inference Engine processes the image and predicts the plant species
5	Prediction result is sent back to the backend API
6	Backend API returns the result to the frontend for user display

---

3. Data Flow Overview

Data Item	Source	Destination	Purpose
Leaf Image	User (Frontend)	Backend API	Image to be identified
Image File	Backend API	ML Inference Engine	Input for prediction

Prediction Result	ML Engine	Backend API	Identified plant species
Result	Backend API	Frontend	Display to user

---

4. Technology Stack

Layer	Technology/Frameworks
Frontend	HTML/CSS/JavaScript (e.g., React or simple JS)
Backend API	Python (Flask or FastAPI)
ML Inference	Python, OpenCV, scikit-learn or TensorFlow/Keras
Data Storage	Local filesystem or lightweight DB (optional)
Image Processing	OpenCV

---

5. Scalability & Reliability

- **Scalability:**
  The system is designed for easy extension; the backend and ML inference can be containerized for deployment. For higher loads, the ML engine can be scaled horizontally.
- **Reliability:**
  Input validation and error handling are implemented at the API level. Uploaded images are sanitized and size-limited.
- **Security:**
  Only image files are accepted; user uploads are validated to prevent malicious content.

End of Document