Leaf Disease Detection API Guide

**Overview**  
This document explains the FastAPI-based leaf disease prediction service, how to interact with it via HTTP POST requests (including Postman), and how to integrate the hosted API into an Android application. The deployed instance is available at   
*https://leaf-disease-api-v3fr.onrender.com*

# 1. Project Summary

FastAPI application that loads a TensorFlow model (keras\_model.h5) to classify plant leaf images into disease categories. Labels are derived from labels.txt and returned with confidence scores. The API exposes two endpoints: a health/info root route and /predict for file uploads.

# 2. Local Setup (Optional)

1. Create virtual environment: python -m venv .venv and activate it.
2. Install dependencies: pip install -r requirements.txt
3. Start server: uvicorn main:app --host 0.0.0.0 --port 8000

# 3. API Endpoints

**GET /**: Returns service status, expected image size, and available labels.

**POST /predict**: Accepts an image upload (multipart/form-data, key "file"). Returns predicted label, confidence score, and per-label confidences.

# 4. Testing the Deployed API with Postman

1. Open Postman and create a new request tab.
2. Set method to POST and URL to https://leaf-disease-api-v3fr.onrender.com/predict.
3. Under Headers add Accept: application/json (optional but recommended).
4. Go to the Body tab, choose form-data, add a key named file, set type to File, and attach a leaf image (JPG/PNG).
5. Click Send. A successful response returns HTTP 200 with JSON containing predicted\_label, confidence, and all\_confidences.
6. If you receive errors: verify the key name (file), ensure the image is valid, and review Render logs.

# 5. Using curl (Optional)

curl -X POST "https://leaf-disease-api-v3fr.onrender.com/predict" -H "Accept: application/json" -F "file=@/path/to/leaf.jpg"

# 6. Android Integration Guide

Call the Render-hosted API from Android using Retrofit or OkHttp. Ensure the request uses multipart/form-data and sends the image as part "file".

* Retrofit outline:
* 1. Add dependencies: implementation "com.squareup.retrofit2:retrofit:2.9.0", implementation "com.squareup.retrofit2:converter-gson:2.9.0", and okhttp multipart support.
* 2. Define API interface:

interface LeafApi {  
 @Multipart  
 @POST("/predict")  
 Call<PredictionResponse> uploadLeaf(  
 @Part MultipartBody.Part file  
 );  
}

* 3. Instantiate Retrofit with base URL https://leaf-disease-api-v3fr.onrender.com
* 4. Convert image file to MultipartBody.Part:

File file = new File(imagePath);  
RequestBody requestFile = RequestBody.create(file, MediaType.parse("image/jpeg"));  
MultipartBody.Part body = MultipartBody.Part.createFormData("file", file.getName(), requestFile);

* 5. Execute the Retrofit call asynchronously and handle PredictionResponse fields (predicted\_label, confidence, all\_confidences).
* 6. Add INTERNET permission in AndroidManifest.xml and handle network errors gracefully.

# 7. Troubleshooting & Maintenance

* HTTP 500/502: Check Render logs; verify TensorFlow model loads and the instance has enough memory.
* Slow responses: large models can cause cold starts; consider upgrading the Render plan.
* Android errors: ensure HTTPS is used and the device/emulator has connectivity.

# 8. Useful Links

**Live API:** https://leaf-disease-api-v3fr.onrender.com  
**Swagger UI:** https://leaf-disease-api-v3fr.onrender.com/docs  
**GitHub Repository:** [**https://github.com/sonamdobriyal-a11y/leaf-disease-api.git**](https://github.com/sonamdobriyal-a11y/leaf-disease-api.git)