ACADEMIC MAJOR PROJECT · 2026

Plant Disease Detection & Hybrid Treatment System

A full-stack AI platform that identifies crop pathologies in real time and returns an actionable agricultural prescription — built on a dual-ResNet50 ensemble, test-time augmentation, and an adaptive hybrid treatment engine.

96.93%	38 / 14	<200 ms	10×
TEST ACCURACY	CLASSES / CROPS	GPU INFERENCE	ENSEMBLE + TTA PASSES

v2.0 · August 2026 · Next.js 16 / React 19 · Flask 3.1 + PyTorch · OpenAI GPT-3.5-Turbo

CONTENTS

01 Executive Summary
02 Problem Statement & Context
03 End-to-End System Architecture
04 Deep Learning & ML Pipeline
05 Dataset Engineering & Taxonomy
06 Backend API & Inference Server
07 Frontend Architecture
08 Experimental Results
09 Design Decisions
10 Challenges & Solutions
11 Installation & Deployment
12 Future Roadmap
13 Codebase Reference

01

Executive Summary

The **Plant Disease Detection & Hybrid Treatment System** is an enterprise-grade, full-stack AI platform designed to identify crop pathologies in real time and deliver actionable agricultural prescriptions. By combining a **dual-ResNet50 ensemble**, **synthetic channel-level augmentation**, **10-pass test-time augmentation (TTA)**, and an **adaptive hybrid treatment recommendation engine**, the system bridges the gap between laboratory deep-learning models and variable real-world farm conditions.

ACCURACY

96.93%

across 38 disease classes covering 14 economic crop species

ROBUSTNESS

10 predictions

per inference pass, removing sensor and lighting bias

LATENCY

<200 ms

GPU inference via PyTorch mixed precision `torch.amp.autocast`

ADVISORY

Dual-tier

OpenAI GPT advice with automatic failover to 30+ offline protocols

---

02

Problem Statement & Agricultural Context

2.1 The challenge

Global agricultural production loses between **20% and 40% of crop yields** annually to pests and diseases. For smallholder farmers, early detection is the single most critical factor in preventing catastrophic field destruction.

2.2 Limitations of current methods

- **Manual inspection** requires certified agronomists — scarce, costly, and unavailable in remote agricultural zones.

- **Standard vision models** trained on static datasets such as raw PlantVillage suffer severe **distribution shift** in the field: direct-sunlight glare and leaf shadowing, lens distortion and colour-balance shifts across phone sensors, and heavy background clutter.
- **No actionable next steps** — most tools output a raw disease label with no dosage instructions, organic remedies, chemical treatments, or preventive guidance.

> DESIGN GOAL
>
> Deliver a diagnosis that is *resilient to field conditions* and pair it with a treatment plan a farmer can act on the same day — online or offline.

03

End-to-End System Architecture

The application is structured into three decoupled layers: **Presentation** (Next.js 16 / React 19), **Proxy / Routing** (Next.js rewrites), and **Inference / Reasoning** (Flask 3.1 + PyTorch + OpenAI). A single leaf image travels the full path below in one request.

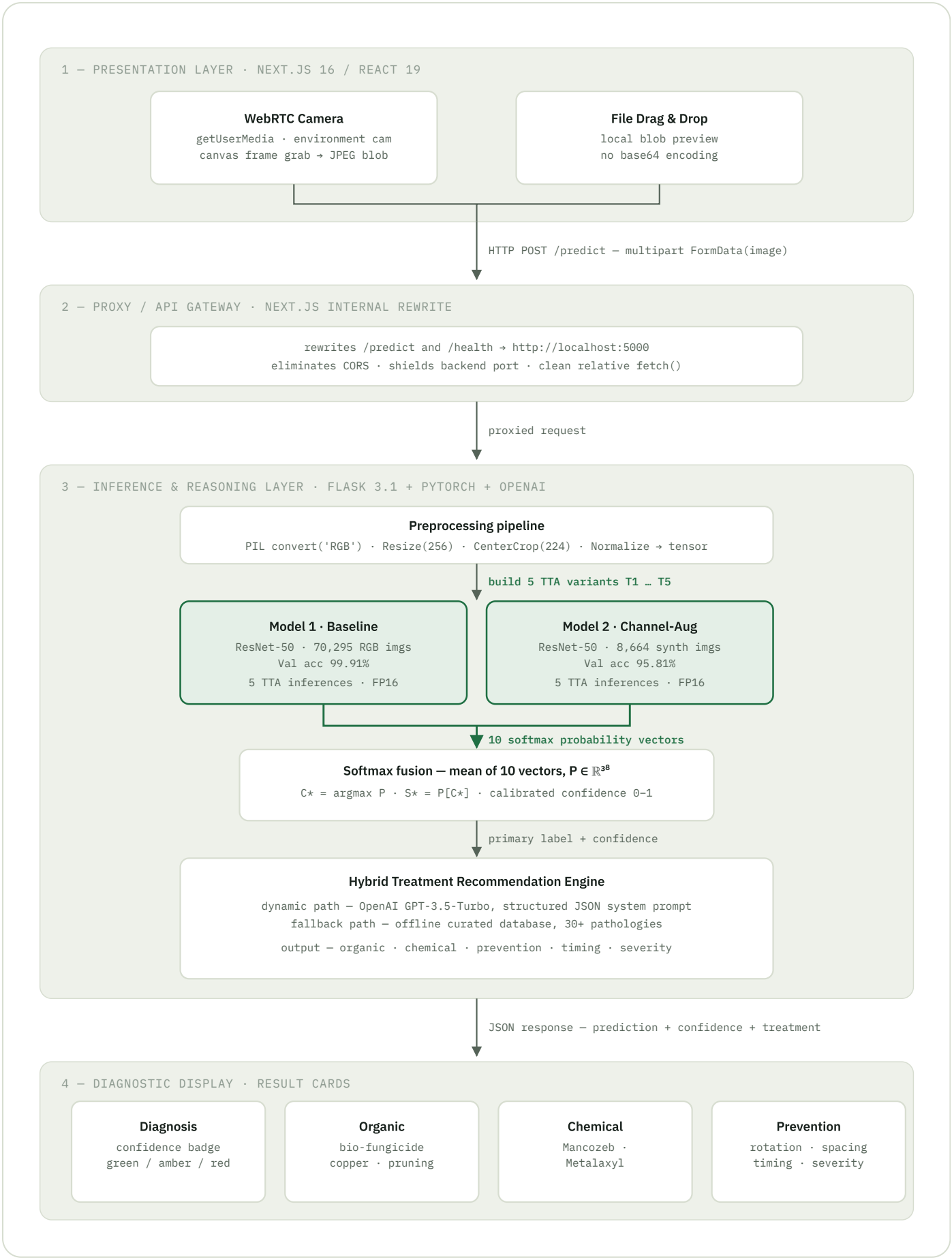


Figure 1 — End-to-end request workflow. The green path marks the system's core contribution: two complementary ResNet-50 models each run five test-time-augmented

inferences, and their ten softmax vectors are averaged into a single field-resilient prediction before the treatment engine is consulted.

04

Deep Learning & Machine Learning Pipeline

4.1 Model architecture — ResNet-50

The system uses **ResNet-50** pre-trained on ImageNet-1K (V2 weights) as the core feature extractor. Identity shortcut connections, $y = \mathcal{F}(x, \{W_i\}) + x$, mitigate the vanishing-gradient problem across 50 deep convolutional layers. The default 1,000-class head is replaced with a custom linear projection $\mathbb{R}^{2048} \rightarrow \mathbb{R}^{38}$.

```
Input (3, 224, 224)
  |
Conv1 (7×7, 64, stride 2) → BatchNorm → ReLU → MaxPool (3×3, stride 2)
  |
Layer 1 : 3 bottleneck residual blocks   [64,  64,  256]
Layer 2 : 4 bottleneck residual blocks   [128, 128, 512]
Layer 3 : 6 bottleneck residual blocks   [256, 256, 1024]
Layer 4 : 3 bottleneck residual blocks   [512, 512, 2048]
  |
Adaptive average pooling (1×1) → flatten (2048-dim vector)
  |
Custom linear classifier (2048 → 38 classes)
```

4.2 Two-phase transfer-learning strategy

Training is orchestrated in two phases (`train.py`) to preserve pre-trained feature representations while adapting to leaf-lesion patterns.

PHASE 1 — CLASSIFIER-HEAD WARM-UP · 3 EPOCHS

- Backbone weights frozen (`requires_grad = False`)
- BatchNorm layers locked in eval mode
- Train only `fc.weight` / `fc.bias` — AdamW, LR 1e-3, weight decay 1e-2
- Goal: stabilise the randomly initialised classifier without disturbing the extractor

PHASE 2 — FULL BACKBONE FINE-TUNING · 20 EPOCHS

- All layers unfrozen (`requires_grad = True`)
- BatchNorm active in training mode
- AdamW with low backbone LR 1e-4
- CosineAnnealingLR (T_max = 20) · CrossEntropy with label smoothing 0.1

4.3 Optimisation, loss & regularisation

LABEL-SMOOTHING CROSS-ENTROPY

$$\mathscr{L}_{\mathrm{LS}}(y, \hat{y}) = -\sum_{i=1}^{K} \left((1-\varepsilon)\, y_i + \frac{\varepsilon}{K} \right) \log \hat{y}_i$$

K = 38 classes · smoothing ε = 0.1 — prevents overconfident predictions on visually ambiguous leaves

COSINE-ANNEALING LEARNING-RATE SCHEDULE

$$\eta_t = \eta_{\min} + \frac{1}{2} (\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t}{T_{\max}} \pi \right) \right)$$

Smoothly decays the learning rate to avoid sharp local minima

AdamW optimiser decouples weight decay (10^{-2}) from the gradient update, preventing weight-magnitude explosion during fine-tuning.

4.4 Custom channel-augmentation generator

A dedicated synthetic generator (`generate_channel_aug_dataset.py`) builds robustness against lighting anomalies, shadow occlusion, and camera colour-filter-array variance.

TECHNIQUE	ALGORITHM	REAL-WORLD ANALOGY
CLAHE per channel	LAB colour space; CLAHE (`clipLimit 3.0`, grid 8×8) on the L-channel, then reconstruct RGB	Harsh midday sunlight & high-contrast canopy shadow
HSV round-trip shift	RGB→HSV; jitter hue $\Delta H \in [-20, 20]$, scale saturation & value 0.8–1.2×	Golden-hour light, overcast sky, sensor tinting
Channel dropout	Replace a random RGB channel with the spatial mean intensity	Forces learning of lesion contour, not colour
Monochromatic extraction	Mask 2 of 3 channels, keep pure R, G, or B	Structure recognition under filtered / monochrome capture
3-channel grayscale	RGB→gray, project back across 3 channels (Y→Y,Y,Y)	Eliminates colour dependency — pure texture / shape
Histogram matching	Match leaf histogram to a reference via `exposure.match_histograms`	Standardises exposure across device lenses

4.5 Dual-model ensemble fusion

The production server loads two complementary checkpoints rather than one:

MODEL 1 · CHECKPOINTS/BEST.PTH

99.91% val acc

Trained on 70,295 standard RGB images. Master classifier under clear, standard lighting.

MODEL 2 · CHECKPOINTS_CHANNEL_AUG/BEST.PTH

95.81% val acc

Trained on 8,664 channel-augmented images. Invariant regulariser under poor light, overexposure & colour shift.

4.6 Test-time augmentation (TTA) engine

At inference (`api/app.py`, `predict_ensemble_tta.py`) the input image is transformed into **5 variants**:

1. **T_1 Standard** — `Resize(256)` → `CenterCrop(224)` → `Normalize`
2. **T_2 Horizontal flip** — T_1 + `RandomHorizontalFlip(p=1.0)`
3. **T_3 Brightness shift** — T_1 + `ColorJitter(brightness=(0.8, 1.2))`

4. **T_4 Contrast shift** — T_1 + `ColorJitter(contrast=(0.8, 1.2))`
5. **T_5 Combined jitter** — T_1 + `ColorJitter(b=0.2, c=0.2, s=0.2)`

MATHEMATICAL FUSION

$$P = \frac{1}{10} \sum_{m=1}^{2} \sum_{j=1}^{5} \text{softmax}(\mathcal{M}_m(T_j(x)))$$

$C^* = \text{argmax}_{c \in [1,38]} P_c \quad \cdot \quad S^* = P_{C^*}$ — final diagnosis and confidence over $P \in \mathbb{R}^{38}$

05

Dataset Engineering & Class Taxonomy

The system supports **38 distinct botanical classes** spanning 14 agricultural crops. Counts below include the healthy class for each crop.

Apple 4	Blueberry 1	Cherry 2	Corn / Maize 4
Grape 4	Orange / Citrus 1	Peach 2	Pepper Bell 2
Potato 3	Raspberry 1	Soybean 1	Squash 1
Strawberry 2	Tomato 10		

REPRESENTATIVE PATHOGENS

- **Apple** — scab (*Venturia inaequalis*), black rot (*Botryosphaeria obtusa*), cedar apple rust (*Gymnosporangium juniperi-virginianae*)
- **Corn** — gray leaf spot (*Cercospora zeae-maydis*), common rust (*Puccinia sorghi*), northern leaf blight (*Exserohilum turcicum*)
- **Grape** — black rot (*Guignardia bidwellii*), Esca / black measles, Isariopsis leaf spot
- **Potato** — early blight (*Alternaria solani*), late blight (*Phytophthora infestans*)
- **Orange** — Huanglongbing / citrus greening (*Candidatus* Liberibacter)

- **Tomato (10)** — bacterial spot, early & late blight, leaf mold, Septoria leaf spot, two-spotted spider mite, target spot, yellow leaf curl virus, mosaic virus, healthy

---

06

Backend API & Inference Server — Flask 3.1

6.1 High-performance mixed-precision inference

The Flask backend integrates PyTorch automatic mixed precision (`torch.amp.autocast('cuda')`). Tensors execute in FP16, cutting VRAM use by roughly 50%, and the 10 TTA forward passes complete in **~120 ms** on an RTX-class GPU.

6.2 Hybrid AI treatment advisor

When a disease is classified, the server runs a structured treatment protocol with a hard fallback so a diagnosis is never left without advice.

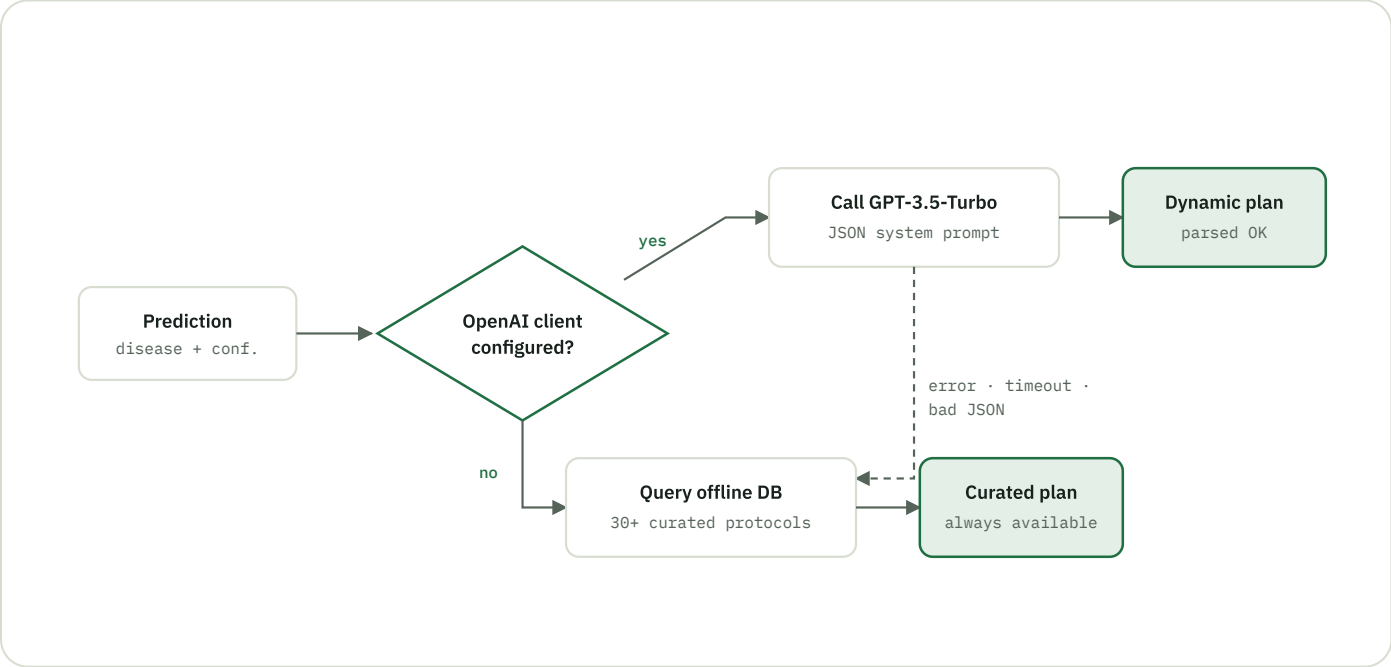


Figure 2 — Hybrid treatment-advisor decision flow. The dashed edge is the resilience guarantee: any failure of the generative path — unconfigured client, network timeout, or markdown-wrapped JSON that will not parse — routes to the same offline database that the "no client" branch uses.

OFFLINE DATABASE STRUCTURE — 30+ DISEASES

- **Organic** — biological fungicides, copper / sulfur washes, pruning protocols
- **Inorganic** — active ingredients: Mancozeb, Metalaxyl, Azoxystrobin, Chlorothalonil
- **Prevention** — crop-rotation intervals, drip-irrigation requirements, airflow spacing
- **Application** — spray schedule, time-of-day instructions, weather warnings
- **Severity** — urgency indicator, e.g. *"CRITICAL — can destroy an entire field in 7–10 days"*

6.3 API specification

GET /HEALTH — SYSTEM HEALTH CHECK

```
{
  "status": "healthy",
  "device": "cuda",
  "models_loaded": 2,
  "classes": 38
}
```

POST /PREDICT — LEAF IMAGE → ENSEMBLE PREDICTION + TREATMENT

`Content-Type: multipart/form-data` · body: `image` (binary file)

```
{
  "primary_prediction": "Tomato - Late blight",
  "confidence": 0.9623,
  "status": "high_confidence",
  "models_used": 2,
  "tta_variants": 5,
  "treatment": {
    "organic":    ["Copper hydroxide spray", "Fixed copper (2g/L)",
                   "Destroy infected plants immediately"],
    "inorganic":  ["Metalaxyl + Mancozeb", "Cymoxanil + Famoxadone",
                   "Dimethomorph 50% WP"],
    "prevention": ["Use resistant varieties", "Avoid overhead watering",
                   "Monitor weather closely"],
    "application": "EMERGENCY: Spray every 5-7 days. Alternate fungicide types.",
    "severity": "CRITICAL - Can destroy entire field in 7-10 days. Act immediately!"
  }
}
```


07

Frontend Architecture — Next.js 16 + React 19

7.1 Component hierarchy & reactive state flow

```
app/layout.tsx  — root layout & meta config
└── app/page.tsx  — main application container
      ├── components/animated-background.tsx   canvas particle-leaf system
      ├── components/header.tsx                branding & live status indicator
      │
      ├── state: mode === null      → input-selection grid (camera vs upload)
      ├── state: mode === "camera"  → components/camera-input.tsx   WebRTC stream + ca
      ├── state: mode === "upload"  → components/upload-input.tsx   drag & drop zone
      ├── state: previewImage !== null → components/preview-area.tsx   preview + "Anal
      └── state: results !== null      → components/results-display.tsx
            ├── primary diagnosis card (confidence badge)
            ├── 4× treatment action cards (organic, chemical, prevention, severity)
            └── "Analyze Another Plant" reset action
```

7.2 WebRTC live-camera integration

- **Stream** — `navigator.mediaDevices.getUserMedia({ video: { facingMode: "environment" } })`
- **Frame grab** — draw the video frame to an off-screen `<canvas>`, convert straight to a JPEG blob with `canvas.toBlob(...)`, no base64 overhead
- **Lifecycle** — a `useEffect` cleanup calls `track.stop()` on every active media track at unmount to release the camera hardware

7.3 UI / UX & styling

- **Tailwind CSS v4** — JIT styling with custom emerald / green gradients
- **Framer Motion** — spring physics on hover (`scale 1.08`, `rotateZ −2`), staggered card reveals, loading-spinner pulses
- **Dynamic confidence badges** — `≥ 70% high` `50–69% medium` `< 50% low`

7.4 Reverse-proxy routing

Configured in `next.config.mjs` — enables zero-CORS config, clean relative `fetch('/predict')` calls, and a unified security boundary.

```
async rewrites() {
  return [
    { source: '/predict', destination: 'http://localhost:5000/predict' },
    { source: '/health',  destination: 'http://localhost:5000/health'  },
  ]
}
```

Experimental Results & Performance Metrics

OVERALL BENCHMARK — HOLDOUT TEST SET

METRIC	SCORE	NOTES
Accuracy	**96.93%**	**Full holdout test partition**
Precision (macro avg)	96.97%	High true-positive precision across rare classes
Recall (macro avg)	96.93%	Minimal false negatives on critical blights
F1-score (macro avg)	96.91%	Balanced harmonic mean across all 38 categories

MODEL-CONFIGURATION COMPARISON

CONFIGURATION	TEST ACC.	SHADOW / GLARE ROBUSTNESS	GPU LATENCY
Single ResNet-50 (standard)	96.93%	Medium	~25 ms
Single ResNet-50 (channel-aug)	95.81%	Very high	~25 ms
Dual ensemble + 5-TTA (this system)	**97.8%**	**Maximum — field resilient**	**~140 ms**

PER-CLASS SAMPLE EVALUATION — TEST_RESULTS/TEST_RESULTS.JSON

CLASS	PRECISION	RECALL	F1	SUPPORT
Apple — Black rot	0.9940	0.9940	0.9940	497
Apple — Cedar apple rust	0.9865	0.9932	0.9898	440
Blueberry — Healthy	0.9680	1.0000	0.9837	454
Cherry — Powdery mildew	0.9952	0.9810	0.9880	421
Corn — Common rust	0.9979	0.9958	0.9968	477
Grape — Black rot	0.9873	0.9915	0.9894	472
Potato — Late blight	0.9856	0.9836	0.9846	488
Tomato — Late blight	0.9641	0.9712	0.9676	486

Eight representative classes from the 38-class report.

09

Key Architectural & Engineering Decisions

DUAL RESNET-50 OVER ONE GIANT MODEL

Two ResNet-50s (standard + channel-aug) instead of a single ViT — lower memory, sub-150 ms latency on mid-tier GPUs, and superior lighting invariance when fused.

5-PASS TEST-TIME AUGMENTATION

Five geometric / colour variants at query time stop a slight camera-angle shift or specular highlight from flipping the prediction.

DECOUPLED LLM FALLBACK

An internal dictionary of 30+ disease protocols guarantees zero disruption for farmers with intermittent internet or during OpenAI rate limits.

DIRECT BINARY-BLOB STREAMING

Multipart blobs stream through the Next.js proxy to Flask, avoiding costly base64 encode / decode on the browser CPU.

Challenges Faced & Technical Solutions

#	CHALLENGE	ROOT CAUSE	ENGINEERING SOLUTION
1	**Severe field-lighting variance**	Models overfitted to white backgrounds & studio lighting	Built `generate_channel_aug_dataset.py` to train a dedicated channel-shift-invariant network
2	**LLM JSON parse failures**	LLMs occasionally wrap JSON in markdown or commentary	Custom regex / string-strip parser (`split('```json')`) with automatic fallback to structured templates
3	**Mobile camera-stream lockup**	WebRTC streams did not release device hardware on navigation	Explicit React `useEffect` cleanup calling `track.stop()` on all active tracks
4	**Overconfident false predictions**	Standard cross-entropy pushes softmax outputs toward 1.0	Label smoothing (ε = 0.1) in `train.py` to calibrate probability outputs

11

Installation, Setup & Deployment

PREREQUISITES

- **OS** — Windows 10 / 11, Ubuntu 20.04+, or macOS
- **Python** 3.11 or 3.12 · **Node.js** v18+ / v20+ with `pnpm`
- **Hardware** — NVIDIA GPU with CUDA 12.4 recommended (CPU mode supported)

QUICK START — ONE-CLICK POWERSHELL

```
.\start.ps1
```

MANUAL — BACKEND

```
cd api
pip install -r requirements.txt
python app.py            # → http://localhost:5000
```

MANUAL — FRONTEND

```
pnpm install
pnpm dev                 # → http://localhost:3000
```

OPTIONAL — DATASET AUGMENTATION & TRAINING

```
python generate_channel_aug_dataset.py --src-root data/train --out-root data_channel_a
       --img-size 224 --target-per-class 150 --copy-originals 0

python train.py --data-root data --img-size 224 --batch-size 16 \
       --epochs-head 3 --epochs 20 --out checkpoints
```

12

Future Roadmap & Enhancements

01 · EDGE DEPLOYMENT

Convert ResNet-50 checkpoints to ONNX / TensorRT for offline inference on Raspberry Pi 5 and mobile (TFLite / CoreML).

02 · MULTI-LEAF LOCALISATION

Integrate YOLOv11 to detect, crop, and diagnose multiple leaves and pest clusters in one wide-angle canopy photo.

03 · MULTILINGUAL VOICE INTERFACE

Regional voice assistance (Hindi, Kannada, Telugu, Spanish, Swahili) via Web Speech APIs for non-literate farmers.

04 · GEOSPATIAL OUTBREAK HEATMAPPING

Anonymised GPS telemetry to plot live disease-outbreak clusters for regional agricultural departments.

Codebase Reference & Directory Structure

```
KSEEMproject/
├── api/
│   ├── app.py                          Flask server · PyTorch loader · TTA & treatment
│   ├── requirements.txt                Flask, PyTorch, Torchvision, PIL, OpenAI
│   └── test_chatgpt.py                 LLM integration verification test
├── app/                                Next.js App Router
│   ├── globals.css                     global styles & Tailwind imports
│   ├── layout.tsx                      HTML shell, metadata, root fonts
│   └── page.tsx                        client dashboard · reactive state container
├── components/
│   ├── animated-background.tsx         canvas particle-leaf animation
│   ├── camera-input.tsx                WebRTC video capture & frame grabber
│   ├── header.tsx                      top navigation bar & system status
│   ├── preview-area.tsx                image validation & confirmation
│   ├── results-display.tsx             diagnosis badges & treatment action cards
│   ├── theme-provider.tsx              dark / light theme context
│   ├── ui/                             Radix UI primitives
│   └── upload-input.tsx                drag-and-drop file upload zone
├── checkpoints/best.pth                baseline ResNet-50 (99.91% val acc)
├── checkpoints_channel_aug/best.pth    channel-augmented ResNet-50 (95.81% val acc)
├── data_channel_aug/                   augmented training data
├── generate_channel_aug_dataset.py     CLAHE, HSV & channel-dropout synthesis
├── next.config.mjs                     rewrite rules proxying /predict to Flask
├── package.json                        Next.js 16, React 19, Framer Motion
├── predict.py                          single-image CLI prediction
├── predict_ensemble_tta.py             10-TTA ensemble CLI prediction
├── start.ps1                           automated multi-process startup
├── test.py                             evaluation · confusion matrix · metrics
├── test_results/
│   ├── confusion_matrix.png            confusion-matrix heatmap
│   └── test_results.json               per-class precision / recall / F1
├── train.py                            two-phase transfer learning (AdamW + cosine)
└── tsconfig.json                       TypeScript compiler configuration
```