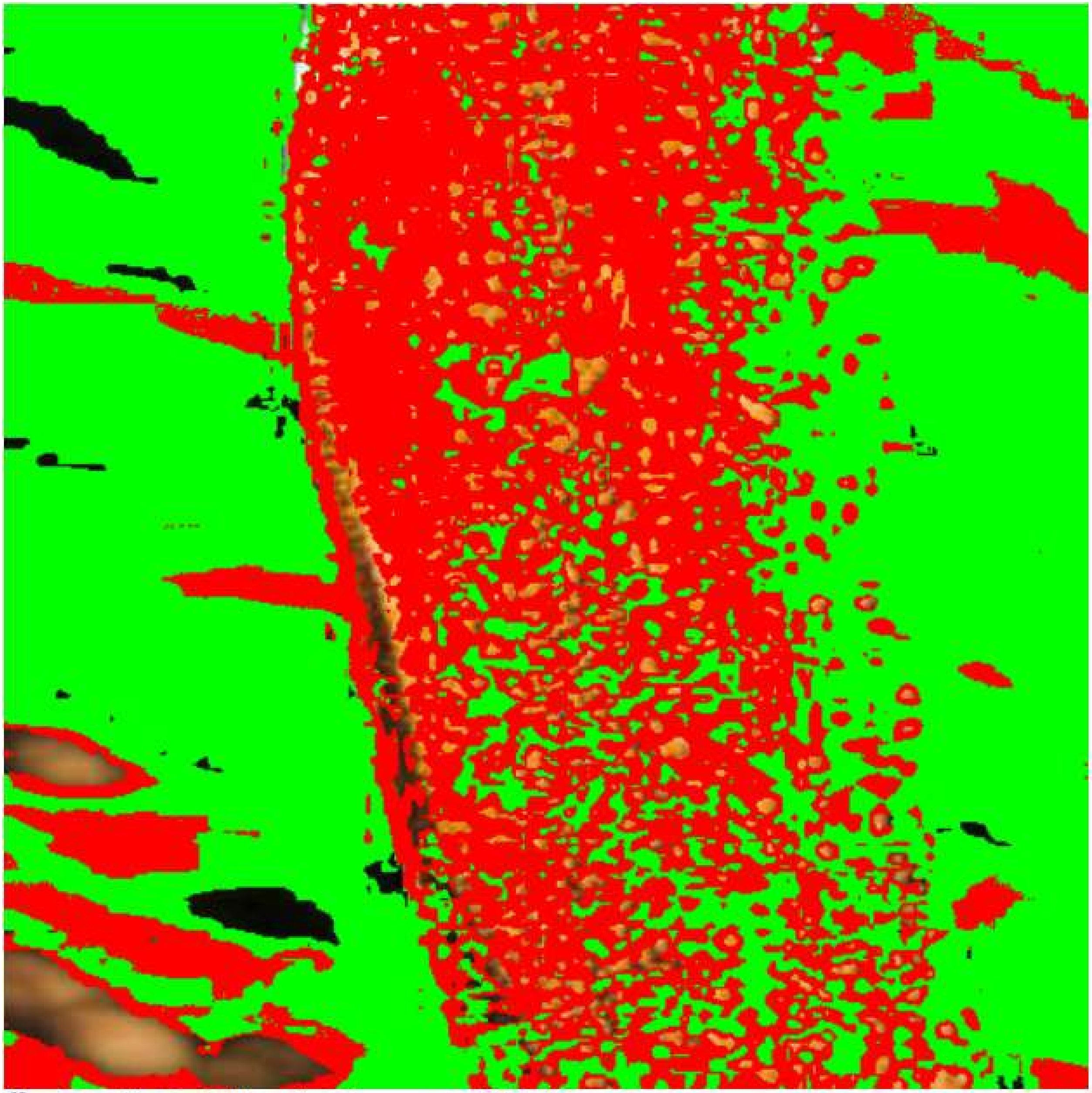


```
🩺 Predicted Wheat Disease: blight
📊 Infection: 43.44% → Moderate Infection
💊 Suggested Pesticide: Mancozeb 2g/L
```

FINAL MODEL CODE DISCRIPTION 👍

CODE ➕

FIRST CONSOLE ;

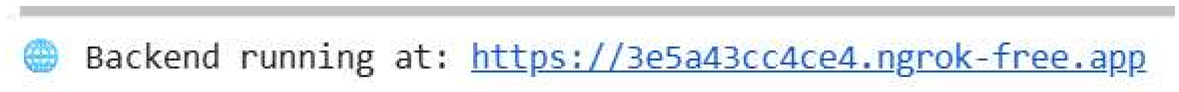
```
!pip install flask pyngrok --quiet

from pyngrok import ngrok
ngrok.set_auth_token("32abAnjhkSAxdxFlB4GuqaAqZ7j_EaHyaAU2n7W8GXit6tpi")
```

SECOND CONSOLE ;

```
from flask import Flask, request, render_template_string
from threading import Thread
from pyngrok import ngrok
import uuid

HTML_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1"/>
<title>Smart Farming Dashboard</title>
<style>
:root{--bg:#f3f4f6;--card:#ffffff;--accent:#16a34a;--muted:#6b7280;}
body{font-family:Inter,system-ui,Segoe
UI,Roboto,Arial;background:var(--bg);margin:0;padding:24px;}
.container{max-width:1100px;margin:0 auto;}
header{display:flex;justify-content:space-between;align-items:center;margi
n-bottom:18px;}
h1{margin:0;font-size:26px;color:#064e3b;}
.sub{color:var(--muted);font-size:14px;}
.grid{display:grid;grid-template-columns:repeat(2,1fr);gap:18px;}
.card{background:var(--card);border-radius:14px;padding:18px;box-shadow:0
6px 18px rgba(15,23,42,0.06);}
.large{font-size:20px;font-weight:700;color:#042a1b;}
.small{color:var(--muted);font-size:13px;}
.percent{font-size:42px;font-weight:800;color:#065f46;}
```

```
.sev{display:inline-block;padding:6px
10px;border-radius:999px;font-weight:700;font-size:13px;}
.sev.healthy{background:#d1fae5;color:#065f46;}
.sev.low{background:#fef9c3;color:#92400e;}
.sev.mod{background:#ffedd5;color:#7c2d12;}
.sev.sev{background:#fecaca;color:#991b1b;}
table{width:100%;border-collapse:collapse;font-size:13px;}
th,td{padding:8px 10px;text-align:left;border-bottom:1px solid #eef2f7;}
.btn{background:var(--accent);color:#fff;padding:8px
12px;border-radius:8px;border:none;cursor:pointer;}
.status-ok{color:#065f46;font-weight:700;}
footer{margin-top:18px;color:var(--muted);font-size:13px;}
@media(max-width:900px){.grid{grid-template-columns:1fr;}}
</style>
</head>
<body>
<div class="container">
  <header>
    <div>
      <h1>🌾 Smart Farming Dashboard</h1>
      <div class="sub">Farmer-friendly view — Disease detection, pesticide
suggestion & sprinkler control</div>
    </div>
    <div>
      <div class="small">Plot: <b>Field #3</b></div>
      <div class="small">Last update: <b>{{time}}</b></div>
    </div>
  </header>

  <div class="grid">
    <!-- Disease Card -->
    <div class="card">
      <div class="large">🦠 Disease Analysis</div>
      <div class="small">Detected leaf preview & results</div>
      <hr style="margin:12px 0;border:none;border-top:1px solid #f1f5f9;">
      <div
style="display:flex;justify-content:space-between;align-items:center;">
        <div>
          <div class="small">Disease</div>
          <div style="font-size:18px;font-weight:800;">{{disease}}</div>
        </div>
        <div style="text-align:right;">
          <div class="small">Infection</div>
          <div class="percent">{{infection}}%</div>
        </div>
      </div>
      <div style="margin-top:12px;">
        <div class="small">Severity</div>
        <div style="margin-top:6px;">
          <span class="sev {{sev_class}}">{{level}}</span>
        </div>
      </div>
    </div>

    <!-- Pesticide Card -->
    <div class="card">
      <div class="large">💊 Pesticide Recommendation</div>
      <div class="small" style="margin-bottom:10px;">Suggested chemical
and dosage</div>
      <div style="display:flex;gap:18px;align-items:center;">
        <div>
          <div class="small">Pesticide</div>
          <div style="font-weight:800;font-size:18px;">{{pesticide}}</div>
        </div>
        <div>
          <div class="small">Dosage</div>
          <div style="font-weight:800;font-size:18px;">{{dosage}}</div>
        </div>
        <div>
          <div class="small">Estimated Spray</div>
          <div
style="font-weight:800;font-size:18px;">{{spray_time}}</div>
        </div>
      </div>
      <div style="margin-top:14px;">
        <button class="btn" onclick="alert('Spray command sent
(demo)')">Send Spray Command</button>
      </div>
    </div>
    <!-- Sprinkler Status -->
    <div class="card">
      <div class="large">🚜 Sprinkler Status</div>
      <div class="small">Controller connection & recent action</div>
      <hr style="margin:12px 0;border:none;border-top:1px solid #f1f5f9;">
      <div
style="display:flex;justify-content:space-between;align-items:center;">
        <div>
          <div class="small">Controller</div>
          <div class="status-ok">Arduino (USB) — Connected</div>
        </div>
        <div>
          <div class="small">Last Action</div>
          <div>3s spray • <span class="status-ok">Success</span></div>
        </div>
      </div>
      <div style="margin-top:14px;">
        <div class="small">Manual Control</div>
        <div style="margin-top:8px;">
          <button class="btn" onclick="alert('Manual ON (demo)')">Manual
ON</button>
          <button style="margin-left:8px;padding:8px
12px;border-radius:8px;" onclick="alert('Manual OFF (demo)')">Manual
OFF</button>
        </div>
      </div>
    </div>

    <!-- Logs -->
    <div class="card">
      <div class="large">📜 Recent Logs</div>
      <div class="small" style="margin-bottom:10px;">Last detection &
spray history</div>
      <table>
        <thead>
          <tr>
            <th>Time</th><th>Disease</th><th>Infection</th><th>Action</th>
          </tr>
        </thead>
        <tbody>
```

```
<tr><td>{{time}}</td><td>{{disease}}</td><td>{{infection}}%</td><td>{{spra
y_time}} Spray</td></tr>
        </tbody>
      </table>
    </div>
  </div>

  <footer>
    Tip: Use the "Send Spray Command" button only after verifying the
pesticide mix. Always wear PPE.
  </footer>
</div>
</body>
</html>
"""

app = Flask(__name__)
results = {}

@app.route("/save", methods=["POST"])
def save():
    data = request.json
    rid = str(uuid.uuid4())[:8]
    results[rid] = data
    return {"link": f"{public_url}/result/{rid}"}

@app.route("/result/<rid>")
def result(rid):
    data = results.get(rid)
    if not data: return "Result not found", 404

    sev_class = "healthy"
    if "Low" in data["level"]: sev_class = "low"
    elif "Moderate" in data["level"]: sev_class = "mod"
    elif "Severe" in data["level"]: sev_class = "sev"

    html = HTML_TEMPLATE
    for k,v in data.items():
        html = html.replace(f"{{{{{k}}}}}", str(v))
    html = html.replace("{{sev_class}}", sev_class)

    return render_template_string(html)

def run():
    app.run(port=5000)

public_url = ngrok.connect(5000).public_url
print("🌐 Backend running at:", public_url)

Thread(target=run, daemon=True).start()
```

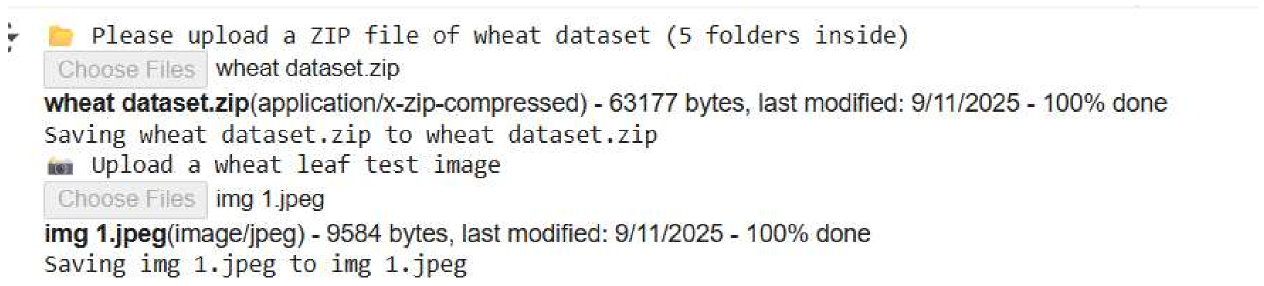


```
🌐 Backend running at: https://3e5a43cc4ce4.ngrok-free.app
```

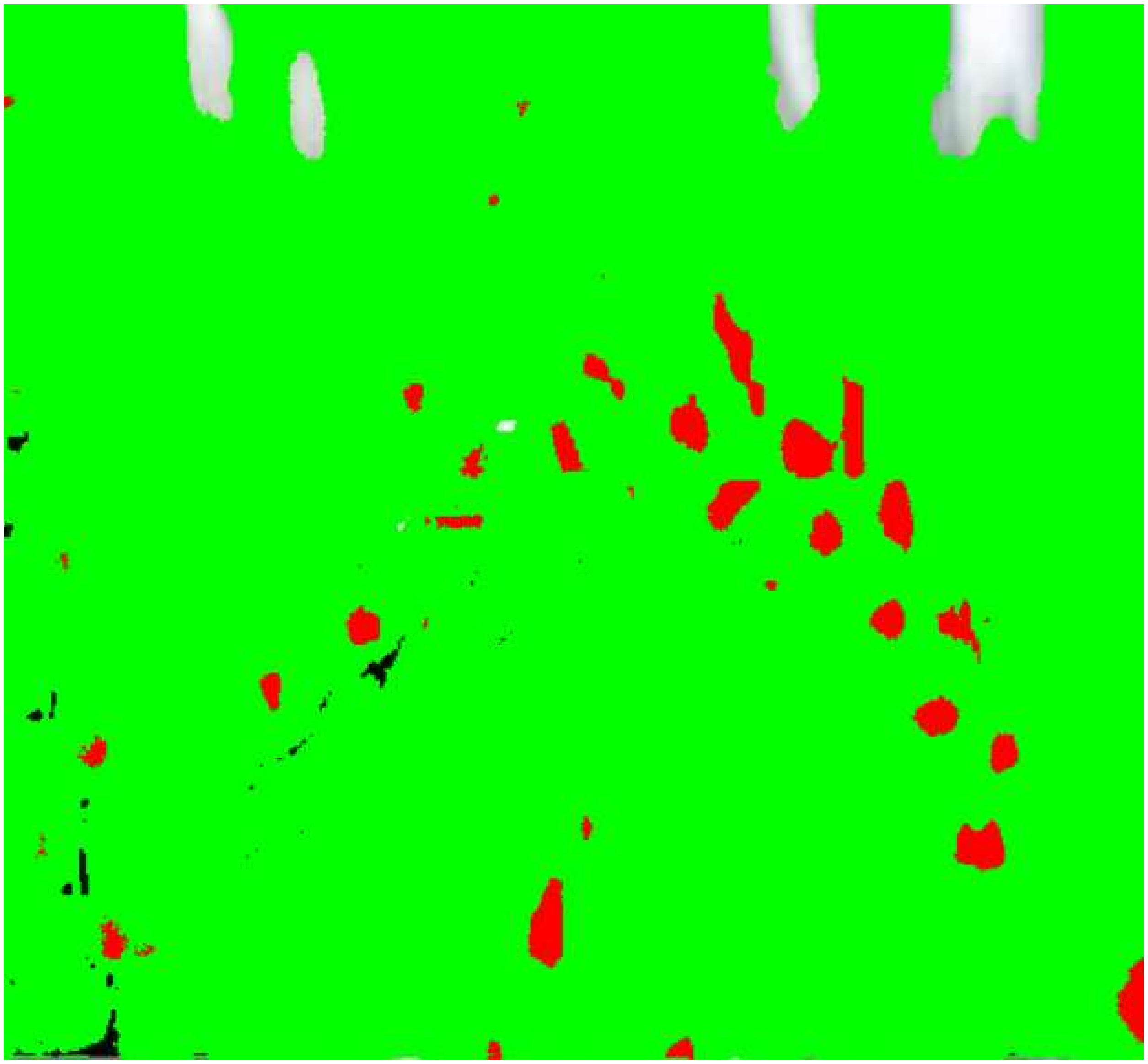
THIRD CONSOLE 🎉

```
import cv2, numpy as np, os, zipfile, requests
from google.colab import files
from google.colab.patches import cv2_imshow

# Upload Wheat Dataset
print("📁 Please upload a ZIP file of wheat dataset (5 folders inside)")
uploaded = files.upload()
zip_path = list(uploaded.keys())[0]
dataset_dir = "/content/wheat_dataset"
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(dataset_dir)

subfolders = os.listdir(dataset_dir)
if len(subfolders) == 1 and os.path.isdir(os.path.join(dataset_dir,
subfolders[0])):
    dataset_dir = os.path.join(dataset_dir, subfolders[0])

# Upload Test Image
print("📷 Upload a wheat leaf test image")
uploaded_test = files.upload()
test_img_path = list(uploaded_test.keys())[0]
```

```python
# Infection Detection
def detect_infection(img):
    img = cv2.resize(img, (500, 500))
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_leaf = np.array([20, 30, 20])
    upper_leaf = np.array([100, 255, 255])
    leaf_mask = cv2.inRange(hsv, lower_leaf, upper_leaf)
    lower_infect = np.array([10, 40, 40])
    upper_infect = np.array([35, 255, 255])
    infect_mask = cv2.inRange(hsv, lower_infect, upper_infect)
    infect_mask = cv2.bitwise_and(infect_mask, leaf_mask)
    total = cv2.countNonZero(leaf_mask)
    infected = cv2.countNonZero(infect_mask)
    infection_percent = (infected / total * 100) if total else 0
    result = img.copy()
    result[leaf_mask > 0] = [0, 255, 0]
    result[infect_mask > 0] = [0, 0, 255]
    return result, infection_percent

# Disease Prediction
def predict_disease(test_img_path, dataset_dir):
    test_img = cv2.imread(test_img_path)
    test_img = cv2.resize(test_img, (500, 500))
    orb = cv2.ORB_create(nfeatures=1000)
    kp1, des1 = orb.detectAndCompute(test_img, None)
    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    best_score = -1
    predicted_disease = None

    for disease in os.listdir(dataset_dir):
        disease_path = os.path.join(dataset_dir, disease)
        if not os.path.isdir(disease_path): continue
        for img_file in os.listdir(disease_path)[:5]:
            ref_img = cv2.imread(os.path.join(disease_path, img_file))
            if ref_img is None: continue
            ref_img = cv2.resize(ref_img, (500, 500))
            kp2, des2 = orb.detectAndCompute(ref_img, None)
            if des1 is None or des2 is None: continue
            matches = bf.match(des1, des2)
            score = len(matches)
            if score > best_score:
                best_score = score
                predicted_disease = disease

    if predicted_disease is None:
        predicted_disease = "Healthy"

    result_img, infection_percent =
detect_infection(cv2.imread(test_img_path))

    if infection_percent < 5:
        level = "Healthy"
    elif infection_percent < 20:
        level = "Low Infection"
    elif infection_percent < 50:
        level = "Moderate Infection"
    else:
        level = "Severe Infection"

    pesticide = "No pesticide required"
    if "rust" in predicted_disease.lower():
        pesticide = "Propiconazole 1ml/L"
    elif "blight" in predicted_disease.lower():
        pesticide = "Mancozeb 2g/L"
    elif "mildew" in predicted_disease.lower():
        pesticide = "Sulphur 3g/L"
    elif "karnal" in predicted_disease.lower():
        pesticide = "Carbendazim 1g/L"

    cv2_imshow(result_img)

    print(f"🩺 Predicted Wheat Disease: {predicted_disease}")
    print(f"📊 Infection: {infection_percent:.2f}% → {level}")
    print(f"💊 Suggested Pesticide: {pesticide}")

    # Send result to backend
    data = {
        "disease": predicted_disease,
        "infection": round(infection_percent,2),
            "level": level,
            "pesticide": pesticide
        }
        r = requests.post(public_url + "/save", json=data)
        print("🔗 Result Link:", r.json()["link"])

predict_disease(test_img_path, dataset_dir)
```



```
📁 Please upload a ZIP file of wheat dataset (5 folders inside)
Choose Files  wheat dataset.zip
wheat dataset.zip(application/x-zip-compressed) - 63177 bytes, last modified: 9/11/2025 - 100% done
Saving wheat dataset.zip to wheat dataset.zip
📷 Upload a wheat leaf test image
Choose Files  img 1.jpeg
img 1.jpeg(image/jpeg) - 9584 bytes, last modified: 9/11/2025 - 100% done
Saving img 1.jpeg to img 1.jpeg
```



```
INFO:werkzeug:127.0.0.1 - - [18/Sep/2025 05:22:57] "POST /save HTTP/1.1" 200
🩺 Predicted Wheat Disease: healthy
📊 Infection: 2.51% → Healthy
💊 Suggested Pesticide: No pesticide required
🔗 Result Link: https://3e5a43cc4ce4.ngrok-free.app/result/ed056247
```