PolePad AI (Team-Friendly MVP Plan)

Goal: Turn messy pole photos into **structured, trustworthy infrastructure data** using AI + a crowd validation loop.

What we’re building (in one sentence)

A system where users upload pole photos → AI extracts the **pole ID** + **pole condition attributes** → people **confirm/dispute** → the system builds **confidence** → then we output a **risk/prediction** at the end.

---

1) What the app does (simple flow)

Step-by-step loop (this is the “working intelligence loop”)

1. **User uploads a photo** (pole or pad-mounted asset).
2. **AI finds and reads the asset tag** (ID).
3. **AI analyzes the full image** to detect key infrastructure attributes (vegetation, safety gear, pole type, etc.).
4. The app shows a **structured result** (like a mini inspection report).
5. Other users can **confirm / dispute** the AI output.
6. The system computes a **confidence score** based on agreement.
7. The system runs a simple **prediction/risk score** using the collected data.

That’s exactly what the prompt calls “Computer Vision + Network Intelligence.”

---

2) Two AI jobs we do at the same time

To avoid confusion: we are not doing “one model.” We’re doing **two tasks** from the same photo.

A) Asset Tag Detection + OCR (Identity)

Purpose: Identify “which asset is this?”

- Use **YOLOv8** to locate the tag region (box around the tag)
- Use **OpenCV** to clean that crop (reduce glare, rotate/deskew, increase contrast)
- Use **EasyOCR** to extract the ID text
- **Normalize** the ID (strip spaces, fix common errors like O/0, S/5)

Output example:

- `pole_id = "625296"` or `asset_id = "PD41459"`

B) Infrastructure Attribute Detection (Context)

Purpose: Determine “what’s going on with the pole?”
Run detection on the **full image** (not the cropped tag).

MVP attributes to detect (keep it realistic):

- **Pole type**: wood vs composite/metal (basic)
- **Vegetation encroachment**: yes/no
- **Safety equipment presence**: guy guard yes/no (or “unknown” if unclear)
- **Visible attachments**: transformer/hardware yes/no (optional)
- **Structural anomaly flag**: “possible damage” yes/no (optional)

Output example:

- `pole_type = wood`

- `vegetation = true`
- `guy_guard = false`
- `hardware_present = true`

---

3) Structured Output (what we store)

Every upload becomes a **structured record**:

```
{
 "asset_id": "625296",
 "ai_read": {
   "pole_type": "wood",
   "vegetation": true,
   "guy_guard": false,
   "hardware_present": true
 },
 "ai_confidence": {
   "tag_conf": 0.78,
   "veg_conf": 0.80,
   "guard_conf": 0.55
 },
 "timestamp": "..."
}
```

This matches the requirement: “turn images into structured, verifiable infrastructure data.”

4) Mesh / Social Validation Layer (our “original” part)

This is how we satisfy the “crowd-powered / mesh-style” requirement without building a full social network.

In the UI, after AI results show:

Users can:

- ✅ Confirm tag number (or correct it)
- ✅ Confirm/dispute vegetation
- ✅ Confirm/dispute guy guard
- 🚩 Flag “possible issue” (free-text note)
- 📷 Upload another image for the same asset

Consensus/Confidence scoring (simple but powerful)

For each asset_id, we store multiple reports and compute agreement:

- If 3 people confirm `625296` → tag confidence ↑
- If 2 say vegetation is present and 1 says not → show “dispute” and set lower confidence

This satisfies:

- “Community confirmation mechanism”
- “Confidence scoring based on consensus”
- “Network improves reliability over time”

5) AI as augmentation (how we show it)

We explicitly show:

- AI **prefills** data
- AI **highlights uncertain characters**
    - Example: `62529?` where the last digit is low-confidence
- AI provides **confidence metrics**
- Humans fix/confirm faster than manual inspections

This matches the guideline: “AI supports the network — does not operate in isolation.”

---

6) Prediction at the end (simple + pitch-friendly)

Once we have attributes + consensus confidence, we run a lightweight prediction.

MVP prediction options (pick 1)

Option A (easiest): Rule-based “Risk Score”

- Missing guy guard → + high risk
- Vegetation present → + medium risk
- Low consensus confidence → + inspection priority

Outputs:

- `risk_level = LOW / MED / HIGH`
- `inspection_priority = Soon / Normal`

Option B (better): XGBoost model (still free)
Inputs:

- vegetation (0/1)
- guy_guard (0/1)
- pole_type encoded
- number_of_reports
- consensus_confidence

Output:

- `failure_risk_probability` (0–1)

Even if trained on “synthetic” data, it demonstrates the concept clearly.

---

7) What we will demo (judge-ready)

We should demo these moments:

1. Upload a pole photo where the tag is small → YOLO finds it
2. OCR reads the tag → we show confidence + allow manual edit
3. Full-image detection shows vegetation and missing guy guard
4. Another teammate “confirms” the result → confidence increases
5. The system outputs a final risk score / inspection priority

That proves:
✅ ingestion + analysis
✅ tag extraction
✅ structured output
✅ mesh validation

✅ network confidence improvement
✅ predictive output

---

8) Team task split (so everyone knows what to do)

- **Person A: YOLO Tag Detector** (tag box + crop)
- **Person B: OCR Pipeline** (OpenCV preprocess + EasyOCR + normalize + confidence)
- **Person C: Context Detector** (vegetation/guy guard/pole type detection on full image)
- **Person D: Mesh Layer + Storage** (store reports, confirmations, consensus scoring)
- **Person E: Streamlit UI + Demo Script** (upload → results → confirm/dispute → score)
- *(Optional)* **Person F: Prediction** (rule-based first, XGBoost if time)

---

9) One-liner pitch for your teammates to remember

"PolePad AI turns messy inspection photos into a living asset registry: AI extracts ID + condition, people verify, confidence grows, then we predict risk and prioritize inspections."