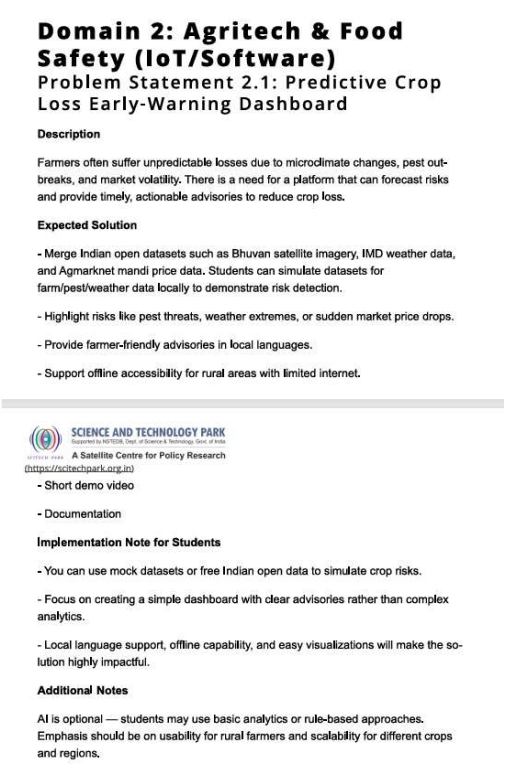
✅ Help you draft your **video script**
✅ Help you prepare your **technical summary**
✅ Help you create your **problem-to-solution mapping**
✅ Review your prototype plan

Just tell me!



Domain 2: Agritech & Food Safety (IoT/Software)

Problem Statement 2.1: Predictive Crop Loss Early-Warning Dashboard

Description

Farmers often suffer unpredictable losses due to microclimate changes, pest outbreaks, and market volatility. There is a need for a platform that can forecast risks and provide timely, actionable advisories to reduce crop loss.

Expected Solution

- Merge Indian open datasets such as Bhuvan satellite imagery, IMD weather data, and Agmarknet mandi price data. Students can simulate datasets for farm/pest/weather data locally to demonstrate risk detection.
- Highlight risks like pest threats, weather extremes, or sudden market price drops.
- Provide farmer-friendly advisories in local languages.
- Support offline accessibility for rural areas with limited internet.

SCIENCE AND TECHNOLOGY PARK
A Satellite Centre for Policy Research
(https://scitechpark.org.in)

- Short demo video
- Documentation

Implementation Note for Students

- You can use mock datasets or free Indian open data to simulate crop risks.
- Focus on creating a simple dashboard with clear advisories rather than complex analytics.
- Local language support, offline capability, and easy visualizations will make the solution highly impactful.

Additional Notes

AI is optional — students may use basic analytics or rule-based approaches. Emphasis should be on usability for rural farmers and scalability for different crops and regions.

KrishiShield_PICT_ PST_a6...
Presentation

This is my topic so give me step by step roadmap for it how to make ?

Thought for 10s >

Great — this is a solid, realistic project choice. Below I've turned your topic into a **practical, step-by-step roadmap** you can follow to build the Predictive Crop Loss Early-Warning Dashboard (KrishiShield). I'll list **ordered implementation steps**, what to build for a Round-

2 prototype/demonstration, technical recommendations, example rule logic, testing & evaluation items, and final submission checklist tied to the hackathon guidelines. (I used your KrishiShield pitch and the Round-2 submission docs to shape this — see notes inline.)

KrishiShield_PICT_ PST_a6fb4321... Prototype_Submission_Guidelines Frequently Asked Questions

---

1 → High-level goal (what your prototype must show)

Make a **simple, robust demo** that:

- Ingests open Indian data + a small local dataset (real or simulated). KrishiShield_PICT_ PST_a6fb4321...
- Runs a **lightweight risk engine** (rule-based ± simple ML) that outputs 3–5 day risk flags (weather, pest, market). KrishiShield_PICT_ PST_a6fb4321...
- Displays farmer-friendly advisories in local language(s) on a **PWA** that works offline / low bandwidth. KrishiShield_PICT_ PST_a6fb4321...
- Is recorded in a 2–5 minute horizontal MP4 demo video plus a code repo and short tech summary for submission. Prototype_Submission_Guidelines Frequently Asked Questions

---

2 → Minimum Viable Prototype (MVP) — required features (priority order)

1. **Data connector / ETL** — fetch / load:
   - IMD forecast (temperature, rainfall) or simulated weather CSV. KrishiShield_PICT_ PST_a6fb4321...
   - Agmarknet daily mandi price for 1–2 commodities (or simulate). KrishiShield_PICT_ PST_a6fb4321...
   - Optional: Bhuvan NDVI/soil layer tiles (use one static layer to demonstrate spatial logic). KrishiShield_PICT_ PST_a6fb4321...
2. **Risk Engine (core)**:
   - Rule-based rules for weather/pest/price risk (see section 6 for concrete examples).
   - Output: Risk score (0–100) + risk type + recommended advisory.
3. **Frontend PWA**:

- Dashboard with a) Location selector, b) “Risk cards” (Weather, Pest, Market), c) Clear textual advisory and suggested farmer actions in local language(s).
- Offline bundle (basic data + advisories) and demo mode.

4. **Repository + README**:
   - Code, sample datasets, and README explaining how to run demo locally.
5. **Demo video (2–5 mins, landscape MP4)** and a **2-page technical summary (optional)** for submission. Prototype_Submission_Guidelines

---

3 → Step-by-step technical roadmap (ordered tasks you can follow)

Step A — Plan & gather

- Pick **1–2 target crops** and **1-2 districts/locations** (keeps scope small).
- Collect or create small sample datasets:
  - Weather: daily forecast (temp, rainfall, humidity) for 7 days.
  - Market: daily mandi price history (last 30 days) for the crop.
  - (Optional) Simulated pest incidence signals or build simple pest probability from weather + NDVI.
- NOTE: You can simulate realistic data if you can’t call live APIs — acceptable for prototype. Frequently Asked Questions

Step B — Build ETL & data model

- Make simple Python scripts (or Node) to:
  - Parse CSV / API JSON into a normalized table: `location, date, temp_max, temp_min, rainfall_mm, humidity, ndvi, price`.
  - Save daily aggregates to a small SQLite or JSON file for demo.
- Keep schema simple; example:

```scss
weather(date, location_id, tmax, tmin, rainfall_mm, humidity)
price(date, location_id, commodity, mandi_price)
ndvi(date, location_id, ndvi)
```

- Add a cron / script to simulate “next 3–5 day forecast” entries if using mock data.

Step C — Implement Risk Engine (rule-based core)

- Make an engine that consumes the data model and outputs:

```yaml
{ location, date, risk_type: [weather|pest|market], score: 0-100, advisory_te
```

- Implement simple, transparent rules (examples next). Log rule triggers for explainability.

Step D — Frontend PWA (user-facing)

- Tech: React (create-react-app) or lightweight Vue + service worker for offline. Build as PWA (manifest + service worker).
- UI components:
  - Risk cards with icon, color (green/yellow/red), short advisory, "Why" and "What to do" buttons.
  - Language toggle (Marathi / Hindi / English). Store translations in JSON.
  - Offline mode: store last synced data in localStorage / IndexedDB; UI should show "demo/offline" banner when offline.
- Keep visuals simple: big text, high contrast, one chart or sparkline per card.

Step E — Backend (lightweight)

- Optional: small Flask / FastAPI backend to serve ETL results; or run risk engine locally in frontend for prototype (client-side JS reads JSON outputs). For hackathon, a client-side approach is simplest for demo.

Step F — Translate advisories

- Write short, action-oriented advisories per risk and crop (examples below). Keep 1–2 sentences per advisory, translated into target languages. Store translations as key:value.

Step G — Testing & evaluation

- Prepare test cases: dry run weather extremes, pest increase scenarios, price crash scenario — verify the engine flags and advisories.
- Create a short checklist of expected outputs per test case.

Step H — Prepare demo materials

- Record a clean horizontal MP4 (2–3 mins recommended): show team intro (≤30s), live demo (show data->engine->dashboard), end with impact & next steps. Test link accessibility per submission rules. Prototype_Submission_Guidelines Frequently Asked Questions

---

4 → Concrete example rules (risk engine) — start here (no heavy ML required)

Use rule thresholds and simple score aggregation.

Weather risk (heat/drought/flood)

- If `rainfall_next3days >= 50 mm` → raise Flood risk. Score = min(100, (rainfall_mm / 50) * 100).
- If `temp_max_next3days >= crop_heat_threshold` → Heat stress. Score proportional to degrees above threshold.
- If `rainfall_last14days < crop_min_rainfall` → Drought risk.

Pest risk (rule from weather + crop stage)

- Certain pests increase after X days of humidity > 80% and temps 20–30°C. Example:
  - If `avg_humidity_next5days > 80% && avg_temp_next5days between 20 and 30 → pest_risk += 50`.
- Use NDVI drop as signal: `if NDVI decreased > 10% over 7 days → stress → higher pest susceptibility`.

Market risk

- If `price_drop_7day_pct <= -10%` → price crash risk. Score = abs(drop_pct) capped at 100.
- If mandi price volatility (std dev / mean over 14 days) > threshold → high volatility alert.

Final risk scoring

- Combine multi-source signals: e.g. `final_score = max(weather_score, pest_score, market_score)` then map to color bands:
  - 0–30 green, 31–60 yellow, 61–100 red.
- Attach an advisory template keyed by `risk_type` and `crop`.

---

5 → Farmer advisories (short actionable templates)

Keep it short and local-language friendly. Example for a tomato crop:

- Flood risk (red): “Delay harvest where possible; move harvested crates to high, dry place; avoid applying fungicide immediately after heavy rain.”
- Pest risk (yellow): “Inspect lower leaf canopy; apply recommended biopesticide at earliest sign; avoid irrigation at dusk.”
- Price drop (yellow): “Consider delaying sale by 3–5 days if storage allows; consult local FPO for pooled sale.”

Store these as small JSON objects for translation and dynamic insertion.

---

6 → Tech stack suggestions (minimal, easy to demo)

- ETL & risk engine: **Python** (Pandas, small scripts) or **Node.js**
- Backend API (optional): **FastAPI** or **Flask**
- Frontend PWA: **React** (Vite/CRA) or **Vue**; use service worker + IndexedDB for offline.
- Storage: small **SQLite** or JSON files for demo.
- Deployment: GitHub repo + host demo with Vercel / Netlify (optional) or run locally for video.
  These choices help keep the prototype lightweight and easy to package for submission. KrishiShield_PICT_ PST_a6fb4321...

---

7 → What to include in your submission pack (per guidelines)

- **Demo Video** (2–3 mins ideal; up to 5 mins). Horizontal MP4. Prototype_Submission_Guidelines
- **GitHub repo link** inside a small `.txt` file and include the repo in the ZIP. README must show how to run the demo. Frequently Asked Questions
- **Optional 1–2 page technical summary** (tech stack, architecture diagram, unique rules). Prototype_Submission_Guidelines
- Name your ZIP following the convention (TeamName_CollegeName_Round2 or TeamName_ProblemID_SciTech2025) and ensure links are viewable.
  Prototype_Submission_Guidelines

---

8 → Demo script outline (what to record in the 2–3 min video)

1. 0:00–0:30 — Team name, project title, and problem statement (brief). Prototype_Submission_Guidelines
2. 0:30–1:30 — Quick show of data sources & where data comes from (or simulated data). KrishiShield_PICT_ PST_a6fb4321...
3. 1:30–2:30 — Live demo: pick a location, show risk cards, click to see advisory and explanation ("why this was flagged").
4. 2:30–3:00 — Impact summary + next steps & contact.
   (Keep narration concise; voice-over is recommended.) Frequently Asked Questions

---

9 → Testing, evaluation & simple metrics

- **Functional tests**: For each test case (heat wave, heavy rain, price crash), verify engine flags expected risk.
- **Accuracy (if you use ML)**: track precision/recall on labelled sample (optional).
- **Usability**: show 2–3 user flows in demo — check that advisories are understandable in translation.
- **Performance**: PWA should load offline demo bundle under poor connectivity (demonstrate in video).

---

10 → Next steps & stretch features (if time permits)

- Add a simple lightweight ML model (e.g., logistic regression) trained on simulated historical signals to refine pest risk probability.
- Add SMS/IVR push alerts for farmers without smartphones.
- Integrate with local extension worker dashboard or FPO portal.
- Add multi-crop profiles and seasonal tuning.

---

11 → Quick checklist you can tick off now

- ☐ Choose 1–2 crops and locations.
- ☐ Prepare / simulate weather, price, and optional NDVI/pest data.

- ☐ Implement ETL to normalized JSON/SQLite.
- ☐ Build rule-based risk engine; log decisions.
- ☐ Build PWA showing Risk Cards + translated advisories + offline demo.
- ☐ Record horizontal MP4 demo (≤5 min) and make repo + README.

Prototype_Submission_Guidelines Frequently Asked Questions

If you want, I can right now:

- write a **starter risk-engine Python script** (rule-based) using the example rules above, or
- draft the **exact JSON schema** for advisories and translations, or
- write a **short 2-page technical summary template** you can include in the submission.

Tell me which of those you want me to produce immediately and I'll generate it. ✅