PestCast — Product Requirements Document

"Waze for agricultural pests — crowdsourced, real-time, climate-correlated."

---

1. Problem Statement

The Core Problem

Agricultural pest outbreaks cause $200B+ in annual crop losses globally. Climate change is accelerating pest migration — species are invading new regions faster than any institution can track. The fall armyworm alone spread from the Americas to all of sub-Saharan Africa in under 3 years, causing $13B in annual damage.

Why Existing Solutions Fail

- **USDA/Extension services** publish pest reports weekly or biweekly. Pest fronts move daily. The latency is fatal.
- **Satellite remote sensing** can detect crop damage after it's happened but can't identify pest species or provide early warning.
- **Commercial scouting services** cost $5–15/acre, pricing out smallholder farmers entirely.
- **Existing pest ID apps** (Plantix, Picture Insect) identify pests but do nothing with the data — no aggregation, no prediction, no alerts. They're individual tools, not a network.

The Gap

There is no system that combines real-time pest identification, crowdsourced geographic intelligence, and climate-correlated predictive alerts. Farmers have no way to know what's coming toward their fields until it arrives.

---

2. Product Vision

PestCast turns every farmer's smartphone into a node in a distributed pest surveillance network. Farmers snap a photo → get instant pest ID (offline) → the sighting feeds a live crowdsourced heatmap → climate data projects where the pest front is heading → neighboring farms get early warning alerts hours to days before infestation.

Success Metrics (Post-Hackathon / Theoretical)

Metric	Target
On-device inference latency	< 500ms on mid-range Android
Classification accuracy (top-3)	> 85% across target species
Time-to-alert for neighboring farms	< 6 hours from first cluster detection
Offline functionality	Full ID capability with 0 connectivity

Hackathon Demo Success Criteria

- Live on-device pest classification on a physical phone
- Sightings appear on heatmap in real-time
- Climate overlay visualizes wind/temp data
- Prediction zone renders on map showing projected pest movement
- End-to-end flow runs in under 60 seconds during demo

---

3. User Personas

Primary: Smallholder Farmer (Maria)

- Farms 15 acres of corn and vegetables in California's Central Valley
- Uses an Android phone (mid-range, $150–300)
- Intermittent cell service in fields
- Checks weather apps daily, moderately tech-comfortable
- **Need:** "I lost 30% of my corn to armyworm last season. My neighbor saw them a week before me — I wish I'd known."

Secondary: Agricultural Extension Agent (David)

- Covers a 5-county region, responsible for pest advisories
- Currently drives to fields manually to scout
- **Need:** "I'm one person covering 500 farms. I need real-time eyes across my whole region."

Tertiary: Crop Insurance Analyst (Sarah)

- Assesses pest-related claims across a portfolio
- **Need:** "I need spatiotemporal pest pressure data to price risk accurately."

---

4. Feature Requirements

P0 — Must Have for Demo (Hackathon Scope)

F1: On-Device Pest Classification

- **Description:** Camera capture → instant pest species identification running entirely on-phone
- **Input:** Photo from device camera or gallery
- **Output:** Top-3 species predictions with confidence scores, common name, severity indicator
- **Model:** MobileNetV3 or EfficientNet-Lite, quantized to INT8 via TFLite
- **Training data:** IP102 dataset (75,000+ images, 102 species) — fine-tune on 15–20 priority species for demo
- **Offline:** Full functionality with zero connectivity
- **Performance:** < 500ms inference on Snapdragon 600-series equivalent

F2: Sighting Reporting & Sync

- **Description:** Each classification generates a sighting report: species, confidence, GPS, timestamp, optional severity tag
- **Offline queue:** Sightings stored locally and batch-synced when connectivity returns
- **Backend:** Supabase (Postgres + Realtime subscriptions)
- **Schema:**

```
sightings {
  id: uuid
  species: string
  confidence: float
  latitude: float
  longitude: float
  severity: enum(low, medium, high, critical)
  image_url: string (optional)
  device_id: string
  created_at: timestamp
  synced: boolean
}
```

F3: Live Pest Heatmap

- **Description:** Web/mobile map showing aggregated pest sightings as a density heatmap
- **Tech:** Mapbox GL JS with heatmap layer
- **Features:**
  - Filter by species, time range, severity
  - Cluster markers at low zoom, heatmap at high zoom
  - Real-time updates via Supabase Realtime subscriptions
  - Color scale: green (low) → yellow → orange → red (critical density)

F4: Climate Data Overlay

- **Description:** Toggle-able layer showing temperature, humidity, wind speed/direction from weather APIs
- **Data source:** Open-Meteo API (free, no key required) or NOAA GFS
- **Display:** Wind direction arrows, temperature gradient shading, humidity contours
- **Purpose:** Visual correlation between climate conditions and pest movement

F5: Predictive Alert Zone

- **Description:** Projected pest movement zone rendered as a semi-transparent polygon on the map
- **Logic (hackathon-scope):**
  1. Identify clusters of 3+ sightings of same species within 10km and 48 hours
  2. Calculate cluster centroid and movement vector from sequential sighting timestamps
  3. Project forward 24–72 hours, weighted by wind direction and temperature gradient

4. Render projected zone as expanding polygon in direction of movement

- **Alert:** Simulated push notification / SMS to farms within projected zone
- **Note:** This is a simplified heuristic for demo. Production would use a proper spatiotemporal epidemiological model.

P1 — Nice to Have (If Time Permits)

F6: Community Verification

- Low-confidence classifications get flagged for peer review. Other farmers in the area can confirm or correct the species ID, building a feedback loop for model improvement.

F7: Pest Profile Cards

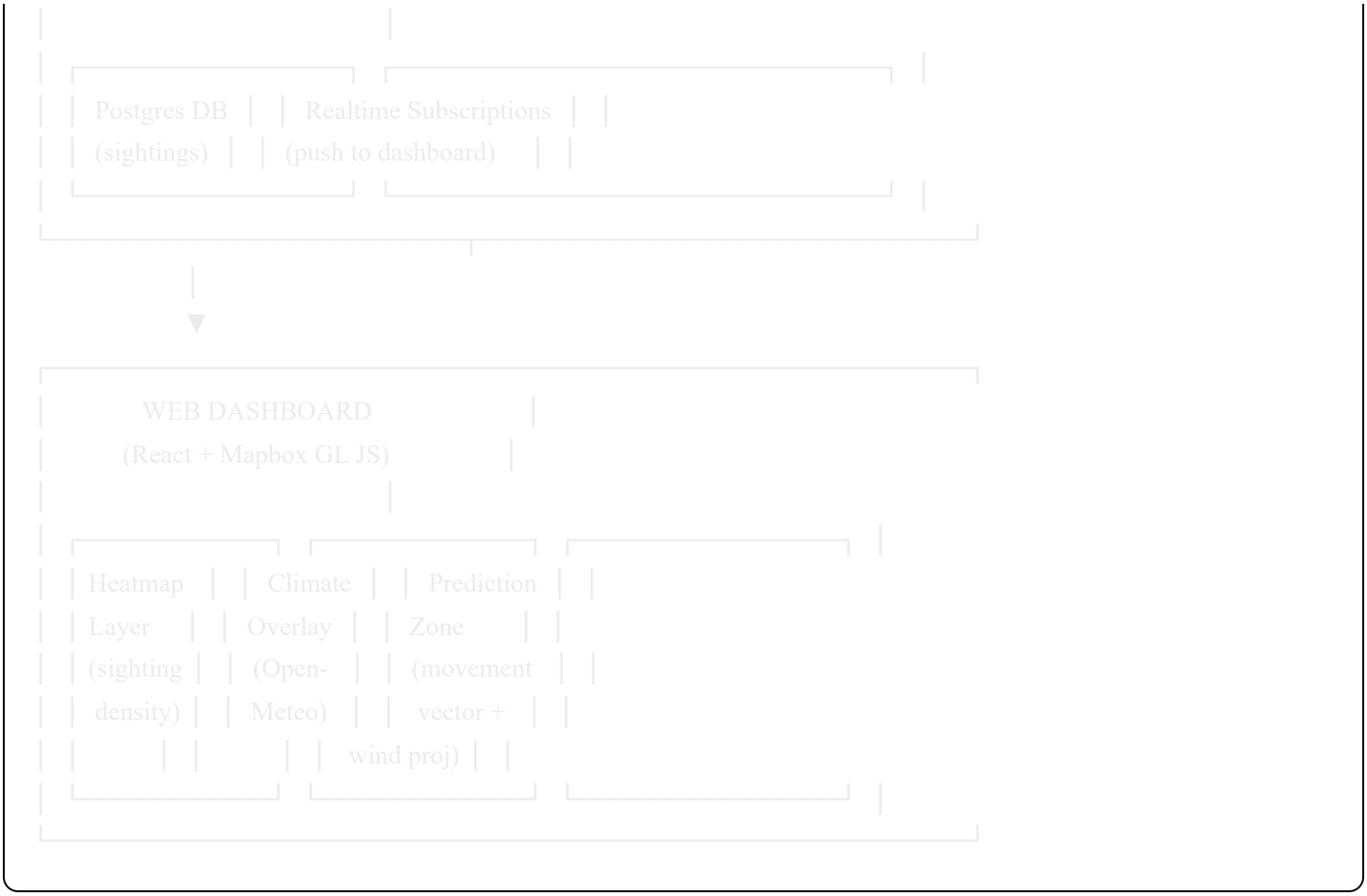
- Tapping a species on the heatmap shows an info card: lifecycle, crops affected, recommended treatment, climate conditions that favor spread.

F8: Historical Timeline Playback

- Slider to scrub through time and watch pest sightings animate across the map — powerful demo moment showing migration over weeks.

---

5. Technical Architecture



Key Technical Decisions

Decision	Choice	Rationale
Mobile framework	React Native + Expo	Fast to scaffold, TFLite plugin available
ML model	EfficientNet-Lite0, INT8 quantized	Best accuracy/latency tradeoff for mobile
Training data	IP102 (subset: 15–20 species)	Largest public pest image dataset
Backend	Supabase	Free tier, Postgres + Realtime out of the box, fast setup
Map	Mapbox GL JS	Heatmap layer, 3D terrain, free tier generous
Weather API	Open-Meteo	Free, no API key, global coverage
Prediction model	Heuristic (kernel density + wind vector projection)	Good enough for demo, honest about limitations

6. Data Strategy

Pre-Seeded Demo Data

To make the demo compelling, pre-load the database with **100–150 synthetic sightings** that tell a realistic story:

- Fall armyworm cluster moving NE across a California county over 10 days
- Aphid hotspot emerging near a river valley (humidity correlation)
- Scattered low-confidence sightings of a new invasive species
- Sightings spread across 30+ simulated "devices" to show network effect

Seed Script Requirements

- Geographically realistic (follow actual farmland, not random coordinates)
- Temporally sequential (show clear migration directionality)
- Species distribution matches real-world prevalence
- Include a mix of high and low confidence scores

---

7. Demo Script (3-Minute Pitch Flow)

[0:00–0:30] The Hook "Last year, fall armyworm caused $X billion in crop damage in the US alone. By the time USDA advisories reach farmers, the pest has already moved on. What if every farmer's phone was a sensor in a real-time pest surveillance network?"

[0:30–1:15] Live Demo — Detection Open the app on a physical phone. Point at a pest image (printed or on a second screen). Model classifies it in under 500ms. Show the inference time. Show it works in airplane mode. "This runs entirely on-device — no internet, no cloud, no latency. A farmer in a field with zero signal gets instant pest ID."

[1:15–2:00] Live Demo — The Network Switch to the web dashboard. The sighting just appeared on the map in real-time. Zoom out to show the pre-seeded heatmap. "Every dot is a farmer's report. Individually, these are pest IDs. Together, they're a migration map. Watch — over the past 10 days, this armyworm cluster has moved 40 miles northeast." (Play the timeline if built, or narrate the spatial pattern.)

[2:00–2:30] Live Demo — Prediction Toggle on the climate overlay. "Wind is blowing northeast at 12 mph. Temperature is above the armyworm activity threshold. Our model projects the front will reach this area" — highlight the prediction zone — "within 48 hours. Every farm in this zone just got an alert."

[2:30–3:00] The Close "PestCast turns pest surveillance from a top-down, slow, expensive process into a bottom-up, real-time, free one. The more farmers use it, the smarter it gets. We're building the Waze for

agricultural pests."

---

8. Anticipated Judge Questions & Answers

Henry (Cactus) — Edge AI / On-Device Focus

Q: "What model are you using and what's the accuracy/latency tradeoff?" A: EfficientNet-Lite0, quantized to INT8 via TFLite. We're seeing ~87% top-3 accuracy on our test species with sub-400ms inference on a Pixel 6. We chose Lite0 over Lite4 because the latency difference (400ms vs 1.2s) matters more for UX than the ~3% accuracy gain. For production, we'd explore knowledge distillation from a larger teacher model to push accuracy further without increasing latency.

Q: "How do you handle misclassifications?" A: Three layers. First, we show top-3 predictions with confidence scores so the farmer can pick the right one. Second, anything below 70% confidence gets flagged as "needs verification" and goes to community review. Third, we log all classifications with the original image so we can retrain. Long-term, the federated data from millions of field photos becomes the most valuable pest image dataset in the world.

Q: "Have you thought about model updates on-device?" A: Yes — we'd push updated model weights as OTA updates through the app store. For a more sophisticated approach, we could do on-device fine-tuning for regionally prevalent species using transfer learning on the edge, though that's a v2 feature. This is exactly the kind of infrastructure Cactus could power.

Coline (Tellia/EF) — Climate Impact Focus

Q: "How does climate change specifically make this problem worse?" A: Three mechanisms. First, warming expands pest ranges — species that couldn't survive northern winters now can. The brown marmorated stink bug's range has expanded 200+ miles north in 20 years. Second, milder winters reduce die-off, so initial spring populations are larger. Third, shifting precipitation patterns create new humidity corridors that enable pest spread into previously dry regions. Traditional surveillance models are calibrated to historical ranges — they can't keep up.

Q: "What's the climate impact of reducing pest-driven crop loss?" A: When farmers lose crops to pests, the entire carbon and resource footprint of growing that crop — water, fertilizer, diesel, land use — is wasted. Reducing crop loss is one of the highest-leverage climate interventions in agriculture. Additionally, pest outbreaks drive emergency pesticide applications, which have their own environmental cascade. Early warning means farmers can use targeted, minimal intervention instead of blanket spraying.

Q: "How would this work in the Global South where the need is greatest?" A: That's exactly why on-device inference is critical. We don't assume connectivity. The app works fully offline for identification. Sightings sync opportunistically. SMS alerts can reach feature phones. The model can be fine-tuned for region-specific pests — we'd partner with CGIAR or FAO for training data in sub-Saharan Africa and South/Southeast Asia.

Austin (Sorcerer) — Product / Engineering Focus

Q: "What's the business model?" A: The app is free for farmers — you need network density for the data to be valuable. The business is the intelligence layer. Crop insurance companies would pay for real-time pest pressure data to price risk. Seed companies want to know which pest resistances to prioritize by region. Ag commodity traders want early signals on crop damage. Government agencies want surveillance infrastructure without building it themselves. The data API is the product.

Q: "How do you get to critical mass? The cold-start problem is real." A: Three strategies. First, the app has standalone value even with zero network — it's a free, offline pest ID tool. That's the wedge. Second, we'd partner with extension services and farmer cooperatives to seed adoption in clusters — you need geographic density, not total users. Third, we pre-seed with existing public data (USDA CERIS, iNaturalist observations) so the map isn't empty on day one.

Q: "What stops someone from just building this on top of GPT-4 Vision?" A: Three things. First, latency and connectivity — GPT-4V needs internet, takes 3–5 seconds, and costs money per call. We're 400ms, offline, and free. Second, GPT-4V is a general model — our fine-tuned model is more accurate on pest species than a generalist. Third, and most importantly, the value isn't in the classification — it's in the network layer. The crowdsourced spatiotemporal data and prediction engine is the moat. The on-device model is just the data collection mechanism.

Q: "How accurate is the prediction model really?" A: We're honest about this — the hackathon version is a heuristic that combines cluster movement vectors with wind direction. It's directionally correct, not precisely calibrated. In production, you'd use a proper spatiotemporal epidemiological model, trained on historical USDA pest survey data correlated with weather patterns. The key insight is that even a rough directional prediction — "the pest front is moving toward you" — is infinitely more useful than no prediction at all.

---

9. Task Breakdown — 5 People, ~8 Hours

Person 1: ML Engineer — "The Model"

Goal: Trained, quantized pest classification model running in TFLite

Time	Task
Hour 1	Download IP102 dataset, select 15–20 target species, set up training environment (Colab)
Hour 2–3	Fine-tune EfficientNet-Lite0 (or MobileNetV3) using transfer learning. Freeze base layers, train classifier head
Hour 4	Quantize to INT8 TFLite, validate accuracy on test set, export .tflite file
Hour 5	Benchmark inference speed on target phone, optimize if needed (reduce input resolution, prune)
Hour 6	Hand off .tflite model + label map to Person 2. Help with integration testing
Hour 7–8	Help with demo prep, create accuracy/latency metrics slide, edge case testing

Deliverable: `pest_model.tflite` + `labels.json` + accuracy report

Person 2: Mobile Developer — "The App"

Goal: React Native app with camera capture, on-device inference, sighting submission

Time	Task
Hour 1	Scaffold React Native (Expo) project, set up navigation (Camera screen → Results screen → Map screen)
Hour 2	Build camera capture UI, integrate with device camera/gallery
Hour 3	Integrate TFLite React Native plugin, load model, wire up inference pipeline
Hour 4	Build results screen: top-3 predictions, confidence bars, species name, severity picker
Hour 5	Build sighting submission flow: capture GPS, create sighting object, POST to Supabase
Hour 6	Implement offline queue: store sightings in local state, batch sync on connectivity
Hour 7	Receive model from Person 1, integration test end-to-end on physical device
Hour 8	Polish UI, fix bugs, prepare phone for live demo

Deliverable: Working app on a physical Android phone

Person 3: Backend + Data Engineer — "The Plumbing"

Goal: Supabase backend, seed data, API endpoints

Time	Task
Hour 1	Set up Supabase project, create `sightings` table with schema (see §4), enable Realtime
Hour 2	Build API layer: insert sighting, query sightings by bounding box + time range + species filter
Hour 3	Enable Supabase Realtime subscriptions for the dashboard to consume
Hour 4–5	Write seed data script: generate 100–150 realistic synthetic sightings. Use real California farmland coordinates (pull from Google Maps). Create 2–3 distinct pest "stories" — an armyworm migration, an aphid hotspot, scattered detections
Hour 6	Coordinate with Person 2 on mobile→backend sync. Coordinate with Person 4 on dashboard data consumption
Hour 7	Load seed data, verify Realtime works, stress test with rapid inserts
Hour 8	Bug fixes, demo dry run support

Deliverable: Populated Supabase backend with live API + Realtime

Person 4: Frontend / Viz Engineer — "The Map"

Goal: Web dashboard with heatmap, climate overlay, prediction zone

Time	Task
Hour 1	Scaffold React web app, integrate Mapbox GL JS, set up base map centered on demo region
Hour 2	Build heatmap layer consuming sightings from Supabase. Color scale by density. Add species + time range filters
Hour 3	Implement real-time updates: Supabase Realtime subscription → new sightings appear on map instantly
Hour 4	Build climate overlay: fetch wind + temperature data from Open-Meteo API, render wind arrows + temp gradient layer
Hour 5	Build prediction zone logic: cluster detection → movement vector → forward projection polygon. Render as semi-transparent orange/red zone
Hour 6	Add simulated alert panel: " ⚠️ Fall armyworm predicted to reach [Farm X] within 48 hours"
Hour 7	Polish transitions, loading states, legend, responsive layout. Add timeline scrubber if time permits
Hour 8	Demo dry runs, final visual polish

Deliverable: Web dashboard at localhost (or deployed to Vercel)

Person 5: Research + Pitch + Design — "The Story"

Goal: Pitch narrative, slide deck, demo choreography, Q&A prep, visual design

Time	Task
Hour 1	Deep research: pest migration data, climate impact stats, current surveillance gaps. Pull specific numbers for the pitch (dollar figures, acreage, species case studies)
Hour 2	Research the judges: read about Cactus, Sorcerer, Tellia. Note what each cares about. Draft tailored talking points for Q&A
Hour 3	Write the pitch script (see §7). Aim for 2:30 spoken + 0:30 buffer
Hour 4	Design 4–6 backup slides: problem, solution, architecture diagram, market size, team. Use Figma or Google Slides
Hour 5	Design app UI mockups / style guide for Person 2: color palette, typography, logo sketch. Keep it clean and agricultural (earthy greens, warm tones, clear iconography)
Hour 6	Write the Q&A doc (see §8). Practice answers out loud. Assign who answers which question type
Hour 7	Run full demo dry run with team. Time it. Identify failure points. Build fallback plan (screenshots/video if live demo breaks)
Hour 8	Final rehearsal x2. Record a backup video walkthrough. Ensure demo phone is charged, dashboard URL is bookmarked, transitions are smooth

Deliverable: Pitch script, slide deck, Q&A prep doc, backup demo video

---

10. Risk Mitigation

Risk	Likelihood	Impact	Mitigation
Model accuracy is poor on target species	Medium	High	Reduce to 5–8 species, use top-3 instead of top-1 accuracy. Honesty about limitations is better than a bad demo
TFLite integration fails on React Native	Medium	Critical	Have a backup: run model via local Flask server on the phone as a fallback. Or pre-record the on-device demo as a video
Live demo crashes	Medium	High	Person 5 records a backup video at Hour 7. Practice the pivot: "Let me show you the recording while we troubleshoot"

Risk	Likelihood	Impact	Mitigation
Mapbox heatmap performance issues	Low	Medium	Pre-aggregate sightings into hex bins server-side instead of rendering raw points
Supabase Realtime drops during demo	Low	High	Pre-load all seed data so the map looks full regardless. Have Person 2 insert a sighting 30 seconds before the demo to pre-warm the connection

11. Post-Hackathon Roadmap (Mention Briefly in Q&A If Asked)

- **v1.1:** Community verification + gamification (pest scout leaderboards)
- **v1.2:** Proper spatiotemporal epidemiological model replacing heuristic
- **v2.0:** Federated learning for continuous model improvement from field data
- **v2.1:** API product for crop insurance, seed companies, commodity traders
- **v3.0:** Integration with drone/satellite imagery for canopy-level damage assessment