**Product Requirements Document (PRD)**

**ComplianceDrone.com - Thermal Inspection Platform**  
**Document Version**: 2.1 (Simplified)  
**Date**: September 16, 2025

**1. Executive Summary**

ComplianceDrone.com is a web platform enabling drone pilots to upload thermal inspection images and receive automated PDF reports using existing computer vision models. The platform wraps Python scripts to detect thermal anomalies in solar panels and infrastructure, eliminating manual report generation. It includes a job board for screened pilots and a simple dashboard with a map displaying thermal anomaly locations.  
**Core Value**: Transform 100+ thermal images into a professional PDF report in under 5 minutes, with pilot job assignments and anomaly mapping.

**2. Current State**

**Existing Assets**

* **Roboflow CV Model**: Trained on 4,000 thermal images to identify:
  + Hot spots
  + Substring failures
  + Module failures
  + Diode failures
  + Cell anomalies
  + Soiling/shading issues
* **Python Pipeline**:
  + roboflow\_processor.py - Runs CV model, creates annotated images
  + process\_anomaly\_images.py - Adds branding to anomaly images
  + ClaudeMain1\_fixed.py - Generates PDF reports from CSV data
* **Infrastructure**:
  + DJI Matrice 300 & M30T drones
  + DJI Cloud API access
  + Mapbox integration in PDF generator

**3. Technical Requirements**

**3.1 Core Functionality**

**Web API Service**

# Required endpoints

POST /api/process-job # Upload images, process, return PDF

GET /api/job/{id}/status # Check processing status

GET /api/job/{id}/report # Download PDF report

POST /api/job/request # Submit job request

GET /api/jobs # List available jobs

POST /api/pilot/register # Pilot registration

**Processing Flow**

1. Pilot uploads batch of thermal images (up to 500)
2. Images processed through Roboflow model
3. Anomalies detected, annotated, and tagged with lat/long
4. CSV summary generated
5. PDF report created with annotated images and anomaly map
6. Report delivered to pilot

**3.2 Frontend Requirements**

**Phase 1: MVP Pages**

* **Landing Page (/)**: Service description, "Request Inspection" and "Become a Pilot" CTAs
* **Upload Interface (/pilot-upload)**: Drag-drop for batch image upload
* **Results Page (/results/[id])**: PDF download, anomaly summary, and map with lat/long pins
* **Simple Dashboard (/dashboard)**: Displays job status, recent uploads, and Mapbox with anomaly lat/long pins (using existing Next.js template)
* **Job Board (/jobs)**: List available inspection jobs
* **Pilot Registration (/pilot-registration)**: Onboarding form with screening questions

**3.3 System Architecture**

[Next.js Frontend]

↓

[FastAPI Service Wrapper]

↓

[Existing Python Scripts]

├── Roboflow CV Model

├── Image Processing

└── PDF Generation

↓

[S3 Storage for Reports]

[Mapbox for Anomaly Mapping]

**4. Implementation Details**

**4.1 API Wrapper Service**

# simple\_api\_wrapper.py structure

@app.post("/process-job")

async def process\_inspection\_job(

job\_id: str,

pilot\_id: str,

location: str,

files: List[UploadFile]

):

# 1. Save uploaded images

# 2. Run roboflow\_processor.py

# 3. Extract lat/long from image metadata

# 4. Run process\_anomaly\_images.py

# 5. Generate CSV summary

# 6. Run ClaudeMain1\_fixed.py with Mapbox integration

# 7. Upload PDF to S3

# 8. Return results with anomaly map data

@app.post("/job/request")

async def request\_job(

customer\_id: str,

location: str,

details: str

):

# Assign to screened pilot

# Notify pilot via email

@app.post("/pilot/register")

async def register\_pilot(

pilot\_data: PilotRegistration

):

# Screen pilot credentials

# Store approved pilot in database

**4.2 Deployment Configuration**

# docker-compose.yml

services:

nextjs:

ports: ["3000:3000"]

environment:

- PYTHON\_SERVICE\_URL=http://python-api:8000

- MAPBOX\_ACCESS\_TOKEN=${MAPBOX\_ACCESS\_TOKEN}

python-api:

build: ./python-services

ports: ["8000:8000"]

volumes:

- ./python-scripts:/app

environment:

- ROBOFLOW\_API\_KEY=${ROBOFLOW\_API\_KEY}

- ROBOFLOW\_MODEL=solar-thermal/1

- S3\_BUCKET=compliance-drone-uploads

- MAPBOX\_ACCESS\_TOKEN=${MAPBOX\_ACCESS\_TOKEN}

**4.3 File Structure**

/project-root

/app # Next.js frontend

/pilot-upload # Upload interface

/dashboard # Dashboard with Mapbox

/jobs # Job board

/pilot-registration # Pilot onboarding

/api/process # API routes

/python-services # Python wrapper

simple\_api\_wrapper.py # FastAPI service

roboflow\_processor.py # Existing CV script

process\_anomaly\_images.py

ClaudeMain1\_fixed.py # PDF generator

/fonts # PDF fonts

Pic\_Logo.png # Company logo

**5. User Flow**

**Primary User Journey (Pilot)**

1. Pilot completes drone inspection flight
2. Navigates to /pilot-upload
3. Drags thermal images into upload zone
4. Clicks "Generate Report"
5. Waits ~2-3 minutes for processing
6. Views /dashboard with job status and Mapbox showing anomaly lat/long pins
7. Downloads PDF report with:
   * Annotated thermal images showing anomalies
   * Summary of issues by type
   * Location information with Mapbox visualization
   * Professional formatting with logos

**Customer Journey**

1. Submits job request via /jobs
2. Screened pilot assigned
3. Receives notification and final report with anomaly map

**6. Development Phases**

**Phase 1: Core Pipeline (Week 1)**

* Set up FastAPI wrapper service
* Create upload endpoint with existing scripts
* Add lat/long extraction for anomaly mapping
* Basic Next.js upload interface and dashboard (using existing template)
* S3 integration for report storage
* Local testing with sample images

**Phase 2: Job Board & Pilot Registration (Week 2)**

* Implement /jobs and /pilot-registration pages
* Add pilot screening logic
* Docker containerization
* Error handling and retries
* Upload progress indicators
* Deploy to cloud (Vercel + Railway/AWS)
* Test with real pilot uploads

**Phase 3: Enhanced Features (Week 3-4)**

* Landing page and branding
* Email notifications for report completion
* DJI pin validation comparison
* Mapbox integration in dashboard and PDF

**7. Technical Specifications**

**Performance Requirements**

* Support 20 concurrent uploads
* Process 100 images in < 3 minutes
* Generate PDF in < 30 seconds
* 99% uptime for API service

**File Handling**

* Max upload: 500 images per batch
* Supported formats: JPG, JPEG, PNG
* Max file size: 20MB per image
* Storage: 7-day retention for reports

**Dependencies**

# Python requirements

fastapi==0.104.1

uvicorn==0.24.0

python-multipart==0.0.6

boto3==1.28.84

roboflow # For CV model

reportlab # For PDF generation

Pillow

pandas

opencv-python

mapbox # For anomaly mapping

**8. Future Enhancements (Not MVP)**

* Enterprise customer dashboard
* Payment processing for job marketplace
* Real-time DJI Flight Hub integration
* Mobile app for field uploads
* Advanced pilot matching algorithm
* White-label options for enterprise
* Thermal temperature extraction from EXIF

**9. Success Metrics**

* **Technical**: < 5% processing failure rate
* **Usage**: 50+ reports generated in first month
* **Quality**: 85%+ accuracy vs manual pin validation
* **Performance**: < 3 min average processing time

**10. Risks & Mitigations**

Risk | Mitigation

-----------------------------|--------------------------------

Roboflow API limits | Cache model locally as backup

Large batch processing fails | Add chunking and retry logic

PDF generation memory issues | Process images in batches

S3 costs | Implement retention policies

Mapbox API limits | Optimize API calls, cache maps

Pilot screening delays | Automate initial screening checks