# FEMA-Oriented Disaster Recovery Plan (Rapid Template)

**Version:** 1.0 (Prepared: Oct 27, 2025)  
**Purpose:** Provide a practical, submission-ready recovery plan aligned with FEMA’s National Disaster Recovery Framework (NDRF), Public Assistance (PA), and Individuals & Households Program (IHP) practices. Use for households, small businesses/PNPs, and local jurisdictions seeking federal/state support after a declared disaster.

## 1) Incident Snapshot

* **Disaster Name / Incident #:**
* **Jurisdiction / Community:**
* **Incident Period:** Start: \_\_\_\_ / End: \_\_\_\_
* **Primary Hazards:** (e.g., wind, storm surge, flooding, landslides)
* **Emergency Declaration / Major Disaster Declaration #:**
* **Designated Counties/Parishes:**

**Executive Summary (3–5 sentences):**  
*[Describe impacts, populations affected, lifeline disruptions (safety & security, food/water/shelter, health & medical, energy, communications, transportation, hazardous materials), and the top recovery priorities for the next 30/90/180 days.]*

## 2) Governance & Coordination

* **Recovery Lead (Overall):** Name, title, organization, contacts
* **Deputy / Continuity Lead:**
* **Finance/Admin (Grant Compliance):**
* **Public Information Officer (PIO):**
* **Damage Assessment Lead:**
* **Debris Task Force Lead:**
* **Procurement/Contracting Officer:**
* **Legal Counsel:**
* **Case Management / Human Services Lead:**

**Coordination Structures:**  
- **Emergency Operations Center (EOC) Link:** Activation level, location, hours  
- **Recovery Task Force / RSFs:** (Community Planning & Capacity Building; Economic; Health & Social Services; Housing; Infrastructure Systems; Natural & Cultural Resources)  
- **DRC(s) / Joint Field Office (if established):** Locations & hours  
- **Mutual Aid / EMAC / MOUs:** Summary and points of contact

**Meetings & Rhythm:**  
- Daily ops brief (time) | Finance/PA sync (time) | Debris sync (time) | Public info update (time)

## 3) Objectives & Measurable Targets

**30-Day Objectives:**  
- Life-safety stabilized; critical access restored to hospitals, shelters, and water/wastewater.  
- 100% initial damage assessments submitted (IA, PA).  
- Debris clearance of priority corridors complete; debris monitoring in place.  
- All eligible applicants registered with FEMA; case mgmt launched.

**90-Day Objectives:**  
- All emergency work documented; 100% of large PA projects in formulation.  
- Interim housing placements matched to unmet needs; landlord engagement active.  
- SBA referrals processed; duplication-of-benefits controls implemented.

**180-Day Objectives:**  
- Permanent work scoping complete; hazard-mitigation opportunities integrated; environmental/historic (EHP) reviews initiated.  
- Long-term recovery plan adopted; capital stack identified (PA, HMGP, CDBG-DR, insurance, philanthropy).

*Metrics:* registrations, inspections completed, debris cubic yards removed, PA projects formulated/obligated, households housed, critical facilities restored, mitigation projects initiated.

## 4) Operational Timeline & Checklists

### A. First 0–72 Hours (Stabilization)

* Account for personnel; confirm safety and continuity (COOP).
* Life-safety priorities; search & rescue, medical, sheltering.
* Establish debris clearance priorities (Tier 1 roadways, emergency routes, critical facilities).
* Start incident documentation: ICS forms, photos/video with GPS/time stamps.
* Open applicant registration channels (households & businesses).

### B. Days 3–14 (Initial Recovery)

* Rapid/initial damage assessments (structures, infrastructure, environmental).
* Stand up Debris Task Force; select debris monitoring solution; draft debris removal strategy (curbside, ROE, haul-out).
* Issue emergency procurement justifications; ensure 2 CFR 200 compliance; avoid cost-plus-percentage-of-cost contracts.
* Initiate unmet-needs/voluntary org coordination; mass care transition plan.
* Launch public communications: how to register with FEMA, DRC locations, SBA, legal aid, crisis counseling, unemployment assistance.

### C. Days 15–60 (Program Delivery)

* Individuals: inspections, eligibility documentation (identity, occupancy, ownership), appeals process; temporary housing/leasing strategy; STEP/repairs if available.
* PA Applicants: kick-off meetings; damage inventory; site inspections; develop scopes of work, cost estimates; EHP screening.
* Debris: execute contracts (include monitoring); daily load tickets, truck certifications, disposal site manifests; monitor T&M caps.
* Finance: establish grant files, cost codes, timekeeping; track donated resources; establish duplication-of-benefits controls (insurance/SBA).

### D. Days 60–180 (Permanent Work & Mitigation)

* Finalize permanent repair scopes; pursue 406 mitigation; align with HMGP 404 and local mitigation plans.
* Begin design/procurement for permanent work; continue case management for vulnerable populations.
* Develop long-term housing strategy; leverage CDBG-DR (if awarded).

### E. 6–24 Months (Long-Term Recovery)

* Obligate large projects; execute construction; maintain change control.
* Track outcomes; publish transparency dashboards; after-action review (AAR) with improvement plan (IP).

## 5) Individuals & Households (IHP) – Action Guide

**Who:** Survivors (homeowners/renters)  
**Key Actions:** - Register for FEMA assistance (online/phone/DRC); meet deadline.  
- Prepare documents: photo ID, SSN (if applicable), disaster address, current contact, insurance info; **Proof of Occupancy** (e.g., utility bill, lease, driver’s license with address) and **Proof of Ownership** (e.g., deed, mortgage, property tax, insurance).  
- Attend inspection; keep all receipts; report changes.  
- If ineligible or amount insufficient, file **appeal within 60 days** with supporting documents.  
- Explore **SBA disaster loans** (homeowners & renters for personal property/vehicle); SBA decisions affect some FEMA aid types.  
- Track **Duplication of Benefits** (insurance, charity, grants).

**Case Management & Wraparound:** Crisis counseling, disaster unemployment, legal aid, disability access and functional needs (AFN), language access, VOADs coordination.

## 6) Public Assistance (PA) – Action Guide (Governments & Eligible PNPs)

**Emergency Work (Cat A/B):** Debris removal; emergency protective measures.  
**Permanent Work (Cat C–G):** Roads/bridges; utilities; public buildings/equipment; water control facilities; parks.

**Applicants:** - Attend applicant briefing; request access to FEMA Grants Portal.  
- Designate Applicant’s Agent & Recovery Finance Lead.  
- Build **Damage Inventory** with locations (GPS), photos, dimensions, cause, pre-disaster condition, estimated costs.  
- Use **force account** and **contract** cost capture (labor, equipment, materials); certify timesheets; maintain procurement files (solicitation, selection, contracts, change orders, invoices).  
- **Debris Operations:** establish monitoring (TTLCs/load tickets), truck certifications, site monitoring, TDSR sites, environmental controls; document volumes/weights and final disposition.  
- **Environmental & Historic (EHP):** do not start permanent work before compliance.  
- Integrate **hazard mitigation** (406) and align with HMGP 404 opportunities.

**Finance & Compliance:**  
- Set up grant files by project; separate accounting codes; capture donated resources; insurance recovery; avoid ineligible contracts (e.g., cost-plus-%).  
- Track deadlines (emergency work 6 months; permanent work 18 months—extensions possible).

## 7) Debris Management Playbook (Quick)

* **Strategy:** Curbside push, haul-out, site pass system, private property ROE program (if authorized), waterways/leaners/hangers.
* **Contracts:** Pre-event if possible; otherwise emergency procurement with competitive quotes when feasible; define unit prices; avoid open-ended T&M after 70 hrs (unless justified/approved).
* **Monitoring:** Independent monitoring; truck certs; load tickets; tower/site monitors; daily reconciliation; photo evidence; GIS mapping of passes.
* **Environmental:** TDSR permitting, segregation (C&D, vegetative, white goods, HHW), final disposal documentation.

## 8) Housing & Human Services

* **Short-Term:** Sheltering, congregate/non-congregate, hoteling, dorms.
* **Interim:** Rentals, repair (e.g., STEP if available), direct housing (manufactured homes) if authorized.
* **Long-Term:** Repair/rebuild, buyout/relocation (with partner programs), accessible units.
* **Equity & AFN:** Prioritize access, reasonable modifications, translation/interpretation, transportation, childcare, elder care.

## 9) Risk, Safety & Communications

* **Lifeline Risks:** Flooding, landslides, hazardous materials, extended power/water outages.
* **Worker Safety:** Heat, debris hazards, carbon monoxide, chainsaw safety, fall protection, PPE.
* **Public Info:** Rumor control, registration how-to, DRC hours/locations, debris pickup maps, permitting guidance, construction fraud warnings.

## 10) Documentation Matrix (What to Save)

| Area | Required Records |
| --- | --- |
| Personnel (Force Account) | Timesheets, activity logs, payroll reports, fringe calcs |
| Equipment | Daily usage logs, rate sheets (FEMA/State published), fuel/maintenance |
| Materials | Invoices, delivery tickets, inventory withdrawals |
| Contracts | Procurement record, bids/quotes, contract, SOW, change orders, invoices, proof of payment |
| Debris | Truck certs, load tickets, route maps, TDSR permits, disposal manifests, monitoring reports |
| Damage | Before/after photos, GPS, dimensions, engineering assessments |
| Finance | General ledger by project, insurance, donations, match sources |

## 11) Funding Strategy & Match

* **Primary Sources:** FEMA IA, FEMA PA (cost share), HMGP, SBA, CDBG-DR (if allocated), insurance, state cost share, philanthropy.
* **Match Plan:** Identify local match; in-kind/donated resources; state legislative aid; foundation grants.
* **Anti-Duplication:** Maintain ledger; verify insurance proceeds; coordinate with SBA/charities.

## 12) Appeals & Closeout

* **IA Appeals:** Submit within 60 days with evidence; keep correspondence.
* **PA Appeals/Requests for Reconsideration:** File within regulatory timelines; maintain audit-ready files.
* **Quarterly Reports:** Progress and financial; variance explanations.
* **Closeout Package:** Final reports, certifications, retention plan (min. 3 years post-closeout or per state requirement).

## 13) Annexes (Fillable)

* **A. Contact Roster** (all roles; 24/7 lines)
* **B. Damage Assessment Forms** (residential, public infrastructure)
* **C. Debris Forms** (truck cert, load ticket templates)
* **D. Procurement Templates** (RFP/IFB, evaluation, contract)
* **E. Public Information Toolkit** (FAQ, SMS scripts, signage)
* **F. Safety Plans** (JHAs, PPE matrices)
* **G. Housing Ops** (shelter to housing flow, AFN checklist)
* **H. Finance/Grant File Index** (binder tabs & naming conventions)

## 14) Signature & Adoption

* **Authorized Official:** Name/Title/Signature/Date
* **Finance/Grants Officer:** Name/Signature/Date
* **Legal Counsel:** Name/Signature/Date

*Maintain version control. Update this plan as declarations, designations, or policies change.*

## 15) Community Commitment Statement

We will show up, stand shoulder to shoulder, and help you grow back even stronger. You will not be left behind or left to starve this time. Our team commits to sustained, dignified support—from immediate relief through long‑term recovery—so every family can rebuild with safety, stability, and hope.

## Appendix I: Equipment Readiness — Drop‑In Code (Single‑File React Component)

Purpose: a **drop‑in, offline‑first panel** to track equipment status (generators, pumps, chainsaws, radios, vehicles), maintenance intervals, checklists, and printable work orders. Works entirely in the browser (localStorage). Supports CSV import/export so ops can sync with spreadsheets when connectivity is limited.

Assumptions: Tailwind CSS available; **shadcn/ui**, **lucide-react**, **framer-motion**, **recharts**, and **qrcode.react** are installed in your project environment.

// File: EquipmentReadinessPanel.tsx  
// Drop‑in, single‑file React component for disaster ops equipment readiness.  
// Dependencies (assumed available per project setup):  
// - react, framer-motion, lucide-react, qrcode.react, recharts, shadcn/ui  
// Styling: Tailwind; minimal, clean aesthetic; offline‑first via localStorage.  
  
import React, { useEffect, useMemo, useState } from "react";  
import { motion } from "framer-motion";  
import {  
 Wrench, Plus, Save, Upload, Download, Trash2, Search, Printer,  
 ClipboardList, CheckCircle2, AlertTriangle, QrCode, RefreshCcw  
} from "lucide-react";  
import { QRCodeCanvas } from "qrcode.react";  
import {  
 Card, CardContent, CardHeader, CardTitle  
} from "@/components/ui/card";  
import { Button } from "@/components/ui/button";  
import { Input } from "@/components/ui/input";  
import { Textarea } from "@/components/ui/textarea";  
import { Label } from "@/components/ui/label";  
import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogTrigger } from "@/components/ui/dialog";  
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table";  
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";  
  
// Types  
export type Equipment = {  
 id: string; // stable id  
 name: string; // e.g., "Generator #1"  
 type: string; // generator | pump | chainsaw | radio | vehicle | other  
 status: "Ready" | "Needs Service" | "Out of Service";  
 hours: number; // runtime hours / odometer equivalent  
 serviceInterval: number; // hours between service  
 lastServiceHours: number; // hours at last service  
 assignedTo?: string; // person/team  
 location?: string; // where stored/deployed  
 notes?: string;  
};  
  
const STORAGE\_KEY = "equip\_readiness\_v1";  
  
function loadInitial(): Equipment[] {  
 try {  
 const raw = localStorage.getItem(STORAGE\_KEY);  
 if (raw) return JSON.parse(raw);  
 } catch {}  
 // Seed with a few examples  
 return [  
 { id: crypto.randomUUID(), name: "Generator #1", type: "generator", status: "Ready", hours: 240, serviceInterval: 100, lastServiceHours: 200, location: "EOC Yard", notes: "Spare spark plugs in case." },  
 { id: crypto.randomUUID(), name: "Trash Pump 2", type: "pump", status: "Needs Service", hours: 87, serviceInterval: 50, lastServiceHours: 50, location: "Depot A", notes: "Replace seals; suction hose worn." },  
 { id: crypto.randomUUID(), name: "Chainsaw 5 (20\" bar)", type: "chainsaw", status: "Ready", hours: 15, serviceInterval: 25, lastServiceHours: 0, assignedTo: "Debris Squad 1" },  
 { id: crypto.randomUUID(), name: "VHF Radio 12", type: "radio", status: "Ready", hours: 0, serviceInterval: 0, lastServiceHours: 0, notes: "Spare battery paired." }  
 ];  
}  
  
function save(items: Equipment[]) {  
 localStorage.setItem(STORAGE\_KEY, JSON.stringify(items));  
}  
  
function hoursUntilService(e: Equipment) {  
 if (e.serviceInterval <= 0) return Infinity;  
 return e.serviceInterval - (e.hours - e.lastServiceHours);  
}  
  
function statusBadge(status: Equipment["status"]) {  
 const base = "inline-flex items-center gap-2 rounded-2xl px-3 py-1 text-sm";  
 if (status === "Ready") return <span className={base + " bg-green-100"}><CheckCircle2 className="h-4 w-4"/>Ready</span>;  
 if (status === "Needs Service") return <span className={base + " bg-yellow-100"}><AlertTriangle className="h-4 w-4"/>Needs Service</span>;  
 return <span className={base + " bg-red-100"}><AlertTriangle className="h-4 w-4"/>Out of Service</span>;  
}  
  
export default function EquipmentReadinessPanel() {  
 const [items, setItems] = useState<Equipment[]>(loadInitial());  
 const [q, setQ] = useState("");  
 const [showForm, setShowForm] = useState(false);  
 const [edit, setEdit] = useState<Equipment | null>(null);  
  
 useEffect(() => { save(items); }, [items]);  
  
 const filtered = useMemo(() => {  
 const s = q.trim().toLowerCase();  
 if (!s) return items;  
 return items.filter(e => [e.name, e.type, e.status, e.location, e.assignedTo, e.notes]  
 .filter(Boolean)  
 .some(v => String(v).toLowerCase().includes(s)));  
 }, [items, q]);  
  
 const dueSoon = useMemo(() => filtered.filter(e => hoursUntilService(e) <= 10), [filtered]);  
  
 function upsert(e: Equipment) {  
 setItems(prev => {  
 const i = prev.findIndex(x => x.id === e.id);  
 if (i === -1) return [e, ...prev];  
 const next = [...prev];  
 next[i] = e; return next;  
 });  
 setShowForm(false); setEdit(null);  
 }  
  
 function remove(id: string) { setItems(prev => prev.filter(e => e.id !== id)); }  
  
 function exportCSV() {  
 const header = ["id","name","type","status","hours","serviceInterval","lastServiceHours","assignedTo","location","notes"];   
 const rows = items.map(e => header.map(h => (e as any)[h] ?? ""));  
 const csv = [header.join(","), ...rows.map(r => r.map(v => `"${String(v).replaceAll('"','""')}"`).join(","))].join("  
");  
 const blob = new Blob([csv], { type: "text/csv" });  
 const url = URL.createObjectURL(blob);  
 const a = document.createElement("a");  
 a.href = url; a.download = "equipment\_readiness.csv"; a.click();  
 URL.revokeObjectURL(url);  
 }  
  
 function importCSV(file: File) {  
 const reader = new FileReader();  
 reader.onload = () => {  
 try {  
 const text = String(reader.result);  
 const [head, ...lines] = text.split(/?  
/).filter(Boolean);  
 const cols = head.split(",").map(h => h.replaceAll('"','').trim());  
 const idx = (k: string) => cols.indexOf(k);  
 const next: Equipment[] = lines.map(line => {  
 const cells = line.match(/("[^"]\*"|[^,]+)/g)?.map(c => c.replace(/^"|"$/g, "").replaceAll('""','"')) ?? [];  
 const get = (k: string) => cells[idx(k)] ?? "";  
 return {  
 id: get("id") || crypto.randomUUID(),  
 name: get("name"),  
 type: get("type"),  
 status: (get("status") as Equipment["status"]) || "Ready",  
 hours: Number(get("hours") || 0),  
 serviceInterval: Number(get("serviceInterval") || 0),  
 lastServiceHours: Number(get("lastServiceHours") || 0),  
 assignedTo: get("assignedTo"),  
 location: get("location"),  
 notes: get("notes"),  
 };  
 });  
 setItems(next);  
 } catch (e) {  
 alert("Import failed: " + (e as Error).message);  
 }  
 };  
 reader.readAsText(file);  
 }  
  
 function resetSample() { setItems(loadInitial()); }  
  
 return (  
 <div className="p-4 md:p-6 max-w-7xl mx-auto">  
 <div className="flex items-center justify-between mb-4">  
 <h1 className="text-2xl md:text-3xl font-semibold flex items-center gap-2"><Wrench className="h-7 w-7"/> Equipment Readiness</h1>  
 <div className="flex gap-2">  
 <Button onClick={() => setShowForm(true)}><Plus className="h-4 w-4 mr-2"/>Add</Button>  
 <Button variant="secondary" onClick={exportCSV}><Download className="h-4 w-4 mr-2"/>Export</Button>  
 <label className="inline-flex items-center gap-2 cursor-pointer">   
 <Upload className="h-4 w-4"/>  
 <span className="sr-only">Import CSV</span>  
 <input type="file" accept=".csv" className="hidden" onChange={(e)=> e.target.files && importCSV(e.target.files[0])}/>  
 <span className="text-sm">Import</span>  
 </label>  
 <Button variant="outline" onClick={resetSample}><RefreshCcw className="h-4 w-4 mr-2"/>Reset</Button>  
 <Button variant="outline" onClick={() => window.print()}><Printer className="h-4 w-4 mr-2"/>Print</Button>  
 </div>  
 </div>  
  
 <Tabs defaultValue="list" className="space-y-4">  
 <TabsList>  
 <TabsTrigger value="list">Inventory</TabsTrigger>  
 <TabsTrigger value="checklists">Checklists</TabsTrigger>  
 <TabsTrigger value="qr">QR Codes</TabsTrigger>  
 </TabsList>  
  
 <TabsContent value="list">  
 <Card className="mb-4">  
 <CardContent className="p-4 flex items-center gap-3">  
 <Search className="h-4 w-4"/>  
 <Input placeholder="Search name, type, status, location…" value={q} onChange={(e)=> setQ(e.target.value)} />  
 </CardContent>  
 </Card>  
 <Card>  
 <CardHeader><CardTitle>Assets</CardTitle></CardHeader>  
 <CardContent className="p-0">  
 <Table>  
 <TableHeader>  
 <TableRow>  
 <TableHead>Name</TableHead>  
 <TableHead>Type</TableHead>  
 <TableHead>Status</TableHead>  
 <TableHead>Hours</TableHead>  
 <TableHead>Next Service (hrs)</TableHead>  
 <TableHead>Assigned</TableHead>  
 <TableHead>Location</TableHead>  
 <TableHead></TableHead>  
 </TableRow>  
 </TableHeader>  
 <TableBody>  
 {filtered.map(e => (  
 <TableRow key={e.id} className={hoursUntilService(e) <= 0 ? "bg-red-50" : hoursUntilService(e) <= 10 ? "bg-yellow-50" : ""}>  
 <TableCell className="font-medium">{e.name}</TableCell>  
 <TableCell className="capitalize">{e.type}</TableCell>  
 <TableCell>{statusBadge(e.status)}</TableCell>  
 <TableCell>{e.hours.toFixed(0)}</TableCell>  
 <TableCell>{Number.isFinite(hoursUntilService(e)) ? hoursUntilService(e).toFixed(0) : "—"}</TableCell>  
 <TableCell>{e.assignedTo || "—"}</TableCell>  
 <TableCell>{e.location || "—"}</TableCell>  
 <TableCell className="text-right">  
 <div className="flex justify-end gap-2">  
 <Button size="sm" variant="secondary" onClick={() => { setEdit(e); setShowForm(true); }}>Edit</Button>  
 <Button size="sm" variant="destructive" onClick={() => remove(e.id)}><Trash2 className="h-4 w-4"/></Button>  
 </div>  
 </TableCell>  
 </TableRow>  
 ))}  
 </TableBody>  
 </Table>  
 </CardContent>  
 </Card>  
  
 {dueSoon.length > 0 && (  
 <Card className="mt-4">  
 <CardHeader><CardTitle>Service Due Soon</CardTitle></CardHeader>  
 <CardContent>  
 <ul className="list-disc pl-6 space-y-1">  
 {dueSoon.map(e => (  
 <li key={e.id}>{e.name} — in {Math.max(0, Math.floor(hoursUntilService(e)))} hrs</li>  
 ))}  
 </ul>  
 </CardContent>  
 </Card>  
 )}  
 </TabsContent>  
  
 <TabsContent value="checklists">  
 <Card>  
 <CardHeader><CardTitle className="flex items-center gap-2"><ClipboardList className="h-5 w-5"/> Printable Checklists</CardTitle></CardHeader>  
 <CardContent>  
 <ChecklistBlock label="Generator Pre‑Start" items={[  
 "Fuel level checked",  
 "Oil level/filters OK",  
 "Battery charged",  
 "Air filter inspected",  
 "Cords/load tested",  
 "Grounding verified",  
 "Spare plugs & fuses available"  
 ]} />  
 <ChecklistBlock label="Pump Deployment" items={[  
 "Intake clear",  
 "Hoses/no leaks",  
 "Strainer fitted",  
 "Prime confirmed",  
 "Gaskets/seals OK",  
 "PPE issued"  
 ]} />  
 <ChecklistBlock label="Chainsaw Safety" items={[  
 "Chain tension/teeth sharp",  
 "Bar oil filled",  
 "Chain brake functional",  
 "Chaps, eye/ear protection",  
 "Clear fall zone",  
 "Spare chain available"  
 ]} />  
 <ChecklistBlock label="Radio Check" items={[  
 "Battery charged + spare",  
 "Channel plan loaded",  
 "Antenna secure",  
 "PTT test successful",  
 "Call signs posted"  
 ]} />  
 </CardContent>  
 </Card>  
 </TabsContent>  
  
 <TabsContent value="qr">  
 <Card>  
 <CardHeader><CardTitle className="flex items-center gap-2"><QrCode className="h-5 w-5"/> Asset QR Codes</CardTitle></CardHeader>  
 <CardContent className="grid md:grid-cols-2 lg:grid-cols-3 gap-4">  
 {items.map(e => (  
 <div key={e.id} className="border rounded-2xl p-4 flex flex-col items-center gap-2">  
 <QRCodeCanvas value={JSON.stringify({ id: e.id, name: e.name })} size={128} includeMargin/>  
 <div className="text-sm text-center">  
 <div className="font-medium">{e.name}</div>  
 <div className="text-gray-600">{e.type} • {e.status}</div>  
 </div>  
 </div>  
 ))}  
 </CardContent>  
 </Card>  
 </TabsContent>  
 </Tabs>  
  
 <EditorDialog open={showForm} onClose={() => { setShowForm(false); setEdit(null); }} onSave={upsert} initial={edit}/>  
 </div>  
 );  
}  
  
function ChecklistBlock({ label, items }: { label: string; items: string[] }) {  
 return (  
 <Card className="mb-4">  
 <CardHeader><CardTitle>{label}</CardTitle></CardHeader>  
 <CardContent>  
 <ul className="grid md:grid-cols-2 gap-2 print:grid-cols-1">  
 {items.map((t, i) => (  
 <li key={i} className="flex items-start gap-2">  
 <input type="checkbox" className="mt-1"/>  
 <span>{t}</span>  
 </li>  
 ))}  
 </ul>  
 <div className="mt-4 flex gap-2">  
 <Button variant="outline" onClick={() => window.print()}><Printer className="h-4 w-4 mr-2"/>Print</Button>  
 </div>  
 </CardContent>  
 </Card>  
 );  
}  
  
function EditorDialog({ open, onClose, onSave, initial }:{  
 open: boolean;  
 onClose: () => void;  
 onSave: (e: Equipment) => void;  
 initial: Equipment | null;  
}) {  
 const [form, setForm] = useState<Equipment>(initial ?? {  
 id: crypto.randomUUID(), name: "", type: "generator", status: "Ready",  
 hours: 0, serviceInterval: 0, lastServiceHours: 0, assignedTo: "", location: "", notes: ""  
 });  
 useEffect(()=>{ if(initial) setForm(initial); }, [initial]);  
  
 function set<K extends keyof Equipment>(k: K, v: Equipment[K]) {  
 setForm(prev => ({ ...prev, [k]: v }));  
 }  
 function submit() {  
 if (!form.name.trim()) { alert("Name is required"); return; }  
 onSave({ ...form, hours: Number(form.hours||0), serviceInterval: Number(form.serviceInterval||0), lastServiceHours: Number(form.lastServiceHours||0) });  
 }  
  
 return (  
 <Dialog open={open} onOpenChange={(v)=> !v && onClose()}>  
 <DialogContent className="max-w-xl">  
 <DialogHeader>  
 <DialogTitle>{initial ? "Edit Asset" : "Add Asset"}</DialogTitle>  
 </DialogHeader>  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-3">  
 <div>  
 <Label>Name</Label>  
 <Input value={form.name} onChange={e=> set("name", e.target.value)} placeholder="e.g., Generator #1"/>  
 </div>  
 <div>  
 <Label>Type</Label>  
 <Input value={form.type} onChange={e=> set("type", e.target.value)} placeholder="generator | pump | …"/>  
 </div>  
 <div>  
 <Label>Status</Label>  
 <select className="w-full border rounded-md h-10 px-3" value={form.status} onChange={e=> set("status", e.target.value as any)}>  
 <option>Ready</option>  
 <option>Needs Service</option>  
 <option>Out of Service</option>  
 </select>  
 </div>  
 <div>  
 <Label>Hours</Label>  
 <Input type="number" value={form.hours} onChange={e=> set("hours", Number(e.target.value))}/>  
 </div>  
 <div>  
 <Label>Service Interval (hrs)</Label>  
 <Input type="number" value={form.serviceInterval} onChange={e=> set("serviceInterval", Number(e.target.value))}/>  
 </div>  
 <div>  
 <Label>Last Service at (hrs)</Label>  
 <Input type="number" value={form.lastServiceHours} onChange={e=> set("lastServiceHours", Number(e.target.value))}/>  
 </div>  
 <div>  
 <Label>Assigned To</Label>  
 <Input value={form.assignedTo || ""} onChange={e=> set("assignedTo", e.target.value)}/>  
 </div>  
 <div>  
 <Label>Location</Label>  
 <Input value={form.location || ""} onChange={e=> set("location", e.target.value)}/>  
 </div>  
 <div className="md:col-span-2">  
 <Label>Notes</Label>  
 <Textarea value={form.notes || ""} onChange={e=> set("notes", e.target.value)} rows={3}/>  
 </div>  
 </div>  
 <div className="flex justify-end gap-2 mt-4">  
 <Button variant="secondary" onClick={onClose}>Cancel</Button>  
 <Button onClick={submit}><Save className="h-4 w-4 mr-2"/>Save</Button>  
 </div>  
 </DialogContent>  
 </Dialog>  
 );  
}

**How to use (2 minutes):** 1. Add this file to your app and route to it (e.g., /readiness).  
2. Ensure Tailwind and shadcn are configured; install lucide-react, framer-motion, qrcode.react, recharts.  
3. Open the page → import your CSV or start adding assets.  
4. Use **Print** for paper checklists; export CSV to share offline.

**Data portability:** CSV headers: id,name,type,status,hours,serviceInterval,lastServiceHours,assignedTo,location,notes.

**Accessibility & Safety:** Keyboard accessible controls, readable print view, and checklists for chainsaw/generator/pump/radio safety. Update to match your SOPs.