```
import { useState, useEffect, useCallback } from "react";

const API = "http://localhost:8000";

const STATUS_COLOR = {
  "確認済み": "#3fb950", "未確認・有力": "#f0b83e",
  "配布済み（追加あり？）": "#58a6ff", "配布済み・高額実績": "#58a6ff",
  "テストネット中": "#a78bfa", "メインネット間近": "#ff6b6b", "未確認": "#6b7a8d",
};
const URGENCY_COLOR = { "高": "#ff6b6b", "中": "#f0b83e", "低": "#3fb950" };
const CATEGORIES = ["全て", "L1", "L2", "DeFi", "インフラ"];

function totalProgress(tasks = []) {
  const done = tasks.filter(t => t.done).reduce((s, t) => s + t.points, 0);
  const total = tasks.reduce((s, t) => s + t.points, 0);
  return { done, total, pct: total > 0 ? Math.round(done / total * 100) : 0 };
}

// ---- Add Airdrop Modal ----
function AddModal({ onClose, onSave }) {
  const [form, setForm] = useState({
    name: "", symbol: "", logo: "🪂", category: "DeFi", chain: "",
    status: "未確認", urgency: "中", deadline: "未定",
    estimated_value: "未定", difficulty: 1, description: "", twitter: "", confirme
  });
  const set = (k, v) => setForm(f => ({ ...f, [k]: v }));
  return (
    <div style={{ position: "fixed", inset: 0, background: "rgba(0,0,0,0.7)", zIn
      <div style={{ background: "#0a1628", border: "1px solid #1e2d42", borderRad
        <div style={{ fontSize: 16, fontWeight: 800, color: "#fff", marginBottom:
        {[
          ["プロジェクト名", "name", "text"], ["シンボル", "symbol", "text"], ["ロゴ（
          ["チェーン", "chain", "text"], ["期限", "deadline", "text"], ["推定価値", '
          ["Twitter", "twitter", "text"], ["説明", "description", "text"],
        ].map(([label, key, type]) => (
          <div key={key} style={{ marginBottom: 12 }}>
            <div style={{ fontSize: 10, color: "#3d5876", marginBottom: 4 }}>{lab
            <input value={form[key]} onChange={e => set(key, e.target.value)} sty
              width: "100%", background: "#05080f", border: "1px solid #1e2d42",
              padding: "8px 12px", color: "#c8d6e5", fontSize: 12, fontFamily: "i
            }} />
          </div>
        ))}
        {[
```

```
          ["カテゴリ", "category", ["L1","L2","DeFi","インフラ","その他"]],
          ["緊急度", "urgency", ["高","中","低"]],
          ["ステータス", "status", ["確認済み","未確認・有力","テストネット中","メインネッ
        ].map(([label, key, opts]) => (
          <div key={key} style={{ marginBottom: 12 }}>
            <div style={{ fontSize: 10, color: "#3d5876", marginBottom: 4 }}>{lab
            <select value={form[key]} onChange={e => set(key, e.target.value)} st
              width: "100%", background: "#05080f", border: "1px solid #1e2d42",
              padding: "8px 12px", color: "#c8d6e5", fontSize: 12, fontFamily: "i
            }}>
              {opts.map(o => <option key={o}>{o}</option>)}
            </select>
          </div>
        ))}
        <div style={{ display: "flex", gap: 10, marginTop: 20 }}>
          <button onClick={onClose} style={{ flex: 1, padding: 10, background: "t
          <button onClick={() => onSave(form)} style={{ flex: 2, padding: 10, bac
        </div>
      </div>
    </div>
  );
}

export default function AirdropDashboard() {
  const [airdrops, setAirdrops] = useState([]);
  const [stats, setStats] = useState({});
  const [notifications, setNotifications] = useState([]);
  const [selected, setSelected] = useState(null);
  const [category, setCategory] = useState("全て");
  const [search, setSearch] = useState("");
  const [sortBy, setSortBy] = useState("urgency");
  const [showNotif, setShowNotif] = useState(false);
  const [showAdd, setShowAdd] = useState(false);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [newTaskLabel, setNewTaskLabel] = useState("");

  const fetchAll = useCallback(async () => {
    try {
      const [ad, st, nf] = await Promise.all([
        fetch(`${API}/airdrops`).then(r => r.json()),
        fetch(`${API}/stats`).then(r => r.json()),
        fetch(`${API}/notifications`).then(r => r.json()),
      ]);
      setAirdrops(ad);
      setStats(st);
      setNotifications(nf);
```

```
      setError(null);
    } catch (e) {
      setError("APIサーバーに接続できません。バックエンドを起動してください。");
    } finally {
      setLoading(false);
    }
  }, []);

  useEffect(() => { fetchAll(); }, [fetchAll]);

  // タスクのチェックを切り替え
  const toggleTask = async (taskId, currentDone) => {
    await fetch(`${API}/tasks/${taskId}`, {
      method: "PATCH",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ done: !currentDone }),
    });
    fetchAll();
  };

  // タスク追加
  const addTask = async (airdropId) => {
    if (!newTaskLabel.trim()) return;
    await fetch(`${API}/airdrops/${airdropId}/tasks`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ label: newTaskLabel, points: 20 }),
    });
    setNewTaskLabel("");
    fetchAll();
  };

  // エアドロ追加
  const saveAirdrop = async (form) => {
    await fetch(`${API}/airdrops`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(form),
    });
    setShowAdd(false);
    fetchAll();
  };

  // エアドロ削除
  const deleteAirdrop = async (id) => {
    if (!confirm("削除しますか？")) return;
    await fetch(`${API}/airdrops/${id}`, { method: "DELETE" });
```

```
  setSelected(null);
  fetchAll();
};

const filtered = airdrops
  .filter(a => category === "全て" || a.category === category)
  .filter(a => a.name.toLowerCase().includes(search.toLowerCase()) || (a.symbol
  .sort((a, b) => {
    if (sortBy === "urgency") return ["高","中","低"].indexOf(a.urgency) - ["高",
    if (sortBy === "progress") {
      return totalProgress(b.tasks).pct - totalProgress(a.tasks).pct;
    }
    return 0;
  });

const sel = selected ? airdrops.find(a => a.id === selected) : null;
const unreadCount = notifications.filter(n => !n.read).length;

if (loading) return (
  <div style={{ background: "#05080f", minHeight: "100vh", display: "flex", ali
    🍄 データ読み込み中...
  </div>
);

return (
  <div style={{ background: "#05080f", minHeight: "100vh", color: "#c8d6e5", fo
    <style>{`
      @import url('https://fonts.googleapis.com/css2?family=Syne:wght@400;600;7
      * { box-sizing: border-box; }
      ::-webkit-scrollbar { width: 4px; } ::-webkit-scrollbar-track { backgroun
      @keyframes glow { 0%,100%{box-shadow:0 0 8px rgba(255,107,107,0.3)} 50%{b
      @keyframes slideIn { from{opacity:0;transform:translateY(-8px)} to{opacit
    `}</style>

    {/* Error banner */}
    {error && (
      <div style={{ background: "rgba(248,81,73,0.1)", borderBottom: "1px solid
        ⚠️ {error} → <code style={{ background: "#0a1628", padding: "2px 8px",
      </div>
    )}

    {/* Header */}
    <div style={{ background: "#0a1628", borderBottom: "1px solid #1e2d42", pad
      <div style={{ display: "flex", alignItems: "center", gap: 12 }}>
        <div style={{ fontSize: 22, width: 36, height: 36, display: "flex", ali
        <div>
          <div style={{ fontSize: 16, fontWeight: 800, color: "#fff", letterSpa
```

```
        <div style={{ fontSize: 9, color: "#3d5876", letterSpacing: 2 }}>SQLi
      </div>
    </div>
    <div style={{ display: "flex", gap: 12, alignItems: "center" }}>
      <input value={search} onChange={e => setSearch(e.target.value)} placeho
      <button onClick={() => setShowAdd(true)} style={{ background: "rgba(63,
      {/* Bell */}
      <div style={{ position: "relative" }}>
        <button onClick={() => setShowNotif(s => !s)} style={{ background: "t
          🔔
          {unreadCount > 0 && <span style={{ position: "absolute", top: -4, r
        </button>
        {showNotif && (
          <div style={{ position: "absolute", right: 0, top: 44, width: 300,
            {notifications.map(n => (
              <div key={n.id} onClick={async () => { await fetch(`${API}/noti
                <span>{n.type === "urgent" ? "🚨" : n.type === "warning" ? "⚠
                <div>
                  <div style={{ fontSize: 11, color: "#c8d6e5", lineHeight: 1
                  <div style={{ fontSize: 9, color: "#3d5876", marginTop: 3 }
                </div>
              </div>
            ))}
          </div>
        )}
      </div>
    </div>
  </div>

  <div style={{ display: "grid", gridTemplateColumns: sel ? "1fr 400px" : "1f
    <div style={{ padding: "24px 28px" }}>
      {/* Stats */}
      <div style={{ display: "grid", gridTemplateColumns: "repeat(4,1fr)", ga
        {[
          { label: "追跡中", value: stats.total, unit: "件", icon: "📡", color
          { label: "アクティブ", value: stats.active, unit: "件", icon: "✅", c
          { label: "緊急", value: stats.urgent, unit: "件", icon: "🚨", color:
          { label: "平均進捗", value: stats.avg_progress, unit: "%", icon: "📊
        ].map((s, i) => (
          <div key={i} style={{ background: "#0a1628", border: `1px solid ${s
            <div style={{ display: "flex", justifyContent: "space-between" }}
              <div>
                <div style={{ fontSize: 10, color: "#3d5876", letterSpacing:
                <div style={{ fontSize: 26, fontWeight: 800, color: s.color }
              </div>
              <span style={{ fontSize: 20 }}>{s.icon}</span>
            </div>
```

```
    </div>
  ))}
</div>

{/* Filters */}
<div style={{ display: "flex", gap: 10, marginBottom: 20, justifyConten
  <div style={{ display: "flex", gap: 6 }}>
    {CATEGORIES.map(c => (
      <button key={c} onClick={() => setCategory(c)} style={{ backgroun
    ))}
  </div>
  <select value={sortBy} onChange={e => setSortBy(e.target.value)} styl
    <option value="urgency">緊急度順</option>
    <option value="progress">進捗順</option>
  </select>
</div>

{/* Cards */}
<div style={{ display: "grid", gridTemplateColumns: sel ? "1fr" : "repe
  {filtered.map(a => {
    const prog = totalProgress(a.tasks);
    const isSelected = selected === a.id;
    return (
      <div key={a.id} onClick={() => setSelected(isSelected ? null : a.
        <div style={{ display: "flex", justifyContent: "space-between",
          <div style={{ display: "flex", gap: 10, alignItems: "center"
            <div style={{ width: 40, height: 40, borderRadius: 10, font
            <div>
              <div style={{ fontSize: 15, fontWeight: 800, color: "#fff
              <div style={{ fontSize: 10, color: "#3d5876" }}>${a.symbo
            </div>
          </div>
          <div style={{ display: "flex", flexDirection: "column", gap:
            <span style={{ fontSize: 10, padding: "2px 8px", borderRadi
            <span style={{ fontSize: 9, padding: "2px 8px", borderRadiu
          </div>
        </div>
        <div style={{ fontSize: 11, color: "#6b7a8d", marginBottom: 12,
        <div>
          <div style={{ display: "flex", justifyContent: "space-between
            <span style={{ fontSize: 10, color: "#3d5876" }}>タスク進捗</
            <span style={{ fontSize: 11, color: prog.pct >= 75 ? "#3fb9
          </div>
          <div style={{ height: 5, background: "#1e2d42", borderRadius:
            <div style={{ height: "100%", borderRadius: 3, width: `${pr
          </div>
        </div>
```

```
          <div style={{ display: "flex", justifyContent: "space-between",
            <span style={{ fontSize: 10, color: "#3d5876" }}>期限: {a.deac
            <span style={{ fontSize: 10, color: "#a78bfa" }}>{a.estimated
          </div>
        </div>
      );
    })}
  </div>
</div>

{/* Detail panel */}
{sel && (
  <div style={{ background: "#0a1628", borderLeft: "1px solid #1e2d42", p
    <div style={{ display: "flex", justifyContent: "space-between", margi
      <div style={{ display: "flex", gap: 12, alignItems: "center" }}>
        <div style={{ fontSize: 28 }}>{sel.logo}</div>
        <div>
          <div style={{ fontSize: 18, fontWeight: 800, color: "#fff" }}>{
          <div style={{ fontSize: 11, color: "#3d5876" }}>{sel.twitter}</
        </div>
      </div>
      <div style={{ display: "flex", gap: 8 }}>
        <button onClick={() => deleteAirdrop(sel.id)} style={{ background
        <button onClick={() => setSelected(null)} style={{ background: "t
      </div>
    </div>

    {/* Info */}
    <div style={{ background: "#05080f", borderRadius: 10, padding: 14, m
      {[["カテゴリ",sel.category],["チェーン",sel.chain],["ステータス",sel.sta
        <div key={k} style={{ display: "flex", justifyContent: "space-bet
          <span style={{ color: "#3d5876" }}>{k}</span>
          <span style={{ color: "#c8d6e5", fontWeight: 600 }}>{v}</span>
        </div>
      ))}
    </div>

    {/* Tasks */}
    <div style={{ marginBottom: 16 }}>
      <div style={{ fontSize: 10, color: "#3d5876", letterSpacing: 2, mar
      {sel.tasks?.map(task => (
        <div key={task.id} onClick={() => toggleTask(task.id, task.done)}
          <div style={{ width: 20, height: 20, borderRadius: 5, flexShrin
          <span style={{ flex: 1, fontSize: 12, color: task.done ? "#3fb9
          <span style={{ fontSize: 10, padding: "2px 6px", borderRadius:
        </div>
      ))}
```

```
              {/* Add task */}
              <div style={{ display: "flex", gap: 8, marginTop: 10 }}>
                <input value={newTaskLabel} onChange={e => setNewTaskLabel(e.targ
                  onKeyDown={e => e.key === "Enter" && addTask(sel.id)}
                  placeholder="新しいタスクを追加..."
                  style={{ flex: 1, background: "#05080f", border: "1px solid #1e
                <button onClick={() => addTask(sel.id)} style={{ background: "rgb
              </div>
            </div>

            {/* Progress summary */}
            {(() => {
              const prog = totalProgress(sel.tasks);
              return (
                <div style={{ background: "#05080f", borderRadius: 10, padding: 1
                  <div style={{ display: "flex", justifyContent: "space-between",
                    <span style={{ fontSize: 11, color: "#3d5876" }}>総合スコア</sp
                    <span style={{ fontSize: 16, fontWeight: 800, color: prog.pct
                  </div>
                  <div style={{ height: 8, background: "#1e2d42", borderRadius: 4
                    <div style={{ height: "100%", borderRadius: 4, width: `${prog
                  </div>
                  <div style={{ fontSize: 11, color: "#3d5876", marginTop: 8, tex
                    {prog.pct >= 80 ? "🎉 エアドロ対象の可能性が高い！" : prog.pct >=
                  </div>
                </div>
              );
            })()}
          </div>
        )}
      </div>

      {showAdd && <AddModal onClose={() => setShowAdd(false)} onSave={saveAirdrop
    </div>
  );
}
```