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

const DEFAULT_NON_NEGOTIABLES = {
  morning: [
    { id: "nn-1", text: "Wake up by 6:00 AM", deadline: "06:00", category: "morni
    { id: "nn-2", text: "Gym / Workout (45 min)", deadline: "08:00", category: "m
    { id: "nn-3", text: "Cold shower", deadline: "08:30", category: "morning" },
    { id: "nn-4", text: "Pray / Meditate (10 min)", deadline: "09:00", category:
  ],
  health: [
    { id: "nn-5", text: "Drink 1 gallon of water", deadline: "21:00", category: "
    { id: "nn-6", text: "Take supplements / vitamins", deadline: "09:00", categor
    { id: "nn-7", text: "Eat a clean meal (no junk)", deadline: "20:00", category
    { id: "nn-8", text: "10,000 steps", deadline: "21:00", category: "health" },
  ],
  evening: [
    { id: "nn-9", text: "Journal / Reflect on the day", deadline: "22:00", catego
    { id: "nn-10", text: "Review tomorrow's plan", deadline: "22:00", category: "
    { id: "nn-11", text: "No screens after 10 PM", deadline: "22:00", category: "
    { id: "nn-12", text: "Lights out by 10:30 PM", deadline: "22:30", category: "
  ],
};

const CATEGORY_CONFIG = {
  morning: { label: "Morning Routine", icon: "☀️", accent: "#F59E0B" },
  health: { label: "Health & Body", icon: "💪", accent: "#10B981" },
  evening: { label: "Evening Wind-Down", icon: "🌙", accent: "#8B5CF6" },
};

const STORAGE_KEYS = {
  nonNegotiables: "zm-non-negotiables",
  dailyTodos: "zm-daily-todos",
  weeklyTodos: "zm-weekly-todos",
  completions: "zm-completions",
  initialized: "zm-initialized",
};

function getToday() {
  return new Date().toISOString().split("T")[0];
}

function getWeekId() {
  const d = new Date();
  const start = new Date(d.getFullYear(), 0, 1);
```

```
  const diff = d - start;
  const week = Math.ceil(diff / (7 * 24 * 60 * 60 * 1000));
  return `${d.getFullYear()}-W${week}`;
}

function genId() {
  return "id-" + Date.now() + "-" + Math.random().toString(36).slice(2, 7);
}

function timeToMinutes(t) {
  if (!t) return null;
  const [h, m] = t.split(":").map(Number);
  return h * 60 + m;
}

function nowMinutes() {
  const n = new Date();
  return n.getHours() * 60 + n.getMinutes();
}

function formatTime12(t) {
  if (!t) return "";
  const [h, m] = t.split(":").map(Number);
  const ampm = h >= 12 ? "PM" : "AM";
  const hr = h % 12 || 12;
  return `${hr}:${m.toString().padStart(2, "0")} ${ampm}`;
}

// Persistent storage helper
function loadData(key, fallback) {
  try {
    const raw = localStorage.getItem(key);
    return raw ? JSON.parse(raw) : fallback;
  } catch {
    return fallback;
  }
}

function saveData(key, data) {
  try {
    localStorage.setItem(key, JSON.stringify(data));
  } catch {}
}

// ——— Components ———

function CheckItem({ item, checked, onToggle, accent, overdue }) {
```

```
return (
  <div
    onClick={onToggle}
    style={{
      display: "flex",
      alignItems: "center",
      gap: 12,
      padding: "14px 16px",
      background: checked ? "rgba(255,255,255,0.03)" : "rgba(255,255,255,0.06)"
      borderRadius: 12,
      cursor: "pointer",
      transition: "all 0.2s",
      border: overdue && !checked ? `1px solid #EF4444` : "1px solid transparen
      opacity: checked ? 0.55 : 1,
      position: "relative",
      overflow: "hidden",
    }}
  >
    <div
      style={{
        width: 24,
        height: 24,
        borderRadius: 7,
        border: checked ? `2px solid ${accent}` : "2px solid rgba(255,255,255,0
        background: checked ? accent : "transparent",
        display: "flex",
        alignItems: "center",
        justifyContent: "center",
        flexShrink: 0,
        transition: "all 0.2s",
      }}
    >
      {checked && (
        <svg width="14" height="14" viewBox="0 0 14 14" fill="none">
          <path d="M3 7.5L5.5 10L11 4" stroke="#000" strokeWidth="2.2" strokeLi
        </svg>
      )}
    </div>
    <div style={{ flex: 1 }}>
      <div
        style={{
          fontSize: 15,
          fontWeight: 500,
          color: checked ? "rgba(255,255,255,0.4)" : "#fff",
          textDecoration: checked ? "line-through" : "none",
          fontFamily: "'DM Sans', sans-serif",
        }}
```

```
        >
          {item.text}
        </div>
        {item.deadline && (
          <div
            style={{
              fontSize: 12,
              color: overdue && !checked ? "#EF4444" : "rgba(255,255,255,0.35)",
              marginTop: 3,
              fontFamily: "'JetBrains Mono', monospace",
            }}
          >
            {overdue && !checked ? "⚠ OVERDUE — " : "Due by "}
            {formatTime12(item.deadline)}
          </div>
        )}
      </div>
      {overdue && !checked && (
        <div
          style={{
            width: 8,
            height: 8,
            borderRadius: 4,
            background: "#EF4444",
            animation: "pulse 1.5s infinite",
            flexShrink: 0,
          }}
        />
      )}
    </div>
  );
}

function AddItemModal({ onAdd, onClose, showDeadline = true, showCategory = false
  const [text, setText] = useState("");
  const [deadline, setDeadline] = useState("");
  const [category, setCategory] = useState("morning");
  const inputRef = useRef(null);

  useEffect(() => {
    setTimeout(() => inputRef.current?.focus(), 100);
  }, []);

  return (
    <div
      style={{
        position: "fixed",
```

```
    inset: 0,
    background: "rgba(0,0,0,0.75)",
    backdropFilter: "blur(8px)",
    zIndex: 1000,
    display: "flex",
    alignItems: "flex-end",
    justifyContent: "center",
    padding: 16,
  }}
  onClick={onClose}
>
  <div
    onClick={(e) => e.stopPropagation()}
    style={{
      background: "#1A1A1E",
      borderRadius: 20,
      padding: 24,
      width: "100%",
      maxWidth: 400,
      border: "1px solid rgba(255,255,255,0.1)",
    }}
  >
    <div style={{ fontSize: 18, fontWeight: 700, color: "#fff", marginBottom:
      Add New Task
    </div>
    <input
      ref={inputRef}
      value={text}
      onChange={(e) => setText(e.target.value)}
      placeholder="What needs to get done?"
      style={{
        width: "100%",
        padding: "14px 16px",
        borderRadius: 12,
        border: "1px solid rgba(255,255,255,0.15)",
        background: "rgba(255,255,255,0.06)",
        color: "#fff",
        fontSize: 15,
        outline: "none",
        fontFamily: "'DM Sans', sans-serif",
        boxSizing: "border-box",
      }}
    />
    {showDeadline && (
      <div style={{ marginTop: 12 }}>
        <label style={{ fontSize: 12, color: "rgba(255,255,255,0.5)", fontFam
          Deadline (optional)
```

```
    </label>
    <input
      type="time"
      value={deadline}
      onChange={(e) => setDeadline(e.target.value)}
      style={{
        width: "100%",
        padding: "12px 16px",
        borderRadius: 12,
        border: "1px solid rgba(255,255,255,0.15)",
        background: "rgba(255,255,255,0.06)",
        color: "#fff",
        fontSize: 15,
        outline: "none",
        marginTop: 6,
        fontFamily: "'JetBrains Mono', monospace",
        boxSizing: "border-box",
        colorScheme: "dark",
      }}
    />
  </div>
)}
{showCategory && (
  <div style={{ marginTop: 12 }}>
    <label style={{ fontSize: 12, color: "rgba(255,255,255,0.5)", fontFam
      Category
    </label>
    <div style={{ display: "flex", gap: 8, marginTop: 6 }}>
      {Object.entries(CATEGORY_CONFIG).map(([key, cfg]) => (
        <button
          key={key}
          onClick={() => setCategory(key)}
          style={{
            flex: 1,
            padding: "10px 8px",
            borderRadius: 10,
            border: category === key ? `2px solid ${cfg.accent}` : "1px s
            background: category === key ? `${cfg.accent}15` : "transpare
            color: category === key ? cfg.accent : "rgba(255,255,255,0.5)
            fontSize: 12,
            cursor: "pointer",
            fontFamily: "'DM Sans', sans-serif",
          }}
        >
          {cfg.icon} {cfg.label.split(" ")[0]}
        </button>
      ))}
```

```
            </div>
          </div>
        )}
        <div style={{ display: "flex", gap: 10, marginTop: 20 }}>
          <button
            onClick={onClose}
            style={{
              flex: 1,
              padding: 14,
              borderRadius: 12,
              border: "1px solid rgba(255,255,255,0.15)",
              background: "transparent",
              color: "rgba(255,255,255,0.6)",
              fontSize: 15,
              cursor: "pointer",
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            Cancel
          </button>
          <button
            onClick={() => {
              if (text.trim()) {
                onAdd({ text: text.trim(), deadline: deadline || null, category }
                onClose();
              }
            }}
            style={{
              flex: 1,
              padding: 14,
              borderRadius: 12,
              border: "none",
              background: "linear-gradient(135deg, #F59E0B, #EF4444)",
              color: "#000",
              fontSize: 15,
              fontWeight: 700,
              cursor: "pointer",
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            Add Task
          </button>
        </div>
      </div>
    </div>
  );
}
```

```
function ConfirmModal({ message, onConfirm, onCancel }) {
  return (
    <div
      style={{
        position: "fixed",
        inset: 0,
        background: "rgba(0,0,0,0.75)",
        backdropFilter: "blur(8px)",
        zIndex: 1000,
        display: "flex",
        alignItems: "center",
        justifyContent: "center",
        padding: 16,
      }}
      onClick={onCancel}
    >
      <div
        onClick={(e) => e.stopPropagation()}
        style={{
          background: "#1A1A1E",
          borderRadius: 20,
          padding: 24,
          width: "100%",
          maxWidth: 340,
          border: "1px solid rgba(255,255,255,0.1)",
          textAlign: "center",
        }}
      >
        <div style={{ fontSize: 16, color: "#fff", marginBottom: 20, fontFamily:
          {message}
        </div>
        <div style={{ display: "flex", gap: 10 }}>
          <button
            onClick={onCancel}
            style={{
              flex: 1, padding: 12, borderRadius: 12, border: "1px solid rgba(255
              background: "transparent", color: "rgba(255,255,255,0.6)", fontSize
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            Cancel
          </button>
          <button
            onClick={onConfirm}
            style={{
              flex: 1, padding: 12, borderRadius: 12, border: "none",
```

```
              background: "#EF4444", color: "#fff", fontSize: 14, fontWeight: 700
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            Delete
          </button>
        </div>
      </div>
    </div>
  );
}

function ProgressRing({ percent, size = 56, accent = "#F59E0B" }) {
  const r = (size - 8) / 2;
  const circ = 2 * Math.PI * r;
  const offset = circ - (percent / 100) * circ;
  return (
    <svg width={size} height={size}>
      <circle cx={size / 2} cy={size / 2} r={r} fill="none" stroke="rgba(255,255,
      <circle
        cx={size / 2} cy={size / 2} r={r} fill="none" stroke={accent} strokeWidth
        strokeDasharray={circ} strokeDashoffset={offset} strokeLinecap="round"
        transform={`rotate(-90 ${size / 2} ${size / 2})`}
        style={{ transition: "stroke-dashoffset 0.6s ease" }}
      />
      <text x={size / 2} y={size / 2} textAnchor="middle" dominantBaseline="centr
        fill="#fff" fontSize={14} fontWeight="700" fontFamily="'JetBrains Mono',
        {Math.round(percent)}%
      </text>
    </svg>
  );
}

function EditItemModal({ item, onSave, onClose, showDeadline = true, showCategory
  const [text, setText] = useState(item.text || "");
  const [deadline, setDeadline] = useState(item.deadline || "");
  const [category, setCategory] = useState(item.category || "morning");
  const inputRef = useRef(null);

  useEffect(() => {
    setTimeout(() => inputRef.current?.focus(), 100);
  }, []);

  return (
    <div
      style={{
        position: "fixed",
```

```
    inset: 0,
    background: "rgba(0,0,0,0.75)",
    backdropFilter: "blur(8px)",
    zIndex: 1000,
    display: "flex",
    alignItems: "flex-end",
    justifyContent: "center",
    padding: 16,
  }}
  onClick={onClose}
>
  <div
    onClick={(e) => e.stopPropagation()}
    style={{
      background: "#1A1A1E",
      borderRadius: 20,
      padding: 24,
      width: "100%",
      maxWidth: 400,
      border: "1px solid rgba(255,255,255,0.1)",
    }}
  >
    <div style={{ fontSize: 18, fontWeight: 700, color: "#fff", marginBottom:
      Edit Task
    </div>
    <input
      ref={inputRef}
      value={text}
      onChange={(e) => setText(e.target.value)}
      placeholder="Task name"
      style={{
        width: "100%",
        padding: "14px 16px",
        borderRadius: 12,
        border: "1px solid rgba(255,255,255,0.15)",
        background: "rgba(255,255,255,0.06)",
        color: "#fff",
        fontSize: 15,
        outline: "none",
        fontFamily: "'DM Sans', sans-serif",
        boxSizing: "border-box",
      }}
    />
    {showDeadline && (
      <div style={{ marginTop: 12 }}>
        <label style={{ fontSize: 12, color: "rgba(255,255,255,0.5)", fontFam
          Deadline (optional)
```

```
    </label>
    <div style={{ display: "flex", gap: 8, alignItems: "center", marginTo
      <input
        type="time"
        value={deadline}
        onChange={(e) => setDeadline(e.target.value)}
        style={{
          flex: 1,
          padding: "12px 16px",
          borderRadius: 12,
          border: "1px solid rgba(255,255,255,0.15)",
          background: "rgba(255,255,255,0.06)",
          color: "#fff",
          fontSize: 15,
          outline: "none",
          fontFamily: "'JetBrains Mono', monospace",
          boxSizing: "border-box",
          colorScheme: "dark",
        }}
      />
      {deadline && (
        <button
          onClick={() => setDeadline("")}
          style={{
            padding: "12px 14px", borderRadius: 12, border: "1px solid rg
            background: "rgba(239,68,68,0.1)", color: "#EF4444", fontSize
            fontFamily: "'DM Sans', sans-serif", fontWeight: 600, whiteSp
          }}
        >
          Clear
        </button>
      )}
    </div>
  </div>
)}
{showCategory && (
  <div style={{ marginTop: 12 }}>
    <label style={{ fontSize: 12, color: "rgba(255,255,255,0.5)", fontFam
      Category
    </label>
    <div style={{ display: "flex", gap: 8, marginTop: 6 }}>
      {Object.entries(CATEGORY_CONFIG).map(([key, cfg]) => (
        <button
          key={key}
          onClick={() => setCategory(key)}
          style={{
            flex: 1,
```

```
              padding: "10px 8px",
              borderRadius: 10,
              border: category === key ? `2px solid ${cfg.accent}` : "1px s
              background: category === key ? `${cfg.accent}15` : "transpare
              color: category === key ? cfg.accent : "rgba(255,255,255,0.5)
              fontSize: 12,
              cursor: "pointer",
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            {cfg.icon} {cfg.label.split(" ")[0]}
          </button>
        ))}
      </div>
    </div>
  )}
  <div style={{ display: "flex", gap: 10, marginTop: 20 }}>
    <button
      onClick={onClose}
      style={{
        flex: 1,
        padding: 14,
        borderRadius: 12,
        border: "1px solid rgba(255,255,255,0.15)",
        background: "transparent",
        color: "rgba(255,255,255,0.6)",
        fontSize: 15,
        cursor: "pointer",
        fontFamily: "'DM Sans', sans-serif",
      }}
    >
      Cancel
    </button>
    <button
      onClick={() => {
        if (text.trim()) {
          onSave({ ...item, text: text.trim(), deadline: deadline || null,
          onClose();
        }
      }}
      style={{
        flex: 1,
        padding: 14,
        borderRadius: 12,
        border: "none",
        background: "linear-gradient(135deg, #3B82F6, #8B5CF6)",
        color: "#fff",
```

```
              fontSize: 15,
              fontWeight: 700,
              cursor: "pointer",
              fontFamily: "'DM Sans', sans-serif",
            }}
          >
            Save
          </button>
        </div>
      </div>
    </div>
  );
}

// ─── Main App ───

export default function DisciplineApp() {
  const [tab, setTab] = useState("daily");
  const [nonNegotiables, setNonNegotiables] = useState([]);
  const [dailyTodos, setDailyTodos] = useState([]);
  const [weeklyTodos, setWeeklyTodos] = useState([]);
  const [completions, setCompletions] = useState({});
  const [showAddModal, setShowAddModal] = useState(null); // 'nn', 'daily', 'week
  const [deleteTarget, setDeleteTarget] = useState(null);
  const [editTarget, setEditTarget] = useState(null); // { item, type: 'nn'|'dail
  const [editMode, setEditMode] = useState(false);
  const [now, setNow] = useState(nowMinutes());

  const today = getToday();
  const weekId = getWeekId();

  // Initialize
  useEffect(() => {
    const inited = loadData(STORAGE_KEYS.initialized, false);
    if (!inited) {
      const allNN = [...DEFAULT_NON_NEGOTIABLES.morning, ...DEFAULT_NON_NEGOTIABL
      saveData(STORAGE_KEYS.nonNegotiables, allNN);
      saveData(STORAGE_KEYS.dailyTodos, []);
      saveData(STORAGE_KEYS.weeklyTodos, []);
      saveData(STORAGE_KEYS.completions, {});
      saveData(STORAGE_KEYS.initialized, true);
      setNonNegotiables(allNN);
    } else {
      setNonNegotiables(loadData(STORAGE_KEYS.nonNegotiables, []));
      setDailyTodos(loadData(STORAGE_KEYS.dailyTodos, []));
      setWeeklyTodos(loadData(STORAGE_KEYS.weeklyTodos, []));
      setCompletions(loadData(STORAGE_KEYS.completions, {}));
```

```
  }
}, []);

// Tick clock for overdue checks
useEffect(() => {
  const i = setInterval(() => setNow(nowMinutes()), 30000);
  return () => clearInterval(i);
}, []);

// Request notification permission
useEffect(() => {
  if ("Notification" in window && Notification.permission === "default") {
    Notification.requestPermission();
  }
}, []);

// Reminder checker
useEffect(() => {
  const check = () => {
    const current = nowMinutes();
    const todayKey = getToday();
    const allItems = [...nonNegotiables, ...dailyTodos];
    allItems.forEach((item) => {
      if (!item.deadline) return;
      const deadlineMin = timeToMinutes(item.deadline);
      const diff = deadlineMin - current;
      const cKey = `${todayKey}-${item.id}`;
      const isComplete = completions[cKey];
      if (!isComplete && diff > 0 && diff <= 15) {
        if ("Notification" in window && Notification.permission === "granted")
          new Notification("⚠️ Task Due Soon!", {
            body: `"${item.text}" is due at ${formatTime12(item.deadline)}`,
            icon: "💥",
            tag: item.id,
          });
        }
      }
    });
  };
  const i = setInterval(check, 60000);
  check();
  return () => clearInterval(i);
}, [nonNegotiables, dailyTodos, completions]);

const saveCompletions = (c) => {
  setCompletions(c);
  saveData(STORAGE_KEYS.completions, c);
```

```
};

const toggleComplete = (id, scope = "daily") => {
  const key = scope === "weekly" ? `${weekId}-${id}` : `${today}-${id}`;
  const next = { ...completions, [key]: !completions[key] };
  if (!next[key]) delete next[key];
  saveCompletions(next);
};

const isComplete = (id, scope = "daily") => {
  const key = scope === "weekly" ? `${weekId}-${id}` : `${today}-${id}`;
  return !!completions[key];
};

const addNonNeg = (item) => {
  const nn = [...nonNegotiables, { ...item, id: genId() }];
  setNonNegotiables(nn);
  saveData(STORAGE_KEYS.nonNegotiables, nn);
};

const addDaily = (item) => {
  const d = [...dailyTodos, { ...item, id: genId() }];
  setDailyTodos(d);
  saveData(STORAGE_KEYS.dailyTodos, d);
};

const addWeekly = (item) => {
  const w = [...weeklyTodos, { ...item, id: genId() }];
  setWeeklyTodos(w);
  saveData(STORAGE_KEYS.weeklyTodos, w);
};

const deleteItem = (id, type) => {
  if (type === "nn") {
    const nn = nonNegotiables.filter((i) => i.id !== id);
    setNonNegotiables(nn);
    saveData(STORAGE_KEYS.nonNegotiables, nn);
  } else if (type === "daily") {
    const d = dailyTodos.filter((i) => i.id !== id);
    setDailyTodos(d);
    saveData(STORAGE_KEYS.dailyTodos, d);
  } else {
    const w = weeklyTodos.filter((i) => i.id !== id);
    setWeeklyTodos(w);
    saveData(STORAGE_KEYS.weeklyTodos, w);
  }
};
```

```
const editItem = (updatedItem, type) => {
  if (type === "nn") {
    const nn = nonNegotiables.map((i) => (i.id === updatedItem.id ? updatedItem
    setNonNegotiables(nn);
    saveData(STORAGE_KEYS.nonNegotiables, nn);
  } else if (type === "daily") {
    const d = dailyTodos.map((i) => (i.id === updatedItem.id ? updatedItem : i)
    setDailyTodos(d);
    saveData(STORAGE_KEYS.dailyTodos, d);
  } else {
    const w = weeklyTodos.map((i) => (i.id === updatedItem.id ? updatedItem : i
    setWeeklyTodos(w);
    saveData(STORAGE_KEYS.weeklyTodos, w);
  }
};

// Stats
const allDailyItems = [...nonNegotiables, ...dailyTodos];
const dailyCompleted = allDailyItems.filter((i) => isComplete(i.id)).length;
const dailyPercent = allDailyItems.length ? (dailyCompleted / allDailyItems.len
const weeklyCompleted = weeklyTodos.filter((i) => isComplete(i.id, "weekly")).l
const weeklyPercent = weeklyTodos.length ? (weeklyCompleted / weeklyTodos.lengt
const overdueCount = allDailyItems.filter(
  (i) => i.deadline && timeToMinutes(i.deadline) < now && !isComplete(i.id)
).length;

const dateStr = new Date().toLocaleDateString("en-US", { weekday: "long", month

return (
  <div
    style={{
      minHeight: "100vh",
      background: "#0D0D0F",
      color: "#fff",
      fontFamily: "'DM Sans', sans-serif",
      maxWidth: 480,
      margin: "0 auto",
      paddingBottom: 100,
      position: "relative",
    }}
  >
    <style>{`
      @import url('https://fonts.googleapis.com/css2?family=DM+Sans:wght@400;50
      @keyframes pulse { 0%, 100% { opacity: 1; } 50% { opacity: 0.3; } }
      @keyframes slideUp { from { transform: translateY(20px); opacity: 0; } to
      * { box-sizing: border-box; margin: 0; padding: 0; -webkit-tap-highlight-
```

```
  body { background: #0D0D0F; }
  ::-webkit-scrollbar { display: none; }
`}</style>

{/* Header */}
<div style={{ padding: "20px 20px 0" }}>
  <div style={{ display: "flex", justifyContent: "space-between", alignItem
    <div>
      <div style={{ fontSize: 13, color: "rgba(255,255,255,0.4)", fontFamil
        {dateStr}
      </div>
      <div style={{ fontSize: 28, fontWeight: 800, marginTop: 6, letterSpac
        DAILY DISCIPLINE
      </div>
    </div>
    <button
      onClick={() => setEditMode(!editMode)}
      style={{
        padding: "8px 14px",
        borderRadius: 10,
        border: editMode ? "1px solid #EF4444" : "1px solid rgba(255,255,25
        background: editMode ? "rgba(239,68,68,0.15)" : "transparent",
        color: editMode ? "#EF4444" : "rgba(255,255,255,0.5)",
        fontSize: 12,
        cursor: "pointer",
        fontFamily: "'DM Sans', sans-serif",
        fontWeight: 600,
      }}
    >
      {editMode ? "Done" : "Edit"}
    </button>
  </div>

  {/* Stats bar */}
  <div style={{ display: "flex", gap: 12, marginTop: 20 }}>
    <div style={{ flex: 1, background: "rgba(255,255,255,0.04)", borderRadi
      <ProgressRing percent={dailyPercent} accent="#F59E0B" />
      <div>
        <div style={{ fontSize: 12, color: "rgba(255,255,255,0.4)" }}>Today
        <div style={{ fontSize: 20, fontWeight: 800, fontFamily: "'JetBrain
          {dailyCompleted}/{allDailyItems.length}
        </div>
      </div>
    </div>
    {overdueCount > 0 && (
      <div style={{ background: "rgba(239,68,68,0.1)", borderRadius: 16, pa
        <div style={{ fontSize: 28 }}>🔥</div>
```

```
        <div>
          <div style={{ fontSize: 12, color: "#EF4444" }}>Overdue</div>
          <div style={{ fontSize: 20, fontWeight: 800, color: "#EF4444", fo
            {overdueCount}
          </div>
        </div>
      </div>
    )}
  </div>

  {/* Tabs */}
  <div style={{ display: "flex", gap: 4, marginTop: 20, background: "rgba(2
    {[
      { key: "daily", label: "Daily" },
      { key: "weekly", label: "Weekly" },
    ].map((t) => (
      <button
        key={t.key}
        onClick={() => setTab(t.key)}
        style={{
          flex: 1,
          padding: "12px 0",
          borderRadius: 11,
          border: "none",
          background: tab === t.key ? "rgba(255,255,255,0.1)" : "transparen
          color: tab === t.key ? "#fff" : "rgba(255,255,255,0.4)",
          fontSize: 14,
          fontWeight: 600,
          cursor: "pointer",
          fontFamily: "'DM Sans', sans-serif",
          transition: "all 0.2s",
        }}
      >
        {t.label}
        {t.key === "weekly" && weeklyTodos.length > 0 && (
          <span style={{ marginLeft: 6, fontSize: 11, background: "rgba(139
            {weeklyCompleted}/{weeklyTodos.length}
          </span>
        )}
      </button>
    ))}
  </div>
</div>

{/* Daily Tab */}
{tab === "daily" && (
  <div style={{ padding: "16px 20px", animation: "slideUp 0.3s ease" }}>
```

```
{/* Non-Negotiables */}
<div style={{ marginBottom: 8, display: "flex", justifyContent: "space-
  <div style={{ fontSize: 11, fontWeight: 700, color: "rgba(255,255,255
    🔒 NON-NEGOTIABLES
  </div>
  {editMode && (
    <button
      onClick={() => setShowAddModal("nn")}
      style={{
        padding: "4px 12px", borderRadius: 8, border: "1px solid rgba(2
        background: "rgba(245,158,11,0.1)", color: "#F59E0B", fontSize:
        fontFamily: "'DM Sans', sans-serif", fontWeight: 600,
      }}
    >
      + Add
    </button>
  )}
</div>

{Object.entries(CATEGORY_CONFIG).map(([catKey, catCfg]) => {
  const items = nonNegotiables.filter((i) => i.category === catKey);
  if (!items.length) return null;
  return (
    <div key={catKey} style={{ marginBottom: 20 }}>
      <div style={{ fontSize: 13, fontWeight: 600, color: catCfg.accent
        {catCfg.icon} {catCfg.label}
      </div>
      <div style={{ display: "flex", flexDirection: "column", gap: 6 }}
        {items.map((item) => (
          <div key={item.id} style={{ display: "flex", alignItems: "cen
            <div style={{ flex: 1 }}>
              <CheckItem
                item={item}
                checked={isComplete(item.id)}
                onToggle={() => !editMode && toggleComplete(item.id)}
                accent={catCfg.accent}
                overdue={item.deadline && timeToMinutes(item.deadline)
              />
            </div>
            {editMode && (
              <>
                <button
                  onClick={() => setEditTarget({ item, type: "nn" })}
                  style={{
                    width: 36, height: 36, borderRadius: 10, border: "1
                    background: "rgba(59,130,246,0.1)", color: "#3B82F6
                    display: "flex", alignItems: "center", justifyConte
```

```
                  }}
                >
                  ✎
                </button>
                <button
                  onClick={() => setDeleteTarget({ id: item.id, type: "
                  style={{
                    width: 36, height: 36, borderRadius: 10, border: "1
                    background: "rgba(239,68,68,0.1)", color: "#EF4444"
                    display: "flex", alignItems: "center", justifyConte
                  }}
                >
                  ×
                </button>
              </>
            )}
          </div>
        ))}
      </div>
    </div>
  );
})}

{/* Daily Todos */}
<div style={{ marginTop: 28, marginBottom: 8, display: "flex", justifyC
  <div style={{ fontSize: 11, fontWeight: 700, color: "rgba(255,255,255
    📋 TODAY'S TO-DOS
  </div>
</div>

{dailyTodos.length === 0 ? (
  <div style={{ textAlign: "center", padding: "30px 20px", color: "rgba
    No extra tasks for today. Tap + to add one.
  </div>
) : (
  <div style={{ display: "flex", flexDirection: "column", gap: 6 }}>
    {dailyTodos.map((item) => (
      <div key={item.id} style={{ display: "flex", alignItems: "center"
        <div style={{ flex: 1 }}>
          <CheckItem
            item={item}
            checked={isComplete(item.id)}
            onToggle={() => !editMode && toggleComplete(item.id)}
            accent="#3B82F6"
            overdue={item.deadline && timeToMinutes(item.deadline) < no
          />
        </div>
```

```
              {editMode && (
                <>
                  <button
                    onClick={() => setEditTarget({ item, type: "daily" })}
                    style={{
                      width: 36, height: 36, borderRadius: 10, border: "1px s
                      background: "rgba(59,130,246,0.1)", color: "#3B82F6", f
                      display: "flex", alignItems: "center", justifyContent:
                    }}
                  >
                    ✎
                  </button>
                  <button
                    onClick={() => setDeleteTarget({ id: item.id, type: "dail
                    style={{
                      width: 36, height: 36, borderRadius: 10, border: "1px s
                      background: "rgba(239,68,68,0.1)", color: "#EF4444", fo
                      display: "flex", alignItems: "center", justifyContent:
                    }}
                  >
                    ×
                  </button>
                </>
              )}
            </div>
          ))}
        </div>
      )}
    </div>
  )}

  {/* Weekly Tab */}
  {tab === "weekly" && (
    <div style={{ padding: "16px 20px", animation: "slideUp 0.3s ease" }}>
      <div style={{ display: "flex", justifyContent: "space-between", alignIt
        <div style={{ fontSize: 11, fontWeight: 700, color: "rgba(255,255,255
          📅 THIS WEEK
        </div>
      </div>

      {weeklyTodos.length > 0 && (
        <div style={{ background: "rgba(139,92,246,0.06)", borderRadius: 14,
          <ProgressRing percent={weeklyPercent} accent="#8B5CF6" size={48} />
          <div>
            <div style={{ fontSize: 13, color: "rgba(255,255,255,0.5)" }}>Wee
            <div style={{ fontSize: 16, fontWeight: 700, fontFamily: "'JetBra
              {weeklyCompleted} of {weeklyTodos.length} done
```

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

{weeklyTodos.length === 0 ? (
  <div style={{ textAlign: "center", padding: "40px 20px", color: "rgba
    No weekly tasks yet. Tap + to add goals for this week.
  </div>
) : (
  <div style={{ display: "flex", flexDirection: "column", gap: 6 }}>
    {weeklyTodos.map((item) => (
      <div key={item.id} style={{ display: "flex", alignItems: "center"
        <div style={{ flex: 1 }}>
          <CheckItem
            item={item}
            checked={isComplete(item.id, "weekly")}
            onToggle={() => !editMode && toggleComplete(item.id, "weekl
            accent="#8B5CF6"
            overdue={false}
          />
        </div>
        {editMode && (
          <>
            <button
              onClick={() => setEditTarget({ item, type: "weekly" })}
              style={{
                width: 36, height: 36, borderRadius: 10, border: "1px s
                background: "rgba(59,130,246,0.1)", color: "#3B82F6", f
                display: "flex", alignItems: "center", justifyContent:
              }}
            >
              ✎
            </button>
            <button
              onClick={() => setDeleteTarget({ id: item.id, type: "week
              style={{
                width: 36, height: 36, borderRadius: 10, border: "1px s
                background: "rgba(239,68,68,0.1)", color: "#EF4444", fo
                display: "flex", alignItems: "center", justifyContent:
              }}
            >
              ×
            </button>
          </>
        )}
      </div>
```

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

{/* FAB */}
<button
  onClick={() => setShowAddModal(tab === "weekly" ? "weekly" : "daily")}
  style={{
    position: "fixed",
    bottom: 28,
    right: "calc(50% - 220px)",
    width: 58,
    height: 58,
    borderRadius: 18,
    border: "none",
    background: "linear-gradient(135deg, #F59E0B, #EF4444)",
    color: "#000",
    fontSize: 28,
    fontWeight: 300,
    cursor: "pointer",
    boxShadow: "0 8px 32px rgba(245,158,11,0.4)",
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
    zIndex: 100,
  }}
>
  +
</button>

{/* Modals */}
{showAddModal === "nn" && (
  <AddItemModal onAdd={addNonNeg} onClose={() => setShowAddModal(null)} sho
)}
{showAddModal === "daily" && (
  <AddItemModal onAdd={addDaily} onClose={() => setShowAddModal(null)} show
)}
{showAddModal === "weekly" && (
  <AddItemModal onAdd={addWeekly} onClose={() => setShowAddModal(null)} sho
)}
{deleteTarget && (
  <ConfirmModal
    message={`Delete "${deleteTarget.text}"?`}
    onConfirm={() => { deleteItem(deleteTarget.id, deleteTarget.type); setD
    onCancel={() => setDeleteTarget(null)}
```

```
        />
      )}
      {editTarget && (
        <EditItemModal
          item={editTarget.item}
          onSave={(updated) => editItem(updated, editTarget.type)}
          onClose={() => setEditTarget(null)}
          showDeadline={editTarget.type !== "weekly"}
          showCategory={editTarget.type === "nn"}
        />
      )}
    </div>
  );
}
```