```
import { create } from "zustand";
import type {
  UiMode,
  RunPhase,
  PanelState,
  NodeId,
  VersionStamps,
  FoundationSnapshot,
  TenCatResult,
  GssResult
} from "./types";

type SureSailState = {
  ui: UiMode;
  versions: VersionStamps;

  activeRunId: string | null;
  phase: RunPhase;
  stageIIUnlocked: boolean;

  panelStates: Record<NodeId, PanelState>;
  lastError?: { message: string; nodeId?: NodeId };

  tenCat?: TenCatResult;
  gss?: GssResult;
  foundation?: FoundationSnapshot;

  // actions
  toggleNeon: () => void;
  setAudio: (mode: UiMode["audio"]) => void;
  setVolume: (v: number) => void;
  toggleZoneOps: () => void;

  setPhase: (p: RunPhase) => void;
  setRunId: (id: string | null) => void;
  setPanelState: (node: NodeId, state: PanelState) => void;

  setTenCat: (r: TenCatResult) => void;
  setGss: (r: GssResult) => void;
  setFoundation: (f: FoundationSnapshot) => void;

  unlockStageII: () => void;
  setError: (message: string, nodeId?: NodeId) => void;
  resetForNewRun: () => void;
};

const allNodes: NodeId[] = [
  "N_CTRL",
  "N_10CAT",
  "N_GSS",
  "N_LABEL",
  "N_FOUNDATION_PROGRESS",
  "N_MICRO",
  "N_ZOOM",
  "N_HIST",
  "N_TH2",
  "N_REPLAY",
  "N_MINDWAKE",
  "N_TOLL",
  "N_STATUS",
  "N_LIVE"
];

function initPanelStates(): Record<NodeId, PanelState> {
  const obj = {} as Record<NodeId, PanelState>;
  for (const n of allNodes) obj[n] = "idle";
  // stage II starts locked
  obj["N_MICRO"] = "locked";
  obj["N_ZOOM"] = "locked";
  obj["N_HIST"] = "locked";
  obj["N_TH2"] = "locked";
  obj["N_MINDWAKE"] = "locked";
  obj["N_TOLL"] = "locked";
  obj["N_REPLAY"] = "idle"; // manual tool; visible but not locked for demo
  return obj;
}

export const useSureSailStore = create<SureSailState>((set, get) => ({
  ui: {
    neon: true,
    audio: "OFF",
    dashboardMode: "execution",
    volume: 0.35
  },
  versions: {
    engine: "v1.0",
    dataContract: "v1.0",
    timeframeCanon: "v1.0",
    thresholdRegistry: "v1.0"
  },

  activeRunId: null,
  phase: "IDLE",
  stageIIUnlocked: false,

  panelStates: initPanelStates(),

  toggleNeon: () => set((s) => ({ ui: { ...s.ui, neon: !s.ui.neon } })),
  setAudio: (mode) => set((s) => ({ ui: { ...s.ui, audio: mode } })),
  setVolume: (v) => set((s) => ({ ui: { ...s.ui, volume: Math.max(0, Math.min(1, v)) } })),
  toggleZoneOps: () =>
    set((s) => ({
      ui: { ...s.ui, dashboardMode: s.ui.dashboardMode === "execution" ? "docs" : "execution" }
    })),

  setPhase: (p) => set({ phase: p }),
  setRunId: (id) => set({ activeRunId: id }),
  setPanelState: (node, state) =>
```

```
    set((s) => ({ panelStates: { ...s.panelStates, [node]: state } })),

  setTenCat: (r) => set({ tenCat: r }),
  setGss: (r) => set({ gss: r }),
  setFoundation: (f) => set({ foundation: f }),

  unlockStageII: () => {
    const ps = { ...get().panelStates };
    (["N_MICRO", "N_ZOOM", "N_HIST", "N_TH2", "N_MINDWAKE", "N_TOLL"] as
NodeId[]).forEach((n) => {
      ps[n] = "idle";
    });
    set({ stageIIUnlocked: true, panelStates: ps });
  },

  setError: (message, nodeId) =>
    set({
      phase: "ERROR",
      lastError: { message, nodeId }
    }),

  resetForNewRun: () => {
    set({
      activeRunId: null,
      phase: "IDLE",
      stageIIUnlocked: false,
      panelStates: initPanelStates(),
      lastError: undefined,
      tenCat: undefined,
      gss: undefined,
      foundation: undefined
    });
  }
}));
```