Prompt: Build my Snowzies Ops home (Next.js app)

I want you to build a single-page “Snowzies Ops Cockpit” web app that I can actually use within 1–2 days.

1. Tech stack
   Use Next.js (App Router) with TypeScript.
   Use Tailwind CSS for styling.
   Structure it so I can deploy easily on Vercel.
   Create two simple API routes:
   GET /api/events
   POST /api/agent
2. Page layout (one main page)
   Create a main page (app/page.tsx) with five clear sections, laid out in a responsive grid:
   Today / Mode (top strip)
   Fields (editable in the UI, stateful on the client):
   Today’s focus (single line text).
   Risk level (1–5 selector).
   Non-negotiables (3 bullet items or small text inputs).
   Show these prominently at the top of the page.
   Alive Wallet / Ops Map (left column card)
   Use this JSON (or TypeScript data) as the source:
   ts

```
const wallets = [
{ layer: 'identity', name: 'ID_MAIN', address: null, profile: 'Comet – profcgrandma', role:
'identity', notes: 'Main identity' },
{ layer: 'vault', name: 'VAULT_TOKENS', address:
'0xcaf7a657AE496Bea23AFcAbB89ee744a755de99a', profile: 'Comet – profcgrandma',
role: 'vault_tokens', notes: 'Token cold vault' },
{ layer: 'vault', name: 'VAULT_NFT', address:
'0x45DA50F7d6d4B0eeEb66D693dC71d62545c563E7', profile: 'Comet – profcgrandma',
role: 'vault_nft', notes: 'NFT vault' },
{ layer: 'ops', name: 'OPS_USER2', address:
'0xf2773C0213E8001c8c1b9C955aC414D0Fc79a696', profile: 'Comet – profb769ops', role:
'ops_user', notes: 'Main ops wallet' },
{ layer: 'ops', name: 'OPS_SB1', address:
'0xba5E19c98b84400B1fA1E845206EFd70D6634689', profile: 'Comet – profb769ops',
role: 'ops_sandbox', notes: 'Degen / experiments' },
{ layer: 'tracking', name: 'TRACKER_ZERION', address: '0xa8ef89128f00fd9bd4159e2aa7c2aab567e28bb3', profile: 'Comet – profcgrandma', role: 'portfolio_tracker', notes: 'Zerion tracker' }
// project layer placeholders will be added later
];
```

Render a table with columns: Layer, Name, Address, Browser Profile, Role, Notes.
Make it easy to visually scan which wallet is what.
Arenas (center card)
Use this arenas data:
ts

```
const arenas = [
{ name: 'Kyan', type: 'btc_options_perps', status: 'live', url: 'https://www.kyan.exchange', comment: 'Main BTC book' },
{ name: 'Polymarket', type: 'prediction', status: 'scouting', url: 'https://polymarket.com', comment: 'Prediction markets' },
{ name: 'Betmoar', type: 'sports_markets', status: 'active', url: 'https://www.betmoar.fun/markets?view=calendar', comment: 'Sports markets + calendar' },
{ name: 'Bullpen', type: 'sports_markets', status: 'active', url: 'https://app.bullpen.fi', comment: 'Sports structured bets' },
{ name: '42', type: 'research', status: 'research', url: 'https://beta.42.space', comment: 'Research / intel' },
{ name: 'Zapper', type: 'portfolio', status: 'tracking', url: 'https://build.zapper.xyz', comment: 'Onchain portfolio overview' },
{ name: 'ENS', type: 'identity', status: 'active', url: 'https://app.ens.domains', comment: 'Names / identity' }
];
```

Render as a table or cards with: Name, Type, Status, Comment, and a button "Open" that links out.
Calendar (right column card)
Use a simple React calendar library (e.g. react-calendar or minimal FullCalendar setup).
Define a small TypeScript type and seed events:
ts

```
type EventType = 'prediction' | 'sports' | 'options' | 'macro' | 'life';

type CockpitEvent = {
id: string;
title: string;
date: string; // YYYY-MM-DD
type: EventType;
source: string;
url?: string | null;
importance: 'low' | 'medium' | 'high';
};

const events: CockpitEvent[] = [
{ id: 'btc-expiry', title: 'BTC options expiry (Kyan)', date: '2025-01-31', type: 'options', source:
```

'Kyan', url: 'https://www.kyan.exchange', importance: 'high' },
{ id: 'poly-1', title: 'Polymarket – key market resolves', date: '2025-02-10', type: 'prediction', source: 'Polymarket', url: 'https://polymarket.com', importance: 'high' },
{ id: 'betmoar-1', title: 'EPL match (Betmoar)', date: '2025-01-14', type: 'sports', source: 'Betmoar', url: 'https://www.betmoar.fun/markets?view=calendar', importance: 'medium' },
{ id: 'fomc-1', title: 'FOMC meeting', date: '2025-02-15', type: 'macro', source: 'macro', importance: 'high' },
{ id: 'no-screens', title: 'No screens day', date: '2025-01-05', type: 'life', source: 'personal', importance: 'high' }
];

Add simple filters (checkboxes) for event types.
On clicking an event:
If url exists, open in a new tab.
Otherwise show a small detail tooltip/modal.
Bottom band – Agent Console & Notes/Tasks
Agent Console (left)
Textarea input (Prompt) and “Ask Agent” button.
On submit, POST /api/agent with JSON:
json
{
"prompt": "user prompt text",
"today": { "focus": "...", "riskLevel": 3, "nonNegotiables": ["...", "...", "..."] },
"wallets": [/* from wallets const / *],*
"arenas": [/ from arenas const */]
}

Render response fields: summary, checklist[], notes.
Notes & Snippets (right)
Tabs: Notes, Snippets, Links.
Notes: a simple list of note cards (title + body), stored in localStorage or simple client state (I don’t need auth yet).
Snippets: list of items with fields title, tags, content (for prompts or code).
Links: fixed list (for now) of key URLs (Bybit, Polymarket, Betmoar, Bullpen, 42, Zapper, ENS, GitHub, Farcaster, email, Discord, Telegram, etc.).
Provide basic “add note/snippet” UI; persistence can be localStorage to keep it simple.
3. API routes
GET /api/events
Return { events: CockpitEvent[] } based on the static array above.
POST /api/agent
For now, just stub it:
Accept the JSON payload as described.
Return a canned JSON response in the same shape:
json
{
"summary": "Stub agent: describe what you would do here based on prompt and context.",
"checklist": [

"Example step 1",
"Example step 2"
],
"notes": "Later, this will call a real model (Claude/Elfa/AutoGen)."
}

Make it easy to replace the internals later with an actual LLM/agent call without changing the frontend.
4. Styling expectations
Use Tailwind to make it look like a clean, dark trading terminal.
Use cards with subtle borders, good spacing, and distinct sections so my brain can parse it quickly.
Responsive layout:
Desktop: Today strip on top; three main columns (Wallet, Arenas + Calendar, Agent/Notes).
Mobile: Stack sections vertically in a sane order.
5. Output
Give me:
The full package.json dependencies.
The Tailwind setup.
The main app/page.tsx code.
The data files for wallets, arenas, events.
The two API route files.
Write everything clean and modular so I can drop it into a repo and run npm install && npm run dev and immediately see my cockpit.

You can drop the files below into a fresh Next.js App Router project and have a working Snowzies Ops Cockpit with dark, terminal-style Tailwind UI, API routes, and client state.[1] [2] [3] [4]

package.json (core deps)

```
{
  "name": "snowzies-ops-cockpit",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "next": "14.2.5",
    "react": "18.3.1",
    "react-dom": "18.3.1",
    "react-calendar": "5.1.0",
    "clsx": "2.1.1"
  },
  "devDependencies": {
    "typescript": "5.6.3",
    "@types/node": "22.7.4",
    "@types/react": "18.3.8",
    "@types/react-dom": "18.3.2",
    "@types/react-calendar": "3.9.0",
    "tailwindcss": "3.4.14",
    "postcss": "8.4.47",
    "autoprefixer": "10.4.20",
    "eslint": "9.13.0",
    "eslint-config-next": "14.2.5"
  }
}
```

Tailwind setup

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

postcss.config.js

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {}
  }
};
```

app/globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

body {
  @apply bg-cockpit-bg text-gray-100 antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
::-webkit-scrollbar-thumb {
  @apply bg-slate-700 rounded-full;
}
```

Data files (wallets, arenas, events)

Create a folder `lib/` (or `data/`) and add these.

lib/wallets.ts

```
export type WalletLayer = "identity" | "vault" | "ops" | "tracking" | "project";

export type WalletRole =
  | "identity"
  | "vault_tokens"
  | "vault_nft"
  | "ops_user"
  | "ops_sandbox"
  | "portfolio_tracker"
  | "project_admin"
  | "project_treasury";

export type Wallet = {
  layer: WalletLayer;
  name: string;
  address: string | null;
  profile: string;
  role: WalletRole;
  notes?: string;
};

export const wallets: Wallet[] = [
  {
    layer: "identity",
    name: "ID_MAIN",
    address: null,
    profile: "Comet – profcgrandma",
    role: "identity",
    notes: "Main identity"
  },
  {
    layer: "vault",
    name: "VAULT_TOKENS",
    address: "0xcaf7a657AE496Bea23AFcAbB89ee744a755de99a",
    profile: "Comet – profcgrandma",
    role: "vault_tokens",
    notes: "Token cold vault"
  },
  {
    layer: "vault",
    name: "VAULT_NFT",
    address: "0x45DA50F7d6d4B0eeEb66D693dC71d62545c563E7",
    profile: "Comet – profcgrandma",
    role: "vault_nft",
    notes: "NFT vault"
  },
  {
    layer: "ops",
    name: "OPS_USER2",
    address: "0xf2773C0213E8001c8c1b9C955aC414D0Fc79a696",
    profile: "Comet – profb769ops",
    role: "ops_user",
    notes: "Main ops wallet"
  },
  {
    layer: "ops",
    name: "OPS_SB1",
    address: "0xba5E19c98b84400B1fA1E845206EFd70D6634689",
    profile: "Comet – profb769ops",
    role: "ops_sandbox",
    notes: "Degen / experiments"
  },
  {
    layer: "tracking",
    name: "TRACKER_ZERION",
    address: "0xa8ef89128f00fd9bd4159e2aa7c2aab567e28bb3",
    profile: "Comet – profcgrandma",
    role: "portfolio_tracker",
    notes: "Zerion tracker"
  }
  // Project layer placeholders can be added later
];
```

lib/arenas.ts

```
export type ArenaStatus = "live" | "active" | "scouting" | "research" | "tracking";

export type ArenaType =
  | "btc_options_perps"
  | "prediction"
  | "sports_markets"
  | "research"
  | "portfolio"
  | "identity";

export type Arena = {
  name: string;
  type: ArenaType;
  status: ArenaStatus;
  url: string;
  comment: string;
};

export const arenas: Arena[] = [
  {
    name: "Kyan",
    type: "btc_options_perps",
    status: "live",
    url: "https://www.kyan.exchange",
    comment: "Main BTC book"
  },
  {
    name: "Polymarket",
    type: "prediction",
    status: "scouting",
    url: "https://polymarket.com",
    comment: "Prediction markets"
  },
  {
    name: "Betmoar",
    type: "sports_markets",
    status: "active",
    url: "https://www.betmoar.fun/markets?view=calendar",
    comment: "Sports markets + calendar"
  },
  {
    name: "Bullpen",
    type: "sports_markets",
    status: "active",
    url: "https://app.bullpen.fi",
    comment: "Sports structured bets"
  },
  {
    name: "42",
    type: "research",
    status: "research",
    url: "https://beta.42.space",
    comment: "Research / intel"
  },
  {
    name: "Zapper",
    type: "portfolio",
    status: "tracking",
    url: "https://build.zapper.xyz",
    comment: "Onchain portfolio overview"
  },
  {
    name: "ENS",
    type: "identity",
    status: "active",
    url: "https://app.ens.domains",
    comment: "Names / identity"
  }
];
```

lib/events.ts

```
export type EventType = "prediction" | "sports" | "options" | "macro" | "life";

export type CockpitEvent = {
  id: string;
  title: string;
  date: string; // YYYY-MM-DD
  type: EventType;
  source: string;
  url?: string | null;
  importance: "low" | "medium" | "high";
};

export const events: CockpitEvent[] = [
  {
    id: "btc-expiry",
    title: "BTC options expiry (Kyan)",
    date: "2025-01-31",
    type: "options",
    source: "Kyan",
    url: "https://www.kyan.exchange",
    importance: "high"
  },
  {
    id: "poly-1",
    title: "Polymarket — key market resolves",
    date: "2025-02-10",
    type: "prediction",
    source: "Polymarket",
    url: "https://polymarket.com",
    importance: "high"
  },
  {
    id: "betmoar-1",
    title: "EPL match (Betmoar)",
    date: "2025-01-14",
    type: "sports",
    source: "Betmoar",
    url: "https://www.betmoar.fun/markets?view=calendar",
    importance: "medium"
  },
  {
    id: "fomc-1",
    title: "FOMC meeting",
    date: "2025-02-15",
    type: "macro",
    source: "macro",
    importance: "high"
  },
  {
    id: "no-screens",
    title: "No screens day",
    date: "2025-01-05",
    type: "life",
    source: "personal",
    importance: "high"
  }
];
```

lib/agent-types.ts

```
import type { Wallet } from "./wallets";
import type { Arena } from "./arenas";

export type AgentRequestPayload = {
  prompt: string;
  today: {
    focus: string;
    riskLevel: number;
    nonNegotiables: string[];
  };
  wallets: Wallet[];
  arenas: Arena[];
};

export type AgentResponsePayload = {
  summary: string;
  checklist: string[];
  notes: string;
};
```

API routes

App Router API routes live under `app/api/.../route.ts`.[4]

app/api/events/route.ts

```
import { NextResponse } from "next/server";
import { events } from "@/lib/events";

export async function GET() {
  return NextResponse.json({ events });
}
```

app/api/agent/route.ts

```
import { NextRequest, NextResponse } from "next/server";
import type { AgentRequestPayload, AgentResponsePayload } from "@/lib/agent-types";

export async function POST(req: NextRequest) {
  const body = (await req.json()) as AgentRequestPayload;

  // Stub logic – later you can call an LLM/agent here.
  const response: AgentResponsePayload = {
    summary:
      "Stub agent: would analyze your prompt, today's focus, risk level, non-negotiables,
    checklist: [
      "Example step 1 – clarify key objectives for this session.",
      "Example step 2 – scan arenas for opportunities aligned with current risk level."
    ],
    notes:
      "Later, this endpoint will call a real model (Claude/Elfa/AutoGen) and return riche
  };

  return NextResponse.json(response);
}
```

Root layout

app/layout.tsx

```
import "./globals.css";
import type { ReactNode } from "react";

export const metadata = {
  title: "Snowzies Ops Cockpit",
  description: "Personal ops home for wallets, arenas, calendar, and agent console."
};

export default function RootLayout({ children }: { children: ReactNode }) {
  return (
    <html lang="en" className="dark">
      <body className="min-h-screen bg-cockpit-bg text-gray-100">
        <div className="min-h-screen flex flex-col">
          {children}
        </div>
      </body>
    </html>
  );
}
```

Main page – app/page.tsx

This is a single file version with clearly separated sections; you can break into components later.

```
"use client";

import { useEffect, useMemo, useState } from "react";
import dynamic from "next/dynamic";
import { clsx } from "clsx";
import { wallets } from "@/lib/wallets";
import { arenas } from "@/lib/arenas";
import type { CockpitEvent, EventType } from "@/lib/events";
import type { AgentResponsePayload } from "@/lib/agent-types";

const ReactCalendar = dynamic(() => import("react-calendar"), { ssr: false });

type ValuePiece = Date | null;
type Value = ValuePiece | [ValuePiece, ValuePiece];

type Note = {
  id: string;
  title: string;
  body: string;
};

type Snippet = {
  id: string;
  title: string;
  tags: string;
  content: string;
};

type TabKey = "notes" | "snippets" | "links";

const NOTES_STORAGE_KEY = "snowzies-notes";
const SNIPPETS_STORAGE_KEY = "snowzies-snippets";

const links: { label: string; url: string }[] = [
  { label: "Bybit", url: "https://www.bybit.com" },
  { label: "Polymarket", url: "https://polymarket.com" },
  { label: "Betmoar", url: "https://www.betmoar.fun/markets?view=calendar" },
  { label: "Bullpen", url: "https://app.bullpen.fi" },
  { label: "42", url: "https://beta.42.space" },
  { label: "Zapper", url: "https://build.zapper.xyz" },
  { label: "ENS", url: "https://app.ens.domains" },
  { label: "GitHub", url: "https://github.com" },
  { label: "Farcaster", url: "https://warpcast.com" },
  { label: "Email", url: "https://mail.google.com" },
  { label: "Discord", url: "https://discord.com/app" },
  { label: "Telegram", url: "https://web.telegram.org" }
];

function Card(props: { title: string; children: React.ReactNode; className?: string }) {
  return (
    <section
      className={clsx(
        "flex flex-col rounded-xl border border-cockpit-border bg-cockpit-card/95",
        "shadow-sm shadow-black/40 backdrop-blur-sm",
        props.className
      )}
    >
      <header className="flex items-center justify-between px-4 py-2 border-b border-cock
        <h2 className="text-sm font-semibold tracking-wide text-gray-200 uppercase">
          {props.title}
        </h2>
      </header>
      <div className="flex-1 px-4 py-3 overflow-hidden">{props.children}</div>
    </section>
  );
}

export default function Page() {
  // Today / Mode state
  const [todayFocus, setTodayFocus] = useState("");
  const [riskLevel, setRiskLevel] = useState(3);
  const [nonNegotiables, setNonNegotiables] = useState<string[]>(["", "", ""]);

  // Calendar state
  const [calendarValue, setCalendarValue] = useState<Value>(new Date());
  const [events, setEvents] = useState<CockpitEvent[]>([]);
  const [eventTypeFilters, setEventTypeFilters] = useState<Record<EventType, boolean>>({
    prediction: true,
    sports: true,
    options: true,
    macro: true,
    life: true
  });
  const [selectedEvent, setSelectedEvent] = useState<CockpitEvent | null>(null);

  // Agent console state
  const [agentPrompt, setAgentPrompt] = useState("");
  const [agentLoading, setAgentLoading] = useState(false);
  const [agentResponse, setAgentResponse] = useState<AgentResponsePayload | null>(null);

  // Notes/snippets
  const [activeTab, setActiveTab] = useState<TabKey>("notes");
  const [notes, setNotes] = useState<Note[]>([]);
  const [snippets, setSnippets] = useState<Snippet[]>([]);
  const [newNoteTitle, setNewNoteTitle] = useState("");
  const [newNoteBody, setNewNoteBody] = useState("");
  const [newSnippetTitle, setNewSnippetTitle] = useState("");
  const [newSnippetTags, setNewSnippetTags] = useState("");
  const [newSnippetContent, setNewSnippetContent] = useState("");
  // Load events from API
  useEffect(() => {
    async function load() {
      try {
        const res = await fetch("/api/events");
        const json = await res.json();
        setEvents(json.events ?? []);
      } catch (e) {
        console.error("Failed to load events", e);
      }
    }
    load();
  }, []);

  // LocalStorage: notes/snippets
  useEffect(() => {
    if (typeof window === "undefined") return;
    try {
      const n = window.localStorage.getItem(NOTES_STORAGE_KEY);
      const s = window.localStorage.getItem(SNIPPETS_STORAGE_KEY);
      if (n) setNotes(JSON.parse(n));
      if (s) setSnippets(JSON.parse(s));
    } catch (e) {
      console.error("Failed to load local storage", e);
    }
  }, []);

  useEffect(() => {
    if (typeof window === "undefined") return;
    window.localStorage.setItem(NOTES_STORAGE_KEY, JSON.stringify(notes));
  }, [notes]);

  useEffect(() => {
    if (typeof window === "undefined") return;
    window.localStorage.setItem(SNIPPETS_STORAGE_KEY, JSON.stringify(snippets));
  }, [snippets]);

  const filteredEvents = useMemo(() => {
    return events.filter((ev) => eventTypeFilters[ev.type]);
  }, [events, eventTypeFilters]);

  const eventsByDate = useMemo(() => {
    const map = new Map<string, CockpitEvent[]>();
    for (const ev of filteredEvents) {
      if (!map.has(ev.date)) map.set(ev.date, []);
      map.get(ev.date)!.push(ev);
    }
    return map;
  }, [filteredEvents]);

  const onToggleType = (type: EventType) => {
    setEventTypeFilters((prev) => ({ ...prev, [type]: !prev[type] }));
  };

  const onCalendarClickDay = (value: Date) => {
    const dateStr = value.toISOString().slice(0, 10);
    const evs = eventsByDate.get(dateStr);
    if (!evs || evs.length === 0) return;
    const first = evs[^1_0];
    if (first.url) {
      window.open(first.url, "_blank", "noopener,noreferrer");
    } else {
      setSelectedEvent(first);
    }
  };

  const handleAskAgent = async () => {
    if (!agentPrompt.trim()) return;
    setAgentLoading(true);
    setAgentResponse(null);
    try {
      const payload = {
        prompt: agentPrompt,
        today: {
          focus: todayFocus,
          riskLevel,
          nonNegotiables
        },
        wallets,
        arenas
      };
      const res = await fetch("/api/agent", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(payload)
      });
      const json = (await res.json()) as AgentResponsePayload;
      setAgentResponse(json);
    } catch (e) {
      console.error("Agent call failed", e);
    } finally {
      setAgentLoading(false);
    }
  };

  const handleAddNote = () => {
    if (!newNoteTitle.trim() && !newNoteBody.trim()) return;
    setNotes((prev) => [
      {
        id: `note-${Date.now()}`,
        title: newNoteTitle || "Untitled",
        body: newNoteBody
      },
      ...prev
    ]);
    setNewNoteTitle("");
    setNewNoteBody("");
  };

  const handleAddSnippet = () => {
    if (!newSnippetTitle.trim() && !newSnippetContent.trim()) return;
    setSnippets((prev) => [
      {
        id: `snip-${Date.now()}`,
        title: newSnippetTitle || "Untitled snippet",
        tags: newSnippetTags,
        content: newSnippetContent
      },
      ...prev
    ]);
    setNewSnippetTitle("");
    setNewSnippetTags("");
    setNewSnippetContent("");
  };

  const todayDateLabel = new Date().toLocaleDateString(undefined, {
    weekday: "short",
    month: "short",
    day: "numeric"
  });

  return (
    <main className="flex-1 max-w-7xl mx-auto w-full px-4 py-4 gap-4 flex flex-col">
      {/* Today / Mode strip */}
      <section className="grid grid-cols-1 md:grid-cols-[minmax(0,2fr)_minmax(0,1fr)_minm
        <Card title="Today’s Focus" className="h-full">
          <div className="flex flex-col gap-2">
            <div className="text-xs text-gray-400 flex justify-between">
              <span>{todayDateLabel}</span>
              <span className="uppercase tracking-wide">Mode</span>
            </div>
            <input
              type="text"
              placeholder="What matters most today?"
              value={todayFocus}
              onChange={(e) => setTodayFocus(e.target.value)}
              className="mt-1 w-full rounded-md bg-black/40 border border-cockpit-border
            />
          </div>
        </Card>

        <Card title="Risk Level" className="h-full">
          <div className="flex flex-col gap-3">
            <input
              type="range"
              min={1}
              max={5}
              value={riskLevel}
              onChange={(e) => setRiskLevel(Number(e.target.value))}
              className="w-full accent-cockpit-accent"
            />
            <div className="flex items-center justify-between text-xs text-gray-400">
              <span>1 (cold)</span>
              <span className="text-sm font-semibold text-cockpit-accent">
                {riskLevel}
              </span>
              <span>5 (hot)</span>
            </div>
          </div>
        </Card>

        <Card title="Non-negotiables" className="h-full">
          <div className="flex flex-col gap-2">
            {nonNegotiables.map((item, idx) => (
              <div key={idx} className="flex items-center gap-2">
                <span className="text-xs text-gray-500 w-4">{idx + 1}.</span>
                <input
                  type="text"
                  value={item}
                  onChange={(e) => {
                    const copy = [...nonNegotiables];
                    copy[idx] = e.target.value;
                    setNonNegotiables(copy);
                  }}
                  className="flex-1 rounded-md bg-black/40 border border-cockpit-border p
                  placeholder="Non-negotiable for today"
                />
              </div>
            ))}
          </div>
        </Card>
      </section>

      {/* Main grid */}
      <section className="grid grid-cols-1 lg:grid-cols-[minmax(0,1.4fr)_minmax(0,2fr)_mi
        {/* Wallets */}
        <Card title="Alive Wallet / Ops Map" className="min-h-[260px]">
          <div className="overflow-auto max-h-[360px] text-xs">
            <table className="w-full border-collapse">
              <thead className="bg-black/40 sticky top-0 z-10">
                <tr>
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Layer</th
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Name</th>
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Address</
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Browser P
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Role</th>
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Notes</th
                </tr>
              </thead>
              <tbody>
                {wallets.map((w) => (
                  <tr
                    key={w.name}
                    className={clsx(
                      "border-b border-cockpit-border/60 hover:bg-cockpit-border/30",
                      w.layer === "vault" && "bg-cockpit-border/20"
                    )}
                  >
                    <td className="px-2 py-2 capitalize text-gray-300">{w.layer}</td>
                    <td className="px-2 py-2 font-semibold text-gray-100">{w.name}</td>
                    <td className="px-2 py-2 font-mono text-[10px] text-gray-300">
                      {w.address ?? "N/A"}
                    </td>
                    <td className="px-2 py-2 text-gray-300">{w.profile}</td>
```

```
                      <td className="px-2 py-2 text-gray-300">{w.role}</td>
                      <td className="px-2 py-2 text-gray-400">{w.notes}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </Card>

          {/* Arenas + Calendar stacked */}
          <div className="flex flex-col gap-4 min-h-0">
            <Card title="Arenas" className="min-h-[160px]">
              <div className="overflow-auto max-h-[200px] text-xs">
                <table className="w-full border-collapse">
                  <thead className="bg-black/40 sticky top-0 z-10">
                    <tr>
                      <th className="px-2 py-2 text-left font-medium text-gray-300">Name</t
                      <th className="px-2 py-2 text-left font-medium text-gray-300">Type</t
                      <th className="px-2 py-2 text-left font-medium text-gray-300">Status<
                      <th className="px-2 py-2 text-left font-medium text-gray-300">Comment
                      <th className="px-2 py-2 text-right font-medium text-gray-300">Open</
                    </tr>
                  </thead>
                  <tbody>
                    {arenas.map((a) => (
                      <tr
                        key={a.name}
                        className="border-b border-cockpit-border/60 hover:bg-cockpit-borde
                      >
                        <td className="px-2 py-2 font-semibold text-gray-100">{a.name}</td>
                        <td className="px-2 py-2 text-gray-300">{a.type}</td>
                        <td className="px-2 py-2">
                          <span
                            className={clsx(
                              "inline-flex items-center rounded-full px-2 py-0.5 text-[10px
                              a.status === "live" || a.status === "active"
                                ? "bg-emerald-500/15 text-emerald-300 border border-emeralc
                                : a.status === "scouting"
                                ? "bg-yellow-500/10 text-yellow-300 border border-yellow-50
                                : "bg-sky-500/10 text-sky-300 border border-sky-500/40"
                            )}
                          >
                            {a.status}
                          </span>
                        </td>
                        <td className="px-2 py-2 text-gray-400">{a.comment}</td>
                        <td className="px-2 py-2 text-right">
                          <button
                            onClick={() =>
                              window.open(a.url, "_blank", "noopener,noreferrer")
                            }
                            className="inline-flex items-center rounded-md border border-cc
                          >
                            Open
                          </button>
                        </td>
                      </tr>
                    ))}
                  </tbody>
                </table>
              </div>
            </Card>

            <Card title="Calendar" className="flex-1 min-h-[220px]">
              <div className="flex flex-col h-full gap-2">
                {/* Filters */}
                <div className="flex flex-wrap gap-2 text-[11px] mb-2">
                  {(["prediction", "sports", "options", "macro", "life"] as EventType[]).ma
                    (t) => (
                      <button
                        key={t}
                        onClick={() => onToggleType(t)}
                        className={clsx(
                          "inline-flex items-center gap-1 rounded-full px-2 py-1 border tex
                          eventTypeFilters[t]
                            ? "border-cockpit-accent bg-cockpit-accentSoft text-cockpit-acc
                            : "border-cockpit-border bg-black/40 text-gray-400"
                        )}
                      >
                        <span className="h-1.5 w-1.5 rounded-full bg-cockpit-accent" />
                        <span>{t}</span>
                      </button>
                    )
                  )}
                </div>

                <div className="grid grid-cols-1 md:grid-cols-[minmax(0,1.3fr)_minmax(0,1fr
                  <div className="rounded-lg border border-cockpit-border bg-black/40 p-2 t
                    <ReactCalendar
                      onChange={setCalendarValue}
                      value={calendarValue}
                      onClickDay={onCalendarClickDay}
                      tileContent={({ date }) => {
                        const key = date.toISOString().slice(0, 10);
                        const evs = eventsByDate.get(key);
                        if (!evs || evs.length === 0) return null;
                        const high = evs.some((e) => e.importance === "high");
                        return (
                          <div className="mt-1 flex justify-center">
                            <span
                              className={clsx(
                                "h-1.5 w-1.5 rounded-full",
                                high ? "bg-rose-400" : "bg-cockpit-accent"
                              )}
                            />
                          </div>
                        );
                      }}
                      className="w-full bg-transparent text-xs text-gray-100 [&_.react-cale
                    />
                  </div>
                  <div className="flex flex-col gap-2 text-xs overflow-auto max-h-[220px]">
                    <h3 className="text-[11px] font-semibold text-gray-300 uppercase tracki
                      Upcoming
                    </h3>
                    {filteredEvents
                      .slice()
                      .sort((a, b) => a.date.localeCompare(b.date))
                      .map((ev) => (
                        <button
                          key={ev.id}
                          onClick={() => {
                            if (ev.url) {
                              window.open(ev.url, "_blank", "noopener,noreferrer");
                            } else {
                              setSelectedEvent(ev);
                            }
                          }}
                          className="w-full text-left rounded-md border border-cockpit-bord
                        >
                          <div className="flex items-center justify-between gap-2">
                            <span className="text-[11px] text-gray-400">
                              {ev.date} · {ev.source}
                            </span>
                            <span
                              className={clsx(
                                "inline-flex items-center rounded-full px-1.5 py-0.5 text-[
                                ev.importance === "high"
                                  ? "bg-rose-500/15 text-rose-300 border border-rose-500/40
                                  : ev.importance === "medium"
                                  ? "bg-amber-500/15 text-amber-300 border border-amber-500
                                  : "bg-slate-500/15 text-slate-300 border border-slate-500
                              )}
                            >
                              {ev.importance}
                            </span>
                          </div>
                          <div className="mt-0.5 text-[11px] text-gray-100">
                            {ev.title}
                          </div>
                        </button>
                      ))}
                    {filteredEvents.length === 0 && (
                      <p className="text-xs text-gray-500">No events with current filters.<
                    )}
                  </div>
                </div>

                {selectedEvent && !selectedEvent.url && (
                  <div className="mt-2 rounded-md border border-cockpit-border bg-black/50
                    <div className="flex items-start justify-between gap-2">
                      <div>
                        <div className="text-gray-300 font-semibold">
                          {selectedEvent.title}
                        </div>
                        <div className="text-gray-400 text-[11px]">
                          {selectedEvent.date} · {selectedEvent.source}
                        </div>
                        <div className="mt-1 text-gray-400">
                          Type: {selectedEvent.type} · Importance:{" "}
                          {selectedEvent.importance}
                        </div>
                      </div>
                      <button
                        onClick={() => setSelectedEvent(null)}
                        className="text-gray-500 hover:text-gray-300 text-xs"
                      >
                        Close
                      </button>
                    </div>
                  </div>
                )}
              </div>
            </Card>
          </div>

          {/* Agent + Notes */}
          <div className="flex flex-col gap-4 min-h-0">
            <Card title="Agent Console" className="min-h-[200px]">
              <div className="flex flex-col h-full gap-3">
                <textarea
                  value={agentPrompt}
                  onChange={(e) => setAgentPrompt(e.target.value)}
                  placeholder="What do you want your agent to help with right now?"
                  className="min-h-[80px] max-h-[180px] w-full rounded-md bg-black/40 borde
                />
                <div className="flex items-center justify-between gap-2">
                  <button
                    disabled={agentLoading}
                    onClick={handleAskAgent}
                    className={clsx(
                      "inline-flex items-center rounded-md px-3 py-1.5 text-xs font-semibol
                      "border border-cockpit-accent bg-cockpit-accent text-black",
                      "hover:bg-cockpit-accent/90 disabled:opacity-50 disabled:cursor-not-a
                    )}
                  >
                    {agentLoading ? "Thinking..." : "Ask Agent"}
                  </button>
                  <span className="text-[10px] text-gray-500">
                    Context: today + wallets + arenas
                  </span>
                </div>

                {agentResponse && (
                  <div className="mt-2 flex flex-col gap-2 rounded-md border border-cockpit
                    <div>
                      <div className="text-[11px] font-semibold text-gray-200">
                        Summary
                      </div>
                      <p className="text-gray-300 mt-0.5">{agentResponse.summary}</p>
                    </div>
                    <div>
                      <div className="text-[11px] font-semibold text-gray-200">
                        Checklist
                      </div>
                      <ul className="mt-1 list-disc list-inside space-y-0.5 text-gray-300">
                        {agentResponse.checklist.map((item, idx) => (
                          <li key={idx}>{item}</li>
                        ))}
                      </ul>
                    </div>
                    <div>
                      <div className="text-[11px] font-semibold text-gray-200">
                        Notes
                      </div>
                      <p className="text-gray-300 mt-0.5">{agentResponse.notes}</p>
                    </div>
                  </div>
                )}
              </div>
            </Card>

            <Card title="Notes · Snippets · Links" className="flex-1 min-h-[220px]">
              <div className="flex flex-col h-full">
                {/* Tabs */}
                <div className="flex gap-2 text-xs mb-2">
                  {([
                    { key: "notes", label: "Notes" },
                    { key: "snippets", label: "Snippets" },
                    { key: "links", label: "Links" }
                  ] as { key: TabKey; label: string }[]).map((t) => (
                    <button
                      key={t.key}
                      onClick={() => setActiveTab(t.key)}
                      className={clsx(
                        "px-3 py-1.5 rounded-md border text-xs font-medium",
                        activeTab === t.key
                          ? "border-cockpit-accent bg-cockpit-accentSoft text-cockpit-accen
                          : "border-cockpit-border bg-black/40 text-gray-400 hover:text-gra
                      )}
                    >
                      {t.label}
                    </button>
                  ))}
                </div>

                {/* Content */}
                <div className="flex-1 min-h-0 overflow-hidden">
                  {activeTab === "notes" && (
                    <div className="flex flex-col h-full gap-3 text-xs">
                      <div className="rounded-md border border-cockpit-border bg-black/40 p
                        <input
                          type="text"
                          value={newNoteTitle}
                          onChange={(e) => setNewNoteTitle(e.target.value)}
                          placeholder="Note title"
                          className="w-full rounded-md bg-black/60 border border-cockpit-bo
                        />
                        <textarea
                          value={newNoteBody}
                          onChange={(e) => setNewNoteBody(e.target.value)}
                          placeholder="Note body"
                          className="w-full rounded-md bg-black/60 border border-cockpit-bo
                        />
                        <div className="flex justify-end">
                          <button
                            onClick={handleAddNote}
                            className="inline-flex items-center rounded-md border border-c
                          >
                            Add note
                          </button>
                        </div>
                      </div>

                      <div className="flex-1 overflow-auto space-y-2">
                        {notes.map((n) => (
                          <div
                            key={n.id}
                            className="rounded-md border border-cockpit-border bg-black/40
                          >
                            <div className="text-[11px] font-semibold text-gray-200">
                              {n.title}
                            </div>
                            <div className="mt-0.5 whitespace-pre-wrap text-gray-300 text-[
                              {n.body}
                            </div>
                          </div>
                        ))}
                        {notes.length === 0 && (
                          <p className="text-[11px] text-gray-500">
                            No notes yet. Capture a quick thought or plan.
                          </p>
                        )}
                      </div>
                    </div>
                  )}

                  {activeTab === "snippets" && (
                    <div className="flex flex-col h-full gap-3 text-xs">
                      <div className="rounded-md border border-cockpit-border bg-black/40
                        <input
                          type="text"
                          value={newSnippetTitle}
                          onChange={(e) => setNewSnippetTitle(e.target.value)}
                          placeholder="Snippet title"
                          className="w-full rounded-md bg-black/60 border border-cockpit-bo
                        />
                        <input
                          type="text"
                          value={newSnippetTags}
                          onChange={(e) => setNewSnippetTags(e.target.value)}
                          placeholder="Tags (comma separated, e.g. prompt, code)"
                          className="w-full rounded-md bg-black/60 border border-cockpit-bo
                        />
                        <textarea
```

```
                        value={newSnippetContent}
                        onChange={(e) => setNewSnippetContent(e.target.value)}
                        placeholder="Snippet content (prompt, code, template...)"
                        className="w-full rounded-md bg-black/60 border border-cockpit-bo
                      />
                      <div className="flex justify-end">
                        <button
                          onClick={handleAddSnippet}
                          className="inline-flex items-center rounded-md border border-co
                        >
                          Add snippet
                        </button>
                      </div>
                    </div>

                    <div className="flex-1 overflow-auto space-y-2">
                      {snippets.map((s) => (
                        <div
                          key={s.id}
                          className="rounded-md border border-cockpit-border bg-black/40
                        >
                          <div className="flex items-center justify-between gap-2 mb-1">
                            <div className="text-[11px] font-semibold text-gray-200">
                              {s.title}
                            </div>
                            {s.tags && (
                              <div className="text-[10px] text-cockpit-accent">
                                {s.tags}
                              </div>
                            )}
                          </div>
                          <pre className="mt-0.5 whitespace-pre-wrap text-gray-300 text-[
                            {s.content}
                          </pre>
                        </div>
                      ))}
                      {snippets.length === 0 && (
                        <p className="text-[11px] text-gray-500">
                          No snippets yet. Save prompts or code you reuse often.
                        </p>
                      )}
                    </div>
                  </div>
                )}

                {activeTab === "links" && (
                  <div className="flex flex-col h-full gap-2 text-xs overflow-auto">
                    <div className="grid grid-cols-2 sm:grid-cols-3 gap-2">
                      {links.map((l) => (
                        <button
                          key={l.label}
                          onClick={() =>
                            window.open(l.url, "_blank", "noopener,noreferrer")
                          }
                          className="flex items-center justify-between gap-2 rounded-md b
                        >
```

```
                          <span className="text-[11px] text-gray-100">
                            {l.label}
                          </span>
                          <span className="text-[9px] text-gray-500">↗</span>
                        </button>
                      ))}
                    </div>
                  </div>
                )}
              </div>
            </div>
          </Card>
        </div>
      </section>
    </main>
  );
}
```

How to run

- `npx create-next-app@latest snowzies-ops-cockpit` and choose App Router, TypeScript.[1]
- Replace `package.json` dependencies with the one above, then `npm install`.[1]
- Add Tailwind configs and `globals.css` as shown, then `npm install -D tailwindcss postcss autoprefixer`.[2] [3]
- Create `lib/` and `app/api/...` files plus `app/page.tsx` and `app/layout.tsx`.
- `npm run dev` and open `http://localhost:3000` to see the cockpit, then push to GitHub and deploy on Vercel.[5] [1]

⁂

I have no idea what im doing here. can I save this repo folder on wsl?

Yes, you can absolutely keep this repo “living” in WSL, and that’s actually the preferred place to work on it.[11] [12]

Where to put the folder

You want the project in your Linux home, not under `/mnt/c/...`:

- Open your WSL terminal (Ubuntu).
- Go to your home folder:

```
cd ~
pwd   # should show something like /home/heart
```

- Create a Dev folder if you like:

```
mkdir -p ~/dev
cd ~/dev
```

Now anything you create here (e.g. `snowzies-ops-cockpit`) is "inside WSL" and safe to use with Node, Next.js, etc.[12]

If the repo is currently on Windows

If you already created the project under Windows (e.g. `C:\Users\you\...`), WSL can see it under `/mnt/c/...`.[11] [12]

Example to move/copy it into WSL:

```
# Example paths – adjust to where the folder actually is
cp -R /mnt/c/Users/yourname/Desktop/snowzies-ops-cockpit ~/dev/
# or move instead of copy:
# mv /mnt/c/Users/yourname/Desktop/snowzies-ops-cockpit ~/dev/
```

Then:

```
cd ~/dev/snowzies-ops-cockpit
ls
```

Now the repo folder is **saved in WSL**.[12]

From here to running the app

Once you are inside that folder in WSL:

```
npm install
npm run dev
```

Then open `http://localhost:3000` in your browser and you should see the cockpit.[13]

If you tell where the folder currently lives (Windows path or WSL path), a few exact commands can be written for your case so you can just paste them.

⁂

These New Gemini Updates Are INSANE! (New Features) - YouTube

Yes. Think of WSL as “its own Linux machine” with its own disk; you want the repo to live there so Node/Next behave nicely.[22]

1. Decide where in WSL you want it

In your Ubuntu/WSL terminal:

```
cd ~          # go to /home/heart (or similar)
mkdir -p dev  # one time
cd dev
pwd           # should show /home/heart/dev
```

Anything inside `/home/heart/dev` is “saved in WSL.”[23]

2. If you haven’t created the Next.js app yet

From that `~/dev` folder:

```
npx create-next-app@latest snowzies-ops-cockpit
cd snowzies-ops-cockpit
```

Then paste/replace the files from the previous message into this folder, run:

```
npm install
npm run dev
```

Now the repo is a normal folder **inside WSL** and you can later `git init` and push to GitHub if you want.[24] [25]

3. If the folder already exists on Windows

If you already made `snowzies-ops-cockpit` somewhere on C:, you can copy/move it into WSL.[26] [23]

Example (adjust path to where it really is):

```
# from your WSL terminal
mkdir -p ~/dev
cp -R "/mnt/c/Users/YourName/Desktop/snowzies-ops-cockpit" ~/dev/
cd ~/dev/snowzies-ops-cockpit
```

Now it lives in WSL; you can run `npm install` and `npm run dev` from there.

If you paste what `pwd` shows where the project currently is, the exact commands for your setup can be written so you only need to copy–paste.

⁂

heart@WIN-1I3FKPL083U:~/dojo/experiments/FC-Base1$ cd ..

```
heart@WIN-1I3FKPL083U:~/dojo/experiments$ cd ..
heart@WIN-1I3FKPL083U:~/dojo$ cd protocols
heart@WIN-1I3FKPL083U:~/dojo/protocols$ dir
snowziesk
heart@WIN-1I3FKPL083U:~/dojo/protocols$ cd snowziesk
heart@WIN-1I3FKPL083U:~/dojo/protocols/snowziesk$ dir
Contracts MyTemplates reporting-workspace snowzies-hana
heart@WIN-1I3FKPL083U:~/dojo/protocols/snowziesk$ cd ..
heart@WIN-1I3FKPL083U:~/dojo/protocols$ cd ..
heart@WIN-1I3FKPL083U:~/dojo$ dir
apps archive bots contracts downloads experiments protocols tools
heart@WIN-1I3FKPL083U:~/dojo$ cd tools
heart@WIN-1I3FKPL083U:~/dojo/tools$ dir
elfa-tools merge_vaults.py sort_dojo.py sort_windows_downloads.py
heart@WIN-1I3FKPL083U:~/dojo/tools$ cd ~
heart@WIN-1I3FKPL083U:~$ dir
dojo
heart@WIN-1I3FKPL083U:~$ cd dojo
heart@WIN-1I3FKPL083U:~/dojo$ dir
apps archive bots contracts downloads experiments protocols tools
heart@WIN-1I3FKPL083U:~/dojo$
```

You're already inside WSL, and you have a perfect place to add this repo: under `~/dojo/apps`. [28] [29]

1. Create a home for the cockpit

From where you are now:

```
cd ~/dojo
cd apps
mkdir -p snowzies-ops-cockpit
cd snowzies-ops-cockpit
pwd
```

`pwd` should now show something like:

```
/home/heart/dojo/apps/snowzies-ops-cockpit
```

That is your **WSL repo folder**.

2. Initialize the Next.js app there

Still in that folder:

```
npx create-next-app@latest .   # note the dot = current folder
# answer the prompts (TypeScript: yes, App Router: yes, Tailwind: no/yes – we’ll overwrit
```

Then:

- Replace `package.json`, `tailwind.config.js`, `postcss.config.js`, `app/layout.tsx`, `app/page.tsx`, `app/globals.css`, `app/api/...`, and `lib/...` with the files from the earlier answer.
- Install deps:

```
npm install
npm run dev
```

Open `http://localhost:3000` and you should see your cockpit running from `~/dojo/apps/snowzies-ops-cockpit` inside WSL.[30]

If you’d like, next step can be: “show me exactly which file to create where from what you sent,” and it can be mapped 1:1 onto this `~/dojo/apps/snowzies-ops-cockpit` path.

⁂

i dont wanna cockpit name it. keep it short sweet is what it is for a while

Totally fine not to call it “cockpit.” Treat it as just your **home** for now and keep the folder name boring and short.[31] [32]

Here are a few clean options that will work nicely with `create-next-app` and Vercel:

- `ops-home`
- `ops-hub`
- `snow-home`
- `snow-ops`
- `hana-home` (if you want to echo `snowzies-hana`)
- `dojo-home`

If you like one, you can do this from WSL:

```
cd ~/dojo/apps
mkdir ops-home
```

```
cd ops-home
npx create-next-app@latest . --typescript
```

Then drop in the files from earlier. You will see it in Vercel as `ops-home` (or whatever you choose), but inside the UI you can still title it “Snowzies” or anything later.[33]

⁂

npx create-next-app@latest . --typescript

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y) where2sOul1!

npm error canceled
npm notice
npm notice New minor version of npm available! 11.6.1 -> 11.7.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.7.0
npm notice To update run: npm install -g npm@11.7.0
npm notice
npm error A complete log of this run can be found in: C:\Users\heart\AppData\Local\npm-cache_logs\2025-12-24T09_45_47_432Z-debug-0.log
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

The prompt `Ok to proceed? (y)` was asking you to type just `y` and press Enter; instead it read your password and treated that as “cancel,” so it stopped.[41]

What to do now

You’re already in `~/dojo/apps/ops-home`, so just run the command again and type `y` only.

```
cd ~/dojo/apps/ops-home

# run this again
npx create-next-app@latest . --typescript
```

When you see:

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y)
```

Type:

```
y
```

and press Enter (no password, nothing else).

Then it will ask a few questions (use the arrow keys / Enter):

- Project name: it will auto-detect `ops-home` – accept.
- TypeScript: yes.
- App Router: yes.
- ESLint: yes.
- Tailwind: yes or no (either is fine; you'll paste in the Tailwind setup after).

When it finishes:

```
npm install
npm run dev
```

Then open `http://localhost:3000` in your browser.

If any question in the wizard confuses you, paste the exact prompt and the options and simple "pick this option" guidance can be given line by line.

⁂

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ cd ~/dojo/apps/ops-home

run this again

```
npx create-next-app@latest . --typescript
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y) y

'\wsl.localhost\Ubuntu\home\heart\dojo\apps\ops-home'
CMD.EXE was started with the above path as the current directory.
UNC paths are not supported. Defaulting to Windows directory.
Could not create a project called "Windows" because of npm naming restrictions:
* name can no longer contain capital letters
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

Windows tried to run part of the command and got confused by the `\\wsl.localhost\...` path, so `create-next-app` broke and tried to call the project "Windows".[42] [43]

You can fix this in two quick steps:

1. Make sure Node/npm are the WSL (Linux) ones

In your WSL terminal, run:

```
which node
which npm
```

You want something like `/usr/bin/node` or `/usr/local/bin/node`, **not** a Windows path like `/mnt/c/Program Files/nodejs/node.exe`.[42]

If they point to Windows, either:

- Install Node via `nvm` in WSL:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
# then restart the WSL shell or run:
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"

nvm install --lts
nvm use --lts
```

Now `which node` should show a Linux path.[42]

2. Run `create-next-app` again from inside WSL

Once Node/npm are coming from WSL, do:

```
cd ~/dojo/apps
rm -rf ops-home   # remove the half-broken folder
mkdir ops-home
cd ops-home

npx create-next-app@latest . --typescript
```

When asked `Ok to proceed? (y)`, type:

```
y
```

Then accept defaults (TypeScript: yes, App Router: yes, Tailwind: yes/no as you prefer).

After it finishes:

```
npm install
npm run dev
```

Then visit `http://localhost:3000`.

If you paste the output of `which node` and `which npm`, the exact next command (with or without nvm) can be given so you can keep going without debugging guesswork.

⁂

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ cd ~

heart@WIN-1I3FKPL083U:~$ which node
which npm
/mnt/c/Program Files/nodejs//npm
heart@WIN-1I3FKPL083U:~$

Right now WSL is using the **Windows** Node/npm, which is what caused the UNC path / "Windows" error. You just need a Linux Node inside WSL (via `nvm`), then rerun the app creation. [44]

1. Install Node in WSL with nvm

Run these commands in your WSL terminal exactly:

```
# 1) Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
```

Close that terminal window, open a **new** WSL terminal, then:

```
# 2) Load nvm (usually added to .bashrc, but do it once manually to be sure)
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"

# 3) Install and use the latest LTS Node
nvm install --lts
nvm use --lts
```

Now check:

```
which node
which npm
```

You should see something like `/home/heart/.nvm/versions/node/...` instead of `/mnt/c/Program Files/...`.[44]

2. Recreate your `ops-home` app

Now that Node/npm are Linux-native:

```
cd ~/dojo/apps
rm -rf ops-home       # delete the broken one
mkdir ops-home
cd ops-home

npx create-next-app@latest . --typescript
```

When it says:

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y)
```

Type `y` and Enter.

Then accept:

- TypeScript: Yes
- App Router: Yes
- ESLint: Yes
- Tailwind: Yes (or No, you'll paste the Tailwind setup anyway)

After it finishes:

```
npm install
npm run dev
```

Open `http://localhost:3000` in your browser.

If you paste the new output of `which node` / `which npm`, it can be double-checked before you run `create-next-app` again.

⁂

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ cd ~

```
heart@WIN-1I3FKPL083U:~$ which node
which npm
/mnt/c/Program Files/nodejs//npm
heart@WIN-1I3FKPL083U:~$
```

WSL is still only seeing the **Windows** npm, and no Linux node at all, so `create-next-app` will keep breaking until you install Node in WSL itself.[45]

Here's the exact sequence to fix it.

1. Install nvm + Node inside WSL

Run these **exact** commands in your WSL terminal:

```
# 1) Install nvm (Node Version Manager) in WSL
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
```

When that finishes:

```
# 2) Start a fresh shell so nvm is available
exec "$SHELL"
```

Now load nvm and install Node:

```
# 3) Load nvm (should also be in your .bashrc now)
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"

# 4) Install and use the latest LTS version of Node
nvm install --lts
nvm use --lts
```

Confirm WSL is using the new Node/npm:

```
which node
which npm
```

You want paths like:

```
/home/heart/.nvm/versions/node/vXX.X.X/bin/node
/home/heart/.nvm/versions/node/vXX.X.X/bin/npm
```

and **not** `/mnt/c/Program Files/nodejs/...`.[45]

2. Recreate the `ops-home` app with the WSL Node

Once `which node` and `which npm` show the nvm paths:

```
cd ~/dojo/apps
rm -rf ops-home
mkdir ops-home
```

```
cd ops-home

npx create-next-app@latest . --typescript
```

When you see:

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y)
```

Type:

```
y
```

and press Enter.

Then answer:

- TypeScript: Yes
- App Router: Yes
- Tailwind: Yes (or No, you'll paste the Tailwind config anyway)

Finally:

```
npm install
npm run dev
```

Then open `http://localhost:3000` in your browser.

If you paste the new output of `which node` and `which npm` after running the nvm part, the next command can be verified before you touch `create-next-app` again.

⁂

which npm

```
/mnt/c/Program Files/nodejs//npm
heart@WIN-1I3FKPL083U:~$ # 1) Install nvm (Node Version Manager) in WSL
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 16563 100 16563 0 0 11232 0 0:00:01 0:00:01 --:--:-- 11236
⇒ Downloading nvm from git to '/home/heart/.nvm'
⇒ Cloning into '/home/heart/.nvm'...
remote: Enumerating objects: 384, done.
remote: Counting objects: 100% (384/384), done.
remote: Compressing objects: 100% (328/328), done.
```

remote: Total 384 (delta 42), reused 181 (delta 28), pack-reused 0 (from 0)
Receiving objects: 100% (384/384), 392.50 KiB | 410.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.

- (HEAD detached at FETCH_HEAD)
  master
  ⇒ Compressing and cleaning up git repository

⇒ nvm source string already in /home/heart/.bashrc
⇒ Appending bash_completion source string to /home/heart/.bashrc
⇒ You currently have modules installed globally with `npm`. These will no
⇒ longer be linked to the active version of Node when you install a new node
⇒ with `nvm`; and they may (depending on how you construct your `$PATH`)
⇒ override the binaries of modules installed with `nvm`:

C:\Users\heart\AppData\Roaming\npm
+-- @anthropic-ai/claude-code@2.0.69
+-- cowsay@1.6.0
+-- cowtalk@
`-- mcp-remote@0.1.29 => If you wish to uninstall them at a later point (or re-install them under your =>` nvm` node installs), you can remove them from the system Node as follows:

```
$ nvm use system
$ npm uninstall -g a_module
```

⇒ Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm''[-s''$NVM_DIR/nvm.sh"] && . "
NVM_DIR/bash_completion"] && . "

Nice, nvm installed correctly; you just need to **activate it, install Node, then re-run create-next-app**.[46]

Do this step by step in your current WSL terminal:

1. Activate nvm and install Node inside WSL

```
# Load nvm into this shell
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"

# Install and use latest LTS Node
nvm install --lts
nvm use --lts
```

Now check:

```
which node
which npm
```

You want paths under `/home/heart/.nvm/versions/node/...`, not `/mnt/c/Program Files/...`.[46]

2. Recreate `ops-home` with the WSL Node

Once `which node` and `which npm` look right:

```
cd ~/dojo/apps
rm -rf ops-home
mkdir ops-home
cd ops-home

npx create-next-app@latest . --typescript
```

When asked:

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y)
```

Type `y` and Enter.

Then:

- TypeScript: Yes
- App Router: Yes
- Tailwind: Yes (or No; you'll paste your Tailwind config after)

Finally:

```
npm install
npm run dev
```

If you paste the output of just:

```
which node
which npm
```

after running the nvm commands, it can be sanity-checked before you run `npx create-next-app` again.

⁂

Computing checksum with sha256sum

```
Checksums matched!
Now using node v24.12.0 (npm v11.6.2)
Creating default alias: default → lts/* (→ v24.12.0)
Now using node v24.12.0 (npm v11.6.2)
heart@WIN-1I3FKPL083U:~$ which node
which npm
/home/heart/.nvm/versions/node/v24.12.0/bin/node
/home/heart/.nvm/versions/node/v24.12.0/bin/npm
heart@WIN-1I3FKPL083U:~$
```

Perfect. You're now using the **WSL Node/npm**, so `create-next-app` will work cleanly.[47]

Here's exactly what to do next:

1. Create the Next.js app in `ops-home`

```
cd ~/dojo/apps
rm -rf ops-home       # clean any broken attempt
mkdir ops-home
cd ops-home

npx create-next-app@latest . --typescript
```

When it shows:

```
Need to install the following packages:
create-next-app@16.1.1
Ok to proceed? (y)
```

type:

```
y
```

and press Enter.

For the prompts, choose:

- TypeScript: **Yes**
- App Router: **Yes**
- ESLint: **Yes**
- Tailwind: **Yes** (easier, you'll still paste your config later)
- Src directory: your choice (Yes or No)
- App Router experimental: default is fine

2. Run it

After the wizard finishes:

```
npm install
npm run dev
```

Then open this in your browser:

```
http://localhost:3000
```

Once you confirm you see the default Next.js starter page, the next step can be “tell me exactly which files to replace with your Snowzies Ops code,” and it can be mapped 1:1 onto `~/dojo/apps/ops-home`.

⁂

[-s "NVM_DIR/nvm.sh"

```
[ -s "NVM_DIR/bash_completion"
```

Install and use latest LTS Node

```
nvm install --lts
nvm use --lts
Installing latest LTS version.
Downloading and installing node v24.12.0...
Downloading https://nodejs.org/dist/v24.12.0/node-v24.12.0-linux-x64.tar.xz...
#######################################################################
######################### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v24.12.0 (npm v11.6.2)
Creating default alias: default → lts/* (→ v24.12.0)
Now using node v24.12.0 (npm v11.6.2)
heart@WIN-1I3FKPL083U:~$ which node
which npm
/home/heart/.nvm/versions/node/v24.12.0/bin/node
/home/heart/.nvm/versions/node/v24.12.0/bin/npm
heart@WIN-1I3FKPL083U:~$ cd ~/dojo/apps
rm -rf ops-home # clean any broken attempt
mkdir ops-home
cd ops-home

npx create-next-app@latest . --typescript
Need to install the following packages:
```

create-next-app@16.1.1
Ok to proceed? (y) y

? Which linter would you like to use? › - Use arrow-keys. Return to submit.
❯ ESLint - More comprehensive lint rules
Biome
None

Nice, everything is working as it should now. Pick these options to keep it simple:

- Linter: **ESLint** (top option)
- Tailwind: **Yes**
- Src directory: **No** (keeps paths closer to what was sent)
- App Router: **Yes**
- Import alias: default is fine (`@/*`)

So, for each prompt:

1. On the linter question you see now:
   - Use the arrow keys to keep `ESLint - More comprehensive lint rules` selected.
   - Press Enter.
2. If it asks about Tailwind:
   - Choose **Yes**.
3. If it asks “Use `src/` directory?”:
   - Choose **No**.
4. If it asks “Use App Router?”:
   - Choose **Yes**.

After the script finishes:

```
npm install
npm run dev
```

Then open `http://localhost:3000` in your browser and confirm the default Next page shows.

Once that works, say “ready” and the next message can be: *“Tell me exactly which files to replace and with what”* mapped directly onto `~/dojo/apps/ops-home`.

react compiler?

For this project, answer **No** to React Compiler; you don’t need it to get your ops home running and it just adds noise right now.[48]

When the prompt appears:

```
Enable React Compiler? (recommended)
```

choose:

- **No**

After that, the wizard will finish creating the app and you can run:

```
npm install
npm run dev
```

Then open `http://localhost:3000` and you’ll see the starter app, ready for us to drop in your Snowzies layout.[48]

⁂

✔ Would you like to customize the import alias (`@/*` by default)? … No / Yes

Creating a new Next.js app in /home/heart/dojo/apps/ops-home.

Using npm.

Initializing project with template: app-tw

Installing dependencies:

- next
- react
- react-dom

Installing devDependencies:

- @tailwindcss/postcss
- @types/node
- @types/react
- @types/react-dom
- eslint
- eslint-config-next
- tailwindcss
- typescript

added 357 packages, and audited 358 packages in 25m

142 packages are looking for funding
run `npm fund` for details

```
found 0 vulnerabilities

Generating route types...
✓ Types generated successfully

Initialized a git repository.

Success! Created ops-home at /home/heart/dojo/apps/ops-home

npm notice
npm notice New minor version of npm available! 11.6.2 → 11.7.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.7.0
npm notice To update run: npm install -g npm@11.7.0
npm notice
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

You're in a good spot: the Next.js + Tailwind app is created cleanly in WSL and ready to turn into your ops home.[49] [50]

What to do next

From inside the project:

```
cd ~/dojo/apps/ops-home
npm run dev
```

Then open `http://localhost:3000` in your browser; you should see the default Tailwind starter page.[50] [49]

Once you confirm that works, the next step is:

- Replace the starter files with the Snowzies layout:
    - `app/layout.tsx`
    - `app/page.tsx`
    - `app/globals.css`
    - `tailwind.config.js`
    - `postcss.config.js`
    - `app/api/events/route.ts`
    - `app/api/agent/route.ts`
    - `lib/wallets.ts`
    - `lib/arenas.ts`
    - `lib/events.ts`
    - `lib/agent-types.ts`

If you say "ready to swap files," a 1:1 mapping can be given like:

- “Open `app/layout.tsx`, paste this…”
- “Create `lib/wallets.ts`, paste this…”

so you can just follow in order without thinking about structure.

⁂

cool, ready.

Good. You have a clean Next + Tailwind app; now you’ll wire in the ops home step by step.[51][52]

Follow these in order inside `~/dojo/apps/ops-home`.

1. Tailwind + globals

`tailwind.config.js`

Replace the existing file with:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

`postcss.config.mjs` **or** `postcss.config.js`

If you have `postcss.config.mjs` from the template, keep it or replace with JS:

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {}
  }
};
```

`app/globals.css`

Replace contents with:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

body {
  @apply bg-cockpit-bg text-gray-100 antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
::-webkit-scrollbar-thumb {
  @apply bg-slate-700 rounded-full;
}
```

2. Root layout

`app/layout.tsx`

Replace the file with:

```
import "./globals.css";
import type { ReactNode } from "react";

export const metadata = {
  title: "Ops Home",
  description: "Personal ops home for wallets, arenas, calendar, and agent console."
};

export default function RootLayout({ children }: { children: ReactNode }) {
  return (
    <html lang="en" className="dark">
      <body className="min-h-screen bg-cockpit-bg text-gray-100">
        <div className="min-h-screen flex flex-col">
          {children}
        </div>
      </body>
    </html>
  );
}
```

3. Data files

Create a `lib` folder:

```
mkdir -p lib
```

`lib/wallets.ts`

```
export type WalletLayer = "identity" | "vault" | "ops" | "tracking" | "project";

export type WalletRole =
  | "identity"
  | "vault_tokens"
  | "vault_nft"
  | "ops_user"
  | "ops_sandbox"
  | "portfolio_tracker"
  | "project_admin"
  | "project_treasury";

export type Wallet = {
  layer: WalletLayer;
  name: string;
  address: string | null;
  profile: string;
  role: WalletRole;
  notes?: string;
};

export const wallets: Wallet[] = [
  {
    layer: "identity",
    name: "ID_MAIN",
    address: null,
    profile: "Comet – profcgrandma",
    role: "identity",
    notes: "Main identity"
  },
  {
    layer: "vault",
    name: "VAULT_TOKENS",
    address: "0xcaf7a657AE496Bea23AFcAbB89ee744a755de99a",
    profile: "Comet – profcgrandma",
    role: "vault_tokens",
    notes: "Token cold vault"
  },
  {
    layer: "vault",
    name: "VAULT_NFT",
    address: "0x45DA50F7d6d4B0eeEb66D693dC71d62545c563E7",
    profile: "Comet – profcgrandma",
    role: "vault_nft",
    notes: "NFT vault"
  },
  {
    layer: "ops",
    name: "OPS_USER2",
    address: "0xf2773C0213E8001c8c1b9C955aC414D0Fc79a696",
    profile: "Comet – profb769ops",
    role: "ops_user",
    notes: "Main ops wallet"
  },
  {
    layer: "ops",
    name: "OPS_SB1",
    address: "0xba5E19c98b84400B1fA1E845206EFd70D6634689",
    profile: "Comet – profb769ops",
    role: "ops_sandbox",
    notes: "Degen / experiments"
  },
  {
    layer: "tracking",
    name: "TRACKER_ZERION",
    address: "0xa8ef89128f00fd9bd4159e2aa7c2aab567e28bb3",
    profile: "Comet – profcgrandma",
    role: "portfolio_tracker",
    notes: "Zerion tracker"
  }
];
```

`lib/arenas.ts`

```
export type ArenaStatus = "live" | "active" | "scouting" | "research" | "tracking";

export type ArenaType =
  | "btc_options_perps"
  | "prediction"
  | "sports_markets"
  | "research"
  | "portfolio"
  | "identity";

export type Arena = {
  name: string;
  type: ArenaType;
  status: ArenaStatus;
  url: string;
  comment: string;
};

export const arenas: Arena[] = [
  {
    name: "Kyan",
    type: "btc_options_perps",
    status: "live",
    url: "https://www.kyan.exchange",
    comment: "Main BTC book"
  },
  {
    name: "Polymarket",
    type: "prediction",
    status: "scouting",
    url: "https://polymarket.com",
    comment: "Prediction markets"
  },
  {
    name: "Betmoar",
    type: "sports_markets",
    status: "active",
    url: "https://www.betmoar.fun/markets?view=calendar",
    comment: "Sports markets + calendar"
  },
  {
    name: "Bullpen",
    type: "sports_markets",
    status: "active",
    url: "https://app.bullpen.fi",
    comment: "Sports structured bets"
  },
  {
    name: "42",
    type: "research",
    status: "research",
    url: "https://beta.42.space",
    comment: "Research / intel"
  },
  {
    name: "Zapper",
    type: "portfolio",
    status: "tracking",
    url: "https://build.zapper.xyz",
    comment: "Onchain portfolio overview"
  },
  {
    name: "ENS",
    type: "identity",
    status: "active",
    url: "https://app.ens.domains",
    comment: "Names / identity"
  }
];
```

`lib/events.ts`

```
export type EventType = "prediction" | "sports" | "options" | "macro" | "life";

export type CockpitEvent = {
  id: string;
  title: string;
  date: string; // YYYY-MM-DD
  type: EventType;
  source: string;
  url?: string | null;
  importance: "low" | "medium" | "high";
};

export const events: CockpitEvent[] = [
  {
    id: "btc-expiry",
    title: "BTC options expiry (Kyan)",
    date: "2025-01-31",
    type: "options",
    source: "Kyan",
    url: "https://www.kyan.exchange",
    importance: "high"
  },
  {
    id: "poly-1",
    title: "Polymarket – key market resolves",
    date: "2025-02-10",
    type: "prediction",
    source: "Polymarket",
    url: "https://polymarket.com",
    importance: "high"
  },
  {
    id: "betmoar-1",
    title: "EPL match (Betmoar)",
    date: "2025-01-14",
    type: "sports",
    source: "Betmoar",
    url: "https://www.betmoar.fun/markets?view=calendar",
    importance: "medium"
  },
  {
    id: "fomc-1",
    title: "FOMC meeting",
    date: "2025-02-15",
    type: "macro",
    source: "macro",
    importance: "high"
  },
  {
    id: "no-screens",
    title: "No screens day",
    date: "2025-01-05",
    type: "life",
    source: "personal",
    importance: "high"
  }
];
```

`lib/agent-types.ts`

```
import type { Wallet } from "./wallets";
import type { Arena } from "./arenas";

export type AgentRequestPayload = {
  prompt: string;
  today: {
    focus: string;
    riskLevel: number;
    nonNegotiables: string[];
  };
  wallets: Wallet[];
  arenas: Arena[];
};

export type AgentResponsePayload = {
  summary: string;
  checklist: string[];
  notes: string;
};
```

4. API routes

Create folders:

```
mkdir -p app/api/events
mkdir -p app/api/agent
```

`app/api/events/route.ts`

```
import { NextResponse } from "next/server";
import { events } from "@/lib/events";

export async function GET() {
  return NextResponse.json({ events });
}
```

`app/api/agent/route.ts`

```
import { NextRequest, NextResponse } from "next/server";
import type { AgentRequestPayload, AgentResponsePayload } from "@/lib/agent-types";

export async function POST(req: NextRequest) {
  const body = (await req.json()) as AgentRequestPayload;

  const response: AgentResponsePayload = {
    summary:
      "Stub agent: would analyze your prompt, today's focus, risk level, non-negotiables,
    checklist: [
      "Example step 1 – clarify key objectives for this session.",
      "Example step 2 – scan arenas for opportunities aligned with current risk level."
    ],
    notes:
      "Later, this endpoint will call a real model (Claude/Elfa/AutoGen) and return riche
  };

  return NextResponse.json(response);
}
```

5. Main page

`app/page.tsx`

Replace the existing `page.tsx` with this full file:

```
"use client";

import { useEffect, useMemo, useState } from "react";
import dynamic from "next/dynamic";
import { clsx } from "clsx";
import { wallets } from "@/lib/wallets";
import { arenas } from "@/lib/arenas";
import type { CockpitEvent, EventType } from "@/lib/events";
import type { AgentResponsePayload } from "@/lib/agent-types";

const ReactCalendar = dynamic(() => import("react-calendar"), { ssr: false });

type ValuePiece = Date | null;
type Value = ValuePiece | [ValuePiece, ValuePiece];

type Note = {
  id: string;
  title: string;
  body: string;
};

type Snippet = {
  id: string;
  title: string;
  tags: string;
  content: string;
};

type TabKey = "notes" | "snippets" | "links";

const NOTES_STORAGE_KEY = "snowzies-notes";
const SNIPPETS_STORAGE_KEY = "snowzies-snippets";

const links: { label: string; url: string }[] = [
  { label: "Bybit", url: "https://www.bybit.com" },
  { label: "Polymarket", url: "https://polymarket.com" },
  { label: "Betmoar", url: "https://www.betmoar.fun/markets?view=calendar" },
  { label: "Bullpen", url: "https://app.bullpen.fi" },
  { label: "42", url: "https://beta.42.space" },
  { label: "Zapper", url: "https://build.zapper.xyz" },
  { label: "ENS", url: "https://app.ens.domains" },
  { label: "GitHub", url: "https://github.com" },
  { label: "Farcaster", url: "https://warpcast.com" },
  { label: "Email", url: "https://mail.google.com" },
  { label: "Discord", url: "https://discord.com/app" },
  { label: "Telegram", url: "https://web.telegram.org" }
];

function Card(props: { title: string; children: React.ReactNode; className?: string }) {
  return (
    <section
      className={clsx(
        "flex flex-col rounded-xl border border-cockpit-border bg-cockpit-card/95",
        "shadow-sm shadow-black/40 backdrop-blur-sm",
        props.className
      )}
    >
      <header className="flex items-center justify-between px-4 py-2 border-b border-cock
        <h2 className="text-sm font-semibold tracking-wide text-gray-200 uppercase">
          {props.title}
        </h2>
      </header>
      <div className="flex-1 px-4 py-3 overflow-hidden">{props.children}</div>
    </section>
  );
}

export default function Page() {
  const [todayFocus, setTodayFocus] = useState("");
  const [riskLevel, setRiskLevel] = useState(3);
  const [nonNegotiables, setNonNegotiables] = useState<string[]>(["", "", ""]);

  const [calendarValue, setCalendarValue] = useState<Value>(new Date());
  const [events, setEvents] = useState<CockpitEvent[]>([]);
  const [eventTypeFilters, setEventTypeFilters] = useState<Record<EventType, boolean>>({
    prediction: true,
    sports: true,
    options: true,
    macro: true,
    life: true
  });
  const [selectedEvent, setSelectedEvent] = useState<CockpitEvent | null>(null);

  const [agentPrompt, setAgentPrompt] = useState("");
  const [agentLoading, setAgentLoading] = useState(false);
  const [agentResponse, setAgentResponse] = useState<AgentResponsePayload | null>(null);

  const [activeTab, setActiveTab] = useState<TabKey>("notes");
  const [notes, setNotes] = useState<Note[]>([]);
  const [snippets, setSnippets] = useState<Snippet[]>([]);
  const [newNoteTitle, setNewNoteTitle] = useState("");
  const [newNoteBody, setNewNoteBody] = useState("");
  const [newSnippetTitle, setNewSnippetTitle] = useState("");
  const [newSnippetTags, setNewSnippetTags] = useState("");
  const [newSnippetContent, setNewSnippetContent] = useState("");

  useEffect(() => {
    async function load() {
      try {
        const res = await fetch("/api/events");
        const json = await res.json();
        setEvents(json.events ?? []);
      } catch (e) {
        console.error("Failed to load events", e);
      }
    }
    load();
  }, []);

  useEffect(() => {
    if (typeof window === "undefined") return;
    try {
      const n = window.localStorage.getItem(NOTES_STORAGE_KEY);
      const s = window.localStorage.getItem(SNIPPETS_STORAGE_KEY);
      if (n) setNotes(JSON.parse(n));
      if (s) setSnippets(JSON.parse(s));
    } catch (e) {
      console.error("Failed to load local storage", e);
    }
  }, []);

  useEffect(() => {
    if (typeof window === "undefined") return;
    window.localStorage.setItem(NOTES_STORAGE_KEY, JSON.stringify(notes));
  }, [notes]);

  useEffect(() => {
    if (typeof window === "undefined") return;
    window.localStorage.setItem(SNIPPETS_STORAGE_KEY, JSON.stringify(snippets));
  }, [snippets]);

  const filteredEvents = useMemo(
    () => events.filter((ev) => eventTypeFilters[ev.type]),
    [events, eventTypeFilters]
  );

  const eventsByDate = useMemo(() => {
    const map = new Map<string, CockpitEvent[]>();
    for (const ev of filteredEvents) {
      if (!map.has(ev.date)) map.set(ev.date, []);
      map.get(ev.date)!.push(ev);
    }
    return map;
  }, [filteredEvents]);

  const onToggleType = (type: EventType) => {
    setEventTypeFilters((prev) => ({ ...prev, [type]: !prev[type] }));
  };
```

```
const onCalendarClickDay = (value: Date) => {
  const dateStr = value.toISOString().slice(0, 10);
  const evs = eventsByDate.get(dateStr);
  if (!evs || evs.length === 0) return;
  const first = evs[^15_0];
  if (first.url) {
    window.open(first.url, "_blank", "noopener,noreferrer");
  } else {
    setSelectedEvent(first);
  }
};

const handleAskAgent = async () => {
  if (!agentPrompt.trim()) return;
  setAgentLoading(true);
  setAgentResponse(null);
  try {
    const payload = {
      prompt: agentPrompt,
      today: {
        focus: todayFocus,
        riskLevel,
        nonNegotiables
      },
      wallets,
      arenas
    };
    const res = await fetch("/api/agent", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(payload)
    });
    const json = (await res.json()) as AgentResponsePayload;
    setAgentResponse(json);
  } catch (e) {
    console.error("Agent call failed", e);
  } finally {
    setAgentLoading(false);
  }
};

const handleAddNote = () => {
  if (!newNoteTitle.trim() && !newNoteBody.trim()) return;
  setNotes((prev) => [
    {
      id: `note-${Date.now()}`,
      title: newNoteTitle || "Untitled",
      body: newNoteBody
    },
    ...prev
  ]);
  setNewNoteTitle("");
  setNewNoteBody("");
};

const handleAddSnippet = () => {
  if (!newSnippetTitle.trim() && !newSnippetContent.trim()) return;
  setSnippets((prev) => [
    {
      id: `snip-${Date.now()}`,
      title: newSnippetTitle || "Untitled snippet",
      tags: newSnippetTags,
      content: newSnippetContent
    },
    ...prev
  ]);
  setNewSnippetTitle("");
  setNewSnippetTags("");
  setNewSnippetContent("");
};

const todayDateLabel = new Date().toLocaleDateString(undefined, {
  weekday: "short",
  month: "short",
  day: "numeric"
});

return (
  <main className="flex-1 max-w-7xl mx-auto w-full px-4 py-4 gap-4 flex flex-col">
    <section className="grid grid-cols-1 md:grid-cols-[minmax(0,2fr)_minmax(0,1fr)_minm
      <Card title="Today’s Focus" className="h-full">
        <div className="flex flex-col gap-2">
          <div className="text-xs text-gray-400 flex justify-between">
            <span>{todayDateLabel}</span>
            <span className="uppercase tracking-wide">Mode</span>
          </div>
          <input
            type="text"
            placeholder="What matters most today?"
            value={todayFocus}
            onChange={(e) => setTodayFocus(e.target.value)}
            className="mt-1 w-full rounded-md bg-black/40 border border-cockpit-border
          />
        </div>
      </Card>

      <Card title="Risk Level" className="h-full">
        <div className="flex flex-col gap-3">
          <input
            type="range"
            min={1}
            max={5}
            value={riskLevel}
            onChange={(e) => setRiskLevel(Number(e.target.value))}
            className="w-full accent-cockpit-accent"
          />
          <div className="flex items-center justify-between text-xs text-gray-400">
            <span>1 (cold)</span>
            <span className="text-sm font-semibold text-cockpit-accent">
              {riskLevel}
            </span>
            <span>5 (hot)</span>
          </div>
        </div>
      </Card>

      <Card title="Non-negotiables" className="h-full">
        <div className="flex flex-col gap-2">
          {nonNegotiables.map((item, idx) => (
            <div key={idx} className="flex items-center gap-2">
              <span className="text-xs text-gray-500 w-4">{idx + 1}.</span>
              <input
                type="text"
                value={item}
                onChange={(e) => {
                  const copy = [...nonNegotiables];
                  copy[idx] = e.target.value;
                  setNonNegotiables(copy);
                }}
                className="flex-1 rounded-md bg-black/40 border border-cockpit-border p
                placeholder="Non-negotiable for today"
              />
            </div>
          ))}
        </div>
      </Card>
    </section>

    <section className="grid grid-cols-1 lg:grid-cols-[minmax(0,1.4fr)_minmax(0,2fr)_mi
      <Card title="Alive Wallet / Ops Map" className="min-h-[260px]">
        <div className="overflow-auto max-h-[360px] text-xs">
          <table className="w-full border-collapse">
            <thead className="bg-black/40 sticky top-0 z-10">
              <tr>
                <th className="px-2 py-2 text-left font-medium text-gray-300">Layer</th
                <th className="px-2 py-2 text-left font-medium text-gray-300">Name</th>
                <th className="px-2 py-2 text-left font-medium text-gray-300">Address</
                <th className="px-2 py-2 text-left font-medium text-gray-300">
                  Browser Profile
                </th>
                <th className="px-2 py-2 text-left font-medium text-gray-300">Role</th>
                <th className="px-2 py-2 text-left font-medium text-gray-300">Notes</th
              </tr>
            </thead>
            <tbody>
              {wallets.map((w) => (
                <tr
                  key={w.name}
                  className={clsx(
                    "border-b border-cockpit-border/60 hover:bg-cockpit-border/30",
                    w.layer === "vault" && "bg-cockpit-border/20"
                  )}
                >
                  <td className="px-2 py-2 capitalize text-gray-300">{w.layer}</td>
                  <td className="px-2 py-2 font-semibold text-gray-100">{w.name}</td>
                  <td className="px-2 py-2 font-mono text-[10px] text-gray-300">
                    {w.address ?? "N/A"}
                  </td>
                  <td className="px-2 py-2 text-gray-300">{w.profile}</td>
                  <td className="px-2 py-2 text-gray-300">{w.role}</td>
                  <td className="px-2 py-2 text-gray-400">{w.notes}</td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </Card>

      <div className="flex flex-col gap-4 min-h-0">
        <Card title="Arenas" className="min-h-[160px]">
          <div className="overflow-auto max-h-[200px] text-xs">
            <table className="w-full border-collapse">
              <thead className="bg-black/40 sticky top-0 z-10">
                <tr>
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Name</t
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Type</t
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Status<
                  <th className="px-2 py-2 text-left font-medium text-gray-300">Comment
                  <th className="px-2 py-2 text-right font-medium text-gray-300">Open</
                </tr>
              </thead>
              <tbody>
                {arenas.map((a) => (
                  <tr
                    key={a.name}
                    className="border-b border-cockpit-border/60 hover:bg-cockpit-borde
                  >
                    <td className="px-2 py-2 font-semibold text-gray-100">{a.name}</td>
                    <td className="px-2 py-2 text-gray-300">{a.type}</td>
                    <td className="px-2 py-2">
                      <span
                        className={clsx(
                          "inline-flex items-center rounded-full px-2 py-0.5 text-[10p
                          a.status === "live" || a.status === "active"
                            ? "bg-emerald-500/15 text-emerald-300 border border-emeral
                            : a.status === "scouting"
                            ? "bg-yellow-500/10 text-yellow-300 border border-yellow-5
                            : "bg-sky-500/10 text-sky-300 border border-sky-500/40"
                        )}
                      >
                        {a.status}
                      </span>
                    </td>
                    <td className="px-2 py-2 text-gray-400">{a.comment}</td>
                    <td className="px-2 py-2 text-right">
                      <button
                        onClick={() =>
                          window.open(a.url, "_blank", "noopener,noreferrer")
                        }
                        className="inline-flex items-center rounded-md border border-c
                      >
                        Open
                      </button>
                    </td>
                  </tr>
                ))}
              </tbody>
            </table>
          </div>
        </Card>

        <Card title="Calendar" className="flex-1 min-h-[220px]">
          <div className="flex flex-col h-full gap-2">
            <div className="flex flex-wrap gap-2 text-[11px] mb-2">
              {(["prediction", "sports", "options", "macro", "life"] as EventType[]).ma
                (t) => (
                  <button
                    key={t}
                    onClick={() => onToggleType(t)}
                    className={clsx(
                      "inline-flex items-center gap-1 rounded-full px-2 py-1 border tex
                      eventTypeFilters[t]
                        ? "border-cockpit-accent bg-cockpit-accentSoft text-cockpit-acc
                        : "border-cockpit-border bg-black/40 text-gray-400"
                    )}
                  >
                    <span className="h-1.5 w-1.5 rounded-full bg-cockpit-accent" />
                    <span>{t}</span>
                  </button>
                )
              )}
            </div>

            <div className="grid grid-cols-1 md:grid-cols-[minmax(0,1.3fr)_minmax(0,1fr
              <div className="rounded-lg border border-cockpit-border bg-black/40 p-2 t
                <ReactCalendar
                  onChange={setCalendarValue}
                  value={calendarValue}
                  onClickDay={onCalendarClickDay}
                  tileContent={({ date }) => {
                    const key = date.toISOString().slice(0, 10);
                    const evs = eventsByDate.get(key);
                    if (!evs || evs.length === 0) return null;
                    const high = evs.some((e) => e.importance === "high");
                    return (
                      <div className="mt-1 flex justify-center">
                        <span
                          className={clsx(
                            "h-1.5 w-1.5 rounded-full",
                            high ? "bg-rose-400" : "bg-cockpit-accent"
                          )}
                        />
                      </div>
                    );
                  }}
                  className="w-full bg-transparent text-xs text-gray-100 [&_.react-cale
                />
              </div>

              <div className="flex flex-col gap-2 text-xs overflow-auto max-h-[220px]">
                <h3 className="text-[11px] font-semibold text-gray-300 uppercase tracki
                  Upcoming
                </h3>
                {filteredEvents
                  .slice()
                  .sort((a, b) => a.date.localeCompare(b.date))
                  .map((ev) => (
                    <button
                      key={ev.id}
                      onClick={() => {
                        if (ev.url) {
                          window.open(ev.url, "_blank", "noopener,noreferrer");
                        } else {
                          setSelectedEvent(ev);
                        }
                      }}
                      className="w-full text-left rounded-md border border-cockpit-bord
                    >
                      <div className="flex items-center justify-between gap-2">
                        <span className="text-[11px] text-gray-400">
                          {ev.date} · {ev.source}
                        </span>
                        <span
                          className={clsx(
                            "inline-flex items-center rounded-full px-1.5 py-0.5 text-[
                            ev.importance === "high"
                              ? "bg-rose-500/15 text-rose-300 border border-rose-500/40
                              : ev.importance === "medium"
                              ? "bg-amber-500/15 text-amber-300 border border-amber-500
                              : "bg-slate-500/15 text-slate-300 border border-slate-500
                          )}
                        >
                          {ev.importance}
                        </span>
                      </div>
                      <div className="mt-0.5 text-[11px] text-gray-100">
                        {ev.title}
                      </div>
                    </button>
                  ))}
                {filteredEvents.length === 0 && (
                  <p className="text-xs text-gray-500">No events with current filters.<
                )}
              </div>
            </div>

            {selectedEvent && !selectedEvent.url && (
              <div className="mt-2 rounded-md border border-cockpit-border bg-black/50
                <div className="flex items-start justify-between gap-2">
                  <div>
                    <div className="text-gray-300 font-semibold">
                      {selectedEvent.title}
                    </div>
                    <div className="text-gray-400 text-[11px]">
                      {selectedEvent.date} · {selectedEvent.source}
                    </div>
                    <div className="mt-1 text-gray-400">
                      Type: {selectedEvent.type} · Importance:{" "}
                      {selectedEvent.importance}
                    </div>
                  </div>
                  <button
                    onClick={() => setSelectedEvent(null)}
                    className="text-gray-500 hover:text-gray-300 text-xs"
                  >
                    Close
                  </button>
                </div>
              </div>
            )}
          </div>
        </Card>
      </div>

      <div className="flex flex-col gap-4 min-h-0">
        <Card title="Agent Console" className="min-h-[200px]">
          <div className="flex flex-col h-full gap-3">
            <textarea
              value={agentPrompt}
              onChange={(e) => setAgentPrompt(e.target.value)}
              placeholder="What do you want your agent to help with right now?"
              className="min-h-[80px] max-h-[180px] w-full rounded-md bg-black/40 borde
            />
            <div className="flex items-center justify-between gap-2">
              <button
                disabled={agentLoading}
                onClick={handleAskAgent}
                className={clsx(
                  "inline-flex items-center rounded-md px-3 py-1.5 text-xs font-semibol
                  "border border-cockpit-accent bg-cockpit-accent text-black",
                  "hover:bg-cockpit-accent/90 disabled:opacity-50 disabled:cursor-not-a
                )}
              >
                {agentLoading ? "Thinking..." : "Ask Agent"}
              </button>
              <span className="text-[10px] text-gray-500">
                Context: today + wallets + arenas
              </span>
            </div>

            {agentResponse && (
              <div className="mt-2 flex flex-col gap-2 rounded-md border border-cockpit
                <div>
                  <div className="text-[11px] font-semibold text-gray-200">
                    Summary
                  </div>
                  <p className="text-gray-300 mt-0.5">{agentResponse.summary}</p>
                </div>
                <div>
                  <div className="text-[11px] font-semibold text-gray-200">
                    Checklist
                  </div>
                  <ul className="mt-1 list-disc list-inside space-y-0.5 text-gray-300">
                    {agentResponse.checklist.map((item, idx) => (
                      <li key={idx}>{item}</li>
                    ))}
                  </ul>
                </div>
                <div>
                  <div className="text-[11px] font-semibold text-gray-200">
                    Notes
                  </div>
                  <p className="text-gray-300 mt-0.5">{agentResponse.notes}</p>
                </div>
              </div>
            )}
          </div>
        </Card>

        <Card title="Notes · Snippets · Links" className="flex-1 min-h-[220px]">
          <div className="flex flex-col h-full">
            <div className="flex gap-2 text-xs mb-2">
              {([
                { key: "notes", label: "Notes" },
                { key: "snippets", label: "Snippets" },
                { key: "links", label: "Links" }
              ] as { key: TabKey; label: string }[]).map((t) => (
                <button
                  key={t.key}
                  onClick={() => setActiveTab(t.key)}
                  className={clsx(
                    "px-3 py-1.5 rounded-md border text-xs font-medium",
                    activeTab === t.key
                      ? "border-cockpit-accent bg-cockpit-accentSoft text-cockpit-accen
                      : "border-cockpit-border bg-black/40 text-gray-400 hover:text-gra
                  )}
                >
                  {t.label}
                </button>
              ))}
            </div>

            <div className="flex-1 min-h-0 overflow-hidden">
              {activeTab === "notes" && (
                <div className="flex flex-col h-full gap-3 text-xs">
                  <div className="rounded-md border border-cockpit-border bg-black/40 p
                    <input
                      type="text"
                      value={newNoteTitle}
                      onChange={(e) => setNewNoteTitle(e.target.value)}
                      placeholder="Note title"
                      className="w-full rounded-md bg-black/60 border border-cockpit-bo
                    />
                    <textarea
                      value={newNoteBody}
                      onChange={(e) => setNewNoteBody(e.target.value)}
                      placeholder="Note body"
                      className="w-full rounded-md bg-black/60 border border-cockpit-bo
                    />
                    <div className="flex justify-end">
                      <button
                        onClick={handleAddNote}
                        className="inline-flex items-center rounded-md border border-c
                      >
                        Add note
                      </button>
                    </div>
                  </div>

                  <div className="flex-1 overflow-auto space-y-2">
                    {notes.map((n) => (
                      <div
                        key={n.id}
                        className="rounded-md border border-cockpit-border bg-black/40
                      >
                        <div className="text-[11px] font-semibold text-gray-200">
                          {n.title}
                        </div>
                        <div className="mt-0.5 whitespace-pre-wrap text-gray-300 text-[
                          {n.body}
                        </div>
                      </div>
                    ))}
                    {notes.length === 0 && (
                      <p className="text-[11px] text-gray-500">
                        No notes yet. Capture a quick thought or plan.
                      </p>
                    )}
                  </div>
                </div>
              )}

              {activeTab === "snippets" && (
                <div className="flex flex-col h-full gap-3 text-xs">
                  <div className="rounded-md border border-cockpit-border bg-black/40
                    <input
                      type="text"
                      value={newSnippetTitle}
                      onChange={(e) => setNewSnippetTitle(e.target.value)}
                      placeholder="Snippet title"
                      className="w-full rounded-md bg-black/60 border border-cockpit-b
                    />
                    <input
                      type="text"
                      value={newSnippetTags}
                      onChange={(e) => setNewSnippetTags(e.target.value)}
                      placeholder="Tags (comma separated, e.g. prompt, code)"
                      className="w-full rounded-md bg-black/60 border border-cockpit-b
                    />
                    <textarea
                      value={newSnippetContent}
                      onChange={(e) => setNewSnippetContent(e.target.value)}
                      placeholder="Snippet content (prompt, code, template...)"
                      className="w-full rounded-md bg-black/60 border border-cockpit-b
```

```
                      />
                      <div className="flex justify-end">
                        <button
                          onClick={handleAddSnippet}
                          className="inline-flex items-center rounded-md border border-c
                        >
                          Add snippet
                        </button>
                      </div>
                    </div>

                    <div className="flex-1 overflow-auto space-y-2">
                      {snippets.map((s) => (
                        <div
                          key={s.id}
                          className="rounded-md border border-cockpit-border bg-black/40
                        >
                          <div className="flex items-center justify-between gap-2 mb-1">
                            <div className="text-[11px] font-semibold text-gray-200">
                              {s.title}
                            </div>
                            {s.tags && (
                              <div className="text-[10px] text-cockpit-accent">
                                {s.tags}
                              </div>
                            )}
                          </div>
                          <pre className="mt-0.5 whitespace-pre-wrap text-gray-300 text-[
                            {s.content}
                          </pre>
                        </div>
                      ))}
                      {snippets.length === 0 && (
                        <p className="text-[11px] text-gray-500">
                          No snippets yet. Save prompts or code you reuse often.
                        </p>
                      )}
                    </div>
                  </div>
                )}

                {activeTab === "links" && (
                  <div className="flex flex-col h-full gap-2 text-xs overflow-auto">
                    <div className="grid grid-cols-2 sm:grid-cols-3 gap-2">
                      {links.map((l) => (
                        <button
                          key={l.label}
                          onClick={() =>
                            window.open(l.url, "_blank", "noopener,noreferrer")
                          }
                          className="flex items-center justify-between gap-2 rounded-md b
                        >
                          <span className="text-[11px] text-gray-100">
                            {l.label}
                          </span>
                          <span className="text-[9px] text-gray-500">↗</span>
                        </button>
                      ))}
                    </div>
                  </div>
                )}
              </div>
            </div>
          </Card>
        </div>
      </section>
    </main>
  );
}
```

6. Install `react-calendar` and `clsx`

From the project root:

```
npm install react-calendar clsx
```

7. Run it

```
npm run dev
```

Go to `http://localhost:3000` and you should see:

- Top strip: Today’s focus, risk slider, non-negotiables.
- Left: wallet map.
- Middle: arenas + calendar with filters.
- Right: agent console + notes/snippets/links.

If you hit any TypeScript or import error, paste the error text and the next step can target that line directly.

⁂

do I stop the npm running?

Yes. Stop the current dev server before editing a lot of files.

- In the terminal where `npm run dev` is running, press:

```
Ctrl + C
```

- Make all the file changes (Tailwind config, `layout.tsx`, `page.tsx`, `lib/*`, API routes).

- Then start it again:

```
npm run dev
```

and reload `http://localhost:3000`.

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir

README.md eslint.config.mjs next.config.ts package-lock.json postcss.config.mjs tsconfig.json app next-env.d.ts node_modules package.json public
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano talwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$

tailwing.config.js wasnt tgere?

The file name from the template is slightly different; that's why nano didn't find it.

In your project root you have:

```
postcss.config.mjs
eslint.config.mjs
next.config.ts
...
```

but no `tailwind.config` yet in that `dir` output. The template usually creates `tailwind.config.ts` or `tailwind.config.js`, but sometimes it's just not obvious.[53]

Do this:

```
cd ~/dojo/apps/ops-home
ls
ls tailwind*
```

If you see something like `tailwind.config.ts`:

1. Open that file:

```
nano tailwind.config.ts
```

2. Replace its contents with this TypeScript version:

```
import type { Config } from "tailwindcss";

const config: Config = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};

export default config;
```

If `ls tailwind*` shows **nothing**, just create a JS config:

```
nano tailwind.config.js
```

Paste:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

Save and exit (`Ctrl+O`, Enter, `Ctrl+X`).

Then move on to editing `app/globals.css` and `app/layout.tsx` as in the previous message, and finally run:

```
npm install react-calendar clsx
npm run dev
```

⁂

README.md eslint.config.mjs next.config.ts package-lock.json postcss.config.mjs tsconfig.json

```
app next-env.d.ts node_modules package.json public
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano talwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json talwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls tailwind*
ls: cannot access 'tailwind*': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

You just have a small typo in the filename: you created `talwind.config.js` instead of `tailwind.config.js`. Tailwind won't see it with the wrong name.[54]

Fix it like this:

```
cd ~/dojo/apps/ops-home

# Rename the file
mv talwind.config.js tailwind.config.js

# Check it
ls tailwind*
```

You should now see:

```
tailwind.config.js
```

Next:

1. Open it:

```
nano tailwind.config.js
```

2. Make sure it contains:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

Save (`Ctrl+O`, Enter) and exit (`Ctrl+X`), then continue with `app/globals.css`, `app/layout.tsx`, etc., and finally:

```
npm install react-calendar clsx
npm run dev
```

⁂

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls tailwind*

```
ls: cannot access 'tailwind*': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ cd ~/dojo/apps/ops-home
```

Rename the file

```
mv talwind.config.js tailwind.config.js
```

Check it

```
ls tailwind*
tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls *tail
ls: cannot access '*tail': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
```

```
app next.config.ts package.json tailwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano tailwind py
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json tailwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls *tail
ls: cannot access '*tail': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

You’re fine now: the important part is that `tailwind.config.js` exists with the right name; the `ls *tail` command isn’t needed and the error there doesn’t matter.[55]

Right now your root shows:

```
tailwind.config.js
postcss.config.mjs
app
...
```

So continue with the remaining steps.

1. Edit `tailwind.config.js`

Open it and paste the config:

```
nano tailwind.config.js
```

Inside, make sure it is exactly:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

Save (`Ctrl+O`, Enter) and exit (`Ctrl+X`).

2. Next edits in order

From `~/dojo/apps/ops-home`:

1. `nano app/globals.css` → replace with the Tailwind + dark styles.
2. `nano app/layout.tsx` → replace with the minimal layout version.
3. `mkdir lib` and create the four `lib/*.ts` files.
4. `mkdir -p app/api/events app/api/agent` and create the two `route.ts` files.
5. `nano app/page.tsx` → replace with the big main page code.
6. Install extra deps:

```
npm install react-calendar clsx
```

7. Run:

```
npm run dev
```

Then open `http://localhost:3000`.

If you tell which file you want to tackle next (e.g. “let’s do globals.css”), the exact content can be re-pasted just for that file so you don’t have to scroll.

⁂

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano talwind.config.js

```
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json talwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls tailwind*
ls: cannot access 'tailwind*': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ cd ~/dojo/apps/ops-home
```

Rename the file

```
mv talwind.config.js tailwind.config.js
```

Check it

```
ls tailwind*
tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls *tail
ls: cannot access '*tail': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json tailwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano tailwind py
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json tailwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls *tail
ls: cannot access '*tail': No such file or directory
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano app/globals.css
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano app/globals.css
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano app/layout.tsx
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package-lock.json public
app next.config.ts package.json tailwind.config.js
eslint.config.mjs node_modules postcss.config.mjs tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ mkdir -p lib
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano lib/wallets.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano lib/arenas.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano lib/events.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano lib/agent-types.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ mkdir -p app/api/events
mkdir -p app/api/agent
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano app/api/events/route.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ nano app/api/agent/route.ts
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ app/page.tsx
-bash: app/page.tsx: Permission denied
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

You did all the file creation perfectly! The last command failed because you used `app/page.tsx` without `nano` – it tried to **execute** the file instead of edit it.

Fix that now:

```
nano app/page.tsx
```

Paste the **big main page code** from earlier (the full `app/page.tsx` with all the sections: Today strip, wallets, arenas, calendar, agent, notes).

Save (`Ctrl+O`, Enter) and exit (`Ctrl+X`).

Then install the missing packages and run:

```
npm install react-calendar clsx @types/react-calendar
npm run dev
```

Open `http://localhost:3000` in your browser.

You should see the full ops home with:

- Dark terminal theme
- Top: Today's focus, risk slider, non-negotiables
- Left: Wallet table
- Middle: Arenas table + calendar with filters
- Right: Agent console + notes/snippets/links tabs

If you get any errors when running `npm run dev`, paste them here and they can be fixed one by one.

You're 95% there – just that last `nano app/page.tsx` and the `npm install`.

API Showcase - Polymarket Documentation

```
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md lib node_modules postcss.config.mjs tsconfig.json
app next-env.d.ts package-lock.json public
eslint.config.mjs next.config.ts package.json tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ npm install react-calendar clsx

added 13 packages, and audited 371 packages in 9m

150 packages are looking for funding
run npm fund for details

found 0 vulnerabilities
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

Everything is in place dependency-wise; next step is just to run the dev server and see it.[56] [57]

1. Make sure `app/page.tsx` is filled

From the project root:

```
cd ~/dojo/apps/ops-home
nano app/page.tsx
```

Confirm that file contains the long React code with:

- `use client` at the top.
- Imports for `dynamic`, `clsx`, `wallets`, `arenas`, `CockpitEvent`, `AgentResponsePayload`.
- A `ReactCalendar = dynamic(() => import("react-calendar"), { ssr: false });` line.
- The big `export default function Page()` component with all five sections.

If it's empty or still the default Next starter, paste the full version from the earlier message into this file and save (`Ctrl+O`, Enter, `Ctrl+X`).

2. Start the dev server

From `~/dojo/apps/ops-home`:

```
npm run dev
```

Then in your browser, go to:

```
http://localhost:3000
```

You should see your dark ops home with:

- Today's focus / risk / non-negotiables on top.
- Wallets table on the left.
- Arenas + calendar in the middle.
- Agent console + notes/snippets/links on the right.[57] [56]

If `npm run dev` prints any error (TypeScript, import, etc.), paste the error message and file/line and it can be fixed line-by-line.

⁂

README.md **lib node_modules postcss.config.mjs tsconfig.json**

app next-env.d.ts package-lock.json public
eslint.config.mjs next.config.ts package.json tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ npm install react-calendar clsx

added 13 packages, and audited 371 packages in 9m

150 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ npm run dev

> ops-home@0.1.0 dev
> next dev

▲ Next.js 16.1.1 (Turbopack)

- Local: http://localhost:3000
- Network: http://10.255.255.254:3000

✓ Starting...
✓ Ready in 4.8s
○ Compiling / ...
Error: Cannot apply unknown utility class `bg-cockpit-bg`. Are you using CSS modules or similar and missing `@reference`? https://tailwindcss.com/docs/functions-and-directives#reference-directive
[at onInvalidCandidate (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:21:1314)]
[at Te (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:16:36652)]
[at /home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:21:355]
[at Rr (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:3:1718)]
[at _ (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:3:1377)]
[at $e (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:21:172)]
[at un (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:38:294)]
[at process.processTicksAndRejections (node:internal/process/task_queues:103:5)]
[at async cn (/home/heart/dojo/apps/ops-home/node_modules/tailwindcss/dist/lib.js:38:631)]
[at async Xr (/home/heart/dojo/apps/ops-home/node_modules/@tailwindcss/node/dist/index.js:10:3417)]
× ./app/layout.tsx:1:9
Parsing ecmascript source code failed

> 1 | kimport "./globals.css";
> | ^^^^^^^^^^^^^^^
> 2 | import type { ReactNode } from "react";
> 3 |
> 4 | export const metadata = {

Expected ';', '}' or <eof>

○ Compiling /_error ...
GET / 500 in 10.0s (compile: 9.3s, render: 640ms)

Two things are going on: a small typo in `app/layout.tsx` and Tailwind not seeing your custom `cockpit` colors yet. Both are quick fixes.

1. Fix `app/layout.tsx` typo

The error:

```
1 | kimport "./globals.css";
      ^^^^^
```

means there's a stray `k` before `import`.

Edit the file:

```
cd ~/dojo/apps/ops-home
nano app/layout.tsx
```

Make sure the **top of the file** is exactly:

```
import "./globals.css";
import type { ReactNode } from "react";

export const metadata = {
  title: "Ops Home",
  description: "Personal ops home for wallets, arenas, calendar, and agent console."
};

export default function RootLayout({ children }: { children: ReactNode }) {
  return (
    <html lang="en" className="dark">
      <body className="min-h-screen bg-cockpit-bg text-gray-100">
        <div className="min-h-screen flex flex-col">
          {children}
        </div>
      </body>
    </html>
  );
}
```

Save (`Ctrl+O`, Enter) and exit (`Ctrl+X`).

2. Make sure Tailwind config has `cockpit` colors

The `bg-cockpit-bg` error means Tailwind doesn’t know that color yet.

Open `tailwind.config.js`:

```
nano tailwind.config.js
```

Ensure it is **exactly**:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: "class",
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {
      colors: {
        cockpit: {
          bg: "#050814",
          card: "#0b1020",
          border: "#1c2333",
          accent: "#00f6b1",
          accentSoft: "#00f6b133"
        }
      }
    }
  },
  plugins: []
};
```

Save and exit.

3. Restart dev server

Stop the current server (if running):

```
# in the dev terminal
Ctrl + C
```

Then:

```
npm run dev
```

Go back to `http://localhost:3000` and reload.

If you still see an error, paste the **new** error message and file/line so it can be fixed next.

⁂

Error evaluating Node.js code

./app/globals.css
Error evaluating Node.js code
CssSyntaxError: tailwindcss: /home/heart/dojo/apps/ops-home/app/globals.css:1:1: Cannot apply unknown utility class `bg-cockpit-bg`. Are you using CSS modules or similar and missing `@reference`? https://tailwindcss.com/docs/functions-and-directives#reference-directive
[at Input.error (turbopack:///[project]/node_modules/postcss/lib/input.js:135:16)]
[at Root.error (turbopack:///[project]/node_modules/postcss/lib/node.js:146:32)]
[at Object.Once (/home/heart/dojo/apps/ops-home/node_modules/@tailwindcss/postcss/dist/index.js:10:6911)]
[at process.processTicksAndRejections (node:internal/process/task_queues:103:5)]
[at async LazyResult.runAsync (turbopack:///[project]/node_modules/postcss/lib/lazy-result.js:293:11)]
[at async transform (turbopack:///[turbopack-node]/transforms/postcss.ts:70:34)]
[at async run (turbopack:///[turbopack-node]/ipc/evaluate.ts:92:23)]

Import trace:
Client Component Browser:
./app/globals.css [Client Component Browser]
./app/layout.tsx [Server Component]

Tailwind is complaining because `bg-cockpit-bg` is being used as a **class in CSS**, not as a Tailwind utility, and the config doesn’t define a matching utility in a way Tailwind understands.

Right now your `app/globals.css` should **not** use `bg-cockpit-bg` directly as a class; it should use `@apply` to apply utilities. Replace it with this minimal version:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

body {
  @apply bg-slate-950 text-gray-100 antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
::-webkit-scrollbar-thumb {
  @apply bg-slate-700 rounded-full;
}
```

That removes `bg-cockpit-bg` from pure CSS and uses built-in Tailwind colors instead.

Then in `app/layout.tsx`, keep:

```
<body className="min-h-screen bg-slate-950 text-gray-100">
```

or:

```
<body className="min-h-screen bg-slate-950 text-gray-100">
  <div className="min-h-screen flex flex-col">
    {children}
  </div>
</body>
```

After you change `app/globals.css`:

1. Stop dev server with `Ctrl+C`.
2. Run:

```
npm run dev
```

3. Reload `http://localhost:3000`.

You'll still get the dark look, just using core Tailwind colors; once everything is stable, you can re-introduce custom `cockpit` colors via `bg-[color]` utilities or keep it as is.

⁂

GET / 500 in 1535ms (compile: 1119ms, render: 417ms)

```
^C

heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ dir
README.md next-env.d.ts package.json tsconfig.json
app next.config.ts postcss.config.mjs
eslint.config.mjs node_modules public
lib package-lock.json tailwind.config.js
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$ ls -l
total 284
-rw-r--r-- 1 heart heart 1450 Dec 24 12:05 README.md
drwxr-xr-x 3 heart heart 4096 Dec 24 22:36 app
```

```
-rw-r--r-- 1 heart heart 465 Dec 24 12:05 eslint.config.mjs
drwxr-xr-x 2 heart heart 4096 Dec 24 19:47 lib
-rw-r--r-- 1 heart heart 251 Dec 24 19:07 next-env.d.ts
-rw-r--r-- 1 heart heart 133 Dec 24 12:05 next.config.ts
drwxr-xr-x 298 heart heart 12288 Dec 24 21:44 node_modules
-rw-r--r-- 1 heart heart 230500 Dec 24 21:53 package-lock.json
-rw-r--r-- 1 heart heart 584 Dec 24 21:53 package.json
-rw-r--r-- 1 heart heart 94 Dec 24 12:05 postcss.config.mjs
drwxr-xr-x 2 heart heart 4096 Dec 24 12:05 public
-rw-r--r-- 1 heart heart 426 Dec 24 22:39 tailwind.config.js
-rw-r--r-- 1 heart heart 666 Dec 24 12:05 tsconfig.json
heart@WIN-1I3FKPL083U:~/dojo/apps/ops-home$
```

The 500 is still there, so something else (likely in `app/page.tsx` or `app/layout.tsx`) is throwing. The quickest way forward is to check those two files for small mistakes.

Do these checks one by one.

1. Confirm `app/globals.css`

Open:

```
nano app/globals.css
```

Make sure it is **exactly**:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

body {
  @apply bg-slate-950 text-gray-100 antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
::-webkit-scrollbar-thumb {
  @apply bg-slate-700 rounded-full;
}
```

Save (Ctrl+O, Enter) and exit (Ctrl+X).

2. Confirm `app/layout.tsx`

```
nano app/layout.tsx
```

Replace the whole file with:

```
import "./globals.css";
import type { ReactNode } from "react";

export const metadata = {
  title: "Ops Home",
  description: "Personal ops home for wallets, arenas, calendar, and agent console."
};

export default function RootLayout({ children }: { children: ReactNode }) {
  return (
    <html lang="en" className="dark">
      <body className="min-h-screen bg-slate-950 text-gray-100">
        <div className="min-h-screen flex flex-col">
          {children}
        </div>
      </body>
    </html>
  );
}
```

Save and exit.

3. Confirm `app/page.tsx` is valid

```
nano app/page.tsx
```

At the very top you should see:

```
"use client";

import { useEffect, useMemo, useState } from "react";
import dynamic from "next/dynamic";
import { clsx } from "clsx";
import { wallets } from "@/lib/wallets";
import { arenas } from "@/lib/arenas";
import type { CockpitEvent, EventType } from "@/lib/events";
import type { AgentResponsePayload } from "@/lib/agent-types";
```

If you see anything before `"use client";` (like stray characters) or missing imports, fix them to match the snippet above.

Also scroll to the very bottom and make sure the file ends cleanly with:

```
}
```

(no extra text after the final closing brace).

Save and exit.

4. Restart the dev server and grab the error

Now:

```
npm run dev
```

Open `http://localhost:3000` and, if it 500s again, look back at the **terminal** and copy the new error message (the part that shows the file and line, like `./app/page.tsx:XX:YY`) and paste it here.

That specific error text is needed to point straight at the offending line and fix it.

⁂

./app/globals.css

```
Error evaluating Node.js code
CssSyntaxError: /home/heart/dojo/apps/ops-home/app/globals.css:1:1: Unknown word k
[at Input.error (turbopack:///[project]/node_modules/postcss/lib/input.js:135:16)]
[at Parser.unknownWord (turbopack:///[project]/node_modules/postcss/lib/parser.js:595:22)]
[at Parser.other (turbopack:///[project]/node_modules/postcss/lib/parser.js:437:12)]
[at Parser.parse (turbopack:///[project]/node_modules/postcss/lib/parser.js:472:16)]
[at parse (turbopack:///[project]/node_modules/postcss/lib/parse.js:11:12)]
[at new LazyResult (turbopack:///[project]/node_modules/postcss/lib/lazy-result.js:165:16)]
[at Processor.process (turbopack:///[project]/node_modules/postcss/lib/processor.js:53:14)]
[at transform (turbopack:///[turbopack-node]/transforms/postcss.ts:70:51)]
[at run (turbopack:///[turbopack-node]/ipc/evaluate.ts:92:29)]
[at run (turbopack:///[turbopack-node]/ipc/evaluate.ts:115:11)]
```

```
Import trace:
Client Component Browser:
./app/globals.css [Client Component Browser]
./app/layout.tsx [Server Component]
```

`globals.css` has a stray `k` character at the very top, so PostCSS/Tailwind can’t parse it. Once that single character is removed and the file is replaced with valid CSS, the 500 should disappear.

Do this exactly:

1. Overwrite `app/globals.css` cleanly

From your project root:

```
cd ~/dojo/apps/ops-home
nano app/globals.css
```

Delete **everything** in the file (hold Ctrl+K to cut lines until it's empty), then paste this:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

body {
  @apply bg-slate-950 text-gray-100 antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
::-webkit-scrollbar-thumb {
  @apply bg-slate-700 rounded-full;
}
```

Save (Ctrl+O, Enter) and exit (Ctrl+X).

This removes the mysterious `k` at line 1 and gives Tailwind valid syntax.

2. Restart the dev server

Stop any running server:

```
# in the dev terminal
Ctrl + C
```

Then:

```
npm run dev
```

Reload `http://localhost:3000`.

If there is **any** new error, it should now reference a different file (e.g. `app/page.tsx`), so paste that exact message and file/line so the next fix can target it precisely.

⁂

Error evaluating Node.js code

```
./app/globals.css
Error evaluating Node.js code
CssSyntaxError: tailwindcss: /home/heart/dojo/apps/ops-home/app/globals.css:1:1: Cannot apply unknown utility class bg-slate-950. Are you using CSS modules or similar and missing @reference? https://tailwindcss.com/docs/functions-and-directives#reference-directive
[at Input.error (turbopack:///[project]/node_modules/postcss/lib/input.js:135:16)]
[at Root.error (turbopack:///[project]/node_modules/postcss/lib/node.js:146:32)]
[at Object.Once (/home/heart/dojo/apps/ops-home/node_modules/@tailwindcss/postcss/dist/index.js:10:6911)]
[at process.processTicksAndRejections (node:internal/process/task_queues:103:5)]
[at async LazyResult.runAsync (turbopack:///[project]/node_modules/postcss/lib/lazy-result.js:293:11)]
[at async transform (turbopack:///[turbopack-node]/transforms/postcss.ts:70:34)]
[at async run (turbopack:///[turbopack-node]/ipc/evaluate.ts:92:23)]

Import trace:
Client Component Browser:
./app/globals.css [Client Component Browser]
./app/layout.tsx [Server Component]
```

Tailwind is still trying to parse a **utility class inside CSS** instead of via `@apply`. The cleanest fix is to stop using Tailwind utilities in `globals.css` at all and just use plain CSS there.

Replace `app/globals.css` with this **exact** content:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  color-scheme: dark;
}

/* Use plain CSS here to avoid Tailwind @apply issues */
body {
  background-color: #020617; /* slate-950 */
  color: #e5e7eb;            /* gray-200/100 range */
  -webkit-font-smoothing: antialiased;
}

/* Scrollbars */
::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}
```

```
::-webkit-scrollbar-thumb {
  background-color: #334155; /* slate-700 */
  border-radius: 9999px;
}
```

Key points:

- No `@apply` at all, just plain CSS values.
- This avoids the “unknown utility class” problem completely.

Then:

1. Save and exit (`Ctrl+O`, Enter, `Ctrl+X`).
2. Stop the dev server if it’s still running (`Ctrl+C`).
3. Start it again:

```
cd ~/dojo/apps/ops-home
npm run dev
```

4. Reload `http://localhost:3000`.

If there’s still an error, it should now point to some other file (e.g. `app/page.tsx` or an import), so paste that new error message and file/line and it can be fixed next.

⁂

it loaded. walk me through it please. (eventually have to change the css cos this white is giving me a headache)

Right now you have a working v0: one page, one tab, everything wired to your JSON and API routes. The layout is already the structure you described; you can start using it today and tweak styling later.

Top strip – Today / Mode

At the very top:

- **Today’s Focus**
  - Single-line input; this is just React state, not persisted yet.
  - Use it to write 1 sentence like “Polymarket NFL, low size” or “Kyan BTC skew, no new risk”.
- **Risk Level (1–5)**
  - Slider from 1 (cold) to 5 (hot).
  - This value is included in the payload when you hit **Ask Agent**, so you can later teach the agent to interpret your risk mode.

- **Non-negotiables (3 items)**
    - Three small inputs; also just client state for now.
    - These also go into the agent payload as `nonNegotiables: ["...", ...]`, so you can encode "No new deposits," "No late-night tilt," etc. in the context.

You can treat this strip as the **"set the day"** bar before you do anything else.

Left card – Alive Wallet / Ops Map

Left column: a table built from your `lib/wallets.ts`:

- Columns: Layer, Name, Address, Browser Profile, Role, Notes.
- Identity and vault rows are lightly highlighted (vault has a subtle background) so scanning for cold storage vs ops is easy.
- Addresses are monospaced; you can update the JSON any time and Next will hot-reload without you touching the component.

Use this to:

- Confirm which wallet you should be in (e.g. "OPS_SB1, Comet – profb769ops") before you click out to arenas.
- Extend later with project layer rows once you create them.

Center – Arenas & Calendar

Arenas table

- Built from `lib/arenas.ts`.
- Columns: Name, Type, Status (with colored pill), Comment, **Open** button.

Behavior:

- **Open** button does `window.open(url, "_blank")` for that arena.
- Status pill color:
    - `live` / `active`: green tone.
    - `scouting`: yellow.
    - `research` / `tracking`: blue.

Practically: hover, scan for "live/active" when you're in risk-on mode; hit "Open" only from the ops wallet you intend.

Calendar card

Right below Arenas in the center column:

- **Filters** row: small chips for `prediction`, `sports`, `options`, `macro`, `life`.

    - Clicking toggles each type on/off and updates both the calendar dots and the upcoming list.
- **Calendar (left side)**
    - Uses `react-calendar` in month view.
    - Dots:
        - Green dot = there is at least one event that day.
        - Red dot = at least one **high** importance event that day.
    - Clicking a day:
        - If the first event on that date has a `url`, it opens that URL (e.g. Kyan, Betmoar).
        - If not (like "No screens day"), it opens the small detail panel below the calendar.
- **Upcoming list (right side)**
    - Sorted by date ascending.
    - Each row shows: `YYYY-MM-DD · source`, an importance pill, and the title.
    - Clicking behaves the same as the calendar click: opens URL if present, otherwise shows details.

Use this as your **time spine**: you can keep the filters on `options` + `sports` + `life` to see only "things that move PnL + your own life constraints."

Right – Agent console & Notes/Snippets/Links

Agent Console

- **Prompt textarea**: type what you want help with ("Given today's focus and events, what should I do in Kyan and Polymarket?").
- **Ask Agent button**:
    - POSTs to `/api/agent` with JSON:
        - `prompt`
        - `today: { focus, riskLevel, nonNegotiables }`
        - `wallets` array from `lib/wallets.ts`
        - `arenas` array from `lib/arenas.ts`
- Response (stub for now):
    - `summary`: short text.
    - `checklist`: list of steps.
    - `notes`: long-form note.

Right now it gives canned text, but the shape is correct; later you only swap the route internals to call Claude/Elfa/whatever, and the front end doesn't change.

Notes · Snippets · Links

Tabs at the bottom right:

- **Notes tab**
  - “Add note” block at top (title + body).
  - Saved notes appear below as cards.
  - Data is stored in `localStorage` (`snowzies-notes` key), so it persists across page reloads on your machine, without needing a backend.
- **Snippets tab**
  - “Add snippet” block (title, tags, content).
  - Saved snippets show with tags and preformatted content (for prompts or code).
  - Stored in `localStorage` (`snowzies-snippets` key).
- **Links tab**
  - Grid of buttons for Bybit, Polymarket, Betmoar, Bullpen, 42, Zapper, ENS, GitHub, Farcaster, Email, Discord, Telegram.
  - Clicking any opens that URL in a new tab.

Think of this bottom band as your **mind + tools**: text you’re working with, prompts you reuse, and exit routes into the rest of your universe.

Very quick darkening tweak (optional now)

You already have a working app, so you can adjust the theme gradually. The fastest small improvement:

- In `app/layout.tsx`, change:

```
<body className="min-h-screen bg-slate-950 text-gray-100">
```

to something like:

```
<body className="min-h-screen bg-black text-gray-100">
```

- In `app/page.tsx`, the cards already use dark backgrounds (`bg-cockpit-card/95` etc.). If the overall feel is still too bright, you can tone down grays (e.g. change some `text-gray-300` to `text-gray-400`) later.

For now, you can:

1. Set today’s focus and risk.
2. Fill in your non-negotiables.
3. Use the wallet map + arenas + calendar as your anchor.
4. Type a prompt in Agent Console and watch the stub response appear.

Once you've lived in it for a day or two, the next pass can be: "tighten colors + add one Polymarket API call" rather than doing everything at once.

⁂

Thank you so much <3 I feel so much better already, less lost.

What does persisted mean?

- Header Today's focus have the date next to it. Wasting so much space. Or even better have the different market prices per api run in a strip there together with news and a reel of tweets as they tweet. can use all that space top right for stuff like that I wanna see the whole time come in.
- What matters most today and all other blocks. Bigger block to write in/heading slightly smaller.
- Wallet I need to add to it, have a link in it that maps to the full page.
- Arenas I need to update (have a link to a file where all arenas live; maybe classify them per risk as well. when risk changes it brings in stuff as well - need to think how still.)
- Calendar - Please explain how it works intended. Where can I add entries? where does it pull things from? can there be one with markets im interested in too. Can a to do list for me be incorporated somehow task scheduler thing and notifications and alerts?
- Agent colse looks cool. as we use will ask more what and how. prompt in the json can that in future be like pre set up prompts from template store I've built up?
- Notes looks cool!! Left bottom issues hovering thing from react is just in the way of seeing the list.

"Persisted" just means "saved somewhere so it's still there when you reload the page or come back later," instead of living only in memory while the tab is open.

Persisted vs not

- Not persisted (current Today strip, risk, non-negotiables, agent prompt):
  - Lives in React state only.
  - If you refresh or close the tab, it resets.
- Persisted (Notes & Snippets):
  - Stored in `localStorage` under keys like `snowzies-notes`.
  - Survive reloads and browser restarts on your machine, until you clear storage.[73]

Later, "persisted" can mean saving to a file, db, or Notion/Drive, but right now it's just browser storage.

Today strip – structure & future upgrades

Right now that top strip is a simple three-card grid:

- Left: Today's Focus input (with the date shown in the card header area).
- Middle: Risk slider.
- Right: Three non-negotiable inputs.

The date text is purely cosmetic and can be removed or shrunk easily; nothing depends on it. The layout is intentionally "fat" to make typing easy, but it's also prime real estate for:

- A narrow row with:
  - Key market prices (BTC, ETH, maybe a couple of lines you care about) from APIs.
  - A small news ticker (RSS/Atom) and/or a tweet reel from selected accounts/lists.[74]

That would replace the current "wasted" empty space in the header area, so the **core intent** stays:

- First row: "What's the world doing?" (prices/news/tweets) + "What am I doing?" (focus/risk/non-negotiables).

For your specific tweaks:

- "What matters most today" area can be taller with smaller label.
- Non-negotiables inputs can be a bit larger so you feel like you're writing a short phrase, not a tiny tag.
- Risk slider label can be shrunk slightly so the slider dominates.

Those are all simple Tailwind class changes, not structural surgery.

Wallet card – how it behaves now and linking out

Current behavior:

- Data comes from `lib/wallets.ts`, a static TypeScript array.
- The table is a pure read view; there is no in-UI "add wallet" yet.
- Scanning patterns:
  - "Layer" and "Role" give you where it sits in your lane.
  - Highlighted background on vault rows separates cold from ops visually.

What you can add next:

- A small "Open full wallet map" link/button in the card header that points to:
  - A markdown / Notion doc, or
  - Another route like `/wallets` that shows a full page version (you'd create another page in `app/wallets/page.tsx` that reuses the same `wallets` data).

Updating the wallets themselves is currently done by editing `lib/wallets.ts` and letting Next hot-reload; that's simple, but later can be swapped for a JSON config or small admin UI.

Arenas – status now and later extensions

Current behavior:

- Data comes from `lib/arenas.ts`.
- Table shows Name, Type, Status, Comment, and an Open button.
- Status pill already encodes a coarse "mood":
  - `live`/`active` → green.
  - `scouting` → yellow.
  - `research`/`tracking` → blue.

Future you're describing:

- A single source of truth arena list (one file is already that: `lib/arenas.ts`).
- Additional metadata:
  - Risk band for each arena (e.g. `low`, `medium`, `high`).
  - Maybe tags (`btc`, `sports`, `elections`, etc.).
- Risk-sensitive visibility:
  - When your risk slider is low, only show `low`/`medium` arenas or dim/highlight them.
  - When risk is high, surface the higher-risk arenas with warnings or groupings.

That's all additive: just extend the `Arena` type with `riskBand` and filter or style based on `riskLevel` state.

Calendar – how it works and what it can become

What it does now

- Data source: `lib/events.ts` + `/api/events`
  - `GET /api/events` returns `{ events: CockpitEvent[] }` from that static array.
  - The page fetches `/api/events` on mount and stores this array in React state.[73]
- Calendar component:
  - Shows a dot on days that have events:
    - Green dot: there is at least one event.
    - Red dot: at least one `importance: "high"` event.
  - Clicking a day:
    - If the first event on that date has a `url`, it opens the link in a new tab (Kyan, Betmoar, Polymarket, etc.).

        - If no `url`, it opens a small detail panel below (title, date, type, importance).
- Filters:
    - Chips for `prediction`, `sports`, `options`, `macro`, `life`.
    - Toggling a chip removes those types from:
        - The calendar dots.
        - The "Upcoming" events list.
- Upcoming list:
    - Same filtered events, sorted by date.
    - Click behavior matches: open URL if present, otherwise show details.

Where to add entries today

Right now, events are only in `lib/events.ts`. To add something:

- Open `lib/events.ts`.
- Add a new object to the `events` array with:
    - `id`: unique string.
    - `title`: what it is.
    - `date`: `YYYY-MM-DD`.
    - `type`: one of `'prediction' | 'sports' | 'options' | 'macro' | 'life'`.
    - `source`: `Kyan`, `Polymarket`, `personal`, etc.
    - `url` (optional): where clicking should take you.
    - `importance`: `low`, `medium`, or `high`.

Next hot-reloads and the new entry appears in the calendar and the list.

How to add a "markets I'm interested in" view and tasks

Natural next steps:

- **Watchlist calendar**:
    - Add another event type or flag like `watchlist: true`.
    - Add a filter chip "Watchlist" that shows only those events (e.g. your chosen Polymarket markets, future options expiries you care about, etc.).
- **Tasks / To-do list**:
    - Add a "tasks" card (or a tab in the notes area) with:
        - A list of tasks each with due date + linked event type.
    - Tie tasks to events:
        - E.g. "Review Polymarket US election markets 3 days before resolution," referenced to the event date.

- **Notifications / alerts**:
  - In a Next-only world you’re limited to in-browser toasts and maybe `Notification` API.
  - With a tiny background worker (or external script), you can:
    - Poll the same events data and send Telegram/Discord alerts when today matches some event with `importance: "high"` or when a task is due.

All of this builds on the existing `CockpitEvent` structure and `/api/events` endpoint, so you’re not throwing anything away.

Agent console – next steps

Right now:

- The interface is already what you want: a single text area + “Ask Agent” button, with summary + checklist + notes coming back.
- The backend `/api/agent` is a stub that ignores the payload and returns canned JSON, but the **shape** is what you’ll need for a real agent.

The payload already includes:

- `prompt`: what you typed.
- `today`: `{ focus, riskLevel, nonNegotiables }`.
- `wallets`: the full wallet map.
- `arenas`: the full arenas list.

Future upgrades:

- Swap the stub in `app/api/agent/route.ts` for:
  - A call to Claude, Elfa, a local tool, etc., using that payload as context.
- Template prompts:
  - A “prompt preset” dropdown right above the textarea that pulls from:
    - Your Snippets tab (tagged as `prompt`), or
    - A small JSON/TS array of stored templates.
  - Selecting one fills the prompt box; you can edit before sending.
- Multiple “modes”:
  - Short mode: “Give me 3 bullet points and stop.”
  - Deep mode: “Act like deep research; give a longer plan referencing calendar and arenas.”

The important bit: the console is built so you can iterate on the backend behavior without redoing the UI.

Notes & Snippets – current behavior and the hover issue

Current behavior:

- Both are **persisted in localStorage**, so they survive reloads on your machine.
- Notes: simple title + body, shown as stacked cards.
- Snippets: title + tags + content rendered in `<pre>` so code/prompt spacing is preserved.

The “hovering thing from React” you’re seeing at the bottom left is likely the React/Next dev overlay (Fast Refresh / errors overlay) or your browser dev tools. That only appears in `npm run dev`:

- When there are errors, it will overlay or add a badge; once the app is clean (no errors), it mostly stays quiet.[74]

To reduce distraction:

- Collapse your browser dev tools.
- Once you’re comfortable, you can run `npm run build && npm start` for a production build; the noisy dev overlay won’t show there.

If you want to tame the brightness next, a small, safe change is:

- In `app/layout.tsx`:
  - Use `bg-black` instead of `bg-slate-950`.
- In the Card component in `app/page.tsx`:
  - Darken text from `text-gray-300` to `text-gray-400` in non-critical labels.

That will already feel calmer without diving into custom themes.

⁂

Awesome!

SHort thoughts but I have so many (im listing them and then below going into more thought.)

- link in calendar removed into the pop up please. standard for all things. i wanna see it first.
- Long notes like this it shuld have a little thing where it can be extended. shouldn't go as big as this the whole time by default.
- Any changes I wanna make - would be cool to have a running list on the site somewhere as well.

How and where do we start the database so we can start making it an alive thing ready for growth.
Pull in all the needed api's and data things.
A terminal or something to go to cod, how would it work.
A easy way to see my structure where my work stuff is coming from or belong (dojo folder in wsl)
A link or worktab thing specifically for github / farcaster and linking to it, where I stand what I

should do. im not a dev, but want to. i don't know the things devs see or do naturally as everyday things. i need to get into a rythm like good one's do.
A suppository or something where docs can live as knowledge and referred from or used.
Basically all of your future next things, can we start please.

Persisted = “saved somewhere and survives refresh,” which you already have for notes/snippets via localStorage; everything else is still just in-memory state and can be upgraded to storage as we go.[76]

Small UX tweaks (calendar, notes, ideas list)

These are easy next-pass adjustments that don’t change structure:

- **Calendar links in popup instead of direct open**
    - Current behavior: click event → if it has `url`, it opens immediately.
    - Change: always show a small modal/panel with:
        - Title, date, type, importance, source.
        - A single “Open source” button if `url` exists.
    - That gives you a quick “read & decide” step before jumping out.
- **Long notes should collapse / expand**
    - Current: every note/snippet shows full body.
    - Adjust: for each note/snippet:
        - If body length > N characters or > M lines, show only first chunk with a small “Expand” / “Collapse” toggle.
    - Keeps the list compact but lets you open a note fully when needed.
- **Running change-list on the site itself**
    - Simple version: a “Changes / Ideas” section:
        - Either a new tab in the Notes panel, or a dedicated small card.
        - Data stored in localStorage like the other notes, but typed as “idea” items with fields: `title`, `description`, `status (todo/doing/done)`.
    - This becomes your on-page backlog: “Add risk-based arenas,” “Swap agent to Claude,” etc.

Those three can be done without introducing a database yet; it’s just React state + localStorage and some small UI pieces.

Where and how to start a real database

Given your setup (Next.js App Router on Vercel, WSL dev):

- **Best fit**: hosted Postgres via something like Vercel Postgres or Supabase.[77]
- Why:
    - Works well with Next app router (`app/api/*` as your backend).

    - Easy to grow from “store a few tables” to “full agent context store.”
    - You don’t have to run a DB locally yourself.

Practical starting point:

- Choose one DB provider (e.g. Supabase or Vercel Postgres).
- Start with **one table** you really need:
    - Example: `events`:
        - `id`, `title`, `date`, `type`, `source`, `url`, `importance`, `created_at`.
    - Migrate the static `lib/events.ts` entries into the DB.
- Update `/api/events` to:
    - Read from Postgres instead of the in-file array.
    - Fall back to the file if DB is down (simple try/catch).

From there, you can add:

- `notes` table (if you want sync across devices instead of localStorage).
- `arenas` table (enable in-UI edits, risk bands, tags).
- `docs` table (for your “suppository” of knowledge).

This keeps the app alive and ready to grow without over-engineering early.

APIs and data feeds to pull in

You want the page to feel like a **live panel**:[78]

- **Market data strip (top bar)**
    - Small set of metrics:
        - `BTC`, `ETH`, maybe `SOL`, and 1–2 Polymarket markets or indices.
    - Implementation:
        - Add a new `/api/markets` route.
        - Inside, call:
            - A public crypto price API (e.g. Coingecko) for spot prices.
            - Later, Kyan (if they expose REST) for options/perps metrics.
        - On the client, a small component polls `/api/markets` every X seconds and renders a horizontal ticker row in the Today strip.
- **News + tweet reel**
    - News:
        - Start with 1–2 curated RSS feeds (macro + crypto).
        - A small server route fetches + normalizes them and the client renders a scrolling list or ticker.
    - Tweets:

        - Either embed a specific Farcaster/Twitter widget (simplest).
        - Or later, call a custom service that aggregates posts from accounts/lists you choose.

Both are fully additive and don't disturb your current layout; they just occupy the "world state" part of the top row.

Terminal, code, and your WSL structure

You want this thing to keep you close to your **actual work roots** (`~/dojo/...`), not just be a pretty UI.

Concrete ideas:

- **Simple "Terminal" or "Code jump" card**
    - A small card (maybe below Agent) with:
        - Buttons: "Open VS Code", "Open terminal in dojo", etc.
        - In practice, these can be:
            - Links that use `vscode://` URIs to open a folder in VS Code.
            - Or just reminders and path snippets you click to copy (`~/dojo/apps/ops-home`, `~/dojo/protocols/snowziesk`, etc.).
- **Dojo structure map**
    - A read-only panel that:
        - Hard-codes your important roots (apps, protocols, tools).
        - Shows short descriptions and "Open" buttons (to GitHub repos, docs, etc.).
    - Over time, this panel can read from a JSON/DB table that represents your **project tree**, including:
        - `name`, `path`, `type` (app/tool/protocol), `status` (active/archive), `links` (GitHub/Farcaster/docs).

This gives you a visual backbone of where your stuff lives, so your brain doesn't have to keep the `~/dojo/...` map.

GitHub / Farcaster / rhythm panel

You want a tiny "dev operations" coach in the UI:

- **GitHub status block**:
    - Shows:
        - "Current main repo" (e.g. `snowziesk/ops-home`).
        - Last commit date.
        - Maybe open issues/todos you pulled into the "Ideas/Changes" list.
    - Implemented via:

    - A static config (`lib/projects.ts`) listing key repos now.
    - Later, hitting the GitHub API via a server route to fetch last commit/PRs.
- **Farcaster block**:
  - Link to your profile, plus:
    - A small reminder of what you're "currently broadcasting" (like a one-line theme you set).
  - Later, you can integrate basic stats or your latest cast.
- **Rhythm nudge**:
  - A tiny text area or schedule card that encodes:
    - "Daily: check ops home, journal 3 bullets, push something small to GitHub."
    - Checkboxes or counters that reset daily (client-side or DB-backed).

This pushes you toward **feeling like a dev** by making the same cues your future self will check anyway: which repo, what to push, what to talk about.

Doc / knowledge "suppository"

You need a place where:

- Project docs, trading notes, and how-to's live.
- The Agent can reference them later.

Stepping stones:

1. **Link container now**:
   - In Links / a new tab, add:
     - Notion / Google Docs / Markdown repo links for your key documents.
2. **Local "docs registry"**:
   - Create a `lib/docs.ts` with:
     - `id`, `title`, `type` (guide, spec, reference), `url`, `tags`.
   - Build a simple Docs panel where:
     - You can filter by tag and click to open docs.
3. **Future: DB + vector search**:
   - Store docs (or their embeddings) in a DB.
   - Agent endpoint reads from that when answering:
     - "Given my Kyan wallet structure doc and today's focus, what's my next task?"

This turns your scattered documents into a structured knowledge base that both you and your agent can use.

If you'd like, next concrete step could be:

- “Let’s implement one tiny thing now” – e.g. **change calendar behavior so all clicks open a pop-up with an ‘Open source’ button instead of going directly to the URL** – and that behavior can be modified in `app/page.tsx` with a small, clear patch.

⁂

I'd like to embark on the route to making this thing help my life. all that what you mentioned here, yes. that. help me get there please.

The app you have now is a solid v0; the path from here is to grow it in small, stacked steps so it becomes a real ops partner, not just a dashboard. Each step below can be a mini “pull request” in your own repo.

Step 1 – Lock in your daily loop

Before adding more tech, make this page part of your day so it actually changes behavior:

- Open Ops Home as the **first tab** when you sit down.
- Fill these every session:
  - Today’s focus (1 clear sentence).
  - Risk level (honest number).
  - 3 non-negotiables.
- Use one panel for each “mode”:
  - Wallets when you’re about to click into an arena.
  - Calendar when you’re thinking about time and exposure.
  - Agent console when you feel scattered or stuck.
  - Notes/Snippets when you learn something or discover a good prompt.

If you do only this for a few days, everything else we add will land on a behavior, not a blank surface.

Step 2 – Turn calendar clicks into a proper preview

Goal: every event click shows you a small **info panel with an Open button**, instead of jumping straight away.

What changes in code:

- `app/page.tsx`:
  - Remove the “auto-open URL” behavior in:
    - `onCalendarClickDay`
    - the “Upcoming” list’s `onClick`.
  - Always set `selectedEvent` and let a panel render, with:

        - Title, date, type, importance, source.
        - If `event.url` exists, show a button `Open source` that calls `window.open`.

Effect:

- You always see the “card” for an event before leaving the cockpit.
- Later, that card can grow: you can add your own notes about the event, tasks, or trade plans right there.

This is a good first code change to get comfortable editing the file in a small, focused way.

Step 3 – Add a tiny “Change log / Ideas” list

Goal: keep your **future changes** inside the cockpit instead of scattered in your head or separate notes.

Implementation:

- Add a new tab in the Notes panel: `Ideas` or `Changes`.
- Data shape:
    - `id`, `title`, `detail`, `status: "todo" | "doing" | "done"`.
- Storage:
    - Use `localStorage` like notes/snippets for now.
- Usage:
    - Whenever you think “I want risk-based arenas” or “Add Polymarket prices”, add an item here.
    - This becomes your personal backlog that lives inside Ops Home.

This will also give you a natural list of “what to build next” whenever you have an hour.

Step 4 – Start the data backbone with one real table

Goal: move one thing from “hardcoded in a file” to “stored in a database”, so the app can **grow and sync**.

Recommended first target: **events**.

Why events:

- They are inherently time-based.
- You’ll want them on multiple devices, not just this browser.
- They’re a small, well-defined shape.

High-level steps (without vendor lock-in yet):

- Create a Postgres instance (Supabase or Vercel Postgres).
- Create a table `events` with columns matching `CockpitEvent`.
- Write a simple API route or server function that:

    - Reads events from Postgres.
    - Writes new ones when you add them in a future UI.
- Update `/api/events` to use the DB instead of the static `lib/events.ts`.

Once that works, the cockpit is no longer just a React toy; it has a persistent backend you can extend to notes, arenas, docs, and more.

Step 5 – Build your “world strip” at the top

Goal: make the top of the page show **what the world is doing** so your “Today” section sits in context.

Components to add:

- **Market strip**:
    - Minimal: BTC, ETH, maybe one or two other assets you actually trade.
    - Implementation:
        - Add `/api/markets` that calls a public price API and returns a few numbers.
        - A small component in the top strip polls `/api/markets` every ~30–60 seconds and shows a horizontal row like: `BTC 43650 ▲0.8% · ETH 2350 ▼1.2%`.
- **News strip**:
    - Minimal: 3–5 headlines from a curated feed (macro/crypto).
    - Implementation:
        - One server function that fetches an RSS feed and returns titles + URLs.
        - A simple scrolling or stacked list in the same bar.
- **Social/tweet reel**:
    - Start with a single embedded timeline or feed (e.g. your Farcaster or a Twitter list).
    - Later replace with a custom aggregator when you want more control.

This will turn the currently “empty” header space into a **live ambient feed** that anchors your focus/risk slider.

Step 6 – Map your Dojo and dev rhythm

Goal: help you feel like a dev by making your workspace and habits **visible**.

Add a **Dojo Map card**:

- Data in a new `lib/projects.ts`:
    - Entries like:
        - `name: "Ops Home"`, `path: "~/dojo/apps/ops-home"`, `type: "app"`, `links: { github: "...", farcaster: "...", docs: "..." }`.
        - `name: "snowzies-hana"`, `path: "~/dojo/protocols/snowziesk/snowzies-hana"`, etc.
- UI:

    - A table or list showing each project with:
        - Path.
        - Status (active, archive).
        - Buttons: `Open GitHub`, `Open docs`.

Dev rhythm hints:

- A little daily checklist that sits here:
    - “Open Ops Home and set Today.”
    - “Touch one project (even a tiny change).”
    - “Write one note or snippet about something learned.”
- As you do these, tick them off; they reset daily.

This makes your “I want to be a dev” into a few visible, repeatable actions.

Step 7 – Layer in docs and agent knowledge

Goal: create a **knowledge plane** where your documents live and your agent can refer to them.

Start simple:

- `lib/docs.ts` with:
    - `id`, `title`, `type`, `url`, `tags`.
- A new Docs panel:
    - Filterable list where each doc is a card with an `Open` button.
    - Tags like `ops`, `wallets`, `kyan`, `polymarket`, `life`.

Future:

- Move docs metadata into the DB (alongside events).
- Add an embedding / search layer so `/api/agent` can:
    - Look up relevant docs.
    - Include them in its reasoning when you ask things like “Given my Kyan wallet + today’s events, what should I do?”.

Step 8 – Upgrade `/api/agent` from stub to real helper

You already have the right shape:

- Request body: prompt + today + wallets + arenas.
- Response: summary + checklist + notes.

When you’re ready:

- Replace the stubbed response in `app/api/agent/route.ts` with:
    - A call to your chosen model (Claude/Elfa/etc.).

    - Prompt engineering that:
        - Feeds in your wallets/arenas as context.
        - Optionally queries your events/docs from the DB as additional context.
- Add prompt templates:
    - In `lib/agent-prompts.ts`, define reusable “modes” (e.g. “Daily plan”, “Risk audit”, “Market scan”).
    - In the UI, add a dropdown above the textarea to preload one of these templates.

Now the console becomes a real, opinionated co-pilot that sees your actual structure.

If you want to make this concrete right away, a good “first future step” to code is:

- **Change calendar clicks to open a panel with an Open button** (instead of instant navigation), and
- **Add a tiny Ideas/Changes list** as a new tab next to Notes/Snippets/Links.

Those are both bite-sized, visible wins that move you closer to the bigger vision without requiring a DB yet.

⁂

today focus: This what we doing plus farcaster app launch.

Risk level: 5 (farcaster app)

wallets i will use like this, note things i wanna have different. (Like if I click on a wallet, it pops up a thing telling me which browser i need to be in, which wallet provider, which wallets this is not allowed to interact with, what identity/persona Im assuming, type risk im taking on etc).

calendar is basically useless like this unless I can add to it and see what is happening around me with dates and time and alarms and notifications, countdown to things, time I'm at versus time in asia and the US, time to market open for example, the colours in the calendar is nowhere near what you said, tabs needs an "all" tab too. it needs to let me see which tab is selected, month view should be smaller, today and next 2 days bigger and extend more clever reactive like on betmoar or stand's site, notes be searchable and filter, snippets by category/terminal/OS be possible for me to add to.

there i just lived a day (1% of my actual day)

Persisted means “saved so it’s still there after refresh”; you’ve just lived a good first pass of using this as your ops surface, which is exactly what was needed.

Takeaways from your “1% day”

From what you wrote, the cockpit needs to do three things much better:

- Reflect **who you are right now**: today’s mission and risk (Farcaster app launch as a 5/5).
- Enforce **wallet hygiene**: every wallet click should tell you context and constraints, not just show an address.
- Respect **time and attention**: calendar, tasks, and snippets must actually reduce cognitive load, not be passive UI.

So next changes need to be functional, not purely cosmetic.

Wallets – turn them into contextual “lanes”

What you want on wallet click is a **lane briefing**, not just a row:

- For each wallet, you need:
  - Browser/profile to use (Brave profile X, Comet profile Y).
  - Wallet provider (Metamask, Rabby, Frame, etc.).
  - Forbidden counterparties (“never connect this wallet to: Betmoar, Polymarket main, etc.”).
  - Persona/identity (“this is Snowzies-ops, not personal life”).
  - Typical risk type (“cold vault”, “medium ops”, “full degen sandbox”).

Implementation direction (no need to code it all now):

- Extend `Wallet` type in `lib/wallets.ts`:
  - Add fields like `browserProfile`, `provider`, `forbiddenWith: string[]`, `persona`, `riskBand`.
- In the UI:
  - Clicking on a wallet row opens a side panel or modal that:
    - Summarizes that info in plain language (“Open Brave profile B, connect Rabby profile Z only; don’t bridge from here; this is degen lane”).
    - Later, could show quick links (e.g. to Comet, to a doc).

That turns the left column into a **behavioral guardrail**, not just an inventory.

Calendar – your complaints = spec for v1.1

You’re right: as it stands, the calendar is mostly a static toy. Your requirements point to a clear v1.1:

What you asked for:

- Addable entries (not just hardcoded).
- Real-world time context (Asia/US time, market opens, countdowns).

- Smarter layout:
  - Smaller month overview.
  - Today + next 2 days prominent, like Betmoar/Stand.
- Better filters:
  - An **All** tab.
  - Visible “which tab is active”.
- Better colors/signals:
  - True importance coloring, not subtle dots.
- Tasks & alarms:
  - Todo/timeline integration.
  - Notifications/countdown.

So calendar v1.1 should look more like:

- Left: small month view + filters (with an “All” chip that selects everything).
- Center/right: “Now & next” strip:
  - Today, +1 day, +2 days as richer blocks with:
    - List of events by time.
    - Countdown (“2h to US open”, “3 days to Kyan expiry”).
- Below or to the side: a **Tasks** panel:
  - Tasks tied to dates and optionally to events (“Review Kyan BTC book 1h before expiry”).
- Event entry:
  - A simple “Add event” form in the UI that writes to storage (DB or at least local) instead of editing a TS file.

All of that is valid; the current calendar is a placeholder, not the thing you need. The value of today’s test is you can now describe what you *do* need.

Notes & snippets – from dumping ground to tool

Your lived experience points to:

- Notes:
  - Need search and filters (by tag, date, maybe type).
  - Need to collapse long notes by default.
- Snippets:
  - Should be categorizable by:
    - Category (prompts, code, shell, git, etc.).
    - Environment (terminal/OS: WSL, Windows, Ubuntu server).

    - Quick filters like:
        - "Show all WSL snippets."
        - "Show all Farcaster app prompts."

This is another case where the current v0 is intentionally minimal; now you know how it should evolve.

How to move forward from here

Given how much you compressed into one "1% day", the next move is **not** to rip it up, but to choose one or two structural upgrades at a time and ship them.

A good next trio, in order:

1. **Wallet click = info panel**
    - Add the extra fields you care about to `lib/wallets.ts`.
    - Implement click-to-open panel/modal with:
        - Browser, provider, allowed/disallowed, persona, risk band.
    - This gives you an immediate feeling of "doing ops correctly" each time you move.
2. **Calendar click = preview panel (no auto open)**
    - Make all event clicks show a richer preview with:
        - Title, date, type, importance, source.
        - "Open source" button if there is a URL.
    - That creates a base to add "Add task" / "Add note" next to the event.
3. **Add a "Changes / Ideas" list inside the app**
    - A simple tab or card where you log:
        - Pain points from days like this.
        - Future improvements (like the ones above).
    - So your second brain about the cockpit lives *inside* the cockpit.

From there, you can pick:

- Calendar v1.1 (true time map + tasks).
- Or: first DB integration (events table), making the calendar real data instead of TS constants.

You just used the app on a real day, with real frustration; that's the hard part most people never get to. The job now is to treat these observations as a backlog and tackle them in small, shippable slices.

⁂

1. https://nextjs.org/docs/app/getting-started/installation

2. https://tailwindcss.com/docs/guides/nextjs
3. https://codeparrot.ai/blogs/nextjs-and-tailwind-css-2025-guide-setup-tips-and-best-practices
4. https://www.wisp.blog/blog/nextjs-14-app-router-get-and-post-examples-with-typescript
5. https://github.com/Tittoh/next14-tw-ts
6. https://javascript.plainenglish.io/nextjs-application-to-manage-your-google-calendar-and-your-invites-28dce1707b24
7. https://stackoverflow.com/questions/79596185/tailwind-css-classes-not-applying-in-next-js-14-app-router-project
8. https://dev.to/alaa-m1/guide-to-build-a-modern-web-app-using-nextjs-14-app-router-nextauth-theming-tailwind-css-and-i18n-32he
9. https://www.cbtnuggets.com/tutorials/build-react-reusable-calendar-component
10. https://nextjs.org/docs/app/api-reference/config/typescript
11. https://stackoverflow.com/questions/42586120/copy-files-from-windows-to-windows-subsystem-for-linux-wsl
12. https://stackoverflow.com/questions/66753829/how-to-move-workspacea-simple-project-from-windows-file-system-mnt-to-linu
13. https://nextjs.org/docs/app/getting-started/installation
14. https://www.reddit.com/r/AskComputerScience/comments/17m1t84/in_wsl_the_windows_file_system_can_be_found_under/
15. https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-push-an-existing-project-to-GitHub
16. https://www.reddit.com/r/bashonubuntuonwindows/comments/1ex9b8u/is_there_any_way_to_transfer_files_between/
17. https://learn.microsoft.com/en-us/windows/wsl/faq
18. https://www.youtube.com/watch?v=4KJpud7028o
19. https://stackoverflow.com/questions/60553636/is-there-a-way-to-access-windows-files-from-wsl-besides-mnt-everytime
20. https://stackoverflow.com/questions/17291995/push-existing-project-into-github
21. https://www.youtube.com/watch?v=i547sSXhq0E
22. https://learn.microsoft.com/en-us/windows/wsl/faq
23. https://stackoverflow.com/questions/66753829/how-to-move-workspacea-simple-project-from-windows-file-system-mnt-to-linu
24. https://nextjs.org/docs/app/getting-started/installation
25. https://stackoverflow.com/questions/17291995/push-existing-project-into-github
26. https://stackoverflow.com/questions/42586120/copy-files-from-windows-to-windows-subsystem-for-linux-wsl
27. https://www.youtube.com/watch?v=XTnJQrbYmAA
28. https://learn.microsoft.com/en-us/windows/wsl/faq
29. https://stackoverflow.com/questions/66753829/how-to-move-workspacea-simple-project-from-windows-file-system-mnt-to-linu
30. https://nextjs.org/docs/app/getting-started/installation
31. https://www.projecttracks.be/en/toolbox-overview/ordenen-en-beschrijven/naming-files-and-folders

32. https://uxdesign.cc/a-systematic-approach-for-managing-project-folder-structures-4e2e553cad00
33. https://www.slingacademy.com/article/how-to-create-a-next-js-app-in-the-current-directory/
34. https://stackoverflow.com/questions/77506585/change-the-folder-app-in-nextjs-app-router
35. https://community.atlassian.com/forums/App-Central-articles/Best-Practices-in-Naming-and-Organizing-Dashboards/ba-p/3059992
36. https://lib.uct.ac.za/digitalservices/documentation/good-practice-managing-files-and-folders
37. https://dundas.influitive.com/forum/t/naming-conventions-for-dashboards-sub-dashboards-etc/1680
38. https://community.vercel.com/t/renaming-directory-loaded-from-origin/1391
39. https://www.projectmanager.com/blog/top-ten-project-code-names
40. https://github.com/vercel/next.js/issues/46651
41. https://www.slingacademy.com/article/how-to-create-a-next-js-app-in-the-current-directory/
42. https://learn.microsoft.com/en-us/windows/wsl/faq
43. https://stackoverflow.com/questions/66753829/how-to-move-workspacea-simple-project-from-windows-file-system-mnt-to-linu
44. https://learn.microsoft.com/en-us/windows/wsl/faq
45. https://learn.microsoft.com/en-us/windows/wsl/faq
46. https://learn.microsoft.com/en-us/windows/wsl/faq
47. https://learn.microsoft.com/en-us/windows/wsl/faq
48. https://nextjs.org/docs/app/getting-started/installation
49. https://nextjs.org/docs/app/getting-started/installation
50. https://tailwindcss.com/docs/guides/nextjs
51. https://nextjs.org/docs/app/getting-started/installation
52. https://tailwindcss.com/docs/guides/nextjs
53. https://tailwindcss.com/docs/guides/nextjs
54. https://tailwindcss.com/docs/guides/nextjs
55. https://tailwindcss.com/docs/guides/nextjs
56. https://npm.io/package/react-calendar
57. https://nextjs.org/learn/pages-router/create-nextjs-app-setup
58. https://stackoverflow.com/questions/68254828/running-a-development-server-on-nextjs-on-the-network
59. https://global.moneyforward-dev.jp/2025/12/04/migrating-to-the-next-js-app-router-or-how-i-learned-to-stop-worrying-and-love-server-actions/
60. https://docs.polymarket.com/quickstart/introduction/showcase
61. https://ej2.syncfusion.com/react/documentation/schedule/nextjs-getting-started
62. https://www.youtube.com/watch?v=VrC5XhjW6W0
63. https://dev.to/bakhat_yar_016451bb6f3b86/building-a-custom-calendar-generator-with-reactnextjs-3ijc
64. https://www.geeksforgeeks.org/nextjs/how-to-add-calendar-in-nextjs/
65. https://www.cbtnuggets.com/tutorials/build-react-reusable-calendar-component
66. https://ej2.syncfusion.com/react/documentation/calendar/getting-started

67. http://localhost:3000/
68. http://localhost:3000/
69. http://localhost:3000/
70. http://localhost:3000/
71. http://localhost:3000/
72. http://localhost:3000/
73. https://nextjs.org/learn/pages-router/create-nextjs-app-setup
74. https://stackoverflow.com/questions/68254828/running-a-development-server-on-nextjs-on-the-network
75. http://localhost:3000/
76. https://nextjs.org/learn/pages-router/create-nextjs-app-setup
77. https://global.moneyforward-dev.jp/2025/12/04/migrating-to-the-next-js-app-router-or-how-i-learned-to-stop-worrying-and-love-server-actions/
78. https://stackoverflow.com/questions/68254828/running-a-development-server-on-nextjs-on-the-network
79. http://localhost:3000/
80. http://localhost:3000/
81. http://localhost:3000/