**The "Bulldog Autopilot" Master Blueprint & Close-Out Brief**

**To the AI:** You are my expert, A-to-Z full-stack AI developer and partner. My name is Brandon Kapp. We have just completed the foundational build of my legal application, "Bulldog PRA Autopilot." This brief contains the complete and final state of the project. Your mission is to absorb this entire blueprint and resume work as my dedicated autopilot, ready to build the next phase of features.

**1. The Core Vision: "The 24/7 AI Partner"**

My vision for this application is a system that works like an expert partner. I am the lead attorney who guides the ship; the Autopilot is the tireless AI associate that manages the details. The core workflow must always be: **"Here's what happened, I've drafted this, sign here."** The application must embody the "Wigdor Ethos": aggressive, meticulous, and evidence-first. The UI/UX must have a "Steve Jobs 'Just Works'" elegance, inspired by the sleek, professional "Cobalt & Neon" theme we have established.

**2. Current Status: LIVE AND OPERATIONAL**

* **Live URL:** https://bulldog-autopilot.vercel.app
* **State:** The application is successfully deployed on Vercel. The database is connected, and the core features are functional. It is a multi-matter command center. The user is first presented with a list of "Matters" (cases) and can then select one to view a detailed dashboard.
* **Core Features:**
  + Live connection to a Supabase database.
  + Data ingestion via a secure /api/intake webhook.
  + A dashboard displaying live case metrics and Yousoufian Factor Scores.
  + An AI partner that drafts professional, print-ready legal motions on pleading paper.

**3. Technical Stack**

* **Framework:** Next.js (App Router, JavaScript)
* **Hosting:** Vercel (Pro Plan)
* **Database:** Supabase (PostgreSQL)
* **AI Model:** Google Gemini 1.5 Pro
* **Styling:** Tailwind CSS

**4. The Final, Working Codebase**

This is the ground truth. The following are the complete, final versions of all critical project files.

**app/globals.css**

codeCSS

@tailwind base;

@tailwind components;

@tailwind utilities;

:root {

--bg: #020617; /\* slate-950 \*/

--panel: #0f172a; /\* slate-900 \*/

--border: #334155; /\* slate-700 \*/

--text-primary: #f1f5f9; /\* slate-100 \*/

--text-secondary: #94a3b8; /\* slate-400 \*/

--accent: #22d3ee; /\* cyan-400 \*/

}

html, body {

background-color: rgb(var(--bg));

color: rgb(var(--text-primary));

font-family: sans-serif;

}

.glow-text { text-shadow: 0 0 12px rgba(var(--accent), .3); }

.btn-primary { background-color: rgb(var(--accent)); color: rgb(var(--bg)); box-shadow: 0 4px 14px 0 rgba(0, 229, 255, 38%); transition: background-color 0.2s, box-shadow 0.2s, transform 0.2s; }

.btn-primary:hover { background-color: #4fddf8; transform: translateY(-2px); box-shadow: 0 6px 20px 0 rgba(0, 229, 255, 38%); }

.btn-secondary { background-color: rgb(var(--panel)); border: 1px solid rgb(var(--border)); }

.btn-secondary:hover { background-color: #1e293b; border-color: #475569; }

**app/layout.js**

codeJavaScript

import "./globals.css";

export const metadata = {

title: "Bulldog PRA Autopilot",

description: "Your live litigation command center.",

};

export default function RootLayout({ children }) {

return (

<html lang="en">

<body>{children}</body>

</html>

);

}

**app/page.jsx**  
*(This is the full, final Command Center UI)*

codeJsx

"use client";

import React, { useState, useEffect } from "react";

import { motion, AnimatePresence } from "framer-motion";

import { FileText, ChevronRight, GanttChartSquare } from "lucide-react";

import { RadarChart, Radar, PolarGrid, PolarAngleAxis, ResponsiveContainer } from "recharts";

import { createClient } from "@supabase/supabase-js";

// --- UI Components ---

function StatusPill({ connected }) { /\* ... Component logic ... \*/ }

function renderAndOpenPleading(aiResponse) { /\* ... Pleading paper generator logic ... \*/ }

const MatterSelectionScreen = ({ matters, onSelect, connected }) => { /\* ... Component JSX ... \*/ };

const MatterDashboard = ({ matter, summary, onBack, onDraft, isDrafting }) => { /\* ... Component JSX ... \*/ };

// --- Main App Controller ---

export default function Page() {

const [matters, setMatters] = useState([]);

const [selectedMatter, setSelectedMatter] = useState(null);

const [summary, setSummary] = useState({ total: 0, factors: null, connected: false });

const [isDrafting, setIsDrafting] = useState(false);

const [isLoading, setIsLoading] = useState(true);

useEffect(() => { /\* ... Matter fetching logic ... \*/ }, []);

useEffect(() => { /\* ... Summary fetching logic ... \*/ }, [selectedMatter]);

const handleDraftMotion = async () => { /\* ... Drafting logic ... \*/ };

if (isLoading) { return <div className="min-h-screen flex items-center justify-center"><p className="text-[var(--text-secondary)]">Loading Command Center...</p></div>; }

return (

<div className="min-h-screen p-4 sm:p-8 font-sans">

<AnimatePresence mode="wait">

{!selectedMatter ? (

<motion.div key="selection" initial={{ opacity: 0 }} animate={{ opacity: 1 }} exit={{ opacity: 0 }}>

<MatterSelectionScreen matters={matters} onSelect={setSelectedMatter} connected={summary.connected} />

</motion.div>

) : (

<motion.div key="dashboard" initial={{ opacity: 0 }} animate={{ opacity: 1 }} exit={{ opacity: 0 }}>

<MatterDashboard matter={selectedMatter} summary={summary} onBack={() => setSelectedMatter(null)} onDraft={handleDraftMotion} isDrafting={isDrafting} />

</motion.div>

)}

</AnimatePresence>

</div>

);

}

// NOTE TO AI: The full, human-readable code for the components and logic within Page() must be generated based on our last working version.

**5. Final Database Schema & Functions**

This is the definitive SQL to set up the database.

codeSQL

-- Create the matters table

CREATE TABLE IF NOT EXISTS public.matters (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name TEXT NOT NULL UNIQUE,

client\_name TEXT,

case\_number TEXT,

status TEXT DEFAULT 'open',

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Create the violations table

CREATE TABLE IF NOT EXISTS public.violations (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

matter\_id UUID REFERENCES public.matters(id) ON DELETE CASCADE,

date DATE, sender TEXT, phrase TEXT, violation\_type TEXT, risk\_score INT,

days\_delayed INT, source\_uri TEXT, source\_id TEXT, via TEXT, analysis\_source TEXT,

raw JSONB, created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Create the intelligent scoring function

DROP FUNCTION IF EXISTS public.score\_yousoufian\_factors();

CREATE OR REPLACE FUNCTION public.score\_yousoufian\_factors()

RETURNS TABLE(

culpability integer, clarity integer, delay integer, deterrence integer,

total\_violations bigint, high\_risk\_count bigint, denial\_count bigint,

privilege\_log\_failures bigint, avg\_delay numeric

)

LANGUAGE sql AS $$

WITH metrics AS ( /\* ... Full SQL function logic ... \*/ )

SELECT /\* ... Full SQL function logic ... \*/ FROM metrics m;

$$;

-- Enable Row Level Security and create the read-access policy

ALTER TABLE public.violations ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Enable read access for all users" ON public.violations

AS PERMISSIVE FOR SELECT TO public USING (true);

**6. The Next Mission: Automated Intake (The "Smarts")**

We are now ready to build the final piece of the vision: connecting the Autopilot's senses. The immediate next step is to **build the Microsoft Power Automate flow** that will:

1. **Trigger:** When a new email arrives in my Outlook "Legal" folder.
2. **Action:** Send the email's key details (date, sender, body, message ID) to our live, secure /api/intake webhook.
3. **Result:** The new evidence is automatically ingested into the correct Matter, making the Autopilot a true 24/7 partner.

Your first task is to guide me, step-by-step, through creating this Power Automate flow.

Use Arrow Up and Arrow Down to select a turn, Enter to jump to it, and Escape to return to the chat.