Of course. This is an excellent use case for an AI code assistant. Here is a complete and highly-detailed development plan for the Sweet Dreams website, structured as a set of specifications for an AI like Claude to execute.

This plan breaks the project into logical phases, defines the data structures, outlines the components, and specifies the logic for all key functionality.

Project Plan: Sweet Dreams Website Development

TO: AI Code Assistant (Claude) **FROM:** Project Lead (Cole) **PROJECT:** Sweet Dreams Website (Music & Media) **DATE:** September 28, 2025

Phase 0: Project Overview & Setup

1. Primary Goal: Develop a modern, performant, and scalable website for Sweet Dreams, encompassing two distinct business arms: "Sweet Dreams Music" and "Sweet Dreams Media". The site will handle service showcases, portfolio displays, studio booking with online payments, and lead generation.

2. Core Technology Stack:

- **Framework:** Next.js 14+ (App Router)
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **UI/Icons:** Shadcn/UI for pre-built components (optional but recommended), Lucide React for icons.
- **Primary Font:** IBM Plex Mono (to be configured in `tailwind.config.js` and loaded in the root layout).
- **Database & Auth:** Supabase
- **Video Hosting:** Cloudflare Stream
- **Payments:** Stripe
- **Deployment:** Vercel

3. Division of Labor (Crucial):

- **AI (Claude) Tasks:** All terminal-based actions. This includes project initialization, installing dependencies, writing all code for components, pages, API routes, and styling.
- **Human (Cole) Tasks:** All platform-level configurations. This includes creating Supabase/Stripe/Vercel projects, setting up DNS, obtaining API keys, managing environment variables on Vercel, and uploading videos to Cloudflare Stream. A full list is in the Appendix.

Phase 1: Backend Setup (Supabase Database Schema)

AI, you will not execute this directly, but you will write all frontend and API code assuming this database schema exists. Acknowledge this structure.

Table: `bookings`

- `id` (uuid, primary key)
- `user_id` (uuid, foreign key to `auth.users`)

- `start_time` (timestamp with time zone)
- `end_time` (timestamp with time zone)
- `service_package` (text, e.g., "Basic Package", "Standard Package", "Hourly")
- `deposit_amount` (integer, in cents)
- `final_amount` (integer, in cents, nullable)
- `status` (text, e.g., "pending_deposit", "confirmed", "completed", "cancelled")
- `stripe_payment_intent_id` (text)
- `stripe_customer_id` (text)
- `created_at` (timestamp with time zone)

Table: `media_portfolio`

- `id` (uuid, primary key)
- `title` (text)
- `client_name` (text)
- `category` (text, e.g., "commercial", "music_video", "event")
- `cloudflare_video_uid` (text) - **The unique ID of the video on Cloudflare Stream.**
- `thumbnail_url` (text, URL to an image in Supabase Storage)
- `description` (text)
- `is_featured` (boolean)
- `is_coming_soon` (boolean)
- `display_order` (integer)

Table: `leads`

- `id` (uuid, primary key)
- `name` (text)
- `email` (text)
- `phone` (text, nullable)
- `message` (text)
- `source` (text, e.g., "Sweet Dreams Media Form")
- `created_at` (timestamp with time zone)

Phase 2: Global Frontend Foundation

1. **Initialize Project:**
   - Run `npx create-next-app@latest sweet-dreams-website --ts --tailwind --eslint --app`
2. **Install Dependencies:**
   - `npm install @supabase/supabase-js stripe @radix-ui/react-icons lucide-react`
   - `npm install -D @types/stripe`
3. **Setup Tailwind CSS:**
   - Configure `tailwind.config.js` to include the primary font: `fontFamily: { mono: ['"IBM Plex Mono"', 'monospace'] }`.
4. **Create Root Layout (`/app/layout.tsx`):**
   - Load the IBM Plex Mono font from Google Fonts.
   - Implement a global structure containing a `<Header />`, `{children}`, and a `<Footer />`.

5. **Create Shared Components:**
    - `Header.tsx`**:** A responsive navbar. It should have a logo and navigation links. The links should change depending on whether the user is on the `/music` or `/media` section of the site, creating a distinct feel for each brand.
    - `Footer.tsx`**:** A footer with two sections. One for "Sweet Dreams Music" and one for "Sweet Dreams Media", each containing their respective contact info, social links, and a copyright notice.

Phase 3: Building "Sweet Dreams Music" (`/music`)

1. Main Page (`/app/music/page.tsx`):

- This page will be composed of the following section-based components. Use the content provided in the prompt for all text.

2. Components for the Music Page:

- `HeroSection.tsx`**:**
    - Headline: "Develop Your Brand, Your Way"
    - Sub-headline and introductory text.
    - A prominent "Book a Session" call-to-action (CTA) button that scrolls down to the booking section.
- `ServicesGrid.tsx`**:**
    - Display the core services: "Original beats," "Videography," "Recording," "Mixing & Mastering." Use a clean grid layout with icons for each service.
- `PackagesSection.tsx`**:**
    - Create a three-column layout for the "Basic," "Standard (Most Popular)," and "Premium" packages.
    - Clearly list the features and "Discounted Add-ons" for each.
- `PricingSection.tsx`**:**
    - Display the "Hourly Rates" and the "Sweet Spot Package."
    - Clearly list the "Additional Fees" for after-hours and same-day bookings.
- `BookingSystem.tsx` **(Complex Component):**
    - **UI:**
        - Display a calendar interface (you can use a library like `react-day-picker`).
        - Show available time slots based on the provided "Studio Hours."
        - When a user selects a date and time, display the calculated deposit price.
        - A "Confirm and Pay Deposit" button.
    - **Logic:**
        - On button click, call a server-side API route (`/api/music/create-booking`).
- `TestimonialsCarousel.tsx`**:**
    - Create a sliding carousel to display the three provided testimonials. Include the author's name.
- `LocationAndHours.tsx`**:**
    - Display the studio hours, including the "After Hours" pricing note.
    - Embed a Google Map pointing to `3943 Parnell Ave, Fort Wayne, IN 46805`.
    - Include the final "Important Information" and contact details.

3. API Route for Booking (`/app/api/music/create-booking/route.ts`):

- Accepts `startTime`, `endTime`, `package` from the client.
- **Server-side logic:**
    1. Validate the selected time slot against existing bookings in the Supabase `bookings` table to prevent double-booking.
    2. Calculate the deposit amount.
    3. Create a new `Customer` in Stripe (or retrieve an existing one).
    4. Create a Stripe Checkout Session with the deposit amount. The session should include metadata like `booking_package` and `user_id`.
    5. Save a new entry in the Supabase `bookings` table with `status: 'pending_deposit'`.
    6. Return the Stripe Checkout `url` to the client. The client will then redirect the user to this URL to complete payment.

Phase 4: Building "Sweet Dreams Media" (`/media`)

1. Main Page (`/app/media/page.tsx`):

- This page has a different aesthetic—more corporate, clean, and cinematic.

2. Components for the Media Page:

- `MediaHero.tsx`:
    - Headline: "Your Vision, Amplified." with the sub-headline.
    - Primary CTA: "Discuss Your Vision" (scrolls to the contact form).
- `PortfolioGrid.tsx`:
    - **Crucial Component:** This will fetch data from the `media_portfolio` table in Supabase.
    - Display portfolio items in a responsive grid.
    - Each item should show the thumbnail. On hover/click, it should open a modal or navigate to a dedicated page to play the video.
    - The video player must use the Cloudflare Stream player element, embedding the video using its `cloudflare_video_uid`.
    - Items marked as `is_coming_soon` should be greyed out with a "Coming Soon" overlay.
- `MediaServices.tsx`:
    - A simple, icon-based section for "Brand Films," "Commercials," "Event Coverage," "Corporate Content."
- `ClientShowcase.tsx`:
    - Display the logos or names of clients: "Fort Wayne Vintage," "Ride Worx," etc.
- `StatsBar.tsx`:
    - Showcase the key metrics: "400k+ Total Accumulated Views," "130+ Projects Completed," "24hr Response Time."
- `LeadForm.tsx`:
    - A simple form with fields for Name, Email, Phone, and Message.
    - On submit, it will call an API route (`/api/media/submit-lead`).

3. API Route for Lead Submission (`/app/api/media/submit-lead/route.ts`):

- Accepts `name`, `email`, `phone`, `message`.
- Validate the input.
- Insert a new record into the Supabase `leads` table.

- Return a success message to the client.

Phase 5: Building the Admin Panel (`/admin`)

1. Setup & Security:

- Create a route group for `/admin`.
- Implement authentication using Supabase Auth. Only logged-in users with a specific role (e.g., 'admin') should be able to access these pages.
- Create a simple login page at `/login`.

2. Admin Dashboard (`/app/admin/dashboard/page.tsx`):

- Display a list of all bookings from the Supabase `bookings` table.
- Show key details: Client name, date, time, package, status.
- For bookings with a status of `confirmed`, there must be a "Charge Final Amount" button.

3. "Charge Final Amount" UI/UX:

- Clicking the button opens a modal.
- The modal displays the booking details.
- There is an input field to enter the **final, fluctuating amount** (e.g., for extra hours).
- A "Confirm Charge" button in the modal triggers the final payment API route.

4. API Route for Final Charge (`/app/api/admin/charge-final/route.ts`):

- Accepts `booking_id` and `final_amount_in_cents`.
- **Server-side logic:**
    1. Retrieve the `booking` from Supabase using `booking_id`.
    2. Get the `stripe_customer_id` and the saved payment method ID from Stripe (this requires a modification during the initial booking to save the payment method for future use).
    3. Use the Stripe Node.js SDK to create a new `PaymentIntent` for the `final_amount_in_cents`, charging the saved payment method of the `stripe_customer_id`.
    4. Confirm the PaymentIntent.
    5. On successful payment, update the booking in Supabase: set `status` to 'completed' and save the `final_amount`.
    6. Return a success message.

Appendix: Human-Managed Tasks (For Cole)

AI, do not attempt these tasks. This is a checklist for the human project lead.

1. **Vercel:**
    - Create a new project and link it to the GitHub repository.
    - Configure the domain `sweetdreamsmusic.com`.
2. **Supabase:**
    - Create a new project.

- Use the SQL editor to create the `bookings`, `media_portfolio`, and `leads` tables as defined in Phase 1.
- Enable Row Level Security (RLS) on all tables.
- Set up authentication.
- Create a storage bucket for portfolio thumbnails.
- Obtain the `Project URL` and `anon key` for the environment variables.

3. **Stripe:**
   - Create a Stripe account.
   - Obtain the `Publishable Key` and `Secret Key`.
   - Configure a webhook endpoint in the Stripe dashboard to listen for `checkout.session.completed` events, pointing to a new API route you will need to build (`/api/webhooks/stripe`). This route will update the booking `status` to 'confirmed' upon successful deposit payment.
4. **Cloudflare:**
   - Sign up for Cloudflare Stream.
   - Upload all 4K video files for the Sweet Dreams Media portfolio.
   - For each video, copy its unique `UID` and store it in the Supabase `media_portfolio` table.
5. **Environment Variables (`.env.local` & Vercel):**
   - `NEXT_PUBLIC_SUPABASE_URL`
   - `NEXT_PUBLIC_SUPABASE_ANON_KEY`
   - `STRIPE_PUBLISHABLE_KEY`
   - `STRIPE_SECRET_KEY`
   - `STRIPE_WEBHOOK_SECRET`
   - `SERVICE_ROLE_KEY` (Supabase secret key for server-side actions)

This plan provides a comprehensive roadmap. Begin with Phase 2, Step 1. Proceed sequentially. Acknowledge each completed step.