Case Study: Building a Portfolio Site with AI-Assisted Development

Overview

I built a full-stack portfolio website in a single day using Next.js, Sanity CMS, and Claude Code as my AI pair-programmer. Starting from a basic Next.js scaffold, we implemented a responsive card-based archive system, integrated four third-party services (Spotify, Cal.com, YouForm, Lemon Squeezy), added keyboard navigation with procedural audio, and deployed to production with a complete CI/CD pipeline. The interesting part wasn't just that it worked—it's that we iterated through real problems together: debugging overlay conflicts, fixing mobile layouts, and designing a hybrid sorting system when the obvious solution wasn't flexible enough. By the end, I had production-grade code, comprehensive documentation, and enough understanding of the architecture to maintain it myself.

Technical Architecture

Stack

Layer	Technology
Framework	Next.js 14 (App Router)
Language	TypeScript
Styling	Tailwind CSS
CMS	Sanity v3
Hosting	Vercel
Payments	Lemon Squeezy

Key Architectural Decisions

Server Components + Client Islands
The page renders server-side via Next.js App Router. Interactive elements (keyboard nav, sound, modals) are isolated in a HomeClient component marked with 'use client'. This keeps the initial bundle small while enabling rich interactivity.

Headless CMS with GROQ
Content lives in Sanity, queried via GROQ. The schema defines 8 content types (Music, Art, Writing, etc.) with type-specific fields. A custom order field enables manual sorting override while preserving chronological defaults.

Dynamic Third-Party Loading
Embed scripts (Cal.com, YouForm, Lemon Squeezy) load conditionally based on content type. Scripts inject only when needed, avoiding unnecessary payload for items that don't require them.

Feature Implementation

1. Responsive Navigation System

Problem: Desktop needed horizontal filter tabs with keyboard shortcuts. Mobile needed touch-friendly targets without the complexity.

Solution: Dual-layout component using Tailwind's sm: breakpoint. Desktop renders a horizontal row with sound toggle and live clock. Mobile renders a 3x3 button grid with the clock relocated to the footer.

```
// Desktop: horizontal row
<div className="hidden sm:flex items-center justify-between">

// Mobile: 3x3 grid
<div className="sm:hidden grid grid-cols-3 gap-2">
```

Sound toggle and keyboard navigation are desktop-only—detected via breakpoint, not user agent.

2. Procedural Audio System

Problem: Wanted subtle hover/navigation sounds without loading audio files.

Solution: Web Audio API generates tones procedurally. A sine wave oscillator (800Hz) with rapid gain decay (30ms) creates a soft click. Sound state persists to localStorage.

```
const oscillator = audioContext.createOscillator()
oscillator.frequency.value = 800
oscillator.type = 'sine'

const gainNode = audioContext.createGain()
gainNode.gain.setValueAtTime(0.015, audioContext.currentTime)
gainNode.gain.exponentialRampToValueAtTime(0.001, audioContext.currentTime + 0.03)
```

3. Keyboard Navigation

Problem: Arrow keys should cycle through filter categories, with visual feedback.

Solution: useEffect listens for ArrowLeft/ArrowRight, updates the active filter, and triggers a 150ms flash on the corresponding arrow indicator in the footer.

```
useEffect(() => {
  const handleKeyDown = (e: KeyboardEvent) => {
    if (selectedItem) return // Don't navigate when modal is open
    const currentIndex = filterCategories.indexOf(activeFilter)
    if (e.key === 'ArrowLeft') {
      const newIndex = currentIndex <= 0 ? filterCategories.length - 1 : currentIndex - 1
      setActiveFilter(filterCategories[newIndex])
    }
    // ...
  }
  window.addEventListener('keydown', handleKeyDown)
  return () => window.removeEventListener('keydown', handleKeyDown)
}, [activeFilter, selectedItem])
```

4. Lemon Squeezy Overlay Checkout

Problem: Lemon Squeezy's overlay checkout conflicted with the site's modal. Opening checkout while the modal was open caused z-index battles and scroll lock conflicts.

Solution: Three-part fix:

1. Use anchor tag with lemonsqueezy-button class (triggers SDK's overlay mode)
2. Close the site modal before checkout opens
3. Use a ref to prevent scroll restoration during the handoff

```
const closingForCheckoutRef = useRef(false)

useEffect(() => {
  document.body.style.overflow = 'hidden'
  return () => {
    if (!closingForCheckoutRef.current) {
      document.body.style.overflow = 'auto'
    }
  }
}, [])

const handleLemonSqueezyClick = () => {
  closingForCheckoutRef.current = true
  onClose() // Close site modal, keep scroll locked for LS overlay
}
```

5. Hybrid Content Ordering

Problem: Wanted newest content first (chronological), but also needed to pin certain items (contact forms) to the bottom.

Solution: Two-field sorting system. An order field (integer) takes priority, with date as secondary sort. Items default to order: 0, sorting by date among themselves. Setting order: 100 pushes items to the end regardless of date.

```
*[_type == "archiveItem"] | order(coalesce(order, 0) asc, date desc)
```

This provides automatic chronological behavior for most content while allowing manual override when needed.

6. Multi-Embed Modal System

Problem: Single modal component needed to handle Spotify players, YouForm contact forms, Cal.com booking calendars, image galleries, and rich text—conditionally.

Solution: Slug-based detection for special embeds, field-based detection for standard embeds.

```
const isContactForm = item.slug?.current === 'send-message'
const isBookingForm = item.slug?.current === 'book-a-call'

// Conditional rendering
{isContactForm && <YouFormEmbed />}
{isBookingForm && <CalEmbed />}
{item.embedUrl && <SpotifyEmbed url={item.embedUrl} />}
{item.gallery?.length > 0 && <Gallery images={item.gallery} />}
```

Each embed type loads its own script dynamically, initializing only when the relevant modal opens.

CMS Schema Design

The archiveItem schema supports 8 content types with shared and type-specific fields:

Field	Type	Purpose
title	string	Display name
slug	slug	URL identifier, special embed detection
type	enum	Category (Music, Art, Shop, etc.)
label	string	Subtype ("Single", "Oil on Canvas")
order	number	Manual sort override
date	date	Chronological sorting
coverImage	image	Card background + modal hero
embedUrl	url	Spotify/Apple Music
externalUrl	url	External links
lemonSqueezyUrl	url	Checkout integration
body	portable text	Rich content for Writing type
gallery	image[]	Additional images

Content Management via CLI

Content can be managed through Sanity Studio or programmatically via the Mutations API. A typical music release workflow:

1. **Upload image asset** → Returns asset reference ID
2. **Create document** → POST mutation with asset reference
3. **Auto-deploy** → Sanity CDN invalidates, site reflects changes

```
# Upload
curl -s "$SANITY_API/assets/images/production" \
  -H "Authorization: Bearer $TOKEN" \
  --data-binary @album-art.jpg

# Create
curl -s "$SANITY_API/data/mutate/production" \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"mutations": [{"create": {...}}]}'
```

Deployment Pipeline

Trigger	Action
git push origin main	Vercel builds + deploys site
npx sanity deploy	Deploys Sanity Studio
Sanity content change	CDN invalidates, site updates

Vercel maintains deployment history with instant rollback. Git provides code versioning. Sanity maintains document revision history. The three systems together create a robust recovery path for any failure mode.

Performance Considerations

- **Server-side rendering**: Initial HTML includes content, no layout shift
- **Image optimization**: Sanity's image pipeline + Next.js <Image> component
- **Conditional script loading**: Third-party SDKs load only when needed
- **Procedural audio**: No audio file requests
- **Tailwind purge**: Only used utilities ship to production

Lessons Learned

1. **Overlay conflicts are subtle**: Z-index and scroll lock interactions between multiple overlay systems (site modal + payment overlay) required careful state management.

2. **Sorting flexibility matters**: A simple "sort by date" seemed sufficient until edge cases emerged. The hybrid order/date system added minimal complexity while solving real problems.

3. **Mobile-first isn't always right**: Starting with the desktop experience and adapting for mobile worked better for this interaction-heavy design. The desktop version defined the feature set; mobile got a thoughtful subset.

4. **Documentation is a feature**: Comprehensive CLAUDE.md and README.md files made context recovery between sessions seamless and will make future maintenance possible.

Metrics

- **Development time**: ~8 hours (single day)
- **Commits**: 15+
- **Third-party integrations**: 4 (Spotify, Cal.com, YouForm, Lemon Squeezy)
- **Content types**: 8
- **Lines of code**: ~1,500 (excluding dependencies)

Repository

- **Live site**: https://samhayek-site.vercel.app
- **Sanity Studio**: https://samhayek.sanity.studio
- **Stack**: Next.js 14 / TypeScript / Tailwind CSS / Sanity v3 / Vercel