

■ MODERN PORTFOLIO WEBSITE

Complete Development Documentation

Next.js + TypeScript + GSAP + Framer Motion

Version:	1.0
Date:	February 08, 2026
Framework:	Next.js 14+
Language:	TypeScript
Styling:	Tailwind CSS

■ Table of Contents

1. Introduction & Overview
2. Design Concepts
3. Technology Stack
4. Project Setup
5. Hero Section Implementation
6. About Section
7. Projects Section
8. Tech Stack Display
9. Journey Timeline
10. GitHub Activity
11. My Favorite Songs
12. Animation Techniques
13. Performance Optimization
14. Deployment Guide

1. Introduction & Overview

This documentation provides a complete guide to building a modern, interactive portfolio website using Next.js, TypeScript, and advanced animation libraries. The portfolio follows a narrative storytelling approach, where each section reveals different aspects of your professional journey through smooth scroll-based animations and interactive elements.

Key Features:

- Parallax scrolling effects for depth and immersion
- Scroll-triggered animations using GSAP and Framer Motion
- Smooth scrolling with Lenis or Locomotive Scroll
- Interactive elements with hover and click effects
- Responsive design optimized for all devices
- Performance-optimized with Next.js Image and lazy loading

2. Design Concepts

2.1 Visual Storytelling Journey

The portfolio is structured as a narrative journey, guiding visitors through your professional story. Each section serves a specific purpose in building your personal brand and showcasing your capabilities.

Section	Purpose	Key Elements
Hero	Dramatic introduction	Parallax landscape, bold typography
About	Personal story	Text reveal, background text
Projects	Portfolio showcase	Alternating layout, 3D tilt
Tech Stack	Skills display	Grid layout, floating icons
Journey	Timeline of growth	Polaroid cards, vertical line
GitHub	Proof of consistency	Contribution graph, stats
Favorites	Personality showcase	Bento grid, hover effects

2.2 Parallax Scrolling Concept

Parallax scrolling creates depth by moving background elements at different speeds than foreground elements. This mimics real-world depth perception and creates an immersive experience.

Layer Speed Configuration:

- Background (sky): 0.1x scroll speed - slowest
- Far mountains: 0.3x scroll speed
- Near trees: 0.5x scroll speed
- UI elements: 1.0x scroll speed - fastest

3. Technology Stack

3.1 Core Technologies

Technology	Version	Purpose
Next.js	14+	React framework with App Router
React	18+	UI component library
TypeScript	5.x	Type-safe JavaScript
Tailwind CSS	3.x	Utility-first CSS framework

3.2 Animation Libraries

Library	Use Case
GSAP 3.x	Advanced scroll-based animations
ScrollTrigger	Trigger animations on scroll
Framer Motion	React-specific animations
Lenis	Smooth scrolling experience

4. Project Setup

4.1 Initialize Next.js Project

```
npx create-next-app@latest portfolio --typescript --tailwind --app cd portfolio
```

Select these options during setup:

- ✓ TypeScript
- ✓ ESLint
- ✓ Tailwind CSS
- ✓ App Router
- ✓ Import alias (@/*)

4.2 Install Dependencies

```
# Animation libraries npm install gsap @studio-freight/lenis framer-motion # UI components npm install @radix-ui/react-slot clsx tailwind-merge lucide-react # Utilities npm install react-intersection-observer react-use
```

4.3 Project Structure

```
portfolio-nextjs/ ├── app/ │ ├── layout.tsx │ ├── page.tsx │ └── ... ├── globals.css ├── components/ │ ├── sections/ │ │ ├── Hero.tsx │ │ ├── About.tsx │ │ ├── Projects.tsx │ │ └── ... │ └── animations/ │ ├── SmoothScroll.tsx │ └── ParallaxSection.tsx ├── ui/ ├── lib/ ├── utils.ts ├── constants.ts ├── public/ ├── images/ └── projects/
```

5. Hero Section Implementation

5.1 Concept Overview

The Hero section creates a dramatic first impression using parallax landscape imagery, bold typography, and smooth entrance animations. It's the visitor's first touchpoint and sets the tone for the entire portfolio.

5.2 Key Components

- Parallax background layers (mountains, trees)
- Large display typography with gradient effects
- Role indicator cards (Frontend, Backend, Available)
- Animated marquee text showing tech stack
- Smooth entrance animations using GSAP

5.3 Implementation Code

```
'use client' import { useEffect, useRef } from 'react' import Image from 'next/image' import { gsap } from 'gsap' export default function Hero() { const heroRef = useRef(null) useEffect(() => { const ctx = gsap.context(() => { gsap.from('.hero-title', { y: 100, opacity: 0, duration: 1, delay: 0.5, ease: 'power3.out', }) }, heroRef) return () => ctx.revert() }, []) return ( /* Parallax layers */ { /* Content */ } ) }
```

7. Projects Section

7.1 Layout Strategy

The Projects section uses an alternating zigzag layout where projects alternate between left and right alignment. This creates visual interest and prevents monotony while showcasing your work effectively.

7.2 Animation Features

- Sequential reveal - projects appear one by one as you scroll
- Directional entrance - odd projects slide from left, even from right
- 3D tilt effect on hover - images rotate based on mouse position
- Scale animation - projects grow slightly when hovered
- Tech tag display - badges showing technologies used

7.3 3D Tilt Implementation

```
document.querySelectorAll('.project-image').forEach(img => {  
  img.addEventListener('mousemove', (e) => { const rect =  
    img.getBoundingClientRect() const x = e.clientX - rect.left const y =  
    e.clientY - rect.top const centerX = rect.width / 2 const centerY =  
    rect.height / 2 const rotateX = ((y - centerY) / centerY) * 15 const rotateY  
    = ((centerX - x) / centerX) * 15 img.style.transform = `perspective(1000px)  
    rotateX(\${rotateX}deg) rotateY(\${rotateY}deg) scale(1.05) ` }) })
```

11. My Favorite Songs Section

11.1 Concept: Interactive Music Universe

This section transforms your music taste into an interactive galaxy where album covers float like stars in space. The center album is highlighted with a colorful glow, while other albums orbit around it. Clicking an album reveals lyrics with a beautiful overlay.

11.2 Visual Components

- Animated starfield background with twinkling stars
- Circular orbital layout with center focus album
- Album covers with colorful glow effects
- Lyrics overlay that appears on click
- Mini music player at the bottom
- Hover effects that scale and brighten albums

11.3 Starfield Background

The starfield creates an immersive cosmic atmosphere. Stars twinkle at different rates and opacities, creating a dynamic background that doesn't distract from the content.

```
const animate = () => { ctx.fillStyle = 'rgba(0, 0, 0, 0.05)' ctx.fillRect(0, 0, canvas.width, canvas.height) stars.forEach((star) => { star.opacity += star.twinkleSpeed if (star.opacity >= 1 || star.opacity <= 0) { star.twinkleSpeed = -star.twinkleSpeed } ctx.fillStyle = `rgba(255, 255, 255, ${star.opacity})` ctx.beginPath() ctx.arc(star.x, star.y, star.size, 0, Math.PI * 2) ctx.fill() }) requestAnimationFrame(animate) }
```

11.4 Circular Album Layout

Albums are positioned in a circular pattern using trigonometry. The `getAlbumPosition` function calculates x,y coordinates based on angle and radius, creating a perfect orbital layout.

```
const getAlbumPosition = (index, total, radius) => { const angle = (index / total) * Math.PI * 2 - Math.PI / 2 return { x: Math.cos(angle) * radius, y: Math.sin(angle) * radius, } } // Usage const position = getAlbumPosition(index, orbitAlbums.length, 400)
```

11.5 Album Hover Effects

When hovering over an album, it scales up and becomes fully opaque while other albums dim. This creates clear visual feedback and focuses attention on the hovered item.

```
setHoveredAlbum(album.id)} onMouseLeave={() => setHoveredAlbum(null)} > { /*  
Album cover */}
```

11.6 Lyrics Overlay

Clicking an album triggers a full-screen overlay with a dark backdrop and centered lyrics. Each line of lyrics fades in sequentially with a stagger effect, creating a reading rhythm.

```
{selectedAlbum.lyrics?.split('\n').map((line, i) => ( {line} ))}
```

11.7 Mini Music Player

A Spotify-inspired mini player sits at the bottom of the screen showing the current song with play/pause controls, skip buttons, and a volume slider. This adds a functional element that enhances the music theme.

11.8 Alternative Layout Options

A. Grid Layout (Simpler)

For a simpler implementation, use a standard grid layout with hover effects. This is easier to implement while still looking professional and allows for easier responsive design.

```
{albums.map((album) => ( {album.title} ))}
```

B. Carousel Layout

Use Swiper.js to create a 3D coverflow effect where albums slide horizontally with a perspective transform. This creates a mobile-friendly alternative to the circular layout.

11.9 Spotify API Integration (Optional)

For dynamic content, integrate the Spotify Web API to fetch your actual top tracks. This requires setting up a Spotify Developer account and implementing OAuth authentication.

Setup Steps:

- Create a Spotify Developer account at developer.spotify.com
- Create a new application and get Client ID & Secret
- Set up refresh token for server-side authentication
- Create API route in Next.js to fetch top tracks
- Display tracks dynamically in your component

```
// lib/spotify.ts export async function getTopTracks() { const { access_token
} = await getAccessToken() return
fetch('https://api.spotify.com/v1/me/top/tracks?limit=15', { headers: {
Authorization: `Bearer \${access_token}`, }, }) } // In component
useEffect(() => { fetch('/api/spotify/top-tracks') .then(res => res.json())
.then(data => setTracks(data.items)) }, [])
```

12. Animation Techniques

12.1 GSAP ScrollTrigger Pattern

ScrollTrigger is the core of scroll-based animations. It watches elements and triggers animations when they enter the viewport. The 'scrub' property ties animations directly to scroll position for smooth, controllable effects.

```
gsap.from(element, { scrollTrigger: { trigger: element, // Element to watch  
start: "top 80%", // Start when top hits 80% of viewport end: "top 30%", //  
End when top hits 30% of viewport scrub: 1, // Smooth scrubbing with 1 second  
lag markers: false, // Debug markers (disable in production) }, opacity: 0,  
y: 100, duration: 1 })
```

12.2 Staggered Animations

Stagger creates sequential animations with a delay between each element. This is perfect for animating lists, grids, or any group of elements where you want a cascading effect.

```
gsap.from('.tech-item', { opacity: 0, y: 50, stagger: { amount: 1.2, // Total  
duration for all staggers grid: [3, 4], // Grid dimensions from: 'start', //  
start/center/end/random }, scrollTrigger: { trigger: '.tech-grid', start:  
'top 70%', } })
```

13. Performance Optimization

13.1 Critical Optimizations

Technique	Implementation	Impact
will-change CSS	Add to animated elements	Prepares GPU rendering
Transform over position	Use translate() not top/left	GPU accelerated
Lazy loading	Next.js Image component	Faster initial load
Code splitting	Dynamic imports	Smaller bundles
Debounce scroll	Use requestAnimationFrame	Reduces calculations

13.2 Mobile Optimization

Disable or simplify heavy animations on mobile devices. Use CSS media queries and JavaScript matchMedia to detect device capabilities and adjust accordingly.

```
ScrollTrigger.matchMedia({ "(min-width: 768px)": function() { // Desktop:
full parallax gsap.to('.parallax', { y: -500, scrub: true }) }, "(max-width:
767px)": function() { // Mobile: simplified gsap.to('.parallax', { y: -200,
scrub: 2 }) } })
```

14. Deployment Guide

14.1 Vercel Deployment (Recommended)

Vercel is the recommended platform for Next.js deployment. It offers automatic deployments, preview URLs for pull requests, and excellent performance with edge caching.

Deployment Steps:

- Push your code to GitHub repository
- Sign up at vercel.com and connect GitHub
- Import your portfolio repository
- Configure build settings (usually auto-detected)
- Click Deploy and wait for build to complete
- Your site will be live at `yourproject.vercel.app`

14.2 Pre-Deployment Checklist

- ✓ Test all animations on desktop and mobile
- ✓ Run Lighthouse audit (aim for 90+ scores)
- ✓ Optimize all images (use WebP format)
- ✓ Test on different browsers (Chrome, Safari, Firefox)
- ✓ Verify all links and external resources
- ✓ Add meta tags for SEO and social sharing
- ✓ Set up custom domain (optional)
- ✓ Configure analytics (Google Analytics, Vercel Analytics)

Conclusion

Building a modern portfolio website is about more than just showcasing projects - it's about creating an experience that tells your story. By combining smooth animations, thoughtful interactions, and clean design, you create something memorable that stands out from traditional portfolio sites. Remember: start simple and iterate. Build the basic structure first, then gradually add animations and interactions. Test frequently on different devices and optimize for performance. Your portfolio is a living project that you can continuously improve as you learn new techniques. The technologies and patterns in this guide provide a solid foundation, but don't be afraid to experiment and add your own creative touches. Your portfolio should reflect your unique personality and style.

Key Takeaways:

- Parallax scrolling creates depth and immersion
- Scroll-triggered animations maintain engagement
- Interactive elements encourage exploration
- Performance optimization is crucial for user experience
- Mobile responsiveness is non-negotiable
- Next.js provides excellent developer experience and performance

Happy coding! ■