# Building Your Portfolio Website — Step-by-Step Guide

## What You're Building

A dark-themed developer portfolio inspired by Gazi Jarin and Brittany Chiang's sites, with an interactive particle silhouette hero section that responds to mouse movement. Built with modern tech that'll actually help your career.

**Tech Stack:**

- **Next.js 14** (React framework — the industry standard)
- **TypeScript** (typed JavaScript — makes code less error-prone)
- **Tailwind CSS** (utility-first CSS — fast styling without writing CSS files)
- **Framer Motion** (animations library)
- **Three.js / React Three Fiber** (3D graphics — for the interactive silhouette)
- **Vercel** (hosting — free, made by the Next.js team)

---

## Phase 0: Setting Up Your Machine

Before writing any code, you need some tools installed. Open your terminal (Command Prompt on Windows, Terminal on Mac/Linux).

### Step 0.1 — Install Node.js

Node.js lets you run JavaScript outside a browser. You need it for everything.

1. Go to https://nodejs.org
2. Download the **LTS** version (not "Current")
3. Run the installer, accept all defaults
4. Verify it worked by opening a terminal and running:

```bash
node --version    # Should show something like v20.x.x
npm --version     # Should show something like 10.x.x
```

### Step 0.2 — Install Git

Git tracks changes to your code and lets you push it to GitHub.

1. Go to https://git-scm.com/downloads
2. Download and install for your OS

3. Verify:

```bash
git --version    # Should show something like git version 2.x.x
```

4. Configure Git with your name and email (this labels your commits):

```bash
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

### Step 0.3 — Install a Code Editor

If you don't already have one, install **VS Code**: https://code.visualstudio.com

Recommended extensions to install (click the Extensions icon on the left sidebar):

- **ESLint** — catches code errors
- **Tailwind CSS IntelliSense** — autocompletes Tailwind classes
- **Prettier** — auto-formats your code
- **TypeScript Vue Plugin** is NOT needed, ignore it

### Step 0.4 — Create a GitHub Account

If you don't have one: https://github.com/signup

You'll push your code here and Vercel will deploy from it.

---

## Phase 1: Creating the Project

### Step 1.1 — Scaffold the Next.js App

Open your terminal, navigate to where you want the project (e.g. `cd ~/Projects`), and run:

```bash
npx create-next-app@latest portfolio
```

It will ask you a series of questions. Answer them like this:

```
✔ Would you like to use TypeScript? → Yes
✔ Would you like to use ESLint? → Yes
✔ Would you like to use Tailwind CSS? → Yes
✔ Would you like your code inside a `src/` directory? → Yes
✔ Would you like to use App Router? → Yes
✔ Would you like to use Turbopack for next dev? → Yes
✔ Would you like to customize the import alias? → No
```

Then:

```bash
cd portfolio
```

### Step 1.2 — Install Extra Dependencies

These are the libraries we'll use on top of what Next.js gives us:

```bash
npm install framer-motion @react-three/fiber @react-three/drei three
npm install -D @types/three
```

What each one does:

- `framer-motion` — makes elements animate (fade in, slide up, etc.)
- `@react-three/fiber` — lets you use Three.js inside React components
- `@react-three/drei` — helper utilities for React Three Fiber
- `three` — the 3D graphics library (needed by the above two)
- `@types/three` — TypeScript type definitions for Three.js

### Step 1.3 — Verify It Works

```bash
npm run dev
```

Open http://localhost:3000 in your browser. You should see the default Next.js page. Press `Ctrl+C` in the terminal to stop the server.

### Step 1.4 — Clean Up the Boilerplate

Next.js generates some default content we don't need. Open the project in VS Code:

```bash
```

```
code .
```

**Delete/clear these files:**

- `src/app/page.tsx` — delete ALL the content inside the `return()` and replace with a simple `<main>Hello</main>`
- `src/app/globals.css` — delete everything EXCEPT the first 3 lines (the `@tailwind` directives)

Your `globals.css` should look just this:

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

Your `page.tsx` should look like just this:

```tsx
export default function Home() {
  return (
    <main>
      Hello
    </main>
  );
}
```

Run `npm run dev` again to check it still works — you should see "Hello" on a blank page.

---

## Phase 2: Project Structure

### Step 2.1 — Create the Folder Structure

Inside `src/`, create these folders:

```
src/
├── app/
│   ├── layout.tsx         (already exists — the root layout)
│   ├── page.tsx           (already exists — the home page)
│   └── globals.css        (already exists — global styles)
├── components/
│   ├── Hero.tsx           (hero section with interactive silhouette)
│   ├── Navbar.tsx         (navigation bar)
│   ├── About.tsx          (about me section)
│   ├── Experience.tsx     (work experience section)
│   ├── Projects.tsx       (projects showcase)
│   ├── Contact.tsx        (contact section)
│   ├── Footer.tsx         (footer)
│   ├── SocialSidebar.tsx   (fixed social links on the side — like Brittany's site)
│   ├── EmailSidebar.tsx    (fixed email on the other side)
│   └── ParticleSilhouette.tsx  (the interactive 3D silhouette effect)
├── lib/
│   └── data.ts            (all your personal data — name, projects, experience, etc.)
└── hooks/
    └── useScrollDirection.ts  (custom hook for hiding/showing navbar on scroll)
```

You can create these manually in VS Code or run this in terminal from the project root:

```bash
mkdir -p src/components src/lib src/hooks
touch src/components/Hero.tsx
touch src/components/Navbar.tsx
touch src/components/About.tsx
touch src/components/Experience.tsx
touch src/components/Projects.tsx
touch src/components/Contact.tsx
touch src/components/Footer.tsx
touch src/components/SocialSidebar.tsx
touch src/components/EmailSidebar.tsx
touch src/components/ParticleSilhouette.tsx
touch src/lib/data.ts
touch src/hooks/useScrollDirection.ts
```

### Step 2.2 — Set Up the Data File

Open `src/lib/data.ts`. This is where ALL your personal content lives. Having it in one file means you only need to edit one place when you want to update your site.

```ts
```

```
export const siteConfig = {
  name: "Your Name",
  role: "Software Engineer",
  email: "hello@example.com",
  location: "London, UK",
  bio: [
    "I'm a software engineer who enjoys building things for the web.",
    "I recently graduated from [Your University] with a degree in Computer Science.",
    "When I'm not coding, you'll find me at the gym or exploring London.",
  ],
  socials: {
    github: "https://github.com/yourusername",
    linkedin: "https://linkedin.com/in/yourusername",
    twitter: "https://twitter.com/yourusername",
  },
  navLinks: [
    { name: "About", url: "#about" },
    { name: "Experience", url: "#experience" },
    { name: "Projects", url: "#projects" },
    { name: "Contact", url: "#contact" },
  ],
  experience: [
    {
      title: "Software Engineer",
      company: "Company Name",
      url: "https://company.com",
      range: "Jan 2024 - Present",
      points: [
        "Built and maintained internal tools using React and TypeScript.",
        "Worked on CI/CD pipelines and Kubernetes deployments.",
        "Collaborated with cross-functional teams on infrastructure projects.",
      ],
    },
    // Add more roles here
  ],
  projects: [
    {
      title: "Project One",
      description:
        "A brief description of what this project does and why it's interesting.",
      tech: ["React", "TypeScript", "Node.js"],
      github: "https://github.com/yourusername/project-one",
      live: "https://project-one.vercel.app",
    },
    {
      title: "Project Two",
      description:
        "Another project description. Keep these to 2-3 sentences max.",
      tech: ["Python", "Docker", "AWS"],
      github: "https://github.com/yourusername/project-two",
      live: "",
    },
    // Add more projects here
  ],
};
```

---

## Phase 3: Building Each Section

Now you build each component one at a time. The order below is the order I'd recommend — start with the layout/navigation, then work top-to-bottom through the page.

### Step 3.1 — Tailwind Config + Global Styles

Open `tailwind.config.ts` and add your custom colour palette:

```
ts
```

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

const config: Config = {
  content: [
    "./src/pages/**/*.{js,ts,jsx,tsx,mdx}",
    "./src/components/**/*.{js,ts,jsx,tsx,mdx}",
    "./src/app/**/*.{js,ts,jsx,tsx,mdx}",
  ],
  theme: {
    extend: {
      colors: {
        navy: {
          DEFAULT: "#0a192f",
          light: "#112240",
          lightest: "#233554",
        },
        slate: {
          DEFAULT: "#8892b0",
          light: "#a8b2d1",
          lightest: "#ccd6f6",
        },
        accent: "#64ffda",      // The green accent — change this to whatever you want
        white: "#e6f1ff",
      },
      fontFamily: {
        mono: ["SF Mono", "Fira Code", "Fira Mono", "Roboto Mono", "monospace"],
        sans: ["Calibre", "Inter", "San Francisco", "SF Pro Text", "system-ui", "sans-serif"],
      },
    },
  },
  plugins: [],
};

export default config;
```

Update `globals.css`:

```
css
```

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

@layer base {
  body {
    @apply bg-navy text-slate antialiased;
  }

  ::selection {
    @apply bg-accent/30 text-slate-lightest;
  }

  /* Smooth scrolling */
  html {
    scroll-behavior: smooth;
  }

  /* Custom scrollbar */
  ::-webkit-scrollbar {
    width: 12px;
  }
  ::-webkit-scrollbar-track {
    @apply bg-navy;
  }
  ::-webkit-scrollbar-thumb {
    @apply bg-navy-lightest rounded-full border-[3px] border-navy;
  }
}

/* Numbered headings for sections */
.numbered-heading {
  @apply flex items-center text-slate-lightest text-2xl font-semibold;
}
.numbered-heading::before {
  @apply text-accent font-mono text-xl mr-2;
  counter-increment: section;
  content: "0" counter(section) ".";
}
.numbered-heading::after {
  @apply block h-px w-full bg-navy-lightest ml-4;
  content: "";
}
```

### Step 3.2 — Root Layout

Open `src/app/layout.tsx`:

```tsx
import type { Metadata } from "next";
import "./globals.css";

export const metadata: Metadata = {
  title: "Your Name | Software Engineer",
  description: "Your Name is a software engineer based in London.",
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <body>
        {children}
      </body>
    </html>
  );
}
```

### Step 3.3 — Build Components (One by One)

From here, you build each component file. I won't write out every single component in this guide (that would be thousands of lines and you'd learn nothing by copy-pasting), but here's what each one should contain and the approach:

**Navbar (`Navbar.tsx`):**

- Fixed to top of page
- Transparent background that gets a subtle blur/background when you scroll down
- Your logo/name on the left
- Navigation links on the right (About, Experience, Projects, Contact)
- A "Resume" button with an accent-coloured border
- On mobile: a hamburger menu that opens a full-screen overlay
- Use `framer-motion` for the staggered fade-in of nav items on page load

**Hero (`Hero.tsx`):**

- Full viewport height section
- Left side: your intro text — "Hi, my name is", your name (big), your tagline, a short blurb, and a CTA button
- Right side (or background): the `ParticleSilhouette` component
- Use `framer-motion` for staggered text animations on load

**ParticleSilhouette (`ParticleSilhouette.tsx`):**

- This is the showpiece. Uses `@react-three/fiber` to create a 3D canvas
- Loads a silhouette image, samples pixels from it, and creates particles at those positions
- Particles react to mouse position (push away from cursor, then drift back)
- This is the most complex component — we'll build this together in detail

**About (`About.tsx`):**

- Section with id="about"
- Numbered heading: "About Me"
- Two columns: text on left, photo on right (with a coloured border offset effect like Brittany's site)
- List your technologies/skills

**Experience (`Experience.tsx`):**

- Section with id="experience"
- Tabbed interface: company names on the left, job details on the right
- Click a company tab to show that role's details
- Use `framer-motion` for tab transitions

**Projects (`Projects.tsx`):**

- Section with id="projects"
- Featured projects as large cards with image, description, tech tags
- Other projects as a grid of smaller cards
- GitHub and external link icons on each card

**Contact (`Contact.tsx`):**

- Section with id="contact"
- Centred text: "Get In Touch"
- A short message and a mailto: button

**SocialSidebar & EmailSidebar:**

- Fixed position, bottom-left and bottom-right
- Vertical line extending down with icons (GitHub, LinkedIn, Twitter) or your email
- These fade in on page load

**Footer (`Footer.tsx`):**

- Simple centred text: "Designed & Built by Your Name"
- Link to your GitHub repo

### Step 3.4 — Assemble in page.tsx

Once your components are built, your `src/app/page.tsx` ties them all together:

```tsx
import Navbar from "@/components/Navbar";
import Hero from "@/components/Hero";
import About from "@/components/About";
import Experience from "@/components/Experience";
import Projects from "@/components/Projects";
import Contact from "@/components/Contact";
import Footer from "@/components/Footer";
import SocialSidebar from "@/components/SocialSidebar";
import EmailSidebar from "@/components/EmailSidebar";

export default function Home() {
  return (
    <>
      <Navbar />
      <SocialSidebar />
      <EmailSidebar />
      <main className="mx-auto max-w-screen-xl px-6 md:px-12 lg:px-24">
        <Hero />
        <About />
        <Experience />
        <Projects />
        <Contact />
      </main>
      <Footer />
    </>
  );
}
```

---

## Phase 4: The Interactive Silhouette (The Hard Part)

This is the standout feature. Here's how it works conceptually:

1. **Load a silhouette image** (a black-on-white or white-on-transparent PNG of a person's outline)
2. **Sample the dark pixels** from that image using a hidden HTML canvas
3. **Create 3D particles** at each sampled pixel position using Three.js
4. **On every frame**, check the mouse position and push nearby particles away from the cursor
5. **Particles drift back** to their original position over time (like elastic)

This component uses `"use client"` at the top (because it needs browser APIs).

**Key concepts you'll learn:**

- HTML Canvas pixel manipulation (`getImageData`)
- Three.js `BufferGeometry` and `Points`
- The `useFrame` hook from React Three Fiber (runs code every animation frame)
- Raycasting / mouse position mapping in 3D space

I'll build the full implementation of this with you when you're ready for it — it's roughly 150-200 lines of code and is the most educational part of the whole project.

---

### Phase 5: Making It Responsive

Once everything works on desktop:

1. **Test at different widths** — use Chrome DevTools (F12 → toggle device toolbar)
2. **Tailwind breakpoints** — use `sm:`, `md:`, `lg:` prefixes for responsive styles
3. **Mobile nav** — hamburger menu that opens a slide-out panel
4. **Silhouette** — either scale it down or hide it on mobile (3D is heavy on phones)
5. **Sidebars** — hide on mobile, show social links in the footer instead

---

### Phase 6: Polish & Animations

#### Scroll Reveal Animations

Wrap each section in a Framer Motion component that fades in when scrolled into view:

```tsx
```

```
<motion.section
  initial={{ opacity: 0, y: 20 }}
  whileInView={{ opacity: 1, y: 0 }}
  transition={{ duration: 0.5 }}
  viewport={{ once: true }}
>
  {/* section content */}
</motion.section>
```

### Page Load Animation

Add a brief loading screen (like Brittany's logo animation), then stagger-reveal the navbar, sidebars, and hero content.

### Hover Effects

- Links: accent colour underline that slides in from left
- Project cards: slight lift with box shadow
- Buttons: background fill animation on hover

---

## Phase 7: Deployment

### Step 7.1 — Push to GitHub

```bash
git init                          # (already done by create-next-app)
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/yourusername/portfolio.git
git push -u origin main
```

(Create the repo on GitHub first: github.com/new — do NOT add a README, .gitignore, or licence when creating it since your project already has these.)

### Step 7.2 — Deploy on Vercel

1. Go to https://vercel.com and sign in with your GitHub account
2. Click "New Project"
3. Import your `portfolio` repo
4. Vercel auto-detects it's a Next.js project — just click "Deploy"
5. Wait ~60 seconds. Your site is now live at `portfolio-yourusername.vercel.app`

**Step 7.3 — Custom Domain (Optional, Later)**

1. Buy a domain (Namecheap, Cloudflare, Google Domains)
2. In Vercel dashboard → your project → Settings → Domains
3. Add your domain, follow the DNS instructions Vercel gives you
4. HTTPS is automatic

---

## Phase 8: What to Build First (Priority Order)

Don't try to build everything at once. Here's the order:

1. **Tailwind config + global styles** (get the colours and fonts right)
2. **Layout + Navbar** (structural skeleton)
3. **Hero section** (just the text, no silhouette yet)
4. **About section**
5. **Experience section**
6. **Projects section**
7. **Contact + Footer**
8. **Sidebars**
9. **Scroll animations** (Framer Motion)
10. **ParticleSilhouette** (save the hardest for last — your site works fine without it)
11. **Mobile responsiveness**
12. **Loading animation**
13. **Deploy**

---

## Useful Commands Reference

```bash
npm run dev        # Start development server (http://localhost:3000)
npm run build      # Build for production (checks for errors)
npm run lint       # Check for code issues
```

---

## Resources to Learn Along the Way

- **Next.js docs**: https://nextjs.org/docs (start with "Getting Started")

- **Tailwind CSS docs**: https://tailwindcss.com/docs (search for any class)
- **Framer Motion docs**: https://www.framer.com/motion/
- **React Three Fiber docs**: https://r3f.docs.pmnd.rs/
- **TypeScript basics**: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

---

## When You Get Stuck

Work through it in this order:

1. Read the error message carefully — it usually tells you exactly what's wrong
2. Search the error message on Google/Stack Overflow
3. Check the docs for whatever library is causing the issue
4. Ask me — paste the error message and the code that's causing it