Portfolio Design System - Vite Setup Guide

📁 File Structure

```
src/
├── styles/
│   ├── globals.css          # Main design system & variables
│   ├── components.css       # Reusable component styles
│   ├── layout.css           # Layout & navigation styles
│   └── terminal.css         # Terminal component styles
├── components/
│   ├── Hero.jsx
│   ├── Skills.jsx
│   ├── Projects.jsx
│   ├── Experience.jsx
│   ├── Contact.jsx
│   ├── Navigation.jsx
│   └── Terminal.jsx
├── hooks/
│   └── useScrollAnimations.js
├── utils/
│   └── animations.js
├── App.jsx
└── main.jsx
```

🚀 Quick Setup

1. Install Dependencies

bash

```bash
# Create Vite project
npm create vite@latest portfolio -- --template react
cd portfolio

# Install additional dependencies (optional)
npm install lucide-react  # For icons
npm install framer-motion # For advanced animations (optional)
```

2. Import Styles in main.jsx

javascript

```jsx
// src/main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'

// Import all styles
import './styles/globals.css'
import './styles/components.css'
import './styles/layout.css'
import './styles/terminal.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

3. Basic App Structure

javascript

```jsx
// src/App.jsx
import { useEffect } from 'react'
import Navigation from './components/Navigation'
import Hero from './components/Hero'
import Skills from './components/Skills'
import Projects from './components/Projects'
import Experience from './components/Experience'
import Contact from './components/Contact'
import Terminal from './components/Terminal'
import { initScrollAnimations, updateScrollProgress } from './utils/animations'

function App() {
  useEffect(() => {
    // Initialize scroll animations
    initScrollAnimations()

    // Setup scroll progress
    const handleScroll = () => updateScrollProgress()
    window.addEventListener('scroll', handleScroll)

    return () => window.removeEventListener('scroll', handleScroll)
  }, [])

  return (
    <div className="App">
      <div className="scroll-progress" id="scrollProgress"></div>
      <Navigation />
      <Hero />
      <Skills />
      <Projects />
      <Experience />
      <Contact />
      <Terminal />
    </div>
  )
}

export default App
```

🎨 Component Examples

Navigation Component

```javascript
```

```jsx
// src/components/Navigation.jsx
import { useState } from 'react'

const Navigation = () => {
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false)

  return (
    <nav className="navbar">
      <div className="container">
        <div className="nav-container">
          <div className="logo">shahid.dev</div>
          <ul className="nav-links">
            <li><a href="#home">Home</a></li>
            <li><a href="#skills">Skills</a></li>
            <li><a href="#projects">Projects</a></li>
            <li><a href="#experience">Experience</a></li>
            <li><a href="#contact">Contact</a></li>
          </ul>
          <button
            className="mobile-menu-btn"
            onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
          >
            {mobileMenuOpen ? '✕' : '☰'}
          </button>
        </div>
      </div>
      <div className={`mobile-menu ${mobileMenuOpen ? 'open' : ''}`}>
        <ul>
          <li><a href="#home" onClick={() => setMobileMenuOpen(false)}>Home</a></li>
          <li><a href="#skills" onClick={() => setMobileMenuOpen(false)}>Skills</a></li>
          <li><a href="#projects" onClick={() => setMobileMenuOpen(false)}>Projects</a></li>
          <li><a href="#experience" onClick={() => setMobileMenuOpen(false)}>Experience</a></li>
          <li><a href="#contact" onClick={() => setMobileMenuOpen(false)}>Contact</a></li>
        </ul>
      </div>
    </nav>
  )
}

export default Navigation
```

Hero Component

javascript

```jsx
// src/components/Hero.jsx
const Hero = () => {
  return (
    <section className="hero" id="home">
      <div className="container">
        <div className="hero-content">
          <div className="hero-text">
            <h1>Hi — I'm <span className="gradient-text">Shahid Parvez</span></h1>
            <p className="hero-subtitle">
              Support Engineer & React-focused Web Developer. I build and support
              AI-first products — from customer journeys to full-stack React apps.
            </p>
            <div className="hero-buttons">
              <a href="#projects" className="btn btn-primary">See work</a>
              <a
                href="https://github.com/mrsnailo"
                target="_blank"
                rel="noopener"
                className="btn btn-secondary"
              >
                View GitHub
              </a>
            </div>
          </div>
          <div className="hero-image">
            <div className="profile-container">
              <div className="profile-glow"></div>
              <div className="profile-border"></div>
              <img
                src="/profile-image.jpg"
                alt="Shahid Parvez"
                className="profile-image"
              />
              <div className="tech-orbit">⚛️</div>
              <div className="tech-orbit">💻</div>
              <div className="tech-orbit">🚀</div>
              <div className="tech-orbit">🤖</div>
              <div className="status-indicator">
                <div className="status-dot"></div>
                <span>Available for hire</span>
              </div>
            </div>
          </div>
```

```
        </div>
      </div>
    </section>
  )
}

export default Hero
```

🔧 Animation Utilities

javascript

```
// src/utils/animations.js
export const initScrollAnimations = () => {
  const observerOptions = {
    threshold: 0.1,
    rootMargin: '0px 0px -50px 0px'
  }

  const observer = new IntersectionObserver((entries) => {
    entries.forEach(entry => {
      if (entry.isIntersecting) {
        entry.target.classList.add('animate')

        // Handle skill bars
        if (entry.target.classList.contains('skill-category')) {
          animateSkillBars(entry.target)
        }

        // Handle counters
        if (entry.target.classList.contains('stat-item')) {
          animateCounter(entry.target.querySelector('.counter'))
        }
      }
    })
  }, observerOptions)

  // Observe all animated elements
  const elements = document.querySelectorAll(
    '.fade-in, .slide-in-left, .slide-in-right, .scale-in, .rotate-in, ' +
    '.skill-category, .stat-item, .timeline-item, .magnetic-card'
  )

  elements.forEach(el => observer.observe(el))
}

export const animateSkillBars = (container) => {
  const progressBars = container.querySelectorAll('.skill-progress')
  progressBars.forEach((bar, index) => {
    setTimeout(() => {
      const width = bar.getAttribute('data-width')
      bar.style.setProperty('--target-width', width + '%')
      bar.style.width = width + '%'
      bar.classList.add('animate')
    }, index * 200)
```

```
  })
}

export const animateCounter = (counter) => {
  if (!counter) return
  const target = parseInt(counter.getAttribute('data-target'))
  const increment = target / 60
  let current = 0

  const updateCounter = () => {
    if (current < target) {
      current += increment
      counter.textContent = Math.ceil(current)
      requestAnimationFrame(updateCounter)
    } else {
      counter.textContent = target
    }
  }

  updateCounter()
}

export const updateScrollProgress = () => {
  const scrollTop = window.scrollY
  const docHeight = document.documentElement.scrollHeight - window.innerHeight
  const scrollPercent = (scrollTop / docHeight) * 100
  const progressBar = document.getElementById('scrollProgress')
  if (progressBar) {
    progressBar.style.width = scrollPercent + '%'
  }
}
```

🎯 Key Features

✅ Design Tokens

- Centralized CSS variables for colors, spacing, typography
- Consistent design system across all components
- Easy theming and customization

✅ Responsive Design

- Mobile-first approach

- Flexible grid systems
- Touch-friendly interactions

✅ Smooth Animations

- Scroll-triggered animations
- Intersection Observer API
- Hardware-accelerated transforms

✅ Performance Optimized

- CSS-only animations where possible
- Efficient scroll handlers
- Minimal JavaScript footprint

✅ Accessibility Ready

- Semantic HTML structure
- ARIA labels and roles
- Keyboard navigation support
- Focus management

🚀 Deployment

```bash
# Build for production
npm run build

# Preview build locally
npm run preview

# Deploy to Vercel/Netlify
# Just connect your GitHub repo!
```

🎨 Customization Tips

1. **Colors**: Modify CSS variables in `globals.css`
2. **Animations**: Adjust timing in keyframes
3. **Layout**: Change grid configurations in `layout.css`

4. **Typography**: Update font stacks and scales
5. **Components**: Mix and match as needed

Your portfolio is now production-ready! 🚀