Capstone Build Book: ShipForge End-to-End

> *Step-by-step rebuild guide from a blank folder to a production-deployed SvelteKit application. Every command, every file, every verification step.*

Table of Contents

Phase 1: Project Setup

Step 1.1: Create the Project

Command:

```
pnpm dlx sv create shipforge
```

Prompts:

- Template: SvelteKit minimal
- Type checking: Yes, using TypeScript (or JSDoc if preferred)
- Add-ons: Prettier, ESLint, Playwright

Verification:

```
cd shipforge
ls -la
# Expected: package.json, svelte.config.js, vite.config.js, src/, static/, etc.
```

Step 1.2: Install Dependencies

Command:

```
pnpm install
```

Verification:

```
ls node_modules/.pnpm | head -5
# Expected: populated node_modules
```

Step 1.3: Install Additional Dependencies

Command:

```
# Animation
pnpm add gsap

# SEO/Meta
pnpm add -D @sveltejs/adapter-vercel
```

Verification:

```
pnpm list --depth 0
# Expected: gsap and adapter listed
```

Step 1.4: Configure Adapter

File: `svelte.config.js`

```
import adapter from '@sveltejs/adapter-vercel';
import { vitePreprocess } from '@sveltejs/vite-plugin-svelte';

const config = {
  preprocess: vitePreprocess(),
  kit: {
    adapter: adapter({
      runtime: 'nodejs22.x'
    }),
    alias: {
      $components: 'src/lib/components',
      $animations: 'src/lib/animations',
      $utils: 'src/lib/utils',
      $styles: 'src/lib/styles'
    }
  }
};

export default config;
```

Verification:

```
pnpm check
# Expected: No errors
```

Step 1.5: Create Directory Structure

Command:

```
mkdir -p src/lib/components/ui
mkdir -p src/lib/components/sections
mkdir -p src/lib/components/layout
mkdir -p src/lib/animations
mkdir -p src/lib/actions
mkdir -p src/lib/utils
mkdir -p src/lib/stores
mkdir -p src/lib/styles
mkdir -p src/lib/data
mkdir -p static/fonts
mkdir -p static/images
mkdir -p static/icons
```

Verification:

```
find src/lib -type d
# Expected: All directories listed
```

Step 1.6: Git Initial Commit

Command:

```
git init -b main
git add -A
git commit -m "chore: initial SvelteKit project setup"
```

Verification:

```
git log --oneline
# Expected: One commit
```

Step 1.7: Start Dev Server

Command:

```
pnpm dev
```

Verification:

- Open `http://localhost:5173`
- See the default SvelteKit welcome page
- Terminal shows no errors

Phase 2: Design System

Step 2.1: Define Design Tokens

File: `src/lib/styles/tokens.css`

```
:root {
  /* Colors */
  --color-primary: #6366f1;
  --color-primary-light: #818cf8;
  --color-primary-dark: #4f46e5;
  --color-secondary: #ec4899;
  --color-accent: #f59e0b;

  --color-gray-50: #f9fafb;
  --color-gray-100: #f3f4f6;
  --color-gray-200: #e5e7eb;
  --color-gray-300: #d1d5db;
  --color-gray-400: #9ca3af;
  --color-gray-500: #6b7280;
  --color-gray-600: #4b5563;
```

```css
--color-gray-700: #374151;
--color-gray-800: #1f2937;
--color-gray-900: #111827;
--color-gray-950: #030712;

--color-success: #10b981;
--color-warning: #f59e0b;
--color-error: #ef4444;

--color-bg: #ffffff;
--color-bg-secondary: var(--color-gray-50);
--color-text: var(--color-gray-900);
--color-text-secondary: var(--color-gray-600);
--color-border: var(--color-gray-200);

/* Typography */
--font-sans: 'Inter', system-ui, -apple-system, sans-serif;
--font-mono: 'JetBrains Mono', 'Fira Code', monospace;

--text-xs: 0.75rem;
--text-sm: 0.875rem;
--text-base: 1rem;
--text-lg: 1.125rem;
--text-xl: 1.25rem;
--text-2xl: 1.5rem;
--text-3xl: 1.875rem;
--text-4xl: 2.25rem;
--text-5xl: 3rem;
--text-6xl: 3.75rem;

--leading-tight: 1.1;
--leading-snug: 1.25;
--leading-normal: 1.5;
--leading-relaxed: 1.75;

--tracking-tight: -0.025em;
--tracking-normal: 0;
--tracking-wide: 0.025em;

/* Spacing */
--space-1: 0.25rem;
--space-2: 0.5rem;
--space-3: 0.75rem;
--space-4: 1rem;
--space-6: 1.5rem;
--space-8: 2rem;
--space-10: 2.5rem;
--space-12: 3rem;
--space-16: 4rem;
--space-20: 5rem;
--space-24: 6rem;
--space-32: 8rem;

/* Layout */
--max-width: 1200px;
--max-width-narrow: 800px;
```

```css
  --max-width-wide: 1400px;

  /* Border Radius */
  --radius-sm: 0.375rem;
  --radius-md: 0.5rem;
  --radius-lg: 0.75rem;
  --radius-xl: 1rem;
  --radius-2xl: 1.5rem;
  --radius-full: 9999px;

  /* Shadows */
  --shadow-sm: 0 1px 2px rgba(0, 0, 0, 0.05);
  --shadow-md: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
  --shadow-lg: 0 10px 15px -3px rgba(0, 0, 0, 0.1);
  --shadow-xl: 0 20px 25px -5px rgba(0, 0, 0, 0.1);

  /* Transitions */
  --transition-fast: 150ms ease;
  --transition-base: 200ms ease;
  --transition-slow: 300ms ease;

  /* Z-Index */
  --z-dropdown: 50;
  --z-sticky: 100;
  --z-overlay: 200;
  --z-modal: 300;
  --z-toast: 400;
}
```

Verification:

- File created at correct path
- CSS variables follow consistent naming convention

Step 2.2: Global Reset and Base Styles

File: `src/lib/styles/global.css`

```css
@import './tokens.css';

/* Reset */
*,
*::before,
*::after {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

html {
  scroll-behavior: smooth;
  -webkit-text-size-adjust: 100%;
  text-size-adjust: 100%;
}

body {
  font-family: var(--font-sans);
  font-size: var(--text-base);
  line-height: var(--leading-normal);
  color: var(--color-text);
  background-color: var(--color-bg);
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

img,
picture,
video,
canvas,
svg {
  display: block;
  max-width: 100%;
}

input,
button,
textarea,
select {
  font: inherit;
  color: inherit;
}

p,
h1,
h2,
h3,
h4,
h5,
h6 {
  overflow-wrap: break-word;
}

h1,
h2,
h3,
h4 {
  line-height: var(--leading-tight);
  letter-spacing: var(--tracking-tight);
}

a {
  color: var(--color-primary);
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

/* Utility Classes */
.container {
  width: 100%;
  max-width: var(--max-width);
  margin: 0 auto;
  padding: 0 var(--space-4);
}

@media (min-width: 640px) {
  .container {
    padding: 0 var(--space-6);
  }
}

@media (min-width: 1024px) {
  .container {
    padding: 0 var(--space-8);
  }
}

.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  white-space: nowrap;
  border-width: 0;
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  *,
  *::before,
  *::after {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
    scroll-behavior: auto !important;
  }
}
```

Step 2.3: Import Styles in Layout

File: `src/routes/+layout.svelte`

```
<script>
  import '$lib/styles/global.css';

  let { children } = $props();
</script>

{@render children()}
```

Step 2.4: Add Fonts

File: `src/app.html`

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" href="%sveltekit.assets%/favicon.png" />
    <link
      rel="preload"
      href="/fonts/inter-var.woff2"
      as="font"
      type="font/woff2"
      crossorigin="anonymous"
    />
    %sveltekit.head%
  </head>
  <body data-sveltekit-preload-data="hover">
    <div style="display: contents">%sveltekit.body%</div>
  </body>
</html>
```

Verification:

```
pnpm dev
# Open browser — text should render with the design system styles
# No console errors
```

Commit:

```
git add -A
git commit -m "feat: add design system with tokens, global styles, and fonts"
```

Phase 3: Core Components

Step 3.1: Button Component

File: `src/lib/components/ui/Button.svelte`

```
<script>
  let {
    variant = 'primary',
    size = 'md',
    href = '',
    disabled = false,
    class: className = '',
    children,
    onclick,
    ...restProps
  } = $props();

  const Tag = href ? 'a' : 'button';
</script>

{#if href}
```

```
  <a
    {href}
    class="button button--{variant} button--{size} {className}"
    class:button--disabled={disabled}
    {...restProps}
  >
    {@render children()}
  </a>
{:else}
  <button
    class="button button--{variant} button--{size} {className}"
    {disabled}
    {onclick}
    {...restProps}
  >
    {@render children()}
  </button>
{/if}

<style>
  .button {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    gap: var(--space-2);
    font-weight: 600;
    border: none;
    border-radius: var(--radius-lg);
    cursor: pointer;
    transition: all var(--transition-fast);
    text-decoration: none;
    white-space: nowrap;
  }

  .button:hover {
    text-decoration: none;
    transform: translateY(-1px);
  }

  .button:active {
    transform: translateY(0);
  }

  /* Variants */
  .button--primary {
    background: var(--color-primary);
    color: white;
    box-shadow: 0 2px 8px rgba(99, 102, 241, 0.3);
  }

  .button--primary:hover {
    background: var(--color-primary-dark);
    box-shadow: 0 4px 12px rgba(99, 102, 241, 0.4);
  }

  .button--secondary {
    background: transparent;
    color: var(--color-text);
    border: 2px solid var(--color-border);
  }

  .button--secondary:hover {
    border-color: var(--color-primary);
    color: var(--color-primary);
  }

  .button--ghost {
    background: transparent;
    color: var(--color-text-secondary);
  }

  .button--ghost:hover {
    background: var(--color-gray-100);
    color: var(--color-text);
  }

  /* Sizes */
  .button--sm {
    padding: var(--space-2) var(--space-4);
    font-size: var(--text-sm);
  }

  .button--md {
    padding: var(--space-3) var(--space-6);
    font-size: var(--text-base);
  }

  .button--lg {
    padding: var(--space-4) var(--space-8);
    font-size: var(--text-lg);
  }

  /* Disabled */
  .button--disabled,
  .button:disabled {
    opacity: 0.5;
    cursor: not-allowed;
    transform: none;
  }
</style>
```

Verification:

```
<!-- Quick test in src/routes/+page.svelte -->
<script>
  import Button from '$lib/components/ui/Button.svelte';
</script>

<Button>Primary</Button>
<Button variant="secondary">Secondary</Button>
```

```
<Button variant="ghost" size="sm">Ghost Small</Button>
<Button href="/about">Link Button</Button>
```

Step 3.2: Section Header Component

File: `src/lib/components/ui/SectionHeader.svelte`

```
<script>
  let {
    badge = '',
    title = '',
    description = '',
    align = 'center'
  } = $props();
</script>

<div class="section-header section-header--{align}">
  {#if badge}
    <span class="section-badge">{badge}</span>
  {/if}
  {#if title}
    <h2 class="section-title">{title}</h2>
  {/if}
  {#if description}
    <p class="section-description">{description}</p>
  {/if}
</div>

<style>
  .section-header {
    margin-bottom: var(--space-12);
  }

  .section-header--center {
    text-align: center;
  }

  .section-header--left {
    text-align: left;
  }

  .section-badge {
    display: inline-block;
    padding: var(--space-1) var(--space-4);
    background: rgba(99, 102, 241, 0.1);
    color: var(--color-primary);
    border-radius: var(--radius-full);
    font-size: var(--text-sm);
    font-weight: 600;
    text-transform: uppercase;
    letter-spacing: var(--tracking-wide);
    margin-bottom: var(--space-4);
  }

  .section-title {
    font-size: var(--text-4xl);
    font-weight: 800;
    color: var(--color-text);
    margin-bottom: var(--space-4);
  }

  .section-description {
    font-size: var(--text-lg);
    color: var(--color-text-secondary);
    max-width: 600px;
    line-height: var(--leading-relaxed);
  }

  .section-header--center .section-description {
    margin: 0 auto;
  }

  @media (min-width: 1024px) {
    .section-title {
      font-size: var(--text-5xl);
    }
  }
</style>
```

Step 3.3: Card Component

File: `src/lib/components/ui/Card.svelte`

```
<script>
  let {
    href = '',
    class: className = '',
    children,
    ...restProps
  } = $props();
</script>

{#if href}
  <a {href} class="card {className}" {...restProps}>
    {@render children()}
  </a>
{:else}
  <div class="card {className}" {...restProps}>
    {@render children()}
  </div>
{/if}

<style>
  .card {
    background: var(--color-bg);
    border: 1px solid var(--color-border);
    border-radius: var(--radius-xl);
    padding: var(--space-8);
    transition: all var(--transition-base);
    text-decoration: none;
    color: inherit;
    display: block;
  }

  a.card:hover {
    text-decoration: none;
    border-color: var(--color-primary);
    box-shadow: var(--shadow-lg);
    transform: translateY(-4px);
  }
</style>
```

Step 3.4: Navigation Component

File: `src/lib/components/layout/Navigation.svelte`

```
<script>
  import { page } from '$app/stores';
  import Button from '$lib/components/ui/Button.svelte';

  const navLinks = [
    { href: '/', label: 'Home' },
    { href: '/features', label: 'Features' },
    { href: '/pricing', label: 'Pricing' },
    { href: '/about', label: 'About' },
    { href: '/blog', label: 'Blog' }
  ];

  let mobileOpen = $state(false);
</script>

<header class="header">
  <nav class="nav container" aria-label="Main navigation">
    <a href="/" class="logo" aria-label="ShipForge Home">
      <strong>ShipForge</strong>
    </a>

    <ul class="nav-links" class:nav-links--open={mobileOpen}>
      {#each navLinks as link}
        <li>
          <a
            href={link.href}
            class="nav-link"
            class:nav-link--active={$page.url.pathname === link.href}
            aria-current={$page.url.pathname === link.href ? 'page' : undefined}
          >
            {link.label}
          </a>
        </li>
      {/each}
    </ul>

    <div class="nav-actions">
      <Button href="/contact" size="sm">Get Started</Button>
    </div>

    <button
      class="mobile-toggle"
      aria-label="Toggle navigation"
      aria-expanded={mobileOpen}
      onclick={() => mobileOpen = !mobileOpen}
    >
      <span class="hamburger" class:hamburger--open={mobileOpen}></span>
    </button>
  </nav>
</header>

<style>
  .header {
    position: sticky;
    top: 0;
    z-index: var(--z-sticky);
    background: rgba(255, 255, 255, 0.9);
    backdrop-filter: blur(12px);
    border-bottom: 1px solid var(--color-border);
  }

  .nav {
    display: flex;
    align-items: center;
    justify-content: space-between;
    height: 64px;
  }

  .logo {
    font-size: var(--text-xl);
    color: var(--color-text);
    text-decoration: none;
  }

  .nav-links {
    display: none;
    list-style: none;
    gap: var(--space-1);
  }

  .nav-link {
    padding: var(--space-2) var(--space-3);
    color: var(--color-text-secondary);
    text-decoration: none;
    border-radius: var(--radius-md);
    font-size: var(--text-sm);
    font-weight: 500;
    transition: all var(--transition-fast);
  }

  .nav-link:hover {
    color: var(--color-text);
    background: var(--color-gray-100);
    text-decoration: none;
  }

  .nav-link--active {
    color: var(--color-primary);
    background: rgba(99, 102, 241, 0.1);
  }

  .nav-actions {
    display: none;
  }

  .mobile-toggle {
    display: flex;
    align-items: center;
    justify-content: center;
    width: 40px;
    height: 40px;
    background: none;
    border: none;
    cursor: pointer;
  }

  .hamburger {
    width: 20px;
    height: 2px;
    background: var(--color-text);
    position: relative;
    transition: all var(--transition-fast);
  }

  .hamburger::before,
  .hamburger::after {
    content: '';
    position: absolute;
    width: 100%;
    height: 2px;
    background: var(--color-text);
    transition: all var(--transition-fast);
  }

  .hamburger::before { top: -6px; }
  .hamburger::after { bottom: -6px; }

  .hamburger--open {
    background: transparent;
  }
  .hamburger--open::before {
    top: 0;
    transform: rotate(45deg);
  }
  .hamburger--open::after {
    bottom: 0;
    transform: rotate(-45deg);
  }

  @media (min-width: 768px) {
    .nav-links {
      display: flex;
    }
    .nav-actions {
      display: flex;
    }
    .mobile-toggle {
      display: none;
    }
  }

  @media (max-width: 767px) {
    .nav-links--open {
      display: flex;
      flex-direction: column;
      position: absolute;
      top: 64px;
      left: 0;
      right: 0;
      background: var(--color-bg);
      border-bottom: 1px solid var(--color-border);
      padding: var(--space-4);
    }
  }
</style>
```

Step 3.5: Footer Component

File: `src/lib/components/layout/Footer.svelte`

```
<script>
  const year = new Date().getFullYear();

  const footerSections = [
    {
      title: 'Product',
      links: [
        { href: '/features', label: 'Features' },
        { href: '/pricing', label: 'Pricing' },
        { href: '/changelog', label: 'Changelog' }
      ]
    },
    {
      title: 'Resources',
      links: [
        { href: '/blog', label: 'Blog' },
        { href: '/docs', label: 'Documentation' },
        { href: '/support', label: 'Support' }
      ]
    },
    {
      title: 'Company',
      links: [
        { href: '/about', label: 'About' },
        { href: '/contact', label: 'Contact' },
        { href: '/privacy', label: 'Privacy' }
      ]
    }
  ];
</script>
```

```
<footer class="footer">
  <div class="container">
    <div class="footer-grid">
      <div class="footer-brand">
        <strong class="footer-logo">ShipForge</strong>
        <p class="footer-tagline">Build and ship modern web apps faster.</p>
      </div>

      {#each footerSections as section}
        <div class="footer-section">
          <h3 class="footer-heading">{section.title}</h3>
          <ul class="footer-links">
            {#each section.links as link}
              <li>
                <a href={link.href} class="footer-link">{link.label}</a>
              </li>
            {/each}
          </ul>
        </div>
      {/each}
    </div>

    <div class="footer-bottom">
      <p>&copy; {year} ShipForge. All rights reserved.</p>
    </div>
  </div>
</footer>

<style>
  .footer {
    background: var(--color-gray-950);
    color: var(--color-gray-400);
    padding: var(--space-16) 0 var(--space-8);
  }

  .footer-grid {
    display: grid;
    grid-template-columns: 1fr;
    gap: var(--space-8);
    margin-bottom: var(--space-12);
  }

  .footer-logo {
    font-size: var(--text-xl);
    color: white;
  }

  .footer-tagline {
    margin-top: var(--space-2);
    font-size: var(--text-sm);
  }

  .footer-heading {
    color: white;
    font-size: var(--text-sm);
    font-weight: 600;
    text-transform: uppercase;
    letter-spacing: var(--tracking-wide);
    margin-bottom: var(--space-4);
  }

  .footer-links {
    list-style: none;
    display: flex;
    flex-direction: column;
    gap: var(--space-2);
  }

  .footer-link {
    color: var(--color-gray-400);
    font-size: var(--text-sm);
    transition: color var(--transition-fast);
  }

  .footer-link:hover {
    color: white;
    text-decoration: none;
  }

  .footer-bottom {
    border-top: 1px solid var(--color-gray-800);
    padding-top: var(--space-8);
    font-size: var(--text-sm);
  }

  @media (min-width: 640px) {
    .footer-grid {
      grid-template-columns: 2fr 1fr 1fr 1fr;
    }
  }
</style>
```

Commit:

```
git add -A
git commit -m "feat: add core UI components (Button, Card, SectionHeader, Navigation, Footer)"
```

Phase 4: Page Architecture

Step 4.1: Update Layout with Navigation and Footer

File: `src/routes/+layout.svelte`

```
<script>
  import '$lib/styles/global.css';
  import Navigation from '$lib/components/layout/Navigation.svelte';
  import Footer from '$lib/components/layout/Footer.svelte';

  let { children } = $props();
</script>
```

```
<Navigation />
<main>
  {@render children()}
</main>
<Footer />

<style>
  main {
    min-height: calc(100vh - 64px);
  }
</style>
```

Step 4.2: Homepage

File: `src/routes/+page.svelte`

```
<script>
  import SEO from '$lib/components/SEO.svelte';
  import Button from '$lib/components/ui/Button.svelte';
  import SectionHeader from '$lib/components/ui/SectionHeader.svelte';
  import Card from '$lib/components/ui/Card.svelte';

  const features = [
    {
      icon: '⚡',
      title: 'Lightning Fast',
      description: 'Built on SvelteKit with SSR, code splitting, and preloading for sub-second
page loads.'
    },
    {
      icon: '🎨',
      title: 'Beautiful Design',
      description: 'A polished design system with responsive components ready for production.'
    },
    {
      icon: '🔒',
      title: 'Production Ready',
      description: 'Authentication, SEO, analytics, and error handling built in from day one.'
    },
    {
      icon: '📱',
      title: 'Mobile First',
      description: 'Every component is responsive and touch-friendly out of the box.'
    },
    {
      icon: '🎬',
      title: 'Smooth Animations',
      description: 'GSAP-powered cinematic animations with ScrollTrigger integration.'
    },
    {
      icon: '🔍',
      title: 'SEO Optimized',
      description: 'Meta tags, Open Graph, structured data, and sitemap generation included.'
    }
  ];
```

```
</script>

<SEO
  title="Build & Ship Faster"
  description="ShipForge is the modern SaaS boilerplate for SvelteKit. Launch your product in
days, not months."
/>

<!-- Hero Section -->
<section class="hero">
  <div class="container">
    <div class="hero-content">
      <span class="hero-badge">Now in Beta</span>
      <h1 class="hero-title">
        Build & Ship<br />
        <span class="hero-gradient">Modern Web Apps</span>
      </h1>
      <p class="hero-description">
        ShipForge is the production-ready SvelteKit boilerplate that gets you
        from idea to launch in days, not months.
      </p>
      <div class="hero-actions">
        <Button href="/pricing" size="lg">Get Started</Button>
        <Button href="/features" variant="secondary" size="lg">Learn More</Button>
      </div>
    </div>
  </div>
</section>

<!-- Features Section -->
<section class="features-section">
  <div class="container">
    <SectionHeader
      badge="Features"
      title="Everything You Need"
      description="From design system to deployment, ShipForge has you covered."
    />

    <div class="features-grid">
      {#each features as feature}
        <Card>
          <span class="feature-icon">{feature.icon}</span>
          <h3 class="feature-title">{feature.title}</h3>
          <p class="feature-description">{feature.description}</p>
        </Card>
      {/each}
    </div>
  </div>
</section>

<!-- CTA Section -->
<section class="cta-section">
  <div class="container">
    <div class="cta-content">
      <h2 class="cta-title">Ready to Ship?</h2>
      <p class="cta-description">
```

```
        Join thousands of developers building with ShipForge.
      </p>
      <Button href="/pricing" size="lg">Start Building Today</Button>
    </div>
  </div>
</section>

<style>
  /* Hero */
  .hero {
    padding: var(--space-20) 0 var(--space-24);
    text-align: center;
  }

  .hero-content {
    max-width: 800px;
    margin: 0 auto;
  }

  .hero-badge {
    display: inline-block;
    padding: var(--space-1) var(--space-4);
    background: rgba(99, 102, 241, 0.1);
    color: var(--color-primary);
    border-radius: var(--radius-full);
    font-size: var(--text-sm);
    font-weight: 600;
    margin-bottom: var(--space-6);
  }

  .hero-title {
    font-size: var(--text-5xl);
    font-weight: 800;
    line-height: var(--leading-tight);
    margin-bottom: var(--space-6);
  }

  .hero-gradient {
    background: linear-gradient(135deg, var(--color-primary), var(--color-secondary));
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
  }

  .hero-description {
    font-size: var(--text-xl);
    color: var(--color-text-secondary);
    max-width: 600px;
    margin: 0 auto var(--space-8);
    line-height: var(--leading-relaxed);
  }

  .hero-actions {
    display: flex;
    gap: var(--space-4);
    justify-content: center;
```

```css
  flex-wrap: wrap;
}

@media (min-width: 1024px) {
  .hero-title {
    font-size: var(--text-6xl);
  }
}

/* Features */
.features-section {
  padding: var(--space-24) 0;
  background: var(--color-bg-secondary);
}

.features-grid {
  display: grid;
  grid-template-columns: 1fr;
  gap: var(--space-6);
}

@media (min-width: 640px) {
  .features-grid {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (min-width: 1024px) {
  .features-grid {
    grid-template-columns: repeat(3, 1fr);
  }
}

.feature-icon {
  font-size: var(--text-3xl);
  margin-bottom: var(--space-4);
  display: block;
}

.feature-title {
  font-size: var(--text-xl);
  font-weight: 700;
  margin-bottom: var(--space-2);
}

.feature-description {
  color: var(--color-text-secondary);
  font-size: var(--text-sm);
  line-height: var(--leading-relaxed);
}

/* CTA */
.cta-section {
  padding: var(--space-24) 0;
}
```

```
  .cta-content {
    text-align: center;
    max-width: 600px;
    margin: 0 auto;
  }

  .cta-title {
    font-size: var(--text-4xl);
    font-weight: 800;
    margin-bottom: var(--space-4);
  }

  .cta-description {
    font-size: var(--text-lg);
    color: var(--color-text-secondary);
    margin-bottom: var(--space-8);
  }
</style>
```

Step 4.3: About Page

File: `src/routes/about/+page.svelte`

```
<script>
  import SEO from '$lib/components/SEO.svelte';
  import SectionHeader from '$lib/components/ui/SectionHeader.svelte';
</script>

<SEO
  title="About"
  description="Learn about the mission and team behind ShipForge."
/>

<section class="about-hero">
  <div class="container">
    <SectionHeader
      badge="About"
      title="Our Mission"
      description="We believe every developer deserves tools that make shipping fast, beautiful
web apps effortless."
    />
  </div>
</section>

<style>
  .about-hero {
    padding: var(--space-20) 0 var(--space-24);
  }
</style>
```

Step 4.4: Static Page Prerendering

File: `src/routes/+layout.js`

```
// Prerender all pages by default
export const prerender = true;
```

Verification:

```
pnpm build
# Expected: Pages are prerendered in build output
```

Commit:

```
git add -A
git commit -m "feat: add page architecture with homepage, about, and layout"
```

Phase 5: Animations with GSAP

Step 5.1: GSAP Reveal Action

File: `src/lib/actions/gsapReveal.js`

```
import gsap from 'gsap';
import { ScrollTrigger } from 'gsap/ScrollTrigger';

gsap.registerPlugin(ScrollTrigger);

export function gsapReveal(node, params = {}) {
  const {
    y = 40,
    opacity = 0,
    duration = 0.6,
    ease = 'power2.out',
    start = 'top 85%',
    delay = 0
  } = params;

  // Check for reduced motion preference
  if (window.matchMedia('(prefers-reduced-motion: reduce)').matches) {
    return { destroy() {} };
  }

  gsap.set(node, { opacity, y });

  const tween = gsap.to(node, {
    opacity: 1,
    y: 0,
    duration,
    ease,
    delay,
    scrollTrigger: {
      trigger: node,
      start,
      toggleActions: 'play none none reverse'
    }
  });
  return {
    destroy() {
      tween.scrollTrigger?.kill();
      tween.kill();
    }
  };
}
```

Step 5.2: Hero Animation

File: `src/lib/animations/heroEntrance.js`

```
import gsap from 'gsap';

export function createHeroEntrance(container) {
  if (window.matchMedia('(prefers-reduced-motion: reduce)').matches) {
    return { play() {}, revert() {} };
  }

  const ctx = gsap.context(() => {
    const tl = gsap.timeline({
      defaults: { ease: 'power3.out' }
    });

    gsap.set('.hero-badge', { opacity: 0, y: 20 });
    gsap.set('.hero-title', { opacity: 0, y: 40 });
    gsap.set('.hero-description', { opacity: 0, y: 30 });
    gsap.set('.hero-actions', { opacity: 0, y: 20 });

    tl.to('.hero-badge', { opacity: 1, y: 0, duration: 0.5 })
      .to('.hero-title', { opacity: 1, y: 0, duration: 0.7 }, '-=0.2')
      .to('.hero-description', { opacity: 1, y: 0, duration: 0.5 }, '-=0.3')
      .to('.hero-actions', { opacity: 1, y: 0, duration: 0.4 }, '-=0.2');
  }, container);

  return {
    revert: () => ctx.revert()
  };
}
```

Step 5.3: Integrate Animations in Homepage

Update `src/routes/+page.svelte` hero section:

```
<script>
  import { browser } from '$app/environment';
  import { gsapReveal } from '$lib/actions/gsapReveal';
  import { createHeroEntrance } from '$lib/animations/heroEntrance';

  let heroRef = $state(null);

  $effect(() => {
    if (!browser || !heroRef) return;
    const anim = createHeroEntrance(heroRef);
    return () => anim.revert();
  });
</script>

<section class="hero" bind:this={heroRef}>
  <!-- ... hero content (classes match animation selectors) ... -->
</section>

<!-- Feature cards with scroll reveal -->
{#each features as feature}
  <div use:gsapReveal={{ delay: 0.1 }}>
    <Card>
      <!-- ... -->
    </Card>
  </div>
{/each}
```

Verification:

```
pnpm dev
# Open browser — hero elements animate in on load
# Scroll down — feature cards reveal on scroll
# No console errors
```

Commit:

```
git add -A
git commit -m "feat: add GSAP animations with hero entrance and scroll reveal"
```

Phase 6: SEO Implementation

Step 6.1: SEO Component

File: `src/lib/components/SEO.svelte`

```
<script>
  import { page } from '$app/stores';

  let {
    title = '',
    description = '',
    image = '',
    type = 'website',
    noindex = false
  } = $props();

  const SITE_NAME = 'ShipForge';
  const SITE_URL = 'https://shipforge.dev';
  const DEFAULT_IMAGE = `${SITE_URL}/og-default.png`;

  const fullTitle = title ? `${title} | ${SITE_NAME}` : SITE_NAME;
  const fullImage = image || DEFAULT_IMAGE;
  const canonical = `${SITE_URL}${$page.url.pathname}`;
</script>
```

```
<svelte:head>
  <title>{fullTitle}</title>
  <meta name="description" content={description} />
  <meta name="robots" content={noindex ? 'noindex, nofollow' : 'index, follow'} />
  <link rel="canonical" href={canonical} />

  <meta property="og:type" content={type} />
  <meta property="og:title" content={fullTitle} />
  <meta property="og:description" content={description} />
  <meta property="og:image" content={fullImage} />
  <meta property="og:url" content={canonical} />
  <meta property="og:site_name" content={SITE_NAME} />

  <meta name="twitter:card" content="summary_large_image" />
  <meta name="twitter:title" content={fullTitle} />
  <meta name="twitter:description" content={description} />
  <meta name="twitter:image" content={fullImage} />
</svelte:head>
```

Step 6.2: JSON-LD Component

File: `src/lib/components/JsonLd.svelte`

```
<script>
  let { schema } = $props();
</script>

<svelte:head>
  {@html `<script type="application/ld+json">${JSON.stringify(schema)}</script>`}
</svelte:head>
```

Step 6.3: Sitemap

File: `src/routes/sitemap.xml/+server.js`

```javascript
const SITE_URL = 'https://shipforge.dev';

const pages = [
  { path: '/', priority: 1.0, changefreq: 'weekly' },
  { path: '/features', priority: 0.8, changefreq: 'monthly' },
  { path: '/pricing', priority: 0.9, changefreq: 'weekly' },
  { path: '/about', priority: 0.7, changefreq: 'monthly' },
  { path: '/blog', priority: 0.9, changefreq: 'daily' },
  { path: '/contact', priority: 0.6, changefreq: 'yearly' }
];

export const prerender = true;

export async function GET() {
  const sitemap = `<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
${pages
  .map(
    (p) => `  <url>
```

```
      <loc>${SITE_URL}${p.path}</loc>
      <lastmod>${new Date().toISOString().split('T')[0]}</lastmod>
      <changefreq>${p.changefreq}</changefreq>
      <priority>${p.priority}</priority>
    </url>`
    )
    .join('\n')}
</urlset>`;

  return new Response(sitemap, {
    headers: {
      'Content-Type': 'application/xml',
      'Cache-Control': 'max-age=3600'
    }
  });
}
```

Step 6.4: robots.txt

File: `src/routes/robots.txt/+server.js`

```
export const prerender = true;

export function GET() {
  return new Response(
    `User-agent: *
Allow: /
Disallow: /api/

Sitemap: https://shipforge.dev/sitemap.xml`,
    {
      headers: { 'Content-Type': 'text/plain' }
    }
  );
}
```

Verification:

```
pnpm build
pnpm preview
# Visit http://localhost:4173/sitemap.xml — valid XML
# Visit http://localhost:4173/robots.txt — valid robots file
# View page source — meta tags present in <head>
```

Commit:

```
git add -A
git commit -m "feat: add SEO components, sitemap, robots.txt, and structured data"
```

Phase 7: Forms & Interactivity

Step 7.1: Contact Form with Server Action

File: `src/routes/contact/+page.svelte`

```
<script>
  import SEO from '$lib/components/SEO.svelte';
  import Button from '$lib/components/ui/Button.svelte';
  import { enhance } from '$app/forms';

  let { form } = $props();
  let submitting = $state(false);
</script>

<SEO
  title="Contact"
  description="Get in touch with the ShipForge team."
/>

<section class="contact-section">
  <div class="container">
    <h1>Contact Us</h1>

    {#if form?.success}
      <div class="alert alert--success">
        Thank you! We'll get back to you soon.
      </div>
    {/if}

    <form
      method="POST"
      use:enhance={() => {
        submitting = true;
        return async ({ update }) => {
          await update();
          submitting = false;
        };
      }}
    >
      <div class="form-field">
        <label for="name">Name</label>
        <input
          type="text"
          id="name"
          name="name"
          required
          value={form?.data?.name ?? ''}
        />
        {#if form?.errors?.name}
          <span class="error">{form.errors.name}</span>
        {/if}
      </div>

      <div class="form-field">
        <label for="email">Email</label>
        <input
          type="email"
          id="email"
          name="email"
          required
          value={form?.data?.email ?? ''}
        />
        {#if form?.errors?.email}
          <span class="error">{form.errors.email}</span>
        {/if}
      </div>

      <div class="form-field">
        <label for="message">Message</label>
        <textarea
          id="message"
          name="message"
          rows="5"
          required
        >{form?.data?.message ?? ''}</textarea>
        {#if form?.errors?.message}
          <span class="error">{form.errors.message}</span>
        {/if}
      </div>

      <Button type="submit" disabled={submitting}>
        {submitting ? 'Sending...' : 'Send Message'}
      </Button>
    </form>
  </div>
</section>

<style>
  .contact-section {
    padding: var(--space-20) 0;
    max-width: var(--max-width-narrow);
    margin: 0 auto;
  }

  h1 {
    margin-bottom: var(--space-8);
  }

  .form-field {
    margin-bottom: var(--space-6);
  }

  label {
    display: block;
    font-weight: 600;
    margin-bottom: var(--space-2);
    font-size: var(--text-sm);
  }

  input,
  textarea {
    width: 100%;
    padding: var(--space-3) var(--space-4);
    border: 1px solid var(--color-border);
    border-radius: var(--radius-md);
    font-size: var(--text-base);
    transition: border-color var(--transition-fast);
  }

  input:focus,
  textarea:focus {
    outline: none;
    border-color: var(--color-primary);
    box-shadow: 0 0 0 3px rgba(99, 102, 241, 0.1);
  }

  .error {
    color: var(--color-error);
    font-size: var(--text-sm);
    margin-top: var(--space-1);
    display: block;
  }

  .alert--success {
    padding: var(--space-4);
    background: rgba(16, 185, 129, 0.1);
    color: var(--color-success);
    border-radius: var(--radius-md);
    margin-bottom: var(--space-6);
    font-weight: 500;
  }
</style>
```

Step 7.2: Form Server Action

File: `src/routes/contact/+page.server.js`

```
import { fail } from '@sveltejs/kit';

export const prerender = false;

export const actions = {
  default: async ({ request }) => {
    const formData = await request.formData();
    const name = formData.get('name')?.toString().trim();
    const email = formData.get('email')?.toString().trim();
    const message = formData.get('message')?.toString().trim();

    const errors = {};

    if (!name || name.length < 2) {
      errors.name = 'Name must be at least 2 characters.';
    }

    if (!email || !email.includes('@')) {
      errors.email = 'Please enter a valid email address.';
    }

    if (!message || message.length < 10) {
      errors.message = 'Message must be at least 10 characters.';
    }

    if (Object.keys(errors).length > 0) {
      return fail(400, {
        errors,
        data: { name, email, message }
      });
    }

    // In production: send email, save to database, etc.
    console.log('Contact form submission:', { name, email, message });

    return { success: true };
  }
};
```

Verification:

```
pnpm dev
# Navigate to /contact
# Submit empty form → validation errors appear
# Submit valid form → success message appears
# Check terminal → form data logged
```

Commit:

```
git add -A
git commit -m "feat: add contact form with server-side validation"
```

Phase 8: Testing

Step 8.1: Configure Playwright

File: `playwright.config.js`

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  webServer: {
    command: 'pnpm build && pnpm preview',
    port: 4173,
    reuseExistingServer: !process.env.CI
  },
  testDir: 'tests',
  testMatch: /(.+\.)?(test|spec)\.[jt]s/
});
```

Step 8.2: SEO Tests

File: `tests/seo.test.js`

```
import { expect, test } from '@playwright/test';

const pages = [
  { path: '/', title: 'Build & Ship Faster' },
  { path: '/about', title: 'About' },
  { path: '/contact', title: 'Contact' }
];

for (const { path, title } of pages) {
  test(`${path} has correct title`, async ({ page }) => {
    await page.goto(path);
    const pageTitle = await page.title();
    expect(pageTitle).toContain(title);
  });

  test(`${path} has meta description`, async ({ page }) => {
    await page.goto(path);
    const desc = await page.getAttribute('meta[name="description"]', 'content');
    expect(desc).toBeTruthy();
    expect(desc.length).toBeGreaterThan(20);
  });

  test(`${path} has canonical URL`, async ({ page }) => {
    await page.goto(path);
    const canonical = await page.getAttribute('link[rel="canonical"]', 'href');
    expect(canonical).toBeTruthy();
  });

  test(`${path} has OG tags`, async ({ page }) => {
    await page.goto(path);
    const ogTitle = await page.getAttribute('meta[property="og:title"]', 'content');
    expect(ogTitle).toBeTruthy();
  });

  test(`${path} has exactly one H1`, async ({ page }) => {
    await page.goto(path);
    const h1Count = await page.locator('h1').count();
    expect(h1Count).toBe(1);
  });
}

test('sitemap.xml returns valid XML', async ({ request }) => {
  const response = await request.get('/sitemap.xml');
  expect(response.ok()).toBeTruthy();
  const body = await response.text();
  expect(body).toContain('<urlset');
  expect(body).toContain('<url>');
});

test('robots.txt is valid', async ({ request }) => {
  const response = await request.get('/robots.txt');
  expect(response.ok()).toBeTruthy();
  const body = await response.text();
  expect(body).toContain('User-agent');
  expect(body).toContain('Sitemap');
});
```

Step 8.3: Navigation Tests

File: `tests/navigation.test.js`

```
import { expect, test } from '@playwright/test';

test('navigation links work', async ({ page }) => {
  await page.goto('/');

  // Click About link
  await page.click('a[href="/about"]');
  await expect(page).toHaveURL('/about');

  // Click Contact link
  await page.click('a[href="/contact"]');
  await expect(page).toHaveURL('/contact');

  // Click logo to go home
  await page.click('.logo');
  await expect(page).toHaveURL('/');
});

test('contact form validates inputs', async ({ page }) => {
  await page.goto('/contact');

  // Submit empty form
  await page.click('button[type="submit"]');

  // Form should show validation (browser native or server)
  const url = page.url();
  expect(url).toContain('/contact');
});

test('contact form submits successfully', async ({ page }) => {
  await page.goto('/contact');

  await page.fill('#name', 'Test User');
  await page.fill('#email', 'test@example.com');
  await page.fill('#message', 'This is a test message from Playwright.');

  await page.click('button[type="submit"]');

  // Wait for success message
  await expect(page.locator('.alert--success')).toBeVisible();
});
```

Step 8.4: Run Tests

Command:

```
# Install browsers
pnpm exec playwright install

# Run all tests
pnpm exec playwright test

# Run with UI
pnpm exec playwright test --headed
```

```
# View report
pnpm exec playwright show-report
```

Verification:

```
pnpm exec playwright test
# Expected: All tests pass
# Output: X passed, 0 failed
```

Commit:

```
git add -A
git commit -m "test: add Playwright tests for SEO, navigation, and contact form"
```

Phase 9: Deployment

Step 9.1: Environment Variables

File: `.env.example`

```
# Public (exposed to client)
PUBLIC_SITE_URL=https://shipforge.dev
PUBLIC_SITE_NAME=ShipForge

# Private (server only)
PRIVATE_CONTACT_EMAIL=hello@shipforge.dev
```

Step 9.2: Pre-Deployment Checklist

```
# 1. Type check
pnpm check
# Expected: No errors

# 2. Lint
pnpm lint
# Expected: No errors

# 3. Build
pnpm build
# Expected: Build succeeds, no warnings

# 4. Preview
pnpm preview
# Expected: Site works locally at localhost:4173

# 5. Test
pnpm exec playwright test
# Expected: All tests pass

# 6. Check bundle size
du -sh build/
```

```
ls -lah build/client/_app/immutable/chunks/ | head -10
# Expected: Reasonable sizes (< 200KB JS total gzipped)
```

Step 9.3: Deploy to Vercel

Command:

```
# Install Vercel CLI
pnpm add -g vercel

# Login
vercel login

# Deploy preview
vercel

# Set environment variables
vercel env add PUBLIC_SITE_URL
# Enter: https://shipforge.dev

vercel env add PRIVATE_CONTACT_EMAIL
# Enter: hello@shipforge.dev

# Deploy to production
vercel --prod
```

Verification:

- Visit the production URL
- Check all pages load correctly
- Test the contact form
- Run Lighthouse audit (aim for 90+ in all categories)
- Verify sitemap.xml and robots.txt are accessible
- Test OG tags with https://opengraph.xyz

Step 9.4: Post-Deployment

```
# Tag the release
git tag -a v1.0.0 -m "Release v1.0.0: Initial production deployment"
git push --tags

# Submit sitemap to Google Search Console
# 1. Go to https://search.google.com/search-console
# 2. Add property (your domain)
# 3. Submit sitemap URL: https://shipforge.dev/sitemap.xml
```

Step 9.5: Monitoring Setup

```
# Set up monitoring:
# 1. Google Search Console - for search performance
# 2. Google Analytics or Plausible - for traffic
# 3. Vercel Analytics - for Web Vitals
# 4. UptimeRobot - for uptime monitoring
# Run periodic checks:
pnpm dlx lighthouse https://shipforge.dev --output html --output-path lighthouse.html
```

Final Project Structure

```
shipforge/
├── src/
│   ├── app.html
│   ├── lib/
│   │   ├── actions/
│   │   │   └── gsapReveal.js
│   │   ├── animations/
│   │   │   └── heroEntrance.js
│   │   ├── components/
│   │   │   ├── layout/
│   │   │   │   ├── Navigation.svelte
│   │   │   │   └── Footer.svelte
│   │   │   ├── sections/
│   │   │   │   └── (section components)
│   │   │   ├── ui/
│   │   │   │   ├── Button.svelte
│   │   │   │   ├── Card.svelte
│   │   │   │   └── SectionHeader.svelte
│   │   │   ├── JsonLd.svelte
│   │   │   └── SEO.svelte
│   │   ├── data/
│   │   │   └── (static data files)
│   │   ├── stores/
│   │   │   └── (Svelte stores)
│   │   ├── styles/
│   │   │   ├── global.css
│   │   │   └── tokens.css
│   │   └── utils/
│   │       └── (utility functions)
│   └── routes/
│       ├── +error.svelte
│       ├── +layout.js
│       ├── +layout.svelte
│       ├── +page.svelte
│       ├── about/
│       │   └── +page.svelte
│       ├── contact/
│       │   ├── +page.server.js
│       │   └── +page.svelte
│       ├── features/
│       │   └── +page.svelte
│       ├── pricing/
│       │   └── +page.svelte
│       ├── blog/
│       │   └── +page.svelte
│       ├── robots.txt/
│       │   └── +server.js
│       └── sitemap.xml/
│           └── +server.js
├── static/
```

```
│   ├── favicon.png
│   ├── fonts/
│   ├── images/
│   └── icons/
├── tests/
│   ├── navigation.test.js
│   └── seo.test.js
├── .env.example
├── package.json
├── pnpm-lock.yaml
├── playwright.config.js
├── svelte.config.js
└── vite.config.js
```

Build Phases Summary

Phase	What You Built	Key Files
1. Setup	Project scaffold, deps, git	`svelte.config.js`, `package.json`
2. Design System	Tokens, reset, typography	`tokens.css`, `global.css`
3. Components	Button, Card, Nav, Footer	`src/lib/components/`
4. Pages	Homepage, About, Layout	`src/routes/`
5. Animations	GSAP reveal, hero entrance	`src/lib/animations/`
6. SEO	Meta tags, sitemap, robots	`SEO.svelte`, `sitemap.xml`
7. Forms	Contact form, validation	`contact/+page.server.js`
8. Testing	Playwright E2E tests	`tests/`
9. Deployment	Vercel, monitoring	Production URL

Git Commit History (Expected)

```
1. chore: initial SvelteKit project setup
2. feat: add design system with tokens, global styles, and fonts
3. feat: add core UI components (Button, Card, SectionHeader, Navigation, Footer)
4. feat: add page architecture with homepage, about, and layout
5. feat: add GSAP animations with hero entrance and scroll reveal
6. feat: add SEO components, sitemap, robots.txt, and structured data
7. feat: add contact form with server-side validation
8. test: add Playwright tests for SEO, navigation, and contact form
9. chore: deploy to production v1.0.0
```

End of Capstone Build Book: ShipForge End-to-End