Write SEO landing page copy for ClawStak.ai

Source: worker-2 fleet

Implementation plan:

Implementation Plan: SEO Landing Page for ClawStak.ai

1. Files to Create or Modify

```
src/
landing/
LandingPage.test.tsx            # Integration tests for full page
components/
HeroSection.tsx                 # Hero headline + primary CTA
FeatureSection.tsx              # Reusable feature block (marketplace, A2A, trust)
SocialProofSection.tsx          # Logos, testimonials, metrics
PricingSection.tsx              # Free / Pro / Enterprise tiers
PricingTierCard.tsx             # Individual pricing card
FAQSection.tsx                  # Accordion FAQ
CTASection.tsx                  # Bottom-of-page call to action
index.ts                        # Barrel exports
landing/
pricing.ts                      # Pricing tiers data
socialProof.ts                  # Testimonials, logos, metrics
types/
hooks/
utils/
structuredData.test.ts
landing/
e2e/
```

2. Key Design Decisions

2.1 Architecture: Content/Presentation Separation

All copy lives in src/data/landing/*.ts -- never hardcoded in components. This is the single most important decision. It enables:

- Non-engineer copy edits without touching component logic

- A/B testing at the data layer

```
// src/types/landing.ts


id: string;

headline: string;

description: string;

icon: string;                    // Icon identifier (e.g., "marketplace", "a2a", "trust")

ctaHref: string;


id: string;

price: string;                   // "$0" | "$49/mo" | "Custom"

description: string;

highlighted: boolean;            // Visual emphasis (Pro tier)

ctaHref: string;


id: string;

answer: string;                  // Supports markdown-light (bold, links)


id: string;

author: string;

company: string;

}


value: string;                   // "10,000+"

}


title: string;

canonicalUrl: string;

keywords: string[];
```

2.2 SEO Strategy: Server-Side Rendering + Structured Data

- SSR/SSG via Next.js generateMetadata -- all meta tags rendered server-side.

- Semantic HTML -- <main>, <section>, <article>, <h1>-><h3> hierarchy. Exactly ONE <h1> on the page.

- loading="lazy" on below-fold images; critical images use priority prop.

```
// src/data/landing/seo.ts



title: 'ClawStak.ai -- The Agent Marketplace with Built-In Trust',

'Discover, deploy, and orchestrate AI agents with ClawStak.ai. ' +

canonicalUrl: 'https://clawstak.ai',

keywords: [

'agent-to-agent protocol',

'trust scoring',

'developer tools',

],
```

2.3 Component API Design

Components are stateless and data-driven. Each section receives its content as props.

```
// Component signatures


interface HeroSectionProps {

subheadline: string;       // The value prop paragraph

secondaryCTA: { text: string; href: string };


interface FeatureSectionProps {

index: number;             // Used for alternating left/right layout


interface SocialProofSectionProps {

testimonials: Testimonial[];

}


interface PricingSectionProps {

tiers: PricingTier[];


interface PricingTierCardProps {

}


interface FAQSectionProps {

items: FAQItem[];


interface CTASectionProps {
subtext: string;

}
```

2.4 Styling Approach

- CSS Modules for scoped styles -- no runtime CSS-in-JS overhead (critical for LCP).

- No Tailwind -- explicit class names are more readable for a content-heavy page and produce smaller CSS bundles for a single page.

2.5 Accessibility

- FAQ uses <details>/<summary> native HTML (progressive enhancement, no JS required for core functionality).

- Color contrast 4.5:1 (WCAG AA).

---

3. Implementation Approach

Phase 1: Data Layer + Types (1-2 hours)

Create types/landing.ts and all files under data/landing/. This is the content authoring pass.

```
  // src/data/landing/features.ts




  {

  slug: 'marketplace',

  subheadline: 'Discover and deploy production-ready AI agents in minutes.',

  'Browse a curated registry of AI agents built by the community. ' +

  bulletPoints: [

  'One-click deployment to any cloud provider',

  'Usage analytics and cost tracking per agent',

  icon: 'marketplace',

  ctaHref: '/marketplace',

  {

  slug: 'a2a',

  subheadline: 'Your agents talk to each other -- securely and natively.',

  'ClawStak implements the A2A protocol so agents negotiate, delegate, ' +

  'discover peers, and orchestrate multi-agent workflows declaratively.',

  'First-class A2A protocol implementation',

  'Capability-based agent discovery',
  ],

  ctaText: 'Read the A2A Docs',

  },

  id: 'trust-scoring',

  headline: 'Trust Scoring You Can Verify',

  description:

  'community reviews, security audits, and uptime history. ' +

  bulletPoints: [

  'Fully queryable via REST and GraphQL APIs',

  'Threshold-based deployment gates for enterprise',

  icon: 'trust',

  ctaHref: '/docs/trust-scoring',

  ];
  // src/data/landing/pricing.ts




  {

  name: 'Free',

  priceSubtext: 'forever',

  features: [

  'Community marketplace access',

  'Public trust scores',

  ],

  ctaText: 'Get Started Free',

  },

  id: 'pro',

  price: '$49',

  description: 'For teams building production agent workflows.',

  'Unlimited deployed agents',

  'Full A2A orchestration engine',

  'Priority support (< 4hr SLA)',

  'CI/CD integration',

  highlighted: true,

  ctaHref: '/signup?plan=pro',

  {

  name: 'Enterprise',

  priceSubtext: 'contact sales',
  features: [
  'Dedicated infrastructure',
  'SSO / SAML / SCIM',
  'Dedicated solutions engineer',
  ],
  ctaText: 'Contact Sales',
  },
```

```
// src/data/landing/faq.ts



{
question: 'What is ClawStak.ai?',
'ClawStak.ai is a developer platform for discovering, deploying, and orchestrating ' +
'and a transparent trust scoring system.',
{
question: 'What is the A2A protocol?',
'A2A (Agent-to-Agent) is a communication protocol that lets AI agents discover each other\'s ' +
'ClawStak provides a first-class implementation.',
{
question: 'How does trust scoring work?',
'Each agent receives a composite trust score derived from runtime behavior analysis, ' +
'fully transparent and queryable via our API.',
{
question: 'What are the limits on the Free tier?',
'The Free tier supports up to 3 deployed agents, community marketplace access, ' +
},
id: 'enterprise-compliance',
answer:
'and optional on-chain trust attestation for auditability.',
{
question: 'Can I self-host ClawStak?',
'Enterprise customers can deploy ClawStak on dedicated infrastructure within their ' +
},
```

Phase 2: Structured Data Utilities (1 hour)

```
// src/utils/structuredData.ts
```

```
* Generates a JSON-LD FAQPage schema from FAQ items.

*/

return {

'@type': 'FAQPage',

'@type': 'Question',

acceptedAnswer: {

text: item.answer,

})),

}


* Generates a JSON-LD SoftwareApplication schema for the product.

export function buildProductSchema(tiers: PricingTier[]): Record<string, unknown> {

'@context': 'https://schema.org',

name: 'ClawStak.ai',

operatingSystem: 'Web',

'@type': 'Offer',

price: tier.price === 'Custom' ? undefined : tier.price.replace('$', ''),

description: tier.description,

};


* Generates a JSON-LD Organization schema.

export function buildOrganizationSchema(): Record<string, unknown> {

'@context': 'https://schema.org',

name: 'ClawStak',

logo: 'https://clawstak.ai/logo.png',

'https://github.com/clawstak',

],

}
```

Phase 3: Components (3-4 hours)

Build bottom-up: PricingTierCard -> PricingSection -> FeatureSection -> etc. -> LandingPage.

```
// src/components/landing/HeroSection.tsx




headline: string;

primaryCTA: { text: string; href: string };

}
```

```
headline,

primaryCTA,

}: HeroSectionProps) {

<section className={styles.hero} aria-labelledby="hero-headline">

<h1 id="hero-headline" className={styles.heroHeadline}>

</h1>

<div className={styles.heroCTAs}>

{primaryCTA.text}

<a href={secondaryCTA.href} className={styles.btnSecondary}>

</a>

</div>

);

// src/components/landing/FeatureSection.tsx


import styles from '@/styles/landing/landing.module.css';


feature: Feature;

}


const isReversed = index % 2 !== 0;


<section

className={${styles.feature} ${isReversed ? styles.featureReversed : ''}}

>

<h2 id={feature-${feature.id}-headline} className={styles.featureHeadline}>

</h2>

<p className={styles.featureDescription}>{feature.description}</p>

{feature.bulletPoints.map((point) => (

))}

<a href={feature.ctaHref} className={styles.btnSecondary}>

</a>

<div className={styles.featureVisual} aria-hidden="true">

</div>

);

// src/components/landing/PricingTierCard.tsx
```

```
import styles from '@/styles/landing/landing.module.css';


tier: PricingTier;


return (

className={${styles.pricingCard} ${tier.highlighted ? styles.pricingCardHighlighted : ''}}

>

<h3 id={pricing-${tier.id}} className={styles.pricingName}>

</h3>

<span className={styles.priceValue}>{tier.price}</span>

<span className={styles.priceSubtext}>{tier.priceSubtext}</span>

</div>

<ul className={styles.pricingFeatures}>

<li key={feature}>{feature}</li>

</ul>

{tier.ctaText}

</article>

}
// src/components/landing/FAQSection.tsx


import styles from '@/styles/landing/landing.module.css';


headline: string;

}


return (

<h2 id="faq-headline" className={styles.sectionHeadline}>

</h2>

{items.map((item) => (

<summary className={styles.faqQuestion}>{item.question}</summary>

</details>

</div>

);

// src/pages/landing/LandingPage.tsx
```

```
import { FeatureSection } from '@/components/landing/FeatureSection';

import { PricingSection } from '@/components/landing/PricingSection';

import { CTASection } from '@/components/landing/CTASection';

import { pricingTiers } from '@/data/landing/pricing';

import { socialProof } from '@/data/landing/socialProof';

import { buildFAQSchema, buildProductSchema, buildOrganizationSchema } from '@/utils/structuredData'


title: seoMetadata.title,

keywords: seoMetadata.keywords,

openGraph: {

description: seoMetadata.description,

images: [seoMetadata.ogImage],

},


const faqSchema = buildFAQSchema(faqItems);

const orgSchema = buildOrganizationSchema();


<>

<script

dangerouslySetInnerHTML={{ __html: JSON.stringify(faqSchema) }}

<script

dangerouslySetInnerHTML={{ __html: JSON.stringify(productSchema) }}

<script

dangerouslySetInnerHTML={{ __html: JSON.stringify(orgSchema) }}


<HeroSection

subheadline="The developer platform for discovering, deploying, and orchestrating AI agents -- with

secondaryCTA={{ text: 'Read the Docs', href: '/docs' }}


<FeatureSection key={feature.id} feature={feature} index={index} />


metrics={socialProof.metrics}

logoUrls={socialProof.logoUrls}




headline="Start building with trusted agents today."
```

```
primaryCTA={{ text: 'Get Started Free', href: '/signup' }}

</main>

);
```

---

4. Test Strategy

Layer 1: Unit Tests (Vitest + React Testing Library)

Test each component in isolation. Focus on content rendering and accessibility.

```
// src/components/landing/HeroSection.test.tsx


import { describe, it, expect } from 'vitest';


headline: 'Test Headline',

primaryCTA: { text: 'Primary', href: '/primary' },

};


it('renders the headline as an h1 element', () => {

const heading = screen.getByRole('heading', { level: 1 });

});


render(<HeroSection {...defaultProps} />);

});


render(<HeroSection {...defaultProps} />);

const secondary = screen.getByRole('link', { name: 'Secondary' });

expect(secondary).toHaveAttribute('href', '/secondary');


const { container } = render(<HeroSection {...defaultProps} />);

expect(h1s).toHaveLength(1);


render(<HeroSection {...defaultProps} />);

expect(section).toBeInTheDocument();

});
```

```
// src/components/landing/PricingTierCard.test.tsx


import { describe, it, expect } from 'vitest';

import type { PricingTier } from '@/types/landing';
```

```
id: 'test',

price: '$49',

description: 'For teams',

highlighted: false,

ctaHref: '/signup?plan=pro',


it('renders the tier name, price, and subtext', () => {

expect(screen.getByRole('heading', { name: 'Pro' })).toBeInTheDocument();

expect(screen.getByText('per seat / month')).toBeInTheDocument();


render(<PricingTierCard tier={baseTier} />);

expect(screen.getByText('Feature B')).toBeInTheDocument();


render(<PricingTierCard tier={baseTier} />);

expect(cta).toHaveAttribute('href', '/signup?plan=pro');


render(<PricingTierCard tier={{ ...baseTier, highlighted: true }} />);

});


render(<PricingTierCard tier={baseTier} />);

});

// src/components/landing/FAQSection.test.tsx


import userEvent from '@testing-library/user-event';

import { FAQSection } from './FAQSection';


{ id: '1', question: 'Question 1?', answer: 'Answer 1.' },

];


it('renders all questions', () => {

expect(screen.getByText('Question 1?')).toBeInTheDocument
```

```
pages/

LandingPage.tsx           # Page-level orchestrator component

index.ts                  # Barrel export

landing/

HeroSection.test.tsx

FeatureSection.test.tsx

SocialProofSection.test.tsx

PricingSection.test.tsx

PricingTierCard.test.tsx

FAQSection.test.tsx

CTASection.test.tsx

data/

features.ts               # Feature content data

faq.ts                    # FAQ Q&A pairs

seo.ts                    # Meta tags, structured data (JSON-LD)

landing.ts                # TypeScript interfaces

useFAQSchema.ts           # Generates FAQPage JSON-LD

structuredData.ts         # JSON-LD builder utilities

styles/

landing.module.css        # Page-specific styles (CSS Modules)

landing.spec.ts           # Playwright E2E tests
```

```
  - Future CMS integration (swap static imports for API calls)

  - i18n readiness
```

```
export interface Feature {

slug: string;                    // URL-friendly ID for anchor links

subheadline: string;

bulletPoints: string[];

ctaText: string;

}
export interface PricingTier {

name: 'Free' | 'Pro' | 'Enterprise';

priceSubtext?: string;           // "forever" | "per seat/month" | "contact sales"

features: string[];

ctaText: string;

}
export interface FAQItem {

question: string;

}
export interface Testimonial {

quote: string;

role: string;

avatarUrl?: string;

export interface SocialProofMetric {

label: string;                   // "Agents Deployed"

export interface SEOMetadata {

description: string;
```

ogImage: string;

}

```
  - JSON-LD structured data -- FAQPage schema for FAQ section, Product schema for pricing, Organizatio
  - Anchor links for each section (#marketplace, #a2a, #trust-scoring, #pricing, #faq).
```

import type { SEOMetadata } from '@/types/landing';
export const seoMetadata: SEOMetadata = {

description:

'Agent-to-agent protocols, trust scoring, and a marketplace built for developers.',

ogImage: 'https://clawstak.ai/og-landing.png',

'AI agent marketplace',

'A2A',

'AI orchestration',

'ClawStak',

};

// HeroSection -- single h1, tagline, dual CTAs

headline: string;   // "Ship Trusted AI Agents. Together."

primaryCTA: { text: string; href: string };

}
// FeatureSection -- renders a single feature with alternating layout

feature: Feature;

}
// SocialProofSection

metrics: SocialProofMetric[];

logoUrls: string[];

// PricingSection -- orchestrates tier cards

headline: string;

}

// PricingTierCard -- individual card

tier: PricingTier;

// FAQSection -- accessible accordion

headline: string;

}

// CTASection -- final conversion block

headline: string;

primaryCTA: { text: string; href: string };

```
  - CSS custom properties for theming (dark developer-friendly palette).

  - Container queries for responsive feature sections.
```

```
  - All interactive elements have visible focus indicators.

  - Pricing comparison is a semantic <ul> per tier, not a table (better screen reader experience for c
```

import type { Feature } from '@/types/landing';

export const features: Feature[] = [

id: 'marketplace',

headline: 'A Marketplace Built for Agents',

description:

```
'Filter by capability, runtime, and trust score. One-click deploy to your stack.',

'Curated agent registry with semantic search',

'Version pinning and rollback support',

],

ctaText: 'Explore the Marketplace',

},

id: 'a2a',

headline: 'Native Agent-to-Agent Protocols',

description:

'and collaborate without custom integration code. Define capabilities, ' +

bulletPoints: [

'Declarative multi-agent workflow orchestration',

'End-to-end encrypted agent communication',

icon: 'a2a',

ctaHref: '/docs/a2a',

{

slug: 'trust-scoring',

subheadline: 'Every agent earns its reputation. Transparent. Auditable. On-chain optional.',

'ClawStak computes a composite trust score from runtime behavior, ' +

'Scores are transparent and queryable via API.',

'Composite score from behavior, reviews, audits, uptime',

'Optional on-chain attestation for compliance',

],

ctaText: 'See How Scoring Works',

},
import type { PricingTier } from '@/types/landing';
export const pricingTiers: PricingTier[] = [

id: 'free',

price: '$0',

description: 'For individual developers exploring agent orchestration.',

'Up to 3 deployed agents',

'Basic A2A protocol support',

'Community support',

highlighted: false,

ctaHref: '/signup?plan=free',

{

name: 'Pro',

priceSubtext: 'per seat / month',

features: [

'Private marketplace listings',

'Advanced trust scoring + custom thresholds',

'Team roles and audit logs',

],

ctaText: 'Start Pro Trial',

},

id: 'enterprise',

price: 'Custom',

description: 'For organizations requiring compliance, SLAs, and dedicated infrastructure.',

'Everything in Pro',

'On-chain trust attestation',

'Custom SLA (up to 99.99%)',
'SOC 2 Type II compliance',

highlighted: false,

ctaHref: '/contact?plan=enterprise',

];


import type { FAQItem } from '@/types/landing';
export const faqItems: FAQItem[] = [

id: 'what-is-clawstak',

answer:

'AI agents. It combines a curated marketplace, native agent-to-agent (A2A) protocols, ' +

},

id: 'what-is-a2a',

answer:

'capabilities, negotiate tasks, and collaborate -- without requiring custom integration code. ' +

},

id: 'how-trust-scoring-works',

answer:

'community reviews, third-party security audits, and historical uptime. Scores are ' +

},

id: 'free-tier-limits',

answer:

'basic A2A support, and public trust scores. No credit card required.',

{

question: 'Do you support SOC 2 and other compliance frameworks?',

'Yes. Our Enterprise tier includes SOC 2 Type II compliance, SSO/SAML/SCIM, ' +

},
id: 'self-host',

answer:

'own cloud environment. Contact sales for details.',

];



import type { FAQItem, PricingTier } from '@/types/landing';
/**

  - @see https://schema.org/FAQPage

export function buildFAQSchema(items: FAQItem[]): Record<string, unknown> {

'@context': 'https://schema.org',

mainEntity: items.map((item) => ({

name: item.question,

'@type': 'Answer',

},

};

/**

*/

return {

'@type': 'SoftwareApplication',

applicationCategory: 'DeveloperApplication',

offers: tiers.map((tier) => ({

name: tier.name,

priceCurrency: 'USD',

})),

}
/**
```

```
*/

return {

'@type': 'Organization',

url: 'https://clawstak.ai',

sameAs: [

'https://twitter.com/clawstak',

};
```

```
import styles from '@/styles/landing/landing.module.css';
interface HeroSectionProps {

subheadline: string;

secondaryCTA: { text: string; href: string };

export function HeroSection({

subheadline,

secondaryCTA,

return (

<div className={styles.heroInner}>

{headline}

<p className={styles.heroSubheadline}>{subheadline}</p>

<a href={primaryCTA.href} className={styles.btnPrimary}>

</a>

{secondaryCTA.text}

</div>

</section>

}
```

```
import type { Feature } from '@/types/landing';

interface FeatureSectionProps {

index: number;

export function FeatureSection({ feature, index }: FeatureSectionProps) {

return (

id={feature.slug}

aria-labelledby={feature-${feature.id}-headline}

<div className={styles.featureContent}>

{feature.headline}

<p className={styles.featureSubheadline}>{feature.subheadline}</p>

<ul className={styles.featureBullets}>

<li key={point}>{point}</li>

</ul>

{feature.ctaText}

</div>

{/ Icon/illustration placeholder -- keyed by feature.icon /}

</section>

}


import type { PricingTier } from '@/types/landing';

interface PricingTierCardProps {

}
export function PricingTierCard({ tier }: PricingTierCardProps) {

<article
aria-labelledby={pricing-${tier.id}}

{tier.highlighted && <span className={styles.pricingBadge}>Most Popular</span>}

{tier.name}

<div className={styles.pricingPrice}>

{tier.priceSubtext && (

)}

<p className={styles.pricingDescription}>{tier.description}</p>

{tier.features.map((feature) => (

))}

<a href={tier.ctaHref} className={tier.highlighted ? styles.btnPrimary : styles.btnSecondary}>

</a>

);
```

```
import type { FAQItem } from '@/types/landing';

interface FAQSectionProps {

items: FAQItem[];

export function FAQSection({ headline, items }: FAQSectionProps) {

<section id="faq" className={styles.faq} aria-labelledby="faq-headline">

{headline}

<div className={styles.faqList}>

<details key={item.id} className={styles.faqItem}>

<p className={styles.faqAnswer}>{item.answer}</p>

))}

</section>

}
```

```
import { HeroSection } from '@/components/landing/HeroSection';

import { SocialProofSection } from '@/components/landing/SocialProofSection';

import { FAQSection } from '@/components/landing/FAQSection';

import { features } from '@/data/landing/features';

import { faqItems } from '@/data/landing/faq';

import { seoMetadata } from '@/data/landing/seo';

import type { Metadata } from 'next';
export const metadata: Metadata = {

description: seoMetadata.description,

alternates: { canonical: seoMetadata.canonicalUrl },

title: seoMetadata.title,

url: seoMetadata.canonicalUrl,

type: 'website',

};
export default function LandingPage() {

const productSchema = buildProductSchema(pricingTiers);

return (

{/ JSON-LD Structured Data /}

type="application/ld+json"

/>

type="application/ld+json"

/>

type="application/ld+json"

/>
<main>

headline="Ship Trusted AI Agents. Together."

primaryCTA={{ text: 'Get Started Free', href: '/signup' }}
/>
{features.map((feature, index) => (

))}
<SocialProofSection

testimonials={socialProof.testimonials}

/>
<PricingSection headline="Simple, Developer-Friendly Pricing" tiers={pricingTiers} />
<FAQSection headline="Frequently Asked Questions" items={faqItems} />
<CTASection

subtext="Free tier. No credit card. Deploy your first agent in under 5 minutes."

/>

</>

}
```

```
import { render, screen } from '@testing-library/react';

import { HeroSection } from './HeroSection';
const defaultProps = {

subheadline: 'Test subheadline text',

secondaryCTA: { text: 'Secondary', href: '/secondary' },

describe('HeroSection', () => {

render(<HeroSection {...defaultProps} />);

expect(heading).toHaveTextContent('Test Headline');

it('renders the subheadline', () => {

expect(screen.getByText('Test subheadline text')).toBeInTheDocument();
```

```
it('renders both CTA links with correct hrefs', () => {

const primary = screen.getByRole('link', { name: 'Primary' });

expect(primary).toHaveAttribute('href', '/primary');

});
it('has exactly one h1 on the page', () => {

const h1s = container.querySelectorAll('h1');

});
it('section has appropriate aria-labelledby', () => {

const section = screen.getByRole('region', { name: 'Test Headline' });

});
```

```
import { render, screen } from '@testing-library/react';

import { PricingTierCard } from './PricingTierCard';

const baseTier: PricingTier = {

name: 'Pro',

priceSubtext: 'per seat / month',

features: ['Feature A', 'Feature B'],

ctaText: 'Start Trial',

};
describe('PricingTierCard', () => {

render(<PricingTierCard tier={baseTier} />);

expect(screen.getByText('$49')).toBeInTheDocument();

});
it('renders all feature items', () => {

expect(screen.getByText('Feature A')).toBeInTheDocument();

});
it('renders CTA link with correct href', () => {
```

const cta = screen.getByRole('link', { name: 'Start Trial' });

});

it('applies highlighted styles when tier.highlighted is true', () => {

expect(screen.getByText('Most Popular')).toBeInTheDocument();

it('does not show badge when not highlighted', () => {

expect(screen.queryByText('Most Popular')).not.toBeInTheDocument();

});

import { render, screen } from '@testing-library/react';

import { describe, it, expect } from 'vitest';

const items = [

{ id: '2', question: 'Question 2?', answer: 'Answer 2.' },

describe('FAQSection', () => {

render(<FAQSection headline="FAQ" items={items} />);

Artifact: implementation.md (code)

FILE: src/types/landing.ts

```
export interface Feature {

slug: string;

subheadline: string;

bulletPoints: string[];

ctaText: string;

}


id: string;

price: string;

description: string;

highlighted: boolean;

ctaHref: string;
```

```
id: string;

answer: string;


id: string;

author: string;

company: string;

}


value: string;

}


metrics: SocialProofMetric[];

logoUrls: string[];


title: string;

canonicalUrl: string;

keywords: string[];


text: string;

}
```

FILE: src/data/landing/seo.ts

```
import type { SEOMetadata } from '@/types/landing';


title: 'ClawStak.ai -- The Agent Marketplace with Built-In Trust',

'Discover, deploy, and orchestrate AI agents with ClawStak.ai. ' +

canonicalUrl: 'https://clawstak.ai',

keywords: [

'agent-to-agent protocol',

'trust scoring',

'developer tools',

],
```

FILE: src/data/landing/features.ts

```
import type { Feature } from '@/types/landing';


{

slug: 'marketplace',
subheadline: 'Discover and deploy production-ready AI agents in minutes.',

'Browse a curated registry of AI agents built by the community. ' +

bulletPoints: [

'One-click deployment to any cloud provider',

'Usage analytics and cost tracking per agent',

icon: 'marketplace',

ctaHref: '/marketplace',

{

slug: 'a2a',

subheadline: 'Your agents talk to each other -- securely and natively.',

'ClawStak implements the A2A protocol so agents negotiate, delegate, ' +

'discover peers, and orchestrate multi-agent workflows declaratively.',

'First-class A2A protocol implementation',

'Capability-based agent discovery',

],

ctaText: 'Read the A2A Docs',

},

id: 'trust-scoring',

headline: 'Trust Scoring You Can Verify',

description:

'community reviews, security audits, and uptime history. ' +

bulletPoints: [

'Fully queryable via REST and GraphQL APIs',

'Threshold-based deployment gates for enterprise',

icon: 'trust',

ctaHref: '/docs/trust-scoring',

];
```

FILE: src/data/landing/pricing.ts

```
import type { PricingTier } from '@/types/landing';


{

name: 'Free',

priceSubtext: 'forever',

features: [

'Community marketplace access',

'Public trust scores',

],
ctaText: 'Get Started Free',

},

id: 'pro',

price: '$49',

description: 'For teams building production agent workflows.',

'Unlimited deployed agents',

'Full A2A orchestration engine',

'Priority support (< 4hr SLA)',

'CI/CD integration',

highlighted: true,

ctaHref: '/signup?plan=pro',

{

name: 'Enterprise',

priceSubtext: 'contact sales',

features: [

'Dedicated infrastructure',

'SSO / SAML / SCIM',

'Dedicated solutions engineer',

],

ctaText: 'Contact Sales',

},
```

FILE: src/data/landing/faq.ts

```
import type { FAQItem } from '@/types/landing';



{

question: 'What is ClawStak.ai?',

'ClawStak.ai is a developer platform for discovering, deploying, and orchestrating ' +

'and a transparent trust scoring system.',

{

question: 'What is the A2A protocol?',

"A2A (Agent-to-Agent) is a communication protocol that lets AI agents discover each other's " +

'ClawStak provides a first-class implementation.',

{

question: 'How does trust scoring work?',

'Each agent receives a composite trust score derived from runtime behavior analysis, ' +

'fully transparent and queryable via our API.',

{
```

```
question: 'What are the limits on the Free tier?',

'The Free tier supports up to 3 deployed agents, community marketplace access, ' +

},

id: 'enterprise-compliance',

answer:

'and optional on-chain trust attestation for auditability.',

{

question: 'Can I self-host ClawStak?',

'Enterprise customers can deploy ClawStak on dedicated infrastructure within their ' +

},
```

FILE: src/data/landing/socialProof.ts

```
import type { SocialProofData } from '@/types/landing';



metrics: [

{ value: '2,500+', label: 'Development Teams' },

{ value: '<200ms', label: 'Median A2A Latency' },

testimonials: [

id: 'testimonial-1',

'ClawStak replaced three internal services and gave us a trust layer we never could have built ourse

role: 'Head of AI Infrastructure',

avatarUrl: '/avatars/sarah-chen.webp',

{

quote:

author: 'Marcus Rodriguez',

company: 'Agentic Labs',

},

id: 'testimonial-3',

'Trust scoring gave our compliance team the auditability they needed. We went from "no AI agents in

role: 'VP of Engineering',

avatarUrl: '/avatars/priya-kapoor.webp',

],

'/logos/meridian-systems.svg',

'/logos/finsecure.svg',

'/logos/synthcore.svg',

],
```

FILE: src/utils/structuredData.ts

```
import type { FAQItem, PricingTier } from '@/types/landing';


* Generates a JSON-LD FAQPage schema from FAQ items.

* @see https://schema.org/FAQPage

*/

if (!items.length) {

}


'@context': 'https://schema.org',

mainEntity: items.map((item) => ({

name: item.question,

'@type': 'Answer',

},

};


* Generates a JSON-LD SoftwareApplication schema with pricing offers.

* @see https://schema.org/SoftwareApplication

*/

if (!tiers.length) {

}


'@context': 'https://schema.org',

name: 'ClawStak.ai',

operatingSystem: 'Web',

.filter((tier) => tier.price !== 'Custom')

'@type': 'Offer',

price: tier.price.replace('$', ''),

description: tier.description,

};


* Generates a JSON-LD Organization schema for brand identity in search.

*/

return {

'@type': 'Organization',

url: 'https://clawstak.ai',

sameAs: [

'https://twitter.com/clawstak',
};

* Generates a JSON-LD WebSite schema with SearchAction for sitelinks searchbox.
*/
return {
'@type': 'WebSite',
url: 'https://clawstak.ai',
'@type': 'SearchAction',
'@type': 'EntryPoint',
},
},
}
```

FILE: src/utils/structuredData.test.ts

```
import { describe, it, expect } from 'vitest';
buildFAQSchema,
buildOrganizationSchema,
} from './structuredData';

it('returns empty object for empty array', () => {
});

const items: FAQItem[] = [
{ id: '2', question: 'How does it work?', answer: 'Like magic.' },


expect(schema['@type']).toBe('FAQPage');

expect(entities[0]).toEqual({
name: 'What is this?',
'@type': 'Answer',
},
expect(entities[1]).toEqual({
name: 'How does it work?',
'@type': 'Answer',
},
});
```

```
const items: FAQItem[] = [

];


const entities = schema.mainEntity as Array<Record<string, unknown>>;

expect.objectContaining({ name: 'Does it support "A2A"?' }),

});


const tiers: PricingTier[] = [

id: 'free',

price: '$0',

description: 'Free tier',

highlighted: false,

ctaHref: '/signup',

{

name: 'Pro',

priceSubtext: 'per month',

features: ['Feature B'],

ctaText: 'Start Trial',

},

id: 'enterprise',

price: 'Custom',

description: 'Enterprise tier',

highlighted: false,

ctaHref: '/contact',

];


expect(buildProductSchema([])).toEqual({});


const schema = buildProductSchema(tiers);


expect(schema['@type']).toBe('SoftwareApplication');

expect(schema.applicationCategory).toBe('DeveloperApplication');


const schema = buildProductSchema(tiers);


expect(offers.map((o) => o.name)).toEqual(['Free', 'Pro']);
```

```
const schema = buildProductSchema(tiers);


expect(offers[1].price).toBe('49');


const schema = buildProductSchema(tiers);


expect(offer.priceCurrency).toBe('USD');

});


it('generates valid Organization schema', () => {


expect(schema['@type']).toBe('Organization');

expect(schema.url).toBe('https://clawstak.ai');

expect(schema.sameAs).toEqual(

'https://github.com/clawstak',

]),

});


it('generates valid WebSite schema with SearchAction', () => {


expect(schema['@type']).toBe('WebSite');


expect(action['@type']).toBe('SearchAction');

});
```

FILE: src/hooks/useFAQSchema.ts

```
import { useMemo } from 'react';

import { buildFAQSchema } from '@/utils/structuredData';


* React hook that memoizes FAQPage JSON-LD schema generation.

*

*/

return useMemo(() => {

return JSON.stringify(schema);

}
```

FILE: src/styles/landing/landing.module.css

```
/* ============================================================

Dark developer-friendly palette with CSS custom properties.
```

```
runtime overhead -- critical for LCP optimization.

============================================================ */


--color-bg: #0a0a0f;

--color-bg-card: #1a1a2e;

--color-surface: #222240;

--color-border-highlighted: #6c63ff;


--color-text-secondary: #a0a0b8;


--color-accent-hover: #7b73ff;



--font-mono: 'JetBrains Mono', 'Fira Code', 'Consolas', monospace;


--space-sm: 0.75rem;

--space-lg: 1.5rem;

--space-2xl: 3rem;

--space-4xl: 6rem;


--radius-md: 10px;


--max-width-narrow: 800px;


--transition-base: 250ms ease;


GLOBAL LAYOUT


background-color: var(--color-bg);

font-family: var(--font-sans);

overflow-x: hidden;


max-width: var(--max-width);

padding: 0 var(--space-lg);


font-size: clamp(1.75rem, 4vw, 2.5rem);
```

```
color: var(--color-text-primary);

margin-bottom: var(--space-3xl);

}


HERO SECTION


padding: var(--space-4xl) 0;

min-height: 70vh;

align-items: center;

background: radial-gradient(

var(--color-accent-subtle) 0%,

);


max-width: var(--max-width-narrow);

padding: 0 var(--space-lg);


font-size: clamp(2.5rem, 6vw, 4rem);

line-height: 1.1;

margin-bottom: var(--space-lg);

-webkit-background-clip: text;

background-clip: text;


font-size: clamp(1.1rem, 2vw, 1.35rem);

line-height: 1.7;

max-width: 640px;

margin-right: auto;


display: flex;

justify-content: center;

}


BUTTONS


.btnSecondary {

align-items: center;

padding: var(--space-sm) var(--space-xl);

font-size: 1rem;
```

```
text-decoration: none;

cursor: pointer;

line-height: 1.4;


background-color: var(--color-accent);

}


background-color: var(--color-accent-hover);

box-shadow: 0 4px 20px rgba(108, 99, 255, 0.35);


outline: 2px solid var(--color-accent);

}


background-color: transparent;

border: 1px solid var(--color-border);


border-color: var(--color-accent);

transform: translateY(-1px);


outline: 2px solid var(--color-accent);

}


FEATURE SECTIONS


padding: var(--space-4xl) 0;

}


border-bottom: none;


max-width: var(--max-width);

padding: 0 var(--space-lg);

grid-template-columns: 1fr 1fr;

align-items: center;


direction: rtl;


direction: ltr;
```

```
direction: ltr;


display: flex;

gap: var(--space-md);


font-size: clamp(1.5rem, 3vw, 2rem);

color: var(--color-text-primary);

}


font-size: 1.1rem;

font-weight: 500;


font-size: 1rem;

line-height: 1.7;


list-style: none;

margin: var(--space-md) 0;

flex-direction: column;

}


display: flex;

gap: var(--space-sm);

font-size: 0.95rem;

}


content: '[x]';

font-weight: 700;

margin-top: 1px;


display: flex;

justify-content: center;

border-radius: var(--radius-lg);

min-height: 320px;

}


width: 80px;
```

```
  border-radius: 50%;

  border: 2px solid var(--color-border);


  SOCIAL PROOF


  padding: var(--space-4xl) 0;

  }


  max-width: var(--max-width);

  padding: 0 var(--space-lg);


  display: grid;
```

import { describe, it, expectTypeOf } from 'vitest';

Feature,

FAQItem,

SocialProofMetric,

SEOMetadata,

} from '@/types/landing';

describe('Landing page types', () => {

it('has the correct shape', () => {

id: string;

headline: string;

description: string;

icon: string;

ctaHref: string;

});

it('does not allow extra properties implicitly via assignment', () => {

```
id: '1',

headline: 'h',

description: 'd',

icon: 'i',

ctaHref: '/cta',

expectTypeOf(feature).toEqualTypeOf<Feature>();

});
describe('PricingTier', () => {

expectTypeOf<PricingTier['name']>().toEqualTypeOf<'Free' | 'Pro' | 'Enterprise'>();


it('makes priceSubtext optional', () => {

});
it('has highlighted as boolean', () => {

});
it('has features as string array', () => {

});


describe('FAQItem', () => {

expectTypeOf<FAQItem>().toMatchObjectType<{

question: string;

}>();

});
describe('Testimonial', () => {

expectTypeOf<Testimonial['avatarUrl']>().toEqualTypeOf<string | undefined>();


it('requires author, role, and company', () => {

id: string;

author: string;

company: string;

});
```

```
describe('SocialProofMetric', () => {

expectTypeOf<SocialProofMetric>().toEqualTypeOf<{

label: string;

});

describe('SocialProofData', () => {

expectTypeOf<SocialProofData>().toMatchObjectType<{

testimonials: Testimonial[];

}>();

});
describe('SEOMetadata', () => {

expectTypeOf<SEOMetadata['keywords']>().toEqualTypeOf<string[]>();

it('has the correct shape', () => {

title: string;

canonicalUrl: string;

keywords: string[];

});

describe('CTALink', () => {

expectTypeOf<CTALink>().toEqualTypeOf<{

href: string;

});

});


import { describe, it, expect } from 'vitest';

import type { SEOMetadata } from '@/types/landing';
describe('seoMetadata', () => {

const metadata: SEOMetadata = seoMetadata;
```

```
});
it('has a non-empty title', () => {

expect(seoMetadata.title.length).toBeGreaterThan(0);


it('title contains the brand name', () => {

});
it('has a non-empty description', () => {

expect(seoMetadata.description.length).toBeGreaterThan(0);


it('description is within reasonable SEO length (under 300 chars)', () => {

});
it('has a valid canonical URL', () => {

});
it('has a valid ogImage URL', () => {

expect(seoMetadata.ogImage).toMatch(/\.(png|jpg|jpeg|webp)$/);


it('has at least one keyword', () => {

});
it('keywords are all non-empty strings', () => {

expect(typeof keyword).toBe('string');

});


it('keywords contain relevant terms', () => {

expect(allKeywords).toContain('ai');

});
it('has no duplicate keywords', () => {

const unique = new Set(lowercased);

});



import { describe, it, expect } from 'vitest';

import type { Feature } from '@/types/landing';
describe('features', () => {
```

```
expect(Array.isArray(features)).toBe(true);

});
it('all items conform to the Feature interface', () => {

expect(feature).toHaveProperty('id');

expect(feature).toHaveProperty('headline');

expect(feature).toHaveProperty('description');

expect(feature).toHaveProperty('icon');

expect(feature).toHaveProperty('ctaHref');

});
it('all features have unique ids', () => {

expect(new Set(ids).size).toBe(ids.length);

it('all features have unique slugs', () => {

expect(new Set(slugs).size).toBe(slugs.length);

it('all ctaHref values start with /', () => {

expect(feature.ctaHref).toMatch(/^\//);

});
it('all features have at least one bullet point', () => {

expect(feature.bulletPoints.length).toBeGreaterThan(0);

});
it('all bullet points are non-empty strings', () => {

feature.bulletPoints.forEach((bp) => {

expect(bp.trim().length).toBeGreaterThan(0);

});

it('contains the marketplace feature', () => {

expect(marketplace).toBeDefined();

});
it('contains the a2a feature', () => {
```

```
expect(a2a).toBeDefined();

});
it('contains the trust-scoring feature', () => {

expect(trust).toBeDefined();

});
it('headlines are non-empty and reasonably short', () => {

expect(feature.headline.length).toBeGreaterThan(0);

});

it('descriptions are non-empty', () => {

expect(feature.description.trim().length).toBeGreaterThan(0);

});
it('ctaText values are non-empty', () => {

expect(feature.ctaText.trim().length).toBeGreaterThan(0);

});
it('icon values are non-empty strings', () => {

expect(typeof feature.icon).toBe('string');

});

});


import { describe, it, expect } from 'vitest';

import type { PricingTier } from '@/types/landing';
describe('pricingTiers', () => {

expect(Array.isArray(pricingTiers)).toBe(true);

});
it('contains exactly three tiers', () => {

});
it('all items conform to the PricingTier interface', () => {

expect(tier).toHaveProperty('id');
expect(tier).toHaveProperty('price');

expect(tier).toHaveProperty('features');

expect(tier).toHaveProperty('ctaText');

});

it('all tiers have unique ids', () => {

expect(new Set(ids).size).toBe(ids.length);

it('tier names are Free, Pro, and Enterprise in order', () => {

});
it('exactly one tier is highlighted', () => {

expect(highlighted).toHaveLength(1);

it('Pro tier is the highlighted tier', () => {

expect(pro).toBeDefined();

});
it('Free tier has price $0', () => {

expect(free).toBeDefined();

});
it('Enterprise tier has Custom pricing', () => {

expect(enterprise).toBeDefined();

});
it('all tiers have at least one feature', () => {

expect(tier.features.length).toBeGreaterThan(0);

});
it('all features are non-empty strings', () => {

tier.features.forEach((feature) => {

expect(feature.trim().length).toBeGreaterThan(0);

});

it('Pro tier has more features than Free tier', () => {
```

```
const pro = pricingTiers.find((t) => t.name === 'Pro')!;

});
it('all ctaHref values start with /', () => {

expect(tier.ctaHref).toMatch(/^\//);

});
it('ctaHref contains plan parameter for non-contact tiers', () => {

const pro = pricingTiers.find((t) => t.name === 'Pro')!;

expect(pro.ctaHref).toContain('plan=pro');


it('Enterprise ctaHref points to contact page', () => {

expect(enterprise.ctaHref).toContain('/contact');


it('all tiers have non-empty descriptions', () => {

expect(tier.description.trim().length).toBeGreaterThan(0);

});
it('priceSubtext is defined for all tiers', () => {

expect(tier.priceSubtext).toBeDefined();

});

});


import { describe, it, expect } from 'vitest';

import type { FAQItem } from '@/types/landing';
describe('faqItems', () => {

expect(Array.isArray(faqItems)).toBe(true);

});
it('all items conform to the FAQItem interface', () => {

expect(item).toHaveProperty('id');

expect(item).toHaveProperty('answer');

});
it('all items have unique ids', () => {
```

```
expect(new Set(ids).size).toBe(ids.length);

it('all questions end with a question mark', () => {

expect(item.question.trim()).toMatch(/\?$/);

});
it('all questions are non-empty', () => {

expect(item.question.trim().length).toBeGreaterThan(0);

});
it('all answers are non-empty', () => {

expect(item.answer.trim().length).toBeGreaterThan(0);

});
it('answers are substantive (at least 20 characters)', () => {

expect(item.answer.length).toBeGreaterThanOrEqual(20);

});
it('contains a "what is ClawStak" FAQ', () => {

expect(found).toBeDefined();

});
it('contains an A2A protocol FAQ', () => {

expect(found).toBeDefined();

});
it('contains a trust scoring FAQ', () => {

expect(found).toBeDefined();

it('contains a free tier limits FAQ', () => {

expect(found).toBeDefined();

});
it('contains enterprise compliance FAQ', () => {

expect(found).toBeDefined();

});
it('contains self-host FAQ', () => {
```

```
expect(found).toBeDefined();

it('ids use kebab-case', () => {

expect(item.id).toMatch(/^[a-z0-9]+(-[a-z0-9]+)*$/);

});


import { describe, it, expect } from 'vitest';

import type { SocialProofData, Testimonial, SocialProofMetric } from '@/types/landing';
describe('socialProof', () => {

const data: SocialProofData = socialProof;

});
describe('metrics', () => {

expect(socialProof.metrics.length).toBeGreaterThan(0);

it('all metrics have value and label', () => {

expect(metric.value.trim().length).toBeGreaterThan(0);

});

it('contains agents deployed metric', () => {

m.label.toLowerCase().includes('agents deployed'),

expect(found).toBeDefined();

it('contains uptime metric', () => {

m.label.toLowerCase().includes('uptime'),

expect(found).toBeDefined();

});
it('contains latency metric', () => {

m.label.toLowerCase().includes('latency'),

expect(found).toBeDefined();

});
it('has exactly 4 metrics', () => {
});

describe('testimonials', () => {

expect(socialProof.testimonials.length).toBeGreaterThan(0);

it('all testimonials have required fields', () => {

expect(testimonial.id).toBeTruthy();

expect(testimonial.author.trim().length).toBeGreaterThan(0);

expect(testimonial.company.trim().length).toBeGreaterThan(0);

});
it('all testimonials have unique ids', () => {

expect(new Set(ids).size).toBe(ids.length);

it('all testimonials have avatarUrl set', () => {

expect(testimonial.avatarUrl).toBeDefined();

});

it('quotes are substantive (at least 50 characters)', () => {

expect(testimonial.quote.length).toBeGreaterThanOrEqual(50);

});
it('has exactly 3 testimonials', () => {

});

describe('logoUrls', () => {

expect(socialProof.logoUrls.length).toBeGreaterThan(0);

it('all logos are SVG files', () => {

expect(url).toMatch(/\.svg$/);

});
it('all logos are under /logos/ path', () => {

expect(url).toMatch(/^\/logos\//);

});
```

```
},

it('has no duplicate logo URLs', () => {

expect(unique.size).toBe(socialProof.logoUrls.length);

it('has at least as many logos as testimonials', () => {

socialProof.testimonials.length,

});

});


import { describe, it, expect } from 'vitest';

buildFAQSchema,

buildOrganizationSchema,

} from '@/utils/structuredData';

/* ------------------------------------------------------------------------

  - ------------------------------------------------------------------------ */
describe('buildFAQSchema', () => {

expect(buildFAQSchema([])).toEqual({});

it('generates valid FAQPage schema for a single item', () => {

{ id: '1', question: 'Q1?', answer: 'A1.' },

const schema = buildFAQSchema(items);
expect(schema['@context']).toBe('https://schema.org');

const entities = schema.mainEntity as Array<Record<string, unknown>>;

expect(entities[0]).toEqual({

name: 'Q1?',

'@type': 'Answer',

},

});
it('generates valid FAQPage schema for multiple items', () => {
```

```
{ id: '1', question: 'What is this?', answer: 'A test.' },

{ id: '3', question: 'Why use it?', answer: 'Because.' },

const schema = buildFAQSchema(items);

expect(entities).toHaveLength(3);

expect(entities[1]).toMatchObject({ name: 'How does it work?' });

});
it('preserves special characters in questions and answers', () => {

{ id: '1', question: 'Does it support "A2A" & <HTML>?', answer: "Yes, it's built-in & ready." },

const schema = buildFAQSchema(items);

expect(entities[0]).toEqual(

);

expect(answer).toBe("Yes, it's built-in & ready.");

it('preserves very long answer text', () => {

const items: FAQItem[] = [

];
const schema = buildFAQSchema(items);

const answer = (entities[0].acceptedAnswer as Record<string, unknown>).text;

});
it('preserves unicode characters', () => {

{ id: '1', question: '', answer: ' ' },

const schema = buildFAQSchema(items);

expect(entities[0]).toMatchObject({ name: '' });

it('preserves item order', () => {

{ id: 'c', question: 'C?', answer: 'C.' },

{ id: 'b', question: 'B?', answer: 'B.' },
const schema = buildFAQSchema(items);

expect(entities.map((e) => e.name)).toEqual(['C?', 'A?', 'B?']);

it('does not include id in the output schema', () => {

{ id: 'test-id', question: 'Q?', answer: 'A.' },

const schema = buildFAQSchema(items);

expect(json).not.toContain('test-id');

it('output is valid JSON-serializable', () => {

{ id: '1', question: 'Q?', answer: 'A.' },

const schema = buildFAQSchema(items);

const parsed = JSON.parse(JSON.stringify(schema));

});

/* -------------------------------------------------------------------------

  - ------------------------------------------------------------------------- */
describe('buildProductSchema', () => {

{

name: 'Free',

priceSubtext: 'forever',

features: ['Feature A'],

ctaText: 'Get Started',

},

id: 'pro',

price: '$49',

description: 'Pro tier',

highlighted: true,

ctaHref: '/signup?plan=pro',
{

name: 'Enterprise',

priceSubtext: 'contact sales',

features: ['Feature C'],

ctaText: 'Contact Sales',

},

it('returns empty object for empty array', () => {

});
it('generates valid SoftwareApplication schema', () => {

expect(schema['@context']).toBe('https://schema.org');

expect(schema.name).toBe('ClawStak.ai');

expect(schema.operatingSystem).toBe('Web');

it('excludes Custom-priced tiers from offers', () => {

const offers = schema.offers as Array<Record<string, unknown>>;
expect(offers).toHaveLength(2);

expect(names).toContain('Free');

expect(names).not.toContain('Enterprise');

it('strips dollar sign from price values', () => {

const offers = schema.offers as Array<Record<string, unknown>>;
expect(offers[0].price).toBe('0');
```