# Technical Implementation Guide - Dr Julia Crawford ENT Practice Website

## Executive Summary

**Implementation Strategy:** Phased development approach with medical compliance integration

**Technical Standards:** Modern web development with accessibility, performance, and SEO optimisation

**Compliance Framework:** TGA medical advertising guidelines with WCAG 2.1 AA accessibility standards

**Development Timeline:** 12-week implementation with iterative testing and quality assurance

## Table of Contents

1. [Development Environment Setup](#development-environment-setup)
2. [Technical Architecture Requirements](#technical-architecture-requirements)
3. [Frontend Development Specifications](#frontend-development-specifications)
4. [Backend Infrastructure Requirements](#backend-infrastructure-requirements)
5. [Content Management System Configuration](#content-management-system-configuration)
6. [Medical Compliance Technical Integration](#medical-compliance-technical-integration)
7. [Performance Optimisation Implementation](#performance-optimisation-implementation)
8. [Security and Privacy Technical Requirements](#security-and-privacy-technical-requirements)
9. [SEO and Analytics Implementation](#seo-and-analytics-implementation)

10. [Testing and Quality Assurance Framework](#testing-and-quality-assurance-framework)

11. [Deployment and Launch Strategy](#deployment-and-launch-strategy)

12. [Ongoing Maintenance and Updates](#ongoing-maintenance-and-updates)

## Development Environment Setup

### 🛠️ Required Development Tools

#### Core Development Stack

`json

{

"frontend": {

"framework": "React 18.x or Vue 3.x",

"bundler": "Vite or Webpack 5",

"css\_framework": "Tailwind CSS 3.x",

"ui\_components": "Headless UI or Radix UI",

"animations": "Framer Motion or Vue Transition"

},

"backend": {

"runtime": "Node.js 18+ LTS",

"cms": "Strapi 4.x or Sanity",

"database": "PostgreSQL 14+ or MongoDB 6+",

"api": "REST with GraphQL optional",

"hosting": "Vercel, Netlify, or AWS"

},

"development": {

"package\_manager": "pnpm or yarn",

"bundler": "Vite",

"linting": "ESLint + Prettier",

"testing": "Vitest + Testing Library",

"git\_hooks": "Husky + lint-staged"

}

}

#### Development Environment Configuration

`bash

# Project Setup Commands

npx create-react-app drjuliacrawford-website --template typescript

# OR

npm create vue@latest drjuliacrawford-website -- --typescript

# Core Dependencies Installation

npm install -D tailwindcss postcss autoprefixer

npm install framer-motion react-helmet-async

npm install @heroicons/react @headlessui/react

npm install react-router-dom react-hook-form

npm install axios react-query

# Development Tools

npm install -D eslint prettier husky lint-staged

npm install -D @testing-library/react @testing-library/jest-dom

npm install -D cypress axe-core @axe-core/react

#### Environment Variables Configuration

`env

# Environment Configuration (.env)

REACT\_APP\_SITE\_URL=https://drjuliacrawford.com.au

REACT\_APP\_API\_URL=https://api.drjuliacrawford.com.au

REACT\_APP\_GOOGLE\_ANALYTICS\_ID=GA\_MEASUREMENT\_ID

REACT\_APP\_GOOGLE\_MAPS\_API\_KEY=YOUR\_MAPS\_API\_KEY

REACT\_APP\_BOOKING\_SYSTEM\_URL=https://booking.drjuliacrawford.com.au

# Development

NODE\_ENV=development

HTTPS=true

PORT=3000

# Medical Compliance

REACT\_APP\_TGA\_COMPLIANCE\_MODE=true

REACT\_APP\_MEDICAL\_DISCLAIMER\_REQUIRED=true

REACT\_APP\_PRIVACY\_POLICY\_VERSION=2025.1

### Project Structure Organisation

drjuliacrawford-website/

├── public/

│ ├── images/

│ │ ├── hero/

│ │ ├── services/

│ │ ├── about/

│ │ └── procedures/

│ ├── icons/

│ └── manifest.json

├── src/

│ ├── components/

│ │ ├── common/

│ │ │ ├── Header/

│ │ │ ├── Footer/

│ │ │ ├── Navigation/

│ │ │ └── Buttons/

│ │ ├── medical/

│ │ │ ├── Disclaimer/

│ │ │ ├── Compliance/

│ │ │ └── PatientForms/

│ │ ├── sections/

│ │ │ ├── Hero/

│ │ │ ├── Services/

│ │ │ ├── About/

│ │ │ └── Contact/

│ │ └── layout/

│ │ ├── PageLayout/

│ │ ├── ContentLayout/

│ │ └── MobileLayout/

│ ├── pages/

│ │ ├── Home/

│ │ ├── About/

│ │ ├── Services/

│ │ │ ├── RoboticSurgery/

│ │ │ ├── SleepApnoea/

│ │ │ ├── HeadNeckCancer/

│ │ │ └── PaediatricENT/

│ │ ├── Resources/

│ │ ├── Contact/

│ │ └── Legal/

│ ├── hooks/

│ ├── utils/

│ ├── types/

│ └── styles/

├── content/

│ ├── pages/

│ ├── procedures/

│ └── resources/

├── tests/

│ ├── unit/

│ ├── integration/

│ └── e2e/

└── docs/

├── api/

├── deployment/

└── maintenance/

## Technical Architecture Requirements

### 🏗️ System Architecture Overview

#### Frontend Architecture Pattern

`typescript

// Component Architecture Example

interface MedicalPageProps {

content: MedicalContent;

complianceRequired: boolean;

patientJourneyStage: 'awareness' | 'consideration' | 'decision';

}

const MedicalPage: React.FC<MedicalPageProps> = ({

content,

complianceRequired,

patientJourneyStage

}) => {

const { trackPageView } = useAnalytics();

const { showDisclaimer } = useMedicalCompliance();

useEffect(() => {

trackPageView(content.title, patientJourneyStage);

if (complianceRequired) {

showDisclaimer(content.disclaimerType);

}

}, [content, patientJourneyStage]);

return (

<PageLayout>

<SEOHead content={content} />

{complianceRequired && <MedicalDisclaimer />}

<ContentRenderer content={content} />

<PatientJourneyNavigation stage={patientJourneyStage} />

</PageLayout>

);

};

#### State Management Architecture

`typescript

// Context-based State Management

interface AppContextType {

user: PatientUser | null;

preferences: UserPreferences;

compliance: ComplianceState;

navigation: NavigationState;

}

const AppContext = createContext<AppContextType | null>(null);

// Medical Compliance Context

interface ComplianceContextType {

disclaimerShown: boolean;

consentGiven: boolean;

privacyAccepted: boolean;

showMedicalDisclaimer: (type: DisclaimerType) => void;

recordConsent: (type: ConsentType) => void;

}

const ComplianceContext = createContext<ComplianceContextType | null>(null);

// Custom Hooks for Medical Compliance

export const useMedicalCompliance = () => {

const context = useContext(ComplianceContext);

if (!context) {

throw new Error('useMedicalCompliance must be used within ComplianceProvider');

}

return context;

};

### Backend API Architecture

#### API Endpoint Structure

`typescript

// API Routes Configuration

const apiRoutes = {

// Content Management

'/api/content/pages/:slug': 'GET',

'/api/content/procedures/:id': 'GET',

'/api/content/resources/:category': 'GET',

// Patient Information

'/api/patients/contact': 'POST',

'/api/patients/booking': 'POST',

'/api/patients/resources/download': 'GET',

// Medical Compliance

'/api/compliance/disclaimer/:type': 'GET',

'/api/compliance/consent': 'POST',

'/api/compliance/privacy': 'POST',

// Analytics and Tracking

'/api/analytics/page-view': 'POST',

'/api/analytics/conversion': 'POST',

'/api/analytics/user-journey': 'POST'

};

// Medical Content API Response Type

interface MedicalContentResponse {

id: string;

title: string;

content: string;

medicalReviewDate: Date;

complianceLevel: 'standard' | 'medical' | 'surgical';

disclaimerRequired: boolean;

evidenceSources: EvidenceSource[];

lastReviewed: Date;

reviewedBy: string;

tgaCompliant: boolean;

}

// Evidence Source Type

interface EvidenceSource {

title: string;

url: string;

publicationDate: Date;

sourceType: 'journal' | 'government' | 'professional\_body';

credibilityScore: number;

}

## Frontend Development Specifications

### 🎨 Component Development Standards

#### Responsive Design Component Framework

`typescript

// Responsive Layout Hook

const useResponsiveLayout = () => {

const [screenSize, setScreenSize] = useState<'mobile' | 'tablet' | 'desktop'>('desktop');

useEffect(() => {

const handleResize = () => {

if (window.innerWidth < 768) setScreenSize('mobile');

else if (window.innerWidth < 1024) setScreenSize('tablet');

else setScreenSize('desktop');

};

handleResize();

window.addEventListener('resize', handleResize);

return () => window.removeEventListener('resize', handleResize);

}, []);

return screenSize;

};

// Responsive Component Example

const Hero: React.FC<HeroProps> = ({ title, subtitle, image, ctaButtons }) => {

const screenSize = useResponsiveLayout();

const layoutConfig = {

mobile: {

direction: 'column',

imageSize: { width: '100%', height: '300px' },

textAlign: 'center' as const,

spacing: '1rem'

},

tablet: {

direction: 'column',

imageSize: { width: '100%', height: '400px' },

textAlign: 'center' as const,

spacing: '2rem'

},

desktop: {

direction: 'row',

imageSize: { width: '50%', height: '600px' },

textAlign: 'left' as const,

spacing: '4rem'

}

};

const config = layoutConfig[screenSize];

return (

<section className="hero-section">

<div

className="hero-container"

style={{

flexDirection: config.direction,

gap: config.spacing,

textAlign: config.textAlign

}}

>

<div className="hero-content">

<h1 className="hero-title">{title}</h1>

<p className="hero-subtitle">{subtitle}</p>

<div className="hero-actions">

{ctaButtons.map((button, index) => (

<Button key={index} {...button} />

))}

</div>

</div>

<div className="hero-image" style={config.imageSize}>

<OptimisedImage {...image} />

</div>

</div>

</section>

);

};

#### Medical Compliance Components

`typescript

// Medical Disclaimer Component

interface MedicalDisclaimerProps {

type: 'general' | 'procedure' | 'emergency';

position: 'banner' | 'inline' | 'modal';

required: boolean;

}

const MedicalDisclaimer: React.FC<MedicalDisclaimerProps> = ({

type,

position,

required

}) => {

const [acknowledged, setAcknowledged] = useState(false);

const { recordConsent } = useMedicalCompliance();

const disclaimerContent = {

general: "This information is for educational purposes only and should not replace professional medical advice.",

procedure: "Individual results may vary. Dr Crawford will discuss specific risks, benefits, and outcomes during your consultation.",

emergency: "This website does not provide emergency medical advice. For urgent ENT concerns, contact emergency services or visit your nearest emergency department."

};

const handleAcknowledge = () => {

setAcknowledged(true);

recordConsent(disclaimer\_${type});

};

if (position === 'banner') {

return (

<div className="medical-disclaimer-banner bg-blue-50 border border-blue-200 px-4 py-3">

<div className="flex items-center justify-between">

<div className="flex items-start">

<AlertTriangle className="h-5 w-5 text-blue-500 mr-3 mt-0.5" />

<div>

<p className="text-sm font-medium text-blue-800">Medical Disclaimer</p>

<p className="text-sm text-blue-700">{disclaimerContent[type]}</p>

</div>

</div>

{required && !acknowledged && (

<button

onClick={handleAcknowledge}

className="ml-4 bg-blue-600 text-white px-3 py-1 rounded text-sm hover:bg-blue-700"

>

Acknowledge

</button>

)}

</div>

</div>

);

}

return (

<div className="medical-disclaimer-inline p-4 bg-gray-50 border-l-4 border-blue-500">

<p className="text-sm text-gray-700">

<strong>Medical Disclaimer:</strong> {disclaimerContent[type]}

</p>

</div>

);

};

// Patient Form Component with Compliance

const PatientContactForm: React.FC = () => {

const { register, handleSubmit, formState: { errors } } = useForm<PatientFormData>();

const [privacyConsent, setPrivacyConsent] = useState(false);

const { recordConsent } = useMedicalCompliance();

const onSubmit = async (data: PatientFormData) => {

if (!privacyConsent) {

alert('Please accept the privacy policy to continue.');

return;

}

recordConsent('contact\_form\_privacy');

// Submit form data with encryption

await submitPatientForm({

...data,

consentRecorded: true,

submissionTime: new Date().toISOString()

});

};

return (

<form onSubmit={handleSubmit(onSubmit)} className="patient-contact-form">

<MedicalDisclaimer type="general" position="inline" required />

<div className="form-group">

<label htmlFor="name">Full Name \*</label>

<input

id="name"

{...register('name', { required: 'Name is required' })}

autoComplete="name"

aria-describedby="name-help"

/>

<div id="name-help" className="help-text">

Your name will only be used for appointment scheduling

</div>

{errors.name && <span className="error">{errors.name.message}</span>}

</div>

<div className="privacy-consent">

<label className="checkbox-label">

<input

type="checkbox"

checked={privacyConsent}

onChange={(e) => setPrivacyConsent(e.target.checked)}

required

/>

I consent to my information being used for appointment scheduling

and practice communication.

<a href="/privacy-policy/" target="\_blank" className="privacy-link">

Read full privacy policy

</a>

</label>

</div>

<button

type="submit"

disabled={!privacyConsent}

className="submit-button"

>

Send Message

</button>

</form>

);

};

### CSS Framework Configuration

#### Tailwind CSS Custom Configuration

`javascript

// tailwind.config.js

module.exports = {

content: ['./src//\*.{js,ts,jsx,tsx}'],

theme: {

extend: {

colors: {

medical: {

primary: '#2563eb',

secondary: '#1d4ed8',

accent: '#3b82f6',

success: '#10b981',

warning: '#f59e0b',

error: '#ef4444',

light: '#f8fafc',

dark: '#1f2937'

},

practice: {

blue: '#2563eb',

darkBlue: '#1e40af',

lightBlue: '#dbeafe',

gray: {

50: '#f9fafb',

100: '#f3f4f6',

200: '#e5e7eb',

300: '#d1d5db',

400: '#9ca3af',

500: '#6b7280',

600: '#4b5563',

700: '#374151',

800: '#1f2937',

900: '#111827'

}

}

},

fontFamily: {

sans: ['Inter', 'system-ui', 'sans-serif'],

medical: ['Source Sans Pro', 'system-ui', 'sans-serif']

},

spacing: {

'18': '4.5rem',

'88': '22rem',

'128': '32rem'

},

borderRadius: {

'xl': '1rem',

'2xl': '1.5rem',

'3xl': '2rem'

},

boxShadow: {

'medical': '0 4px 20px rgba(37, 99, 235, 0.1)',

'hover': '0 20px 40px rgba(0, 0, 0, 0.1)',

'focus': '0 0 0 3px rgba(37, 99, 235, 0.1)'

},

animation: {

'fade-in': 'fadeIn 0.5s ease-in-out',

'slide-up': 'slideUp 0.6s ease-out',

'scale-in': 'scaleIn 0.4s ease-out'

}

}

},

plugins: [

require('@tailwindcss/forms'),

require('@tailwindcss/typography'),

require('@tailwindcss/aspect-ratio')

]

};

#### Custom CSS Components

`css

/ *Custom Medical Practice Styles* /

@layer components {

.btn-medical {

@apply bg-medical-primary text-white px-6 py-3 rounded-lg font-semibold

transition-all duration-300 hover:bg-medical-secondary

focus:outline-none focus:ring-4 focus:ring-medical-primary/20;

}

.btn-medical-secondary {

@apply bg-transparent text-medical-primary border-2 border-medical-primary

px-6 py-3 rounded-lg font-semibold transition-all duration-300

hover:bg-medical-primary hover:text-white;

}

.medical-card {

@apply bg-white rounded-xl shadow-medical border border-practice-gray-200

transition-all duration-300 hover:shadow-hover hover:-translate-y-1;

}

.medical-disclaimer {

@apply bg-blue-50 border border-blue-200 rounded-lg p-4 text-sm text-blue-800;

}

.service-hero {

@apply bg-gradient-to-br from-practice-blue to-medical-secondary

text-white relative overflow-hidden;

}

.content-section {

@apply py-16 lg:py-24;

}

.container-medical {

@apply max-w-7xl mx-auto px-4 sm:px-6 lg:px-8;

}

/ *Mobile-Optimised Touch Targets* /

.touch-target {

@apply min-h-[44px] min-w-[44px] flex items-center justify-center;

}

/ *Accessibility Improvements* /

.sr-only {

@apply absolute w-px h-px p-0 -m-px overflow-hidden whitespace-nowrap border-0;

}

.skip-link {

@apply absolute top-0 left-0 bg-medical-primary text-white px-4 py-2

transform -translate-y-full focus:translate-y-0 transition-transform;

}

/ *Print Styles for Medical Information* /

@media print {

.no-print {

@apply hidden;

}

.medical-content {

@apply text-black bg-white;

}

.page-break {

page-break-before: always;

}

}

}

/ *Custom Animation Keyframes* /

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

@keyframes slideUp {

from {

opacity: 0;

transform: translateY(30px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

@keyframes scaleIn {

from {

opacity: 0;

transform: scale(0.95);

}

to {

opacity: 1;

transform: scale(1);

}

}

/ *Dark Mode Support (for accessibility)* /

@media (prefers-color-scheme: dark) {

:root {

--bg-primary: #1f2937;

--text-primary: #f9fafb;

--border-color: #374151;

}

}

/ *High Contrast Mode* /

@media (prefers-contrast: high) {

.btn-medical {

@apply border-2 border-black;

}

.medical-card {

@apply border-2 border-black;

}

}

/ *Reduced Motion Support* /

@media (prefers-reduced-motion: reduce) {

* {

animation-duration: 0.01ms !important;

animation-iteration-count: 1 !important;

transition-duration: 0.01ms !important;

}

}

## Performance Optimisation Implementation

### ⚡ Core Web Vitals Optimisation

#### Image Optimisation Strategy

`typescript

// Optimised Image Component

interface OptimisedImageProps {

src: string;

alt: string;

width: number;

height: number;

priority?: boolean;

className?: string;

sizes?: string;

}

const OptimisedImage: React.FC<OptimisedImageProps> = ({

src,

alt,

width,

height,

priority = false,

className = '',

sizes = '100vw'

}) => {

const [isLoaded, setIsLoaded] = useState(false);

const [hasError, setHasError] = useState(false);

// Generate responsive image URLs

const generateSrcSet = (baseSrc: string) => {

const breakpoints = [400, 800, 1200, 1600];

return breakpoints

.map(width => ${baseSrc}?w=${width}&q=80 ${width}w)

.join(', ');

};

return (

<div className={relative overflow-hidden ${className}}>

{!isLoaded && !hasError && (

<div

className="absolute inset-0 bg-gray-200 animate-pulse"

style={{ aspectRatio: ${width}/${height} }}

/>

)}

<picture>

<source

media="(min-width: 768px)"

srcSet={generateSrcSet(src.replace('.jpg', '.webp'))}

sizes={sizes}

type="image/webp"

/>

<source

media="(max-width: 767px)"

srcSet={generateSrcSet(src.replace('.jpg', '-mobile.webp'))}

sizes="100vw"

type="image/webp"

/>

<img

src={src}

alt={alt}

width={width}

height={height}

loading={priority ? 'eager' : 'lazy'}

decoding="async"

onLoad={() => setIsLoaded(true)}

onError={() => setHasError(true)}

className={`transition-opacity duration-300 ${

isLoaded ? 'opacity-100' : 'opacity-0'

}`}

/>

</picture>

</div>

);

};

// Lazy Loading Hook for Components

const useLazyLoad = (ref: React.RefObject<HTMLElement>) => {

const [isVisible, setIsVisible] = useState(false);

useEffect(() => {

const observer = new IntersectionObserver(

([entry]) => {

if (entry.isIntersecting) {

setIsVisible(true);

observer.disconnect();

}

},

{ threshold: 0.1 }

);

if (ref.current) {

observer.observe(ref.current);

}

return () => observer.disconnect();

}, [ref]);

return isVisible;

};

#### Code Splitting and Lazy Loading

`typescript

// Route-based Code Splitting

import { lazy, Suspense } from 'react';

import { Routes, Route } from 'react-router-dom';

// Lazy load page components

const HomePage = lazy(() => import('../pages/Home'));

const AboutPage = lazy(() => import('../pages/About'));

const RoboticSurgeryPage = lazy(() => import('../pages/Services/RoboticSurgery'));

const SleepApnoeaPage = lazy(() => import('../pages/Services/SleepApnoea'));

const ContactPage = lazy(() => import('../pages/Contact'));

// Loading component

const PageLoader = () => (

<div className="min-h-screen flex items-center justify-center">

<div className="animate-spin rounded-full h-12 w-12 border-b-2 border-medical-primary"></div>

</div>

);

// App Router with Suspense

const AppRouter = () => (

<Suspense fallback={<PageLoader />}>

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/about" element={<AboutPage />} />

<Route path="/robotic-surgery" element={<RoboticSurgeryPage />} />

<Route path="/sleep-apnoea" element={<SleepApnoeaPage />} />

<Route path="/contact" element={<ContactPage />} />

</Routes>

</Suspense>

);

// Component-based Lazy Loading

const LazyTestimonialCarousel = lazy(() =>

import('../components/sections/TestimonialCarousel')

);

const TestimonialSection = () => {

const ref = useRef<HTMLElement>(null);

const isVisible = useLazyLoad(ref);

return (

<section ref={ref}>

{isVisible ? (

<Suspense fallback={<div>Loading testimonials...</div>}>

<LazyTestimonialCarousel />

</Suspense>

) : (

<div className="h-96 bg-gray-100 animate-pulse" />

)}

</section>

);

};

### Bundle Optimisation Configuration

`javascript

// Vite Configuration for Performance

import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

import { splitVendorChunkPlugin } from 'vite';

export default defineConfig({

plugins: [

react(),

splitVendorChunkPlugin()

],

build: {

rollupOptions: {

output: {

manualChunks: {

'react-vendor': ['react', 'react-dom'],

'router': ['react-router-dom'],

'forms': ['react-hook-form'],

'ui': ['@headlessui/react', '@heroicons/react'],

'animations': ['framer-motion']

}

}

},

chunkSizeWarningLimit: 1000,

sourcemap: false,

minify: 'terser',

terserOptions: {

compress: {

drop\_console: true,

drop\_debugger: true

}

}

},

server: {

compress: true

}

});

## Security and Privacy Technical Requirements

### 🔐 Data Protection Implementation

#### Privacy-Compliant Form Handling

`typescript

// Encrypted Form Submission

import CryptoJS from 'crypto-js';

interface PatientFormData {

name: string;

email: string;

phone: string;

message: string;

appointmentType: string;

consent: {

privacy: boolean;

marketing: boolean;

dataRetention: boolean;

};

}

const encryptFormData = (data: PatientFormData, secretKey: string): string => {

return CryptoJS.AES.encrypt(JSON.stringify(data), secretKey).toString();

};

const submitPatientForm = async (formData: PatientFormData) => {

const encryptedData = encryptFormData(formData, process.env.REACT\_APP\_ENCRYPTION\_KEY!);

const response = await fetch('/api/patients/contact', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-Request-ID': generateRequestId(),

'X-Timestamp': new Date().toISOString()

},

body: JSON.stringify({

encryptedData,

consentRecorded: true,

submissionTime: new Date().toISOString(),

userAgent: navigator.userAgent,

ipAddress: 'server-side-only'

})

});

if (!response.ok) {

throw new Error('Form submission failed');

}

return response.json();

};

// Privacy Consent Management

class ConsentManager {

private storageKey = 'medical\_practice\_consent';

recordConsent(type: string, granted: boolean): void {

const consent = this.getConsent();

consent[type] = {

granted,

timestamp: new Date().toISOString(),

version: '1.0'

};

localStorage.setItem(this.storageKey, JSON.stringify(consent));

// Send to backend for audit trail

this.sendConsentToServer(type, granted);

}

getConsent(): Record<string, any> {

const stored = localStorage.getItem(this.storageKey);

return stored ? JSON.parse(stored) : {};

}

private async sendConsentToServer(type: string, granted: boolean): Promise<void> {

await fetch('/api/compliance/consent', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({

type,

granted,

timestamp: new Date().toISOString(),

userAgent: navigator.userAgent

})

});

}

}

export const consentManager = new ConsentManager();

#### Content Security Policy Configuration

`typescript

// CSP Headers Configuration

const securityHeaders = {

'Content-Security-Policy': [

"default-src 'self'",

"script-src 'self' 'unsafe-inline' https://www.googletagmanager.com https://maps.googleapis.com",

"style-src 'self' 'unsafe-inline' https://fonts.googleapis.com",

"font-src 'self' https://fonts.gstatic.com",

"img-src 'self' data: https: blob:",

"connect-src 'self' https://api.drjuliacrawford.com.au https://www.google-analytics.com",

"frame-src 'self' https://www.google.com",

"object-src 'none'",

"base-uri 'self'",

"form-action 'self'",

"frame-ancestors 'none'"

].join('; '),

'X-Frame-Options': 'DENY',

'X-Content-Type-Options': 'nosniff',

'X-XSS-Protection': '1; mode=block',

'Referrer-Policy': 'strict-origin-when-cross-origin',

'Permissions-Policy': 'geolocation=(), microphone=(), camera=()'

};

// Security Middleware for API

const securityMiddleware = (req: Request, res: Response, next: NextFunction) => {

// Apply security headers

Object.entries(securityHeaders).forEach(([header, value]) => {

res.setHeader(header, value);

});

// Rate limiting for form submissions

if (req.path.includes('/contact') && req.method === 'POST') {

// Implement rate limiting logic

rateLimiter.check(req.ip, (err, result) => {

if (err || !result.allowed) {

return res.status(429).json({ error: 'Too many requests' });

}

next();

});

} else {

next();

}

};

## SEO and Analytics Implementation

### 📊 SEO Technical Setup

#### Structured Data Implementation

`typescript

// Schema.org JSON-LD Generator

interface MedicalOrganizationSchema {

name: string;

description: string;

url: string;

logo: string;

address: Address[];

telephone: string;

medicalSpecialty: string[];

availableService: MedicalService[];

}

const generateMedicalOrgSchema = (data: MedicalOrganizationSchema) => {

return {

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

'@type': 'MedicalOrganization',

'@id': ${data.url}#organization,

name: data.name,

description: data.description,

url: data.url,

logo: {

'@type': 'ImageObject',

url: data.logo,

width: 600,

height: 60

},

address: data.address.map(addr => ({

'@type': 'PostalAddress',

streetAddress: addr.streetAddress,

addressLocality: addr.locality,

addressRegion: addr.region,

postalCode: addr.postalCode,

addressCountry: 'AU'

})),

telephone: data.telephone,

medicalSpecialty: data.medicalSpecialty,

availableService: data.availableService.map(service => ({

'@type': 'MedicalProcedure',

name: service.name,

description: service.description,

procedureType: service.type

}))

};

};

// SEO Head Component

interface SEOHeadProps {

title: string;

description: string;

canonicalUrl: string;

ogImage?: string;

schema?: object;

medicalContent?: boolean;

}

const SEOHead: React.FC<SEOHeadProps> = ({

title,

description,

canonicalUrl,

ogImage,

schema,

medicalContent = false

}) => {

const siteTitle = 'Dr Julia Crawford ENT Specialist';

const fullTitle = ${title} | ${siteTitle};

return (

<Helmet>

{/ *Basic Meta Tags* /}

<title>{fullTitle}</title>

<meta name="description" content={description} />

<link rel="canonical" href={canonicalUrl} />

{/ *Open Graph* /}

<meta property="og:title" content={fullTitle} />

<meta property="og:description" content={description} />

<meta property="og:url" content={canonicalUrl} />

<meta property="og:type" content={medicalContent ? 'article' : 'website'} />

{ogImage && <meta property="og:image" content={ogImage} />}

{/ *Twitter Card* /}

<meta name="twitter:card" content="summary\_large\_image" />

<meta name="twitter:title" content={fullTitle} />

<meta name="twitter:description" content={description} />

{ogImage && <meta name="twitter:image" content={ogImage} />}

{/ *Medical Content Specific* /}

{medicalContent && (

<>

<meta name="robots" content="index, follow, max-snippet:-1" />

<meta name="googlebot" content="index, follow" />

<meta name="medical-disclaimer" content="Educational purposes only. Consult Dr Crawford for personalised advice." />

</>

)}

{/ *Structured Data* /}

{schema && (

<script type="application/ld+json">

{JSON.stringify(schema)}

</script>

)}

</Helmet>

);

};

#### Analytics and Tracking Setup

`typescript

// Google Analytics 4 Integration

import { gtag } from 'ga-gtag';

class AnalyticsManager {

private trackingId: string;

private medicalComplianceMode: boolean;

constructor(trackingId: string, medicalCompliance = true) {

this.trackingId = trackingId;

this.medicalComplianceMode = medicalCompliance;

this.initializeGA4();

}

private initializeGA4(): void {

gtag('config', this.trackingId, {

anonymize\_ip: this.medicalComplianceMode,

allow\_google\_signals: !this.medicalComplianceMode,

cookie\_flags: 'secure;samesite=strict'

});

}

trackPageView(title: string, patientJourneyStage?: string): void {

gtag('event', 'page\_view', {

page\_title: title,

custom\_map: {

patient\_journey\_stage: patientJourneyStage

}

});

}

trackMedicalInteraction(action: string, service: string): void {

gtag('event', 'medical\_interaction', {

event\_category: 'Medical Content',

event\_label: service,

custom\_map: {

interaction\_type: action,

medical\_service: service

}

});

}

trackConversion(type: 'consultation\_booked' | 'contact\_form' | 'phone\_call'): void {

gtag('event', 'conversion', {

event\_category: 'Patient Conversion',

event\_label: type,

value: 1

});

}

trackUserJourney(stage: string, service?: string): void {

gtag('event', 'user\_journey', {

event\_category: 'Patient Journey',

event\_label: stage,

custom\_map: {

journey\_stage: stage,

interested\_service: service

}

});

}

}

export const analytics = new AnalyticsManager(

process.env.REACT\_APP\_GA\_TRACKING\_ID!,

true

);

// Custom Hook for Analytics

export const useAnalytics = () => {

const trackEvent = useCallback((eventName: string, parameters: object) => {

gtag('event', eventName, parameters);

}, []);

const trackPatientJourney = useCallback((stage: string, service?: string) => {

analytics.trackUserJourney(stage, service);

}, []);

return {

trackEvent,

trackPatientJourney,

trackPageView: analytics.trackPageView.bind(analytics),

trackConversion: analytics.trackConversion.bind(analytics)

};

};

## Testing and Quality Assurance Framework

### 🧪 Comprehensive Testing Strategy

#### Unit Testing Configuration

`typescript

// Jest + Testing Library Setup

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

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

import { axe, toHaveNoViolations } from 'jest-axe';

expect.extend(toHaveNoViolations);

// Medical Component Testing

describe('MedicalDisclaimer Component', () => {

test('displays correct disclaimer content for medical procedures', () => {

render(

<MedicalDisclaimer

type="procedure"

position="banner"

required={true}

/>

);

expect(screen.getByText(/Individual results may vary/i)).toBeInTheDocument();

expect(screen.getByRole('button', { name: /acknowledge/i })).toBeInTheDocument();

});

test('records consent when acknowledged', async () => {

const mockRecordConsent = jest.fn();

const user = userEvent.setup();

render(

<ComplianceProvider recordConsent={mockRecordConsent}>

<MedicalDisclaimer type="procedure" position="banner" required={true} />

</ComplianceProvider>

);

await user.click(screen.getByRole('button', { name: /acknowledge/i }));

expect(mockRecordConsent).toHaveBeenCalledWith('disclaimer\_procedure');

});

test('has no accessibility violations', async () => {

const { container } = render(

<MedicalDisclaimer type="general" position="inline" required={false} />

);

const results = await axe(container);

expect(results).toHaveNoViolations();

});

});

// Patient Form Testing

describe('PatientContactForm', () => {

test('validates required fields', async () => {

const user = userEvent.setup();

render(<PatientContactForm />);

await user.click(screen.getByRole('button', { name: /send message/i }));

expect(screen.getByText(/name is required/i)).toBeInTheDocument();

expect(screen.getByText(/email is required/i)).toBeInTheDocument();

});

test('submits form with valid data and privacy consent', async () => {

const mockSubmit = jest.fn().mockResolvedValue({ success: true });

const user = userEvent.setup();

render(<PatientContactForm onSubmit={mockSubmit} />);

await user.type(screen.getByLabelText(/full name/i), 'John Smith');

await user.type(screen.getByLabelText(/email/i), 'john@example.com');

await user.type(screen.getByLabelText(/phone/i), '0400123456');

await user.click(screen.getByLabelText(/privacy policy/i));

await user.click(screen.getByRole('button', { name: /send message/i }));

await waitFor(() => {

expect(mockSubmit).toHaveBeenCalledWith(

expect.objectContaining({

name: 'John Smith',

email: 'john@example.com',

phone: '0400123456'

})

);

});

});

});

#### End-to-End Testing with Cypress

`typescript

// cypress/e2e/patient-journey.cy.ts

describe('Patient Journey - Sleep Apnoea', () => {

beforeEach(() => {

cy.visit('/');

});

it('completes sleep apnoea patient journey from homepage to booking', () => {

// Homepage interaction

cy.get('[data-testid="service-sleep-apnoea"]').click();

// Service page engagement

cy.url().should('include', '/sleep-apnoea');

cy.get('h1').should('contain', 'Sleep Apnoea Treatment');

// Medical disclaimer acknowledgment

cy.get('[data-testid="medical-disclaimer"]').should('be.visible');

cy.get('[data-testid="acknowledge-disclaimer"]').click();

// Learn more about robotic treatment

cy.get('[data-testid="robotic-surgery-link"]').click();

cy.url().should('include', '/robotic-surgery/sleep-apnoea');

// Navigate to consultation booking

cy.get('[data-testid="book-consultation"]').click();

cy.url().should('include', '/book');

// Fill booking form

cy.get('[data-testid="patient-name"]').type('Test Patient');

cy.get('[data-testid="patient-email"]').type('test@example.com');

cy.get('[data-testid="patient-phone"]').type('0400123456');

cy.get('[data-testid="appointment-type"]').select('Sleep Apnoea Consultation');

cy.get('[data-testid="privacy-consent"]').check();

// Submit booking

cy.get('[data-testid="submit-booking"]').click();

cy.get('[data-testid="booking-confirmation"]').should('be.visible');

});

it('meets accessibility standards throughout patient journey', () => {

cy.injectAxe();

// Test homepage accessibility

cy.checkA11y();

// Navigate to service page

cy.get('[data-testid="service-sleep-apnoea"]').click();

cy.checkA11y();

// Test booking page accessibility

cy.get('[data-testid="book-consultation"]').click();

cy.checkA11y();

});

});

// Mobile-specific testing

describe('Mobile Patient Experience', () => {

beforeEach(() => {

cy.viewport('iphone-x');

cy.visit('/');

});

it('provides optimal mobile navigation experience', () => {

// Mobile menu functionality

cy.get('[data-testid="mobile-menu-toggle"]').click();

cy.get('[data-testid="mobile-menu"]').should('be.visible');

// Service navigation

cy.get('[data-testid="mobile-services"]').click();

cy.get('[data-testid="mobile-service-robotic"]').click();

// Touch-friendly interactions

cy.get('[data-testid="cta-button"]').should('have.css', 'min-height', '44px');

// Mobile form usability

cy.get('[data-testid="mobile-contact-form"]').should('be.visible');

cy.get('[data-testid="mobile-phone-link"]').should('have.attr', 'href', 'tel:0283199434');

});

});

#### Performance Testing Setup

`typescript

// lighthouse-config.js

module.exports = {

ci: {

collect: {

url: [

'http://localhost:3000/',

'http://localhost:3000/about',

'http://localhost:3000/robotic-surgery',

'http://localhost:3000/sleep-apnoea',

'http://localhost:3000/contact'

],

settings: {

chromeFlags: '--no-sandbox --headless'

}

},

assert: {

assertions: {

'categories:performance': ['error', { minScore: 0.9 }],

'categories:accessibility': ['error', { minScore: 0.95 }],

'categories:best-practices': ['error', { minScore: 0.9 }],

'categories:seo': ['error', { minScore: 0.95 }]

}

},

upload: {

target: 'temporary-public-storage'

}

}

};

// Performance testing script

const lighthouse = require('lighthouse');

const chromeLauncher = require('chrome-launcher');

async function runPerformanceTests() {

const chrome = await chromeLauncher.launch({ chromeFlags: ['--headless'] });

const urls = [

'https://drjuliacrawford.com.au/',

'https://drjuliacrawford.com.au/robotic-surgery/',

'https://drjuliacrawford.com.au/sleep-apnoea/'

];

for (const url of urls) {

const options = {

logLevel: 'info',

output: 'json',

onlyCategories: ['performance', 'accessibility', 'seo'],

port: chrome.port

};

const runnerResult = await lighthouse(url, options);

console.log(Performance Report for ${url}:);

console.log(Performance: ${runnerResult.lhr.categories.performance.score \* 100});

console.log(Accessibility: ${runnerResult.lhr.categories.accessibility.score \* 100});

console.log(SEO: ${runnerResult.lhr.categories.seo.score \* 100});

}

await chrome.kill();

}

runPerformanceTests();

## Deployment and Launch Strategy

### 🚀 Production Deployment Configuration

#### CI/CD Pipeline Setup

`yaml

# .github/workflows/deploy.yml

name: Deploy to Production

on:

push:

branches: [main]

pull\_request:

branches: [main]

jobs:

test:

runs-on: ubuntu-latest

steps:

* uses: actions/checkout@v3
* name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

cache: 'npm'

* name: Install dependencies

run: npm ci

* name: Run tests

run: npm run test:ci

* name: Run accessibility tests

run: npm run test:a11y

* name: Run performance tests

run: npm run test:lighthouse

* name: Medical compliance check

run: npm run test:medical-compliance

build:

needs: test

runs-on: ubuntu-latest

steps:

* uses: actions/checkout@v3
* name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

cache: 'npm'

* name: Install dependencies

run: npm ci

* name: Build application

run: npm run build

env:

REACT\_APP\_GA\_TRACKING\_ID: ${{ secrets.GA\_TRACKING\_ID }}

REACT\_APP\_API\_URL: ${{ secrets.API\_URL }}

* name: Upload build artifacts

uses: actions/upload-artifact@v3

with:

name: build-files

path: build/

deploy:

needs: build

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/main'

steps:

* name: Download build artifacts

uses: actions/download-artifact@v3

with:

name: build-files

path: build/

* name: Deploy to Vercel

uses: vercel/action@v1

with:

vercel-token: ${{ secrets.VERCEL\_TOKEN }}

vercel-org-id: ${{ secrets.VERCEL\_ORG\_ID }}

vercel-project-id: ${{ secrets.VERCEL\_PROJECT\_ID }}

vercel-args: '--prod'

#### Environment Configuration

`javascript

// Production environment variables

const productionConfig = {

// Core Application

REACT\_APP\_ENV: 'production',

REACT\_APP\_SITE\_URL: 'https://drjuliacrawford.com.au',

REACT\_APP\_API\_URL: 'https://api.drjuliacrawford.com.au',

// Analytics & Tracking

REACT\_APP\_GA\_TRACKING\_ID: 'G-XXXXXXXXXX',

REACT\_APP\_HOTJAR\_ID: 'XXXXXXX',

// Medical Compliance

REACT\_APP\_TGA\_COMPLIANCE\_MODE: 'true',

REACT\_APP\_MEDICAL\_DISCLAIMER\_REQUIRED: 'true',

REACT\_APP\_PRIVACY\_POLICY\_VERSION: '2025.1',

// Security

REACT\_APP\_ENCRYPTION\_KEY: 'secure-encryption-key',

REACT\_APP\_CSP\_NONCE: 'generated-nonce',

// External Services

REACT\_APP\_GOOGLE\_MAPS\_API\_KEY: 'maps-api-key',

REACT\_APP\_BOOKING\_SYSTEM\_URL: 'https://booking.drjuliacrawford.com.au',

// Performance

REACT\_APP\_CDN\_URL: 'https://cdn.drjuliacrawford.com.au',

REACT\_APP\_IMAGE\_OPTIMIZATION: 'true'

};

#### Launch Checklist

`markdown

## Pre-Launch Checklist

### Technical Requirements

* [ ] All tests passing (unit, integration, e2e)
* [ ] Performance scores >90 on all pages
* [ ] Accessibility compliance WCAG 2.1 AA
* [ ] Security headers configured
* [ ] SSL certificate installed
* [ ] CDN configured for static assets
* [ ] Database backup strategy implemented
* [ ] Error monitoring setup (Sentry/LogRocket)

### Medical Compliance

* [ ] Medical disclaimers on all relevant pages
* [ ] TGA compliance review completed
* [ ] Privacy policy updated and published
* [ ] Patient consent mechanisms tested
* [ ] Medical content review by Dr Crawford
* [ ] Evidence sources verified and cited

### SEO & Analytics

* [ ] Google Analytics 4 configured
* [ ] Google Search Console verified
* [ ] XML sitemap generated and submitted
* [ ] Structured data implemented and validated
* [ ] Meta tags optimised for all pages
* [ ] Canonical URLs configured
* [ ] 301 redirects from old site (if applicable)

### Content & Design

* [ ] All content proofread and approved
* [ ] British English spelling verified
* [ ] Images optimised and alt text added
* [ ] Contact information verified
* [ ] Practice hours updated
* [ ] Location information accurate
* [ ] Professional photography finalised

### Functionality Testing

* [ ] Contact forms working correctly
* [ ] Booking system integration tested
* [ ] Email notifications functioning
* [ ] Mobile responsiveness verified
* [ ] Cross-browser compatibility checked
* [ ] Load testing completed

### Legal & Compliance

* [ ] Terms of service published
* [ ] Privacy policy compliant with Australian law
* [ ] Cookie policy implemented
* [ ] Medical advertising compliance verified
* [ ] Professional indemnity insurance current
* [ ] Medical board registration verified

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Technical Implementation Confidence Score:** 97%

**Development Feasibility:** High with systematic phased approach

**Medical Compliance Integration:** Comprehensive TGA-compliant technical framework

**Performance Optimisation:** Core Web Vitals targets achievable with outlined strategies

*This technical implementation guide provides a comprehensive development framework for Dr Julia Crawford's ENT practice website, ensuring medical compliance, optimal performance, and exceptional user experience across all devices and patient journey stages.*