SEO Tag Helper Tool - Product Requirements Document (Zero-Cost Edition)

Executive Summary

The SEO Tag Helper Tool is a web-based application that automatically scans websites and generates optimized SEO metadata recommendations. This zero-cost edition operates entirely on free hosting tiers (Vercel, Render.com, Supabase) while delivering professional SEO analysis through downloadable Word documents.

Product Overview

Problem Statement

Businesses and marketers struggle to maintain consistent, SEO-optimized metadata across their websites. Manual optimization is time-consuming and often inconsistent, leading to missed opportunities in search rankings.

Solution

An automated tool that crawls websites, analyzes content, and generates SEO-optimized recommendations while operating entirely on free hosting services with specific limitations designed around free tier constraints.

Key Constraints (Zero-Cost Approach)

- **Infrastructure**: Must operate within free tier limits
- **Processing**: Limited to single-instance processing with 1 concurrent job
- **Storage**: Temporary in-memory storage with 3-hour auto-cleanup
- **Scale**: Maximum 50 pages per scan, 3 levels deep

Core Features

1. Website Scanner

Feature Specification

Aspect	Specification	TDD Specification
Input Method	User enters website URL or single page URL	Single scan at a time
Crawl Depth	Up to 3 levels maximum	`depth > 3` check in scraper
Max Pages	50 pages per scan	`maxPages = 50` in LightweightScraper
Page Discovery	Maximum 10 internal links per page	`slice(0, 10)` in page evaluation
Timeout Handling	30-second timeout per page	`timeout: 30000` in page.goto
Progress Tracking	Updates every 5 pages	`pagesProcessed % 5 === 0`
Technology	Puppeteer with resource blocking	Blocks images, stylesheets, fonts, media
Concurrency	Single browser instance	`maxConcurrent = 1` in queue

Resource Optimization (from TDD)

```
Browser args: [
  '--no-sandbox',
  '--disable-setuid-sandbox',
  '--disable-dev-shm-usage',
  '--disable-gpu',
  '--single-process',
  '--no-zygote',
  '--disable-web-security'
]
```

2. Content Analyzer

Extraction Specifications (from TDD)

Element	Extraction Method	Storag
Page Title	`document.querySelector('title')?.textContent`	In scan JSONl
Meta Description	`querySelector('meta[name="description"]')?.getAttribute('content')`	In scan JSONl
Headings	All H1, H2, H3 tags mapped to arrays	`head` `{ h1: h2: h3:`
Images	`querySelectorAll('img')` with src and alt	Array curren
Word Count	`body.textContent.split(/\s+/).filter(word => word.length > 0).length`	Per-pa metric
Content Check	`wordCount > 50` determines hasContent	Boolea

3. SEO Tag Generator

Optimization Rules (from TDD)

Element	Character Limits	Generation Logic
Meta Title	30-60 characters	If < 30: Use H1 + " | Your Brand" If > 60: Truncate to 57 chars + "..."
Meta Description	120-160 characters	If < 120: "Learn more about {title}. {current}" If > 160: Truncate to 157 chars + "..."
Image Alt Text	Maximum 100 characters	Clean filename + " on " + page title

Priority Calculation (from TDD)

```typescript
- Title issues (< 30 or > 60 chars): +1 point
- Description issues (< 120 chars): +1 point
- No content (hasContent = false): +2 points
- Images without alt text: +1 point

Score >= 3: 'high'
Score >= 1: 'medium'
Score = 0: 'low'
```

4. Brand Customization

Configuration Storage (from TDD)

Feature	Implementation	Storage Location
Brand Colors	3 optional colors	`report_config` JSONB column
Color Format	Primary, secondary, tertiary	Passed to ReportGenerator
Storage Duration	3 hours	Auto-cleanup via `expires_at`

5. Report Generator

Word Document Specifications (from TDD)

Section	Content	Implementation
Cover Page	SEO Analysis Report URL Generation date	Using docx library
Executive Summary	Total pages analyzed High priority count Pages skipped	Calculated from scan_data
Page-by-Page Analysis	Current title (with char count) Recommended title Current description Recommended description Priority level	Iterate through pages array
Image Optimization	Only images missing alt text Recommended alt for each	Filter where !currentAlt

Report Caching

- In-memory Map storage: `reportCache.set(sessionId, buffer)`
- 3-hour auto-cleanup: `setTimeout(() => reportCache.delete(sessionId), 3 * 60 * 60 * 1000)`
- Single report per session

Technical Specifications

Performance Limits (from TDD)

Metric	Specification	Implementation
Max Pages per Scan	50	`this.maxPages = 50`
Page Load Strategy	DOM content loaded	`waitUntil: 'domcontentloaded'`
Page Timeout	30 seconds	`timeout: 30000`
Queue Concurrency	1 job at a time	`maxConcurrent = 1`
Retry Attempts	3 attempts	`if (job.attempts < 3)`
Retry Delay	5 seconds	`await this.delay(5000)`
Report Generation	< 45 seconds for 100 pages	Target metric
Data Retention	3 hours	`expires_at` timestamp

Technology Stack (from TDD)

Component	Technology	Free Tier Limits
Frontend	React with TypeScript	Vercel: 100GB bandwidth/month
Backend	Node.js + Express	Render.com: 750 hours/month
Database	PostgreSQL	Supabase: 500MB storage
Queue	In-memory JavaScript Map	No external service
Scraping	Puppeteer (lightweight)	Single browser instance
Report Generation	docx library	In-memory processing
Email (Optional)	Resend	100 emails/day
File Storage	Temporary server memory	3-hour cleanup

Database Schema (from TDD)

```sql
-- Main table using JSONB for flexibility
CREATE TABLE scan_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    url VARCHAR(2048) NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    scan_data JSONB DEFAULT '{}',
    report_config JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP + INTERVAL '3 hours'
);

-- Email collection (optional feature)
CREATE TABLE email_list (
    email VARCHAR(255) PRIMARY KEY,
    website_url VARCHAR(2048),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Indexes
CREATE INDEX idx_sessions_status ON scan_sessions(status);
CREATE INDEX idx_sessions_expires ON scan_sessions(expires_at);
```

API Endpoints (from TDD)

Method	Endpoint	Purpose	Rate Limit
POST	`/api/scan/start`	Start new scan	20 requests/minute per IP
GET	`/api/scan/:id/status`	Get scan progress	20 requests/minute per IP
GET	`/api/scan/:id/results`	Get scan results	20 requests/minute per IP
POST	`/api/report/generate`	Generate Word report	20 requests/minute per IP
GET	`/api/report/:id/download`	Download report	20 requests/minute per IP
POST	`/api/report/:id/email`	Email report (optional)	20 requests/minute per IP
GET	`/health`	Keep Render.com awake	No limit

User Experience Flow

1. Landing Page

- URL input field with validation (must start with http:// or https://)
- "Start Scanning" button

- Maximum 20 requests per minute per IP

2. Configuration Screen

- Optional brand colors (HEX, RGB, or color picker as per TDD)
- All three input methods supported unlike simplified version

3. Scanning Progress

- Real-time status updates showing:
  - Current scan status
  - Pages processed count
  - Total pages found
  - Current page being scanned
  - Error count

4. Download Screen

- Direct download link (active for 3 hours)
- Report expires at timestamp
- Optional email delivery (if Resend configured)

Error Handling (from TDD)

Scan Errors

Error Type	Handling	User Message
Invalid URL	400 status	"Invalid URL"
Page timeout	Skip page, continue scan	Logged in errors array
Visit limit	Stop at 50 pages	Reflected in pagesProcessed
Memory limit	400MB heap warning	Health check monitoring

Server Cold Starts

- Retry interceptor with exponential backoff
- Maximum 3 retries
- Delays: 2s, 4s, 8s (max 10s)

Free Tier Monitoring

Resource Usage (from TDD Table)

Service	Free Tier Limit	Expected Usage	Buffer
Vercel	100GB/month	~10GB/month	90GB
Render.com	750 hours/month	24/7 running	OK
Supabase	500MB storage	~100MB	400MB
Resend	100 emails/day	Optional feature	100/day

Health Monitoring

- Memory check: Alert at 400MB (512MB limit)
- Database connectivity check
- Hourly cleanup of expired sessions
- Self-ping every 14 minutes during business hours (9-17)

Scaling Path (from TDD)

Tier	Price	Features
Free	$0	Current limitations
Starter	$7/month	Render.com Starter (no spin-down)
Pro	$25/month	+ Supabase Pro (8GB storage)
Business	$50/month	+ Redis for proper queuing
Enterprise	$100+/month	Multiple workers, CDN, monitoring

Environment Variables (from TDD)

Backend (.env)

```bash
NODE_ENV=production
PORT=3000
SUPABASE_URL=https://your-project.supabase.co
SUPABASE_ANON_KEY=your-anon-key
FRONTEND_URL=https://your-app.vercel.app
BACKEND_URL=https://your-backend.onrender.com
RESEND_API_KEY=your-resend-key # Optional
```

Frontend (.env)

```bash
REACT_APP_API_URL=https://your-backend.onrender.com
```

Deployment Configuration

Vercel (Frontend)

```json
{
  "buildCommand": "npm run build",
  "outputDirectory": "build",
  "framework": "create-react-app",
  "rewrites": [{"source": "/(.*)", "destination": "/index.html"}]
}
```

Render.com (Backend)

```yaml
services:
  - type: web
    name: seo-tool-backend
    env: node
    plan: free
    buildCommand: npm install && npm run build
    startCommand: node dist/server.js
```

Compliance & Security

- **Data Retention**: Automatic deletion after 3 hours via `expires_at`
- **HTTPS Only**: Enforced by hosting platforms
- **No Persistent Storage**: Scan data deleted after session expires
- **Rate Limiting**: 20 requests/minute per IP address
- **Robots.txt Compliance**: Checked before crawling