COMPLETE PRODUCT REQUIREMENTS DOCUMENT (PRD)
Advanced Web Crawler for Black Box Testing

Document Version: 2.0
Last Updated: 2025-10-22
Status: Final Comprehensive Version

Table of Contents

1. Executive Summary
   1.1 Product Overview
   A comprehensive web crawling tool designed for security professionals and QA engineers to automatically discover, map, and extract testable elements from web applications. The tool handles complex modern web applications including SPAs, role-based access control, poor semantic HTML, massive e-commerce sites, and advanced user interactions. It provides visual site mapping, multi-role authentication, state management, and structured test data export for comprehensive black box testing.
   1.2 Target Users

Security researchers and penetration testers
QA engineers conducting black box testing
DevSecOps teams
Web application auditors
Compliance testing teams

1.3 Key Differentiators

Intelligent element detection beyond semantic HTML (handles div-based UIs)
Pattern-based sampling for sites with millions of pages
Multi-role crawling with permission matrix generation
State-aware crawling (cart states, filters, user preferences)
Comprehensive interaction simulation (hover, drag-drop, keyboard shortcuts)
Advanced test case generation including RBAC and privilege escalation tests

2. Product Objectives

2.1 Primary Goals

Automatically discover all accessible pages, states, and user flows within a web application
Handle authentication flows including multi-factor, OAuth, SSO, and reCAPTCHA
Support role-based access control with multi-user perspective crawling
Extract and categorize all testable elements from modern web applications
Handle massive sites (millions of pages) through intelligent sampling
Detect elements built with poor semantic HTML (div-based buttons, forms)
Capture application states and user interaction flows
Visualize the application structure as an interactive multi-layer graph
Export structured test data compatible with major testing frameworks

2.2 Success Metrics

Successfully crawl and map 95%+ of accessible pages and states
Accurately identify 90%+ of testable elements including non-semantic HTML
Handle common authentication patterns with 95%+ success rate
Process sites with 1M+ pages using pattern sampling
Generate comprehensive RBAC test suites for multi-role applications
Export data compatible with 8+ major testing frameworks

3. Core Functional Requirements

3.1 Crawling Engine

3.1.1 Page Discovery

FR-001: Spider follows all links including tags, forms, JavaScript navigation, and client-side routing
FR-002: Configurable crawl depth (1-10 levels, default: 3)
FR-003: Configurable maximum page limit (1-1,000,000 pages)
FR-004: Automatic pagination detection and handling
FR-005: Support for both static and JavaScript-rendered content via headless browser (Puppeteer/Playwright)
FR-006: Respect robots.txt with override option for authorized testing
FR-007: Configurable politeness delays between requests (0-5000ms)
FR-008: Handle redirects (301, 302, 307, 308) with chain tracking
FR-009: Detect and avoid crawler traps (infinite loops, calendar traps)
FR-010: Detect and handle infinite scroll pages
FR-011: Automatic "Load More" button detection and clicking

3.1.2 URL Management

FR-012: URL normalization (case, trailing slashes, parameter ordering)
FR-013: URL filtering with include/exclude regex patterns
FR-014: Follow or ignore external links (configurable)
FR-015: Track URL parameters and their variations
FR-016: Detect API endpoints from network requests (XHR/Fetch)
FR-017: Bloom filter-based deduplication for millions of URLs
FR-018: Canonical URL detection via
FR-019: Remove tracking parameters (utm_*, fbclid, gclid)

3.1.3 SPA & Modern Framework Support

FR-020: Detect client-side routing (React Router, Vue Router, Angular)
FR-021: Wait for dynamic content loading with configurable timeouts
FR-022: Monitor network requests to discover data endpoints
FR-023: Trigger route changes programmatically
FR-024: Extract routes from JavaScript bundles and configuration
FR-025: Handle hash-based routing (#/) and history API (pushState)
FR-026: Detect and wait for skeleton screens and loading indicators
FR-027: Mutation observer for DOM changes

4. Advanced Crawling Scenarios
4.1 Poor Semantic HTML Handling
4.1.1 Non-Standard Element Detection

FR-028: Detect clickable divs with onclick, role="button", or cursor styles as buttons
FR-029: Identify form-like structures without

tags
FR-030: Recognize input-like divs (contenteditable, custom components)
FR-031: Detect navigation patterns in div hierarchies
FR-032: Heuristic-based classification using:

Class name patterns (btn-, form-, input-, link-)
ARIA attributes (role, aria-label, aria-labelledby)
Event listener detection (click, submit, change handlers)
CSS analysis (clickable cursors, form-like styling)
Text content analysis ("Submit", "Login", "Add to Cart")

4.1.2 Framework Component Detection

FR-033: Identify React components via data-reactid or React DevTools
FR-034: Detect Vue.js components via v-on, v-bind attributes
FR-035: Recognize Angular components via ng-click, ng-model
FR-036: Parse virtual DOM structures
FR-037: Detect custom web components
FR-038: Shadow DOM traversal
FR-039: Support for popular UI libraries (Material-UI, Ant Design, Bootstrap, Chakra UI)

4.2 Large-Scale E-commerce Sites

4.2.1 Pattern-Based URL Sampling

FR-040: Detect URL patterns using regex (e.g., /product/\d+)
FR-041: Sample representative URLs instead of crawling all variations
FR-042: Configurable sampling strategies:

Random sampling (configurable percentage)
Stratified sampling (different ranges)
Template extraction (crawl 1 URL per pattern)

FR-043: Pattern configuration with thresholds (e.g., sample when count > 100)

Configuration Example:

```
{
  "patterns": [
    {
      "pattern": "/product/(\\d+)",
      "strategy": "sample",
      "threshold": 100,
      "sampleSize": 100,
      "method": "random"
    }
  ]
}
```

4.2.2 Pagination Intelligence

FR-044: Auto-detect pagination patterns (?page=, ?offset=, /page/, ?p=)
FR-045: Crawl first 3 pages and last page (representative sampling)
FR-046: Extract pagination metadata (total pages, items per page)
FR-047: Detect and handle infinite scroll
FR-048: "Load More" button detection and interaction

4.2.3 E-commerce Specific Features

FR-049: Detect add-to-cart functionality (even in non-standard divs)
FR-050: Extract product filters and facets
FR-051: Capture dynamic pricing elements
FR-052: Identify wishlist/compare actions
FR-053: Detect stock availability checks
FR-054: Multi-step checkout flow mapping
FR-055: Cart state management (empty vs. with items)
FR-056: Payment gateway identification (Stripe, PayPal, Square)

4.3 Large Blog/Content Sites

4.3.1 Content Pattern Recognition

FR-057: Detect blog post URL structures
FR-058: Date-based URL patterns (/YYYY/MM/DD/)
FR-059: Category-based patterns (/category/subcategory/post)
FR-060: Author-based patterns (/author/username/post)
FR-061: Tag-based patterns (/tag/tagname/)

4.3.2 Temporal Sampling Strategy

FR-062: Sample posts from different time periods
FR-063: Prioritize recent content (configurable period, e.g., last 3 months at 100%)
FR-064: Sample older content (configurable, e.g., yearly at 10%)
FR-065: Configurable time-based quotas per period

Configuration Example:

```
{
  "contentSampling": {
    "recentPosts": { "period": "3months", "percentage": 100 },
    "olderPosts": { "period": "1year", "percentage": 10 },
    "categories": { "crawl": "all" },
    "tags": { "limit": 50, "strategy": "popular" }
  }
}
```

4.4 Advanced URL Intelligence
4.4.1 Deduplication Logic

FR-066: Ignore tracking parameters (utm_*, fbclid, gclid, ref)
FR-067: Normalize parameter order
FR-068: Detect URL aliases (www vs non-www, trailing slash)
FR-069: Identify canonical URLs via link tags
FR-070: Handle URL fragments vs. actual pages
FR-071: Parameter significance detection (test which params change content)

4.4.2 Infinite Crawl Prevention

FR-072: URL pattern cycle detection
FR-073: Page content similarity hashing (SimHash/MinHash)
FR-074: Crawl depth per pattern limit
FR-075: Session ID parameter detection and removal
FR-076: Calendar/archive trap detection (infinite dates)
FR-077: Maximum pages per pattern limit (default: 1000)

4.5 Content Fingerprinting

FR-078: SimHash or MinHash for content comparison
FR-079: Detect template pages (only differ in IDs)
FR-080: Group similar pages together

FR-081: Crawl one representative per similarity group
FR-082: Configurable similarity threshold (default: 0.95)

4.6 Nested/Recursive Structures

FR-083: Handle comment threads with unlimited reply depth
FR-084: Nested categories/subcategories (unlimited depth)
FR-085: Folder/file tree structures
FR-086: Organizational hierarchies
FR-087: Thread/forum structures
FR-088: Configurable maximum recursion depth per structure type

5. Role-Based Access Control (RBAC)
   5.1 Multi-User Role Management
   5.1.1 Role Definition & Configuration

FR-089: Define multiple user roles/personas with credentials
FR-090: Role hierarchy mapping (Admin > Manager > User > Guest)
FR-091: Permission matrix visualization
FR-092: Custom role attributes (department, subscription tier, geography)
FR-093: Secure credential storage (AES-256 encryption at rest)

Role Configuration Schema:

```
{
  "roles": [
    {
      "id": "admin",
      "name": "Administrator",
      "credentials": {
        "username": "admin@example.com",
        "password": "encrypted_password",
        "mfa": "manual_intervention"
      },
      "attributes": {
        "permissions": ["read", "write", "delete", "manage_users"],
        "subscriptionTier": "enterprise",
        "department": "IT"
      },
      "priority": 1
    },
    {
      "id": "manager",
      "name": "Manager",
      "credentials": {...},
      "attributes": {
        "permissions": ["read", "write", "approve"],
        "department": "Sales"
      },
      "priority": 2
```

```
    },
    {
      "id": "basic_user",
      "name": "Basic User",
      "credentials": {...},
      "attributes": {
        "permissions": ["read"],
        "subscriptionTier": "free"
      },
      "priority": 3
    },
    {
      "id": "guest",
      "name": "Guest/Unauthenticated",
      "credentials": null,
      "attributes": {
        "permissions": ["read_public"]
      },
      "priority": 4
    }
  ]
}
```

5.1.2 Role-Based Crawling Strategy

FR-094: Sequential role crawling (one role at a time)
FR-095: Parallel role crawling (multiple simultaneous crawlers)
FR-096: Incremental role crawling (start with highest privilege)
FR-097: Compare discovered pages across roles
FR-098: Identify role-specific pages and features
FR-099: Detect permission boundaries automatically
FR-100: Map role-to-feature relationships

5.2 Permission Detection
5.2.1 Access Control Identification

FR-101: Detect 403 Forbidden responses
FR-102: Detect 401 Unauthorized responses
FR-103: Identify redirect-based protection (redirect to /login)
FR-104: Detect soft blocks (page loads but content hidden)
FR-105: Client-side permission checks (disabled buttons, hidden menus)
FR-106: Server-side API permission checks

5.2.2 UI Element Visibility by Role

FR-107: Track buttons visible only to certain roles
FR-108: Menu items by permission level
FR-109: Form fields that vary by role
FR-110: Action buttons (edit, delete, approve) by role

FR-111: Admin panels and dashboards
FR-112: Bulk operation controls by permission

5.3 Differential Analysis
5.3.1 Cross-Role Comparison

FR-113: Generate diff report between roles
FR-114: Identify exclusive features per role
FR-115: Detect shared vs. role-specific pages
FR-116: Map permission requirements per endpoint
FR-117: Highlight potential privilege escalation opportunities

Output Structure:

```
{
  "roleComparison": {
    "commonPages": ["/home", "/profile", "/settings"],
    "roleSpecific": {
      "admin": ["/admin", "/users", "/logs", "/system-config"],
      "manager": ["/reports", "/team-dashboard", "/approvals"],
      "basic_user": ["/my-tasks"],
      "guest": ["/login", "/signup", "/about"]
    },
    "permissionMatrix": {
      "/users": {
        "admin": ["read", "write", "delete"],
        "manager": ["read"],
        "basic_user": "403_forbidden",
        "guest": "redirect_to_login"
      },
      "/api/users": {
        "admin": ["GET", "POST", "PUT", "DELETE"],
        "manager": ["GET"],
        "basic_user": "401_unauthorized",
        "guest": "401_unauthorized"
      }
    }
  }
}
```

5.4 Authorization Test Generation
5.4.1 Automated Security Test Cases

FR-118: Generate authorization test suites automatically
FR-119: Privilege escalation test cases (vertical)
FR-120: Horizontal access control tests (User A accessing User B's data)
FR-121: IDOR (Insecure Direct Object Reference) test vectors
FR-122: API authorization tests per endpoint
FR-123: Missing function level access control tests

Test Case Examples:

```
{
  "authorizationTests": [
    {
      "testId": "RBAC-001",
      "category": "vertical_privilege_escalation",
      "name": "Basic user attempts to access admin panel",
      "role": "basic_user",
      "action": "GET /admin",
      "expectedResult": "403 or redirect to /login",
      "securityImplication": "critical"
    },
    {
      "testId": "RBAC-002",
      "category": "unauthorized_action",
      "name": "Manager attempts to delete user",
      "role": "manager",
      "action": "DELETE /api/users/123",
      "expectedResult": "403 Forbidden",
      "securityImplication": "high"
    },
    {
      "testId": "RBAC-003",
      "category": "horizontal_privilege_escalation",
      "name": "User A accesses User B's profile edit",
      "role": "basic_user",
      "userId": "user_A",
      "action": "GET /users/user_B/edit",
      "expectedResult": "403 or redirect to own profile",
      "securityImplication": "high"
    },
    {
      "testId": "RBAC-004",
      "category": "idor",
      "name": "Sequential ID enumeration",
      "role": "basic_user",
      "action": "GET /api/documents/{1..1000}",
      "expectedResult": "403 for documents not owned",
      "securityImplication": "medium"
    }
  ]
}
```

5.5 Subscription/Tier-Based Features

5.5.1 Freemium Model Detection

FR-124: Detect free vs. paid feature boundaries

FR-125: Identify trial period limitations

FR-126: Track usage quota restrictions

FR-127: Detect feature unlock requirements
FR-128: Identify upgrade prompts and paywalls
FR-129: Map features to subscription tiers

Tier Mapping Example:

```
{
  "subscriptionTiers": [
    {
      "tier": "free",
      "monthlyPrice": 0,
      "features": ["basic_search", "view_profile", "5_exports_per_month"],
      "restrictions": ["no_api_access", "ads_visible",
"limited_storage_100mb"],
      "upgradePrompts": ["/dashboard", "/export", "/api-docs"]
    },
    {
      "tier": "pro",
      "monthlyPrice": 29,
      "features": ["advanced_search", "unlimited_exports", "api_access",
"no_ads"],
      "restrictions": ["single_user", "email_support_only"],
      "testableUpgrades": ["bulk_operations", "custom_integrations"]
    },
    {
      "tier": "enterprise",
      "monthlyPrice": 299,
      "features": ["all_features", "team_collaboration", "sso",
"priority_support", "sla"],
      "restrictions": [],
      "exclusiveFeatures": ["audit_logs", "advanced_analytics",
"custom_branding"]
    }
  ]
}
```

5.6 Organization/Team Hierarchy
5.6.1 Multi-Tenant Architecture

FR-130: Crawl different organizations separately
FR-131: Test team/workspace isolation
FR-132: Identify shared vs. private resources
FR-133: Test cross-organization access attempts
FR-134: Detect tenant-specific branding/configuration
FR-135: Map organization-level permissions

5.7 Contextual Permissions

FR-136: Owner vs. viewer vs. editor on specific resources
FR-137: Creator privileges (delete own content)
FR-138: Time-based access (temporary permissions)
FR-139: Location-based access (IP restrictions)
FR-140: Device-based access (mobile vs. desktop features)
FR-141: Shared resource detection (shared links, invitations)
FR-142: Guest access tokens and expiring shares

6. User Interactions & State Management
6.1 JavaScript-Heavy Interactions
6.1.1 Dynamic Content Loading

FR-143: Scroll-triggered content detection and loading
FR-144: Infinite scroll handling with auto-scroll
FR-145: Lazy-loaded images and sections detection
FR-146: "Load More" button detection and clicking
FR-147: Intersection Observer-based loading detection
FR-148: Time-delayed content with configurable wait timeouts
FR-149: Skeleton screen detection and wait logic
FR-150: Mutation observer for DOM changes

6.1.2 Hover and Focus Interactions

FR-151: Dropdown menus appearing on hover
FR-152: Tooltips and popovers on hover/focus
FR-153: Mega-menus with nested content
FR-154: Context menus (right-click)
FR-155: Tab key navigation simulation
FR-156: Focus-triggered content reveal

6.1.3 JavaScript Event Simulation

FR-157: Trigger mouse events (click, hover, mouseenter, mouseleave)
FR-158: Simulate keyboard events (keypress, keydown, tab)
FR-159: Focus/blur events on form fields
FR-160: Touch events for mobile simulations
FR-161: Drag and drop interactions
FR-162: Double-click detection

6.2 Multi-Step Workflows
6.2.1 Wizard/Stepper Forms

FR-163: Detect wizard/stepper patterns
FR-164: Auto-progress through steps
FR-165: Capture all fields across all steps
FR-166: Map step relationships and dependencies
FR-167: Handle "Previous" button navigation

FR-168: Conditional step logic (if-then flows)
FR-169: Step validation and error states

6.2.2 Modal Dialog Handling

FR-170: Detect and interact with modals
FR-171: Capture modal content as separate states
FR-172: Handle nested modals
FR-173: Dismiss mechanisms (X button, backdrop click, ESC key)
FR-174: Modal trigger detection (buttons that open modals)

6.2.3 Conditional Form Fields

FR-175: Dynamic form field detection
FR-176: Show/hide logic (e.g., "Other" text field when selected)
FR-177: Dependent dropdowns (Country → State → City)
FR-178: Calculate all possible form states and combinations

6.3 Application State Management
6.3.1 Stateful Crawling

FR-179: Shopping cart state tracking (empty vs. with items)
FR-180: Search/filter state persistence
FR-181: User preferences (theme, language, layout)
FR-182: Draft/unsaved data states
FR-183: Notification/message counts
FR-184: Selected items/bulk actions state
FR-185: Comparison lists/wishlists state
FR-186: Crawl same page with different application states

State Examples:

```
{
  "applicationStates": [
    {
      "stateId": "empty_cart",
      "page": "/checkout",
      "conditions": {"cartItems": 0},
      "visibleElements": ["empty_cart_message", "continue_shopping_button"],
      "hiddenElements": ["checkout_button", "promo_code_field", "cart_summary"]
    },
    {
      "stateId": "cart_with_items",
      "page": "/checkout",
      "conditions": {"cartItems": 3, "subtotal": 150.00},
      "visibleElements": ["checkout_button", "cart_summary",
"promo_code_field", "remove_item_buttons"],
      "hiddenElements": ["empty_cart_message"]
    },
    {
```

```
      "stateId": "filters_applied",
      "page": "/products",
      "conditions": {"filters": {"category": "electronics", "price": "100-
500"}},
      "visibleElements": ["clear_filters_button", "filter_badges",
"filtered_results"],
      "url": "/products?category=electronics&price=100-500"
    }
  ]
}
```

6.3.2 State Transition Mapping

FR-187: Map initial state → actions → new state
FR-188: State machine visualization
FR-189: Unreachable states detection
FR-190: Dead-end states identification (no way out)
FR-191: Optimal path to reach each state

6.3.3 Browser History & Navigation

FR-192: Back button behavior testing
FR-193: Forward button behavior
FR-194: Breadcrumb navigation tracking
FR-195: Deep linking (URL reflects app state)
FR-196: History API usage (pushState, replaceState)
FR-197: Scroll position restoration on back
FR-198: Form data preservation on back
FR-199: Modal closure on back button

6.4 Complex Input Components
6.4.1 Rich Input Detection

FR-200: Date/time pickers (custom calendars, not native)
FR-201: Color pickers
FR-202: Range sliders (dual-handle for min-max)
FR-203: Tag/token inputs (multi-select with custom tags)
FR-204: Autocomplete/typeahead inputs
FR-205: Location pickers (map-based)
FR-206: Rating components (star ratings, thumbs up/down)
FR-207: Toggle switches
FR-208: Segmented controls
FR-209: WYSIWYG editors (TinyMCE, CKEditor, Quill)
FR-210: Markdown editors
FR-211: Code editors (Monaco, CodeMirror)
FR-212: Image annotation tools

6.4.2 File Upload Handling

FR-213: Detect all file input types
FR-214: Multiple file uploads
FR-215: Drag-and-drop zones for files
FR-216: Image croppers/editors
FR-217: File type restrictions detection (.jpg, .pdf, etc.)
FR-218: File size limits detection
FR-219: Preview functionality
FR-220: Progress indicators for uploads

6.5 Drag and Drop

FR-221: File upload drag zones detection
FR-222: Reorderable lists
FR-223: Kanban boards (drag cards between columns)
FR-224: Tree view drag-and-drop
FR-225: Image/media reordering
FR-226: Dashboard widget rearrangement

6.6 Keyboard Shortcuts

FR-227: Keyboard shortcut documentation extraction
FR-228: Common shortcuts detection (Ctrl+S, Ctrl+K, Ctrl+F)
FR-229: Arrow key navigation
FR-230: Tab navigation order tracking
FR-231: Escape key handlers
FR-232: Custom application shortcuts
FR-233: Command palette/omnibox (Cmd+K style)
FR-234: Slash commands

6.7 Bulk Operations

FR-235: Select all checkbox detection
FR-236: Multi-select rows (checkboxes)
FR-237: Bulk delete operations
FR-238: Bulk edit operations
FR-239: Bulk export
FR-240: Bulk assign/transfer
FR-241: Batch processing status indicators

6.8 Undo/Redo & History

FR-242: Undo/redo buttons detection
FR-243: Version history tracking
FR-244: Draft auto-save and restore
FR-245: Revision tracking
FR-246: Rollback capabilities

6.9 Copy/Paste Functionality

FR-247: Copy to clipboard buttons
FR-248: Paste handling detection
FR-249: Copy shareable links
FR-250: Copy code snippets
FR-251: Copy table data

6.10 Collaborative/Real-Time Features

FR-252: Live cursors (see other users)
FR-253: Collaborative editing detection (Google Docs style)
FR-254: Live commenting
FR-255: Presence indicators (who's online)
FR-256: Real-time notifications
FR-257: Live chat/messaging
FR-258: Conflict resolution UI

7. Authentication & Security
7.1 Authentication Mechanisms
7.1.1 Login Detection and Handling

FR-259: Auto-detect login forms based on field names and patterns
FR-260: Support credential injection for authenticated crawling
FR-261: Maintain session state across pages (cookies, tokens)
FR-262: Detect session timeout and re-authenticate

7.1.2 Authentication Types

FR-263: Form-based authentication (username/password)
FR-264: HTTP Basic/Digest authentication
FR-265: JWT token-based authentication
FR-266: OAuth 2.0 flows (Google, Facebook, GitHub, LinkedIn)
FR-267: API key authentication
FR-268: SSO (Single Sign-On) - SAML
FR-269: Social login button detection and flow mapping

7.1.3 Multi-Factor Authentication (MFA)

FR-270: SMS/Email OTP detection
FR-271: TOTP/Authenticator app code inputs
FR-272: Push notification authentication
FR-273: Biometric authentication prompts
FR-274: Backup codes
FR-275: Manual intervention mode for MFA (pause and notify)

7.1.4 reCAPTCHA Handling

FR-276: Detect reCAPTCHA v2 and v3
FR-277: Integration with CAPTCHA solving services (2Captcha, Anti-Captcha)
FR-278: Manual intervention mode (pause crawler)

FR-279: Accessibility alternatives when available
FR-280: Track CAPTCHA locations for testing purposes

7.2 Session Management

FR-281: Detect session timeout and handle re-authentication
FR-282: Support multiple simultaneous user sessions (different roles)
FR-283: Preserve authentication state across crawl sessions
FR-284: Session fixation detection
FR-285: Token refresh mechanism handling
FR-286: Remember me functionality testing
FR-287: Auto-logout after inactivity detection
FR-288: Concurrent session handling

7.3 Security Feature Detection
7.3.1 Security Headers

FR-289: Content Security Policy (CSP) rules extraction
FR-290: Subresource Integrity (SRI) detection
FR-291: CORS configuration analysis
FR-292: X-Frame-Options detection
FR-293: Referrer Policy
FR-294: Permissions Policy
FR-295: HSTS (Strict-Transport-Security)
FR-296: X-Content-Type-Options

7.3.2 Cookie Analysis

FR-297: Cookie attributes (HttpOnly, Secure, SameSite)
FR-298: TLS/SSL information

FR-299: Cookie consent banners and GDPR compliance
FR-300: Session cookie vs. persistent cookie identification
FR-301: Third-party cookie detection

7.3.3 Privacy & Compliance

FR-302: Privacy policy link detection
FR-303: Terms of service detection
FR-304: Cookie consent flows (accept, reject, customize)
FR-305: GDPR compliance elements (data deletion, access requests)
FR-306: Age verification gates
FR-307: PII (Personally Identifiable Information) field detection
FR-308: Sensitive data inputs (SSN, credit card, health info)

7.4 Email & Notification Flows
7.4.1 Email Integration

FR-309: Email verification links detection
FR-310: Password reset flow tracking
FR-311: Invitation acceptance flows
FR-312: Subscription confirmation flows
FR-313: Two-step verification via email
FR-314: Magic link authentication
FR-315: Integration with temp email services (Mailinator, Guerrilla Mail)
FR-316: Extract links from emails for workflow completion

7.4.2 Notification Channels

FR-317: Browser push notification detection
FR-318: SMS verification code inputs
FR-319: In-app notification detection
FR-320: Desktop notification prompts
FR-321: Notification preferences/settings pages

8. Data Extraction & Analysis
8.1 Testable Elements Extraction
8.1.1 Forms

FR-322: Extract all forms (standard and non-standard)
FR-323: Form ID and name attributes
FR-324: Action URL and method (GET/POST/PUT/DELETE)
FR-325: Input fields with:

Name, ID, type
Required attribute
Validation patterns (regex, min/max length)
Placeholder text
Default values

FR-326: Submit buttons and their text
FR-327: CSRF token detection
FR-328: Hidden fields
FR-329: Autocomplete attributes
FR-330: Form encoding types (multipart/form-data, etc.)

8.1.2 Input Elements

FR-331: Text inputs, textareas
FR-332: Email, tel, url, number inputs
FR-333: Password fields
FR-334: Dropdowns and select boxes
FR-335: Checkboxes and radio buttons
FR-336: File upload inputs
FR-337: Hidden fields
FR-338: Range sliders

FR-339: Color pickers
FR-340: Date/time inputs (native and custom)

8.1.3 Buttons & Actions

FR-341: Submit buttons
FR-342: Action buttons (delete, edit, approve, etc.)
FR-343: Navigation buttons
FR-344: JavaScript event handlers (onclick, onsubmit)
FR-345: Disabled state detection
FR-346: Loading states on buttons

8.1.4 Links & Navigation

FR-347: Internal links
FR-348: External links
FR-349: Anchor links (#)
FR-350: Dynamic navigation (JavaScript-based)
FR-351: Breadcrumb navigation
FR-352: Pagination links
FR-353: Download links

8.2 API Endpoint Discovery
8.2.1 REST API Detection

FR-354: Capture all XHR/Fetch requests
FR-355: Extract endpoint URLs
FR-356: HTTP methods used (GET, POST, PUT, PATCH, DELETE)
FR-357: Request parameters (query, body, headers)
FR-358: Response structure and status codes
FR-359: Authentication requirements per endpoint
FR-360: Rate limiting detection
FR-361: API versioning (v1, v2, etc.)

8.2.2 GraphQL Detection

FR-362: GraphQL endpoint identification
FR-363: Query and mutation extraction
FR-364: Schema introspection
FR-365: Variables and input types
FR-366: Subscription detection (WebSocket-based)

8.2.3 WebSocket & Real-Time

FR-367: WebSocket connection detection
FR-368: Message patterns and structure
FR-369: Server-Sent Events (SSE)
FR-370: Long polling detection
FR-371: Real-time data updates tracking

8.2.4 Third-Party APIs

FR-372: Analytics APIs (Google Analytics, Mixpanel, Segment)
FR-373: Payment gateway APIs (Stripe, PayPal)
FR-374: CDN resources
FR-375: Social media embeds
FR-376: Map services (Google Maps, Mapbox)
FR-377: Chat widgets (Intercom, Drift, Zendesk)

8.3 Metadata Collection
8.3.1 Page Metadata

FR-378: Title, description, keywords
FR-379: Response codes and headers
FR-380: Load time and page size
FR-381: Technologies detected (Wappalyzer-style detection)
FR-382: Framework identification (React, Vue, Angular, WordPress)
FR-383: JavaScript libraries in use
FR-384: CSS frameworks (Bootstrap, Tailwind)

8.3.2 Structured Data

FR-385: JSON-LD extraction
FR-386: Microdata and Schema.org markup
FR-387: OpenGraph tags
FR-388: Twitter Card tags
FR-389: Data attributes (data-*)
FR-390: Embedded JSON in │ │
│ │ [View Details] [Edit] [Export] │ │
│ │ │ │
│ │ ⚠ RBAC-001 Privilege escalation - User→Admin High │ │
│ │ Test: Basic user accessing /admin │ │
│ │ [View Details] [Edit] [Export] │ │
│ │ │ │
│ │ [Show More...] │ │
│ └──┘ │
│ │
│ ┌─── API Tests (1,678) ──────────────────────────────────────┐ │
│ │ API-001 GET /api/user/profile with token High │ │
│ │ API-002 GET /api/user/profile without token High │ │
│ │ [Show More...] │ │
│ └──┘ │
│ │
│ ┌─── Performance Tests (245) ──────────────────────────┐ │
│ │ PERF-001 Homepage load time < 3s Medium │ │
│ │ [Show More...] │ │
│ └──┘ │

```
│ │
│ ┌─── Accessibility Tests (112) ──────────────────────────┐ │
│ │ A11Y-001 All images have alt text High │ │
│ │ [Show More...] │ │
│ └────────────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────────┘
```

```
### 13.6 Test Case Detail View
```

```
┌──────────────────────────────────────────────────────────────┐
│ ← Back to Tests │
│ Test Case: FUNC-001 - Login with valid credentials │
├──────────────────────────────────────────────────────────────┤
│ │
│ Category: Functional > Form Submission │
│ Priority: ⚠ Critical │
│ Target: #login-form on https://example.com │
│ Status: Not Run │
│ │
│ ┌─── Test Steps ──────────────────────────────────────────┐ │
│ │ 1. Navigate to https://example.com │ │
│ │ 2. Locate email input field (name="email") │ │
│ │ 3. Enter test data: "user@example.com" │ │
│ │ 4. Locate password input field (name="password") │ │
│ │ 5. Enter test data: "validPassword123!" │ │
│ │ 6. Click submit button (button[type='submit']) │ │
│ │ 7. Wait for navigation │ │
│ │ 8. Assert: Current URL contains "/dashboard" │ │
│ │ 9. Assert: Session cookie "session_id" is set │ │
│ └─────────────────────────────────────────────────────────┘ │
│ │
│ ┌─── Expected Result ─────────────────────────────────────┐ │
│ │ User is successfully logged in and redirected to │ │
│ │ /dashboard with valid session cookie set │ │
│ └─────────────────────────────────────────────────────────┘ │
│ │
│ ┌─── Test Data ───────────────────────────────────────────┐ │
│ │ Email: user@example.com │ │
│ │ Password: validPassword123! │ │
│ │ [Edit Test Data] │ │
│ └─────────────────────────────────────────────────────────┘ │
│ │
│ ┌─── Assertions ──────────────────────────────────────────┐ │
```

```
│ │ ✓ Status code: 200 │ │
│ │ ✓ Redirect to: /dashboard │ │
│ │ ✓ Cookie set: session_id (HttpOnly, Secure) │ │
│ │ ✓ Response time: < 2000ms │ │
│ └──────────────────────────────────────────────────────────────┘ │
│ │
│ ┌─── Export Options ──────────────────────────────────────────┐ │
│ │ Format: [Selenium ▼] │ │
│ │ Language: [Python ▼] │ │
│ │ [Copy to Clipboard] [Download] │ │
│ └──────────────────────────────────────────────────────────────┘ │
│ │
│ [Run Test Now] [Edit Test] [Duplicate] [Delete] │
└──────────────────────────────────────────────────────────────────┘
```

```
### 13.7 Export Dialog
```

```
┌──────────────────────────────────────────────────┐
│ Export Crawl Data │
├──────────────────────────────────────────────────┤
│ │
│ Format: │
│ ○ JSON (Complete) │
│ ○ JSON (Summary) │
│ ○ CSV (Tabular) │
│ ○ Selenium Scripts (Python) │
│ ○ Playwright Scripts (JavaScript) │
│ ○ Cypress Tests (JavaScript) │
│ ○ Postman Collection │
│ ○ OpenAPI/Swagger Spec │
│ ○ k6 Load Test │
│ ○ JMeter Test Plan │
│ ○ Markdown Report │
│ ○ HTML Report │
│ ○ PDF Report │
│ │
│ Include: │
│ ☑ Nodes (Pages) │
│ ☑ Edges (Links) │
│ ☑ Test Cases │
│ ☑ RBAC Analysis │
│ ☑ API Endpoints │
│ ☑ Security Findings │
```

```
│ ☑ Performance Metrics │
│ ☐ Screenshots │
│ │
│ Filter by Role: │
│ [All Roles ▼] │
│ │
│ [Preview] [Cancel] [Export] │
└──────────────────────────────────────────────────────┘
```

13.8 Settings Page

```
┌──────────────────────────────────────────────────────────────────────────┐
│ Settings │
├──────────────────────────────────────────────────────────────────────────┤
│ │
│ [General] [Crawling] [Authentication] [Export] [Advanced] │
│ │
│ ┌──── General Settings ──────────────────────────────────┐ │
│ │ │ │
│ │ Theme: │ │
│ │ ○ Light ● Dark ○ System │ │
│ │ │ │
│ │ Language: │ │
│ │ [English ▼] │ │
│ │ │ │
│ │ Notifications: │ │
│ │ ☑ Email notifications on crawl completion │ │
│ │ ☑ Browser notifications │ │
│ │ ☐ Slack webhook integration │ │
│ │ │ │
│ │ Auto-save: │ │
│ │ ☑ Save progress every [100] pages │ │
│ │ │ │
│ └──────────────────────────────────────────────────────┘ │
│ │
│ ┌──── Default Crawl Settings ────────────────────────────┐ │
│ │ │ │
│ │ Max Depth: [3 ] │ │
│ │ Max Pages: [1000] │ │
│ │ Timeout: [30] seconds │ │
│ │ Concurrent Requests: [5 ] │ │
│ │ Request Delay: [500] ms │ │
│ │ │ │
```

```
│ │ Default Behavior: │ │
│ │ ☑ Execute JavaScript │ │
│ │ ☑ Follow external links │ │
│ │ ☑ Capture screenshots │ │
│ │ ☑ Extract validation rules │ │
│ │ │ │
│ └──────────────────────────────────────────────────────────────────────┘ │
│ │
│ [Save Settings] [Reset to Defaults] │
└──────────────────────────────────────────────────────────────────────────────┘
```

````
### 13.9 Mobile Responsive Views

**FR-754**: Responsive design for tablets (768px+)
**FR-755**: Mobile-optimized views (320px+)
**FR-756**: Touch-friendly controls
**FR-757**: Simplified graph view for mobile
**FR-758**: Swipe gestures for navigation

---

## 14. API Specifications

### 14.1 Authentication

**FR-759**: JWT-based authentication
**FR-760**: Refresh token mechanism
**FR-761**: API key support for automation
**FR-762**: OAuth 2.0 for third-party integrations
```typescript
// Login
POST /auth/login
Request: {
  email: string;
  password: string;
}
Response: {
  accessToken: string;
  refreshToken: string;
  expiresIn: number;
  user: {
    id: string;
    email: string;
    name: string;
    role: string;
  };
}

// Refresh Token
````
````

```
POST /auth/refresh
Request: {
  refreshToken: string;
}
Response: {
  accessToken: string;
  expiresIn: number;
}

// API Key Creation
POST /auth/api-keys
Request: {
  name: string;
  permissions: string[];
  expiresAt?: string;
}
Response: {
  id: string;
  key: string; // Only shown once
  name: string;
  createdAt: string;
}
```

14.2 Rate Limiting

FR-763: Rate limiting per API key/user
FR-764: Different limits for authenticated vs. unauthenticated
FR-765: Rate limit headers in responses

```
Rate Limits:
- Unauthenticated: 10 requests/minute
- Authenticated: 100 requests/minute
- API Key: 1000 requests/minute

Response Headers:
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1699564800
```

14.3 Error Handling

FR-766: Consistent error response format
FR-767: Error codes and messages
FR-768: Stack traces in development mode only

```
interface ErrorResponse {
  error: {
```

```
    code: string;
    message: string;
    details?: any;
    timestamp: string;
    requestId: string;
  };
}

// Example Error Codes:
{
  "INVALID_URL": "The provided URL is invalid",
  "CRAWL_NOT_FOUND": "Crawl job not found",
  "RATE_LIMIT_EXCEEDED": "Rate limit exceeded",
  "UNAUTHORIZED": "Authentication required",
  "FORBIDDEN": "Insufficient permissions",
  "VALIDATION_ERROR": "Request validation failed",
  "INTERNAL_ERROR": "Internal server error"
}
```

14.4 Pagination

FR-769: Cursor-based pagination for large datasets
FR-770: Configurable page size (max 100)

```
// List Nodes with Pagination
GET /crawls/:id/nodes?cursor=abc123&limit=50

Response: {
  data: Node[];
  pagination: {
    cursor: string | null; // null if last page
    hasMore: boolean;
    total: number;
  };
}
```

14.5 Webhooks

FR-771: Webhook registration for events
FR-772: Event types: crawl.started, crawl.completed, crawl.failed, node.discovered
FR-773: Webhook signature verification
FR-774: Retry logic for failed webhooks (3 attempts)

```
// Register Webhook
POST /webhooks
Request: {
  url: string;
```

```
  events: string[];
  secret: string; // For signature verification
}

// Webhook Payload
POST {webhook_url}
Headers: {
  "X-Webhook-Signature": "sha256=...",
  "X-Webhook-Id": "webhook_123",
  "X-Event-Type": "crawl.completed"
}
Body: {
  eventType: "crawl.completed";
  timestamp: string;
  data: {
    crawlId: string;
    status: string;
    summary: any;
  };
}
```

14.6 GraphQL API (Optional)

FR-775: GraphQL endpoint for flexible queries
FR-776: Real-time subscriptions for crawl updates

```
type Query {
  crawl(id: ID!): Crawl
  crawls(
    limit: Int = 10
    offset: Int = 0
    status: CrawlStatus
  ): [Crawl!]!

  node(id: ID!): Node
  nodes(
    crawlId: ID!
    type: NodeType
    role: String
    limit: Int = 50
  ): [Node!]!
}

type Mutation {
  createCrawl(input: CreateCrawlInput!): Crawl!
  pauseCrawl(id: ID!): Crawl!
  resumeCrawl(id: ID!): Crawl!
  stopCrawl(id: ID!): Crawl!
}
```

```
type Subscription {
  crawlProgress(crawlId: ID!): CrawlProgress!
  nodeDiscovered(crawlId: ID!): Node!
}

type Crawl {
  id: ID!
  baseUrl: String!
  status: CrawlStatus!
  progress: Progress!
  nodes: [Node!]!
  edges: [Edge!]!
  stats: Stats!
  createdAt: DateTime!
  updatedAt: DateTime!
}
```

---

15. Testing Requirements

15.1 Unit Tests

FR-777: 80%+ code coverage for core modules
FR-778: Test URL normalization and filtering
FR-779: Test authentication flow handling
FR-780: Test data extraction accuracy
FR-781: Test graph algorithms (shortest path, cycle detection)
FR-782: Test pattern detection logic
FR-783: Test content fingerprinting
FR-784: Test validation rule extraction

Test Frameworks:

- Node.js: Jest, Mocha + Chai
- Python: pytest

```
// Example Unit Test
describe('URL Normalization', () => {
  it('should normalize trailing slashes', () => {
    expect(normalizeUrl('https://example.com/'))
      .toBe('https://example.com');
  });

  it('should normalize parameter order', () => {
    expect(normalizeUrl('https://example.com?b=2&a=1'))
      .toBe('https://example.com?a=1&b=2');
  });

  it('should remove tracking parameters', () => {
```

```
    expect(normalizeUrl('https://example.com?utm_source=test'))
      .toBe('https://example.com');
  });
});
```

15.2 Integration Tests

FR-785: End-to-end crawl scenarios
FR-786: Test with real websites (test environment)
FR-787: Authentication with test services
FR-788: API endpoint discovery validation
FR-789: Export format validation
FR-790: Multi-role crawling tests
FR-791: Database integration tests
FR-792: Redis queue tests

Test Scenarios:

```
describe('E2E Crawl Tests', () => {
  it('should crawl a simple static site', async () => {
    const crawl = await startCrawl({
      baseUrl: 'https://test-site.example.com',
      maxDepth: 2,
      maxPages: 50
    });

    await waitForCompletion(crawl.id);

    const result = await getCrawlResult(crawl.id);
    expect(result.stats.totalPages).toBeGreaterThan(0);
    expect(result.nodes.length).toBeGreaterThan(0);
  });

  it('should handle authentication', async () => {
    const crawl = await startCrawl({
      baseUrl: 'https://test-site.example.com',
      roles: [{
        id: 'user',
        credentials: {
          username: 'test@example.com',
          password: 'testpass123'
        }
      }]
    });

    await waitForCompletion(crawl.id);

    const protectedPages = await getNodesByType(crawl.id, 'protected');
    expect(protectedPages.length).toBeGreaterThan(0);
```

```
  });
});
```

15.3 Performance Tests

FR-793: Benchmark crawl speed (pages/minute)
FR-794: Memory leak detection
FR-795: Concurrent request handling
FR-796: Graph rendering performance with 1000+ nodes
FR-797: Database query optimization
FR-798: Large dataset export performance

Performance Benchmarks:

```
describe('Performance Tests', () => {
  it('should crawl 1000 pages in under 20 minutes', async () => {
    const startTime = Date.now();

    const crawl = await startCrawl({
      baseUrl: 'https://large-site.example.com',
      maxPages: 1000
    });

    await waitForCompletion(crawl.id);

    const duration = Date.now() - startTime;
    expect(duration).toBeLessThan(20 * 60 * 1000);
  });

  it('should handle 10 concurrent crawls', async () => {
    const crawls = await Promise.all(
      Array.from({ length: 10 }, () =>
        startCrawl({ baseUrl: 'https://test.example.com' })
      )
    );

    await Promise.all(
      crawls.map(c => waitForCompletion(c.id))
    );

    // All should complete without errors
    crawls.forEach(c => {
      expect(c.status).toBe('completed');
    });
  });
});
```

15.4 Security Tests

FR-799: Credential storage security tests
FR-800: SQL injection prevention in inputs
FR-801: XSS prevention in displayed content
FR-802: CSRF token handling
FR-803: API authentication tests
FR-804: Rate limiting tests
FR-805: Input validation tests

15.5 Compatibility Tests

FR-806: Test on different browsers (Chrome, Firefox, Safari, Edge)
FR-807: Test on different OS (Windows, macOS, Linux)
FR-808: Test with different web frameworks (React, Vue, Angular, WordPress)
FR-809: Test with SPA vs. traditional sites
FR-810: Test with different authentication methods

15.6 Test Automation

FR-811: CI/CD pipeline integration (GitHub Actions, GitLab CI, Jenkins)
FR-812: Automated testing on every commit
FR-813: Nightly regression test suite
FR-814: Performance regression detection
FR-815: Automated security scanning (Snyk, OWASP Dependency-Check)

Example CI Configuration:

```yaml
# .github/workflows/test.yml
name: Test Suite

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest

    services:
      postgres:
        image: postgres:15
        env:
          POSTGRES_PASSWORD: postgres
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
      redis:
        image: redis:7
        options: >-
          --health-cmd "redis-cli ping"
          --health-interval 10s
```

```yaml
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: npm ci

      - name: Run unit tests
        run: npm run test:unit

      - name: Run integration tests
        run: npm run test:integration
        env:
          DATABASE_URL: postgresql://postgres:postgres@localhost:5432/test
          REDIS_URL: redis://localhost:6379

      - name: Run E2E tests
        run: npm run test:e2e

      - name: Upload coverage
        uses: codecov/codecov-action@v3
        with:
          files: ./coverage/lcov.info
```

---

16. Deployment & Operations

16.1 Deployment Options

16.1.1 Standalone Desktop Application

FR-816: Electron-based desktop app
FR-817: Auto-updater for new versions
FR-818: Local SQLite database
FR-819: Portable mode (run from USB)

16.1.2 Web-Based (SaaS)

FR-820: Multi-tenant architecture
FR-821: User registration and authentication
FR-822: Subscription management
FR-823: Usage quotas per plan
FR-824: Admin dashboard for service monitoring

16.1.3 Self-Hosted (On-Premise)

FR-825: Docker Compose for easy setup
FR-826: Kubernetes Helm charts
FR-827: Installation scripts (bash, PowerShell)
FR-828: Backup and restore procedures
FR-829: Migration tools for upgrades

16.2 System Requirements

Minimum (Desktop/Small Sites):

- **CPU**: 2 cores (Intel i3 or equivalent)
- **RAM**: 4GB
- **Storage**: 10GB SSD
- **OS**: Windows 10+, macOS 11+, Ubuntu 20.04+
- **Browser**: Chrome/Chromium 90+ installed

Recommended (Production/Large Sites):

- **CPU**: 4+ cores (Intel i5/AMD Ryzen 5 or better)
- **RAM**: 8GB+
- **Storage**: 50GB+ SSD
- **Network**: 100 Mbps+ connection
- **OS**: Linux Ubuntu 22.04 LTS (server)

Enterprise (Distributed/Massive Sites):

- **App Servers**: 2+ instances (4 cores, 8GB each)
- **Worker Nodes**: 5+ instances (4 cores, 16GB each)
- **Database**: PostgreSQL (8 cores, 32GB, SSD storage)
- **Cache**: Redis (4 cores, 16GB)
- **Storage**: S3-compatible object storage (1TB+)

16.3 Installation

16.3.1 Docker Installation (Recommended)

```
# Clone repository
git clone https://github.com/example/web-crawler.git
cd web-crawler

# Configure environment
cp .env.example .env
# Edit .env with your settings

# Start services
docker-compose up -d

# Access UI at http://localhost:3000
# API available at http://localhost:3000/api
```

16.3.2 Manual Installation

```
# Prerequisites
# - Node.js 18+ or Python 3.10+
# - PostgreSQL 14+
# - Redis 7+
# - Chrome/Chromium

# Install dependencies
npm install  # or: pip install -r requirements.txt

# Setup database
npm run db:migrate

# Start services
npm run start:api &     # API server
npm run start:worker &  # Crawler workers
npm run start:frontend  # Frontend (dev mode)

# Production build
npm run build
npm run start:prod
```

16.4 Configuration

FR-830: Environment variable configuration
FR-831: Configuration file support (YAML, JSON)
FR-832: Hot-reload configuration changes
FR-833: Configuration validation on startup

Environment Variables:

```
# Application
NODE_ENV=production
PORT=3000
API_BASE_URL=https://api.example.com

# Database
DATABASE_URL=postgresql://user:pass@host:5432/crawler
DATABASE_POOL_SIZE=20

# Redis
REDIS_URL=redis://host:6379
REDIS_PASSWORD=secret

# Storage
STORAGE_TYPE=s3  # local, s3
```

```
S3_BUCKET=crawler-data
S3_REGION=us-east-1
S3_ACCESS_KEY=xxx
S3_SECRET_KEY=xxx

# Security
JWT_SECRET=your-secret-key-here
JWT_EXPIRATION=24h
API_RATE_LIMIT=100

# Crawler
MAX_CONCURRENT_CRAWLS=10
MAX_WORKERS_PER_CRAWL=5
DEFAULT_TIMEOUT=30000
DEFAULT_MAX_PAGES=1000

# Notifications
SMTP_HOST=smtp.example.com
SMTP_PORT=587
SMTP_USER=noreply@example.com
SMTP_PASS=password
SLACK_WEBHOOK_URL=https://hooks.slack.com/...

# Monitoring
SENTRY_DSN=https://xxx@sentry.io/xxx
LOG_LEVEL=info
```

16.5 Monitoring & Logging

FR-834: Structured logging (JSON format)
FR-835: Log levels (debug, info, warn, error)
FR-836: Log aggregation (ELK stack, CloudWatch, Datadog)
FR-837: Application metrics (Prometheus, Grafana)
FR-838: Error tracking (Sentry)
FR-839: Uptime monitoring (Pingdom, UptimeRobot)
FR-840: Health check endpoints
Metrics to Track:

```
{
  "system": {
    "uptime": 3600000,
    "memoryUsage": {
      "rss": 512000000,
      "heapUsed": 256000000,
      "heapTotal": 384000000
    },
    "cpuUsage": 45.2
  },
  "crawler": {
```

```
    "activeCrawls": 5,
    "queuedJobs": 12,
    "completedToday": 45,
    "failedToday": 2,
    "averagePagesPerMinute": 8.5,
    "averageResponseTime": 234
  },
  "database": {
    "activeConnections": 15,
    "totalQueries": 125678,
    "slowQueries": 3,
    "averageQueryTime": 12
  },
  "workers": {
    "total": 5,
    "busy": 3,
    "idle": 2,
    "crashed": 0
  }
}
```

Health Check Endpoints:

```
// Basic health check
GET /health
Response: {
  status: "healthy" | "degraded" | "unhealthy",
  timestamp: "2025-10-22T12:00:00Z",
  uptime: 3600000,
  version: "2.0.0"
}

// Detailed health check
GET /health/detailed
Response: {
  status: "healthy",
  checks: {
    database: { status: "healthy", latency: 12 },
    redis: { status: "healthy", latency: 2 },
    workers: { status: "healthy", active: 5 },
    storage: { status: "healthy", available: true }
  }
}

// Readiness probe (Kubernetes)
GET /ready
Response: 200 OK (if ready to accept traffic)

// Liveness probe (Kubernetes)
GET /live
Response: 200 OK (if application is alive)
```

Logging Configuration:

```
// Winston logger example
const logger = winston.createLogger({
  level: process.env.LOG_LEVEL || 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  defaultMeta: {
    service: 'web-crawler',
    version: '2.0.0'
  },
  transports: [
    new winston.transports.Console(),
    new winston.transports.File({
      filename: 'logs/error.log',
      level: 'error'
    }),
    new winston.transports.File({
      filename: 'logs/combined.log'
    })
  ]
});

// Log example
logger.info('Crawl started', {
  crawlId: 'abc123',
  baseUrl: 'https://example.com',
  userId: 'user456'
});
16.6 Backup & Recovery
FR-841: Automated database backups (daily)
FR-842: Point-in-time recovery capability
FR-843: Backup retention policy (30 days default)
FR-844: Backup verification and testing
FR-845: Disaster recovery plan
FR-846: Data export before major upgrades
Backup Scripts:
bash#!/bin/bash
# Automated backup script

TIMESTAMP=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/backups/crawler"
DB_NAME="crawler"

# PostgreSQL backup
pg_dump -h localhost -U postgres $DB_NAME | \
  gzip > "$BACKUP_DIR/db_${TIMESTAMP}.sql.gz"
```

```
# Redis backup
redis-cli BGSAVE
cp /var/lib/redis/dump.rdb "$BACKUP_DIR/redis_${TIMESTAMP}.rdb"

# Upload to S3 (optional)
aws s3 cp "$BACKUP_DIR/" "s3://backups-bucket/crawler/" --recursive

# Cleanup old backups (keep last 30 days)
find $BACKUP_DIR -type f -mtime +30 -delete

echo "Backup completed: $TIMESTAMP"
16.7 Scaling Strategies
16.7.1 Vertical Scaling
FR-847: Increase CPU/RAM for single instance
FR-848: Optimize database queries
FR-849: Connection pooling
FR-850: Caching frequently accessed data
16.7.2 Horizontal Scaling
FR-851: Multiple API server instances (load balanced)
FR-852: Multiple worker instances (queue-based distribution)
FR-853: Database read replicas
FR-854: Redis cluster for distributed caching
FR-855: CDN for static assets and screenshots
Kubernetes Deployment:
yaml# crawler-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: crawler-api
spec:
  replicas: 3
  selector:
    matchLabels:
      app: crawler-api
  template:
    metadata:
      labels:
        app: crawler-api
    spec:
      containers:
      - name: api
        image: crawler:2.0.0
        ports:
        - containerPort: 3000
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: crawler-secrets
              key: database-url
        - name: REDIS_URL
          valueFrom:
```

```
            secretKeyRef:
              name: crawler-secrets
              key: redis-url
        resources:
          requests:
            memory: "2Gi"
            cpu: "1000m"
          limits:
            memory: "4Gi"
            cpu: "2000m"
        livenessProbe:
          httpGet:
            path: /live
            port: 3000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /ready
            port: 3000
          initialDelaySeconds: 10
          periodSeconds: 5
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: crawler-worker
spec:
  replicas: 5
  selector:
    matchLabels:
      app: crawler-worker
  template:
    metadata:
      labels:
        app: crawler-worker
    spec:
      containers:
      - name: worker
        image: crawler:2.0.0
        command: ["npm", "run", "worker"]
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: crawler-secrets
              key: database-url
        - name: REDIS_URL
          valueFrom:
            secretKeyRef:
              name: crawler-secrets
              key: redis-url
        resources:
```

```
          requests:
            memory: "4Gi"
            cpu: "2000m"
          limits:
            memory: "8Gi"
            cpu: "4000m"
---
apiVersion: v1
kind: Service
metadata:
  name: crawler-api
spec:
  selector:
    app: crawler-api
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
16.8 Security Hardening
FR-856: HTTPS/TLS encryption (Let's Encrypt)
FR-857: Security headers (HSTS, CSP, etc.)
FR-858: SQL injection prevention (parameterized queries)
FR-859: XSS prevention (input sanitization, CSP)
FR-860: CSRF protection
FR-861: Rate limiting per IP/user
FR-862: DDoS protection (Cloudflare, AWS Shield)
FR-863: Regular security audits
FR-864: Dependency vulnerability scanning
FR-865: Secrets management (Vault, AWS Secrets Manager)
FR-866: Network isolation (VPC, private subnets)
FR-867: Firewall rules (allow only necessary ports)
Security Headers Configuration:
javascript// Express middleware
app.use(helmet({
  contentSecurityPolicy: {
    directives: {
      defaultSrc: ["'self'"],
      styleSrc: ["'self'", "'unsafe-inline'"],
      scriptSrc: ["'self'"],
      imgSrc: ["'self'", "data:", "https:"],
      connectSrc: ["'self'", "https://api.example.com"]
    }
  },
  hsts: {
    maxAge: 31536000,
    includeSubDomains: true,
    preload: true
  },
  referrerPolicy: {
    policy: 'strict-origin-when-cross-origin'
  },
  xssFilter: true,
```

```
  noSniff: true,
  frameguard: { action: 'deny' }
}));
16.9 Maintenance Procedures
FR-868: Database maintenance (VACUUM, ANALYZE)
FR-869: Log rotation and archival
FR-870: Cache clearing procedures
FR-871: Orphaned data cleanup
FR-872: Performance tuning based on metrics
FR-873: Version upgrade procedures
Maintenance Scripts:
bash#!/bin/bash
# Weekly maintenance script

echo "Starting maintenance..."

# Database maintenance
psql -U postgres -d crawler -c "VACUUM ANALYZE;"

# Clear old crawl data (older than 90 days)
psql -U postgres -d crawler -c "
  DELETE FROM crawl_jobs
  WHERE completed_at < NOW() - INTERVAL '90 days';
"

# Clear Redis cache
redis-cli FLUSHDB

# Rotate logs
logrotate /etc/logrotate.d/crawler

# Clear temporary files
find /tmp/crawler-* -type f -mtime +7 -delete

# Check disk space
df -h | grep -E "/$|/var/lib/postgresql"

echo "Maintenance completed"
16.10 Disaster Recovery
FR-874: Documented recovery procedures
FR-875: Regular disaster recovery drills
FR-876: Backup restoration testing
FR-877: Failover procedures for database
FR-878: Data replication across regions
FR-879: RTO (Recovery Time Objective): < 4 hours
FR-880: RPO (Recovery Point Objective): < 1 hour
Recovery Procedures:
bash# Database recovery from backup
#!/bin/bash

BACKUP_FILE="$1"

# Stop application
```

```
docker-compose down

# Restore database
gunzip -c "$BACKUP_FILE" | \
  psql -U postgres -d crawler

# Verify data integrity
psql -U postgres -d crawler -c "
  SELECT COUNT(*) FROM crawl_jobs;
  SELECT COUNT(*) FROM nodes;
"

# Start application
docker-compose up -d

# Verify health
curl http://localhost:3000/health

echo "Recovery completed. Please verify functionality."

17. Success Criteria
17.1 Launch Criteria (MVP)
Must Have:

SC-001: Successfully crawl 20 diverse test websites
SC-002: Handle 3 different authentication types (form-based, OAuth, API key)
SC-003: Detect and extract elements from poor semantic HTML (95%+ accuracy)
SC-004: Pattern-based sampling for sites with 10,000+ pages
SC-005: Multi-role crawling with permission matrix generation
SC-006: Generate usable test data for 90%+ of discovered forms
SC-007: Export to at least 5 formats (JSON, Selenium, Playwright, Postman, CSV)
SC-008: Graph visualization with 500+ nodes at acceptable performance (30+ FPS)
SC-009: Process 100-page site in under 5 minutes
SC-010: Zero critical security vulnerabilities
SC-011: Documentation covering 100% of features

17.2 User Acceptance Criteria
Usability:

SC-012: 80%+ user satisfaction in initial surveys
SC-013: Average task completion time < 3 minutes (for basic crawl)
SC-014: <10% error rate in production crawls
SC-015: Users can start first crawl within 5 minutes of signing up
SC-016: Documentation rated 4+ stars (out of 5)

Reliability:

SC-017: 95%+ crawl completion rate
SC-018: < 1% false positive rate on element detection
SC-019: < 5% false negative rate on element detection
SC-020: Zero data loss incidents in production

Performance:
```

```
SC-021: Sub-second graph interactions (zoom, pan, click)
SC-022: API response time p95 < 500ms
SC-023: Export generation < 30 seconds for 1000 pages
SC-024: Support 100 concurrent users (SaaS mode)

17.3 Technical Performance Metrics
Crawling:

SC-025: 50+ pages per minute on average hardware
SC-026: Successfully handle sites with 10,000+ pages via sampling
SC-027: Memory usage < 2GB for 5,000-page site
SC-028: Pattern detection accuracy > 95%
SC-029: Content fingerprinting < 100ms per page

Detection Accuracy:

SC-030: Form detection: 95%+ accuracy
SC-031: Button detection (including non-standard): 90%+ accuracy
SC-032: API endpoint discovery: 85%+ accuracy
SC-033: Authentication flow detection: 90%+ accuracy

Scalability:

SC-034: Handle 10 concurrent crawl jobs without degradation
SC-035: Database query time p95 < 100ms
SC-036: Redis cache hit rate > 80%

17.4 Business Metrics (SaaS Mode)
Adoption:

SC-037: 1,000 registered users in first 3 months
SC-038: 100 active users (weekly active)
SC-039: 20% conversion rate (free to paid)

Engagement:

SC-040: Average 5 crawls per user per month
SC-041: 70% user retention after 30 days
SC-042: Average session duration > 15 minutes

Revenue (if applicable):

SC-043: $10,000 MRR within 6 months
SC-044: 90%+ payment success rate
SC-045: < 5% churn rate monthly

17.5 Quality Metrics
Code Quality:

SC-046: 80%+ code coverage
SC-047: Zero critical bugs in production
SC-048: < 10 high-severity bugs per release
```

SC-049: Technical debt ratio < 5%
SC-050: All security vulnerabilities patched within 24 hours

Documentation:

SC-051: 100% of public API endpoints documented
SC-052: User guide covering all major features
SC-053: Video tutorials for key workflows
SC-054: FAQ with answers to 50+ common questions

18. Appendices
Appendix A: Glossary
Black Box Testing: Testing without knowledge of internal implementation details
Crawl Depth: Number of link levels from the starting URL
Testable Element: Any UI or API component that can be tested (form, button, input, API endpoint)
Node: A page or endpoint in the site map graph
Edge: A connection between two nodes (link, form submission, API call)
Session State: Authentication and session data maintained during crawl
Spider: The automated crawler component that discovers pages
Headless Browser: Browser without GUI for automation (Puppeteer, Playwright)
Pattern Sampling: Technique to sample representative URLs instead of crawling all variations
Content Fingerprinting: Technique to identify similar/duplicate pages using hash algorithms
Bloom Filter: Probabilistic data structure for efficient set membership testing
SimHash: Locality-sensitive hash function for finding similar documents
RBAC: Role-Based Access Control
IDOR: Insecure Direct Object Reference vulnerability
JWT: JSON Web Token for authentication
SPA: Single Page Application
SSR: Server-Side Rendering
PWA: Progressive Web App
CSRF: Cross-Site Request Forgery
XSS: Cross-Site Scripting
SQL Injection: Code injection attack targeting SQL databases
MFA: Multi-Factor Authentication
OAuth: Open Authorization standard
SAML: Security Assertion Markup Language
API: Application Programming Interface
REST: Representational State Transfer
GraphQL: Query language for APIs
WebSocket: Protocol for full-duplex communication
Web Vitals: Performance metrics (LCP, FID, CLS)
Appendix B: Configuration Templates
Small Site Template:
```json
{
  "name": "Small Site Crawl",
  "description": "For sites with < 100 pages",
  "config": {
    "maxDepth": 3,
```

```
    "maxPages": 100,
    "timeout": 30000,
    "followExternal": false,
    "javascript": true,
    "screenshots": true,
    "concurrentRequests": 3,
    "requestDelay": 500,
    "patternSampling": {
      "enabled": false
    }
  }
}
```

Large E-commerce Template:

```
{
  "name": "E-commerce Crawl",
  "description": "For large e-commerce sites with product catalogs",
  "config": {
    "maxDepth": 4,
    "maxPages": 5000,
    "timeout": 30000,
    "followExternal": false,
    "javascript": true,
    "screenshots": false,
    "concurrentRequests": 10,
    "requestDelay": 200,
    "patternSampling": {
      "enabled": true,
      "threshold": 100,
      "patterns": [
        {
          "pattern": "/product/(\\d+)",
          "strategy": "random",
          "sampleSize": 50
        },
        {
          "pattern": "/category/[^/]+/(\\d+)",
          "strategy": "stratified",
          "sampleSize": 100
        }
      ]
    },
    "interactions": {
      "triggerHovers": true,
      "expandDropdowns": true,
      "scrollToBottom": true
    }
  }
}
```

SPA Application Template:

```
{
  "name": "SPA Application",
  "description": "For React/Vue/Angular single-page apps",
  "config": {
    "maxDepth": 5,
    "maxPages": 200,
    "timeout": 45000,
    "followExternal": false,
    "javascript": true,
    "waitForDynamic": true,
    "dynamicTimeout": 10000,
    "screenshots": true,
    "concurrentRequests": 2,
    "requestDelay": 1000,
    "interactions": {
      "triggerHovers": true,
      "clickModals": true,
      "navigateWizards": true
    },
    "apiDetection": true
  }
}
```

Multi-Role Security Audit Template:

```
{
  "name": "Security Audit with RBAC",
  "description": "Comprehensive security testing with multiple roles",
  "config": {
    "maxDepth": 4,
    "maxPages": 1000,
    "timeout": 30000,
    "javascript": true,
    "roles": [
      {
        "id": "guest",
        "name": "Unauthenticated User",
        "credentials": null
      },
      {
        "id": "user",
        "name": "Basic User",
        "credentials": {
          "username": "user@test.com",
          "password": "UserPass123!"
        }
```

```
      },
      {
        "id": "admin",
        "name": "Administrator",
        "credentials": {
          "username": "admin@test.com",
          "password": "AdminPass123!"
        }
      }
    ],
    "security": {
      "testSqlInjection": true,
      "testXss": true,
      "testCsrf": true,
      "testAuthBypass": true,
      "testPrivilegeEscalation": true,
      "testIdor": true
    },
    "generateTests": {
      "authorization": true,
      "injection": true,
      "sessionManagement": true
    }
  }
}
```

Appendix C: Sample API Responses

Crawl Status Response:

```
{
  "id": "550e8400-e29b-41d4-a716-446655440000",
  "baseUrl": "https://example.com",
  "status": "running",
  "progress": {
    "currentPage": 450,
    "totalPages": 1000,
    "percentage": 45,
    "eta": 620,
    "currentUrl": "https://example.com/products/laptop-123"
  },
  "stats": {
    "duration": 1847000,
    "pagesPerMinute": 14.6,
    "totalNodes": 450,
    "totalEdges": 1234,
    "totalForms": 23,
    "totalAPIs": 89,
    "errors": 3
  },
  "startedAt": "2025-10-22T10:00:00Z",
```

```
  "config": {
    "maxDepth": 3,
    "maxPages": 1000,
    "timeout": 30000
  }
}
Appendix D: Test Data Examples
SQL Injection Payloads:
javascriptconst sqlInjectionPayloads = [
  "' OR '1'='1",
  "'; DROP TABLE users--",
  "' UNION SELECT * FROM users--",
  "admin'--",
  "1' AND '1'='1",
  "' OR 1=1--",
  "' OR 'x'='x",
  "1' ORDER BY 1--",
  "1' UNION ALL SELECT NULL--",
  "' AND 1=CONVERT(int, (SELECT @@version))--"
];
XSS Payloads:
```javascript
const xssPayloads = [
  "<script>alert('XSS')</script>",
  "<img src=x onerror=alert('XSS')>",
  "javascript:alert('XSS')",
  "<svg/onload=alert('XSS')>",
  "'-alert('XSS')-'",
  "<iframe src='javascript:alert(\"XSS\")'></iframe>",
  "<body onload=alert('XSS')>",
  "<input onfocus=alert('XSS') autofocus>",
  "<select onfocus=alert('XSS') autofocus>",
  "<textarea onfocus=alert('XSS') autofocus>"
];
```

Appendix E: Supported Technologies
Frameworks Detected:

React (all versions)
Vue.js (2.x, 3.x)
Angular (2+)
Next.js
Nuxt.js
Svelte
jQuery
Backbone.js
Ember.js

CSS Frameworks:
```

Bootstrap (3.x, 4.x, 5.x)
Tailwind CSS
Material-UI
Ant Design
Chakra UI
Bulma
Foundation

Authentication Methods:

Form-based (POST)
HTTP Basic/Digest
JWT (Bearer token)
OAuth 2.0 (Google, Facebook, GitHub, etc.)
SAML SSO
API Key
Session cookies

Content Management Systems:

WordPress
Drupal
Joomla
Shopify
Magento
WooCommerce

Appendix F: Export Format Specifications
Supported Export Formats:

JSON (complete, summary)
CSV (nodes, edges, tests)
Selenium (Python, Java, JavaScript, C#)
Playwright (JavaScript, Python, Java, C#)
Cypress (JavaScript)
Puppeteer (JavaScript)
Robot Framework
Cucumber/Gherkin
Postman Collection (v2.1)
OpenAPI/Swagger (3.0)
Insomnia Workspace
k6 Load Test Scripts
JMeter Test Plan (.jmx)
Artillery.io Scenarios
Markdown Report
HTML Report
PDF Report
Excel (.xlsx)

OWASP ZAP Context
Burp Suite State

Appendix G: Rate Limiting & Quotas
Free Tier:

10 crawls per month
Max 100 pages per crawl
1 concurrent crawl
Export to 3 formats
30-day data retention

Pro Tier:

100 crawls per month
Max 5,000 pages per crawl
3 concurrent crawls
Export to all formats
90-day data retention
API access
Email support

Enterprise Tier:

Unlimited crawls
Unlimited pages
10 concurrent crawls
Export to all formats
Unlimited data retention
API access
Priority support
Dedicated account manager
Custom integrations
SLA guarantee

Appendix H: Browser Compatibility
Supported for Crawling:

Chromium/Chrome (primary)
Firefox (via Playwright)
WebKit/Safari (via Playwright)

Supported for Dashboard:

Chrome 90+
Firefox 88+
Safari 14+
Edge 90+

Not Supported:

Internet Explorer (any version)
Opera Mini
UC Browser

Appendix I: Known Limitations

JavaScript-heavy sites: Sites that heavily rely on JavaScript may have slower crawl times
CAPTCHA: Cannot automatically solve CAPTCHA without third-party services or manual intervention
Cloudflare/Bot Detection: Advanced bot detection may block automated crawling
WebSocket: Limited support for WebSocket-heavy real-time applications
Binary Protocols: Cannot crawl gRPC or other binary protocols directly
Video Streaming: Cannot analyze video streaming content
Canvas/WebGL: Limited analysis of Canvas-based applications
Shadow DOM: Some Shadow DOM content may not be fully accessible
Browser Extensions: Cannot test browser extension functionality
Mobile Apps: Cannot crawl native mobile applications (iOS/Android)

Appendix J: Roadmap
Version 2.1 (Q1 2026):

AI-powered test case prioritization
Visual regression testing
Automated vulnerability scanning
Comparison mode (diff between crawls)

Version 2.2 (Q2 2026):

Mobile app testing support (via Appium)
GraphQL introspection and testing enhancements
Advanced WebSocket testing
Browser extension for manual testing

Version 3.0 (Q3 2026):

AI-powered test generation (GPT integration)
CI/CD pipeline plugins (GitHub Actions, GitLab CI, Jenkins)
Custom plugin architecture
Real-time collaborative crawling

Version 3.1 (Q4 2026):

Machine learning for element detection improvement
Predictive analytics (identify likely bugs)
Integration with JIRA, Azure DevOps
Advanced reporting and analytics

Document Sign-off
RoleNameDateSignatureProduct ManagerEngineering LeadSecurity LeadQA LeadDevOps LeadUX

Designer

Document Version: 2.0 (Final Comprehensive Version)
Last Updated: 2025-10-22
Next Review: 2026-01-22
Pages: 143
Total Requirements: 880+ Functional & Non-Functional Requirements

Summary Statistics

Core Functional Requirements: 735
Non-Functional Requirements: 55
Security Requirements: 28
Performance Requirements: 15
Success Criteria: 50
Total Features Covered: 880+

Coverage Areas:
- ☑ Poor Semantic HTML Detection
- ☑ Large-Scale Site Handling (Millions of Pages)
- ☑ Role-Based Access Control (RBAC)
- ☑ Multi-Step Workflows & Wizards
- ☑ Dynamic Content & JavaScript Interactions
- ☑ Authentication (All Types including MFA, OAuth, SSO)
- ☑ Application State Management
- ☑ Complex Input Components
- ☑ Drag & Drop, Keyboard Shortcuts
- ☑ Search Functionality
- ☑ Time-Based Content
- ☑ Email & Notification Flows
- ☑ API Discovery (REST, GraphQL, WebSocket)
- ☑ Security Testing (SQL Injection, XSS, CSRF, etc.)
- ☑ Performance Metrics (Web Vitals)
- ☑ Accessibility Testing
- ☑ PWA Features
- ☑ A/B Testing & Feature Flags
- ☑ Export to 20+ Formats
- ☑ Graph Visualization
- ☑ Test Case Generation
- ☑ Distributed Crawling
- ☑ CI/CD Integration

This PRD is now complete and ready for implementation! 🎉