Black-Box Crawler Master Reference Document

This document consolidates all crawlable objects, potential issues, and handling strategies for an AI-driven black-box testing crawler for modern web applications.

---

1. Crawlable Objects / Features

1.1 Navigation & Structural Coverage

- `<a>` links (internal & external, anchors)
- Buttons triggering navigation or forms (JS, SPA routing)
- Menus, hover/click-triggered, sidebars
- iFrames / embedded components
- SPA client-side routing without URL changes

1.2 Forms & Inputs

- Text, password, email, number fields
- Radio buttons, checkboxes, dropdowns, multi-selects
- Multi-step forms (wizard-style)
- Hidden fields, dynamic IDs, CSRF tokens
- File uploads / downloads

1.3 Dynamic & Interactive Components

- Modals, popups, alerts, accordions, tabs
- Sliders, toggles, switches, carousels
- Drag-and-drop elements, resizable panels

- Hover menus, right-click/context menus
- Copy-paste / clipboard interactions

1.4 Client-Side Rendering / DOM Changes

- React, Angular, Vue, SPAs
- Shadow DOM & Web Components
- Lazy-loaded content, infinite scroll
- Dynamic content / A/B testing / feature flags

1.5 Event Handling

- Click, double-click, right-click/context menus
- Keyboard events (Enter, Tab, Arrow keys, shortcuts)
- Drag/drop, touch gestures (swipe, pinch)
- Clipboard events

1.6 Authentication & Sessions

- Login/logout flows, token-based sessions (JWT, cookies)
- Session expiration & re-login handling
- Role-based UI variations (admin, user)

1.7 API & Network Interactions

- XHR/fetch requests, event-driven API calls
- Capture request/response pairs for bug analysis
- Trigger API-induced UI changes

1.8 Media & Special Elements

- Videos, audio, interactive charts, maps
- Third-party widgets (social media feeds, payment widgets)
- File uploads/downloads

1.9 Error & Edge Cases

- 404 / 500 pages, unhandled JS exceptions
- Forms with invalid or extreme inputs
- Empty states (no products, empty lists)
- Infinite redirects / reload loops
- Missing media or broken scripts

1.10 Accessibility & Hidden Elements

- ARIA roles, off-screen focusable elements
- Elements revealed only via hover, focus, or keyboard navigation

1.11 Timing & Async Behaviors

- AJAX / fetch updates, network idle detection
- Loading spinners, animations, transitions
- Lazy-loaded images and content

1.12 Device & Viewport Variations

- Responsive layouts (desktop, tablet, mobile)
- Orientation changes (portrait/landscape)
- Mobile-only interactions (touch gestures)

1.13 Advanced Product / Feature Variations

- Stock availability (in/out of stock)
- Size / color / variant selectors
- Wishlist / add-to-cart / recommendation buttons
- Promotions, seasonal banners, geo-targeted content

1.14 State & Storage Objects

- SessionStorage, LocalStorage, IndexedDB
- Cookies affecting UI or feature flags
- Global state effects (cart, wishlist, theme, language)

---

2. Potential Issues / Pitfalls

1. State explosion from thousands of similar pages
2. Hidden or lazy-loaded elements missed
3. Dynamic IDs / randomized attributes inflating graph nodes
4. SPA routing without URL changes
5. Async behavior causing missed DOM updates
6. Multi-step or conditional flows not fully traversed
7. Session expiry / authentication / role-based UI challenges
8. Client-side storage affecting UI states
9. Error / boundary states (404, empty lists, invalid inputs)
10. Shadow DOM / web components inaccessible without special queries
11. Responsive / viewport-specific content missed

12. Rate limiting / anti-bot measures (CAPTCHA, WAF)
13. Dependencies between pages (cart, wishlist, preferences)
14. Randomized / dynamic content (ads, recommendations, rotating banners)

---

3. Handling Strategies / Best Practices

3.1 Feature-Aware Deduplication

- Hash DOM + meaningful feature vector (stock status, selectors, buttons)
- Ignore cosmetic differences (names, images)

3.2 Event Combination Sampling

- Trigger all meaningful events per state
- Prioritize critical elements first (navigation, submit, add-to-cart)

3.3 State Abstraction & Clustering

- Cluster pages by functional equivalence
- Sample representative pages per cluster to avoid redundancy

3.4 Async & Timing Management

- Wait for network idle before hashing or moving on
- Detect and wait for spinners, loaders, animations

3.5 Session & Authentication Handling

- Maintain and refresh sessions automatically
- Re-login on expiry

- Crawl multiple roles if needed

3.6 Coverage Metrics

- Track % of interactive elements triggered
- Track % of unique states reached
- Stop crawling when marginal gain is below threshold

3.7 Graph Representation & Logging

- Nodes = unique functional UI states
- Edges = events/interactions
- Metadata = screenshots, DOM snapshots, console logs, network logs
- Use graph pruning to avoid redundant nodes

3.8 Parallelization & Scalability

- Multi-tab / multi-browser crawling
- Distributed crawling for very large sites
- Batch processing for clusters of similar pages

---

4. Out-of-Scope / Not Crawlable

- CAPTCHAs / reCAPTCHAs
- Multi-factor authentication codes (OTP/SMS/Email)
- Strictly protected / encrypted content (banking apps, secure APIs)
- Anti-bot protected sites (Cloudflare/WAF), unless explicitly allowed

---

This document serves as a **master reference for building a black-box testing crawler** with maximum UI coverage and minimal redundancy. It covers **crawlable objects, potential issues, and strategies to handle them**, providing a roadmap for **iteration 1 through final implementation** of your FYP.