Adhyan POM Studio — Consolidated Phases (Previous vs Upgraded)

This document collects the requirements, prior implementation notes, and upgraded code for each phase derived from the chat. Each phase includes: Requirements, Previous (short), Upgraded code (representative), Explanation, and suggested next steps.

Note: Code blocks are provided as plain text for copy/paste. The 'copy to clipboard' button is not executable inside Word — use the code blocks directly or copy manually.

Phase 1 — Core helpers & xpathLiteral

Requirements:

Requirements: Robust xpath literal builder, label fetcher, skip/dedupe helpers.

Previous (short):

```
// Previous (simplified)
function xpathLiteral(s){ return '"' + s + '"'; }
function getLabelFor(el){ return el && el.textContent ? el.textContent.trim() :
null; }
```

Upgraded (representative code):

```
// Upgraded (representative)
function xpathLiteral(s){
  s = String(s);
  if (s.indexOf('"') === -1) return '"' + s + '"';
  if (s.indexOf("'") === -1) return "'" + s + "'";
  const parts = s.split('"');
  const out = [];
  for (let i = 0; i < parts.length; i++){
    if (parts[i] !== '') out.push('"' + parts[i].replace(/\\/g,'\\\\') + '"');
    if (i < parts.length - 1) out.push('\'"\'');
  }
  return 'concat(' + out.join(',') + ')';
}

function getLabelFor(el){
  try { if (typeof bestLabelFor === 'function') return bestLabelFor(el); }
catch(e){}
  const id = el && el.getAttribute ? el.getAttribute('id') : null;
  if (id) {
    const sel = 'label[for="' + id.replace(/([#.;,[\\]()>+~=:\"'\\])/g,'\\$1') +
'"]';
    const lab = (window.CURRENT_DOC || document).querySelector(sel);
    if (lab && lab.textContent) return lab.textContent.trim();
  }
  const parentLabel = el && el.closest ? el.closest('label') : null;
  if (parentLabel && parentLabel.textContent) return
parentLabel.textContent.trim();
  return null;
}
```

Explanation / What it does / Why upgrade:

Explanation: The upgraded xpathLiteral handles both single and double quotes by building a concat(...) expression when needed. getLabelFor defers to user-provided bestLabelFor and falls back to label[for] and closest label.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 2 — Collect candidate elements & shouldSkip

Requirements:

Requirements: Wide but safe selector set, early filtering for noise, de-duplication.

Previous (short):

```
// Previous (simplified)
function collectCandidateElements(doc){
  return Array.from(doc.querySelectorAll('button,input,a'));
}
function shouldSkip(el){ return false; }
```

Upgraded (representative code):

```
// Upgraded (representative)
function isHiddenByInlineStyle(el){
  const s = (el.getAttribute('style')||'').toLowerCase();
  return
/display\s*:\s*none|visibility\s*:\s*hidden|opacity\s*:\s*0|pointer-events\s*:\s
*none/.test(s);
}
function hasUsefulLabel(el){
  if (el.getAttribute('aria-label')) return true;
  if (el.getAttribute('placeholder')) return true;
  if (el.getAttribute('name')) return true;
  const t = (el.textContent||'').trim();
  return t.length > 0;
}
function shouldSkip(el){
  if(!el || el.nodeType !== 1) return true;
  if(el.hasAttribute('hidden')) return true;
  if(el.getAttribute('aria-hidden') === 'true') return true;
  if(isHiddenByInlineStyle(el)) return true;
  if(el.hasAttribute('disabled')) return true;
  if(!hasUsefulLabel(el)) return true;
  return false;
}

function collectCandidateElements(doc){
  const basics = Array.from(doc.querySelectorAll([
    'input:not([type="hidden"])','button','a','select','textarea',
    '[role="button"]','[data-ctl]','[data-qa-locator]','lightning-input'
  ].join(',')));
  let filtered = basics.filter(el => !shouldSkip(el));
  const seen = new Set();
  filtered = filtered.filter(el=>{
    const key = (el.tagName + '|' + (el.id||'') + '|' +
(el.getAttribute('name')||''));
    if (seen.has(key)) return false;
    seen.add(key);
    return true;
  });
  return filtered;
}
```

Explanation / What it does / Why upgrade:

Explanation: The upgraded collector broadens selectors (framework-specific tags), filters noise early with shouldSkip, and de-duplicates by a signature. This reduces false positives and keeps the list manageable.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 3 — XPath generator family (basic, wildcards, axes, functions)

Requirements:

Requirements: Provide multiple XPath families and a 'best' chooser.

Previous (short):

```
// Previous (simplified)
function genBasicXPath(el){ return '//' + el.tagName.toLowerCase(); }
```

Upgraded (representative code):

```
// Upgraded (representative)
function genBasicXPath(el){
  const tag = el.tagName.toLowerCase();
  if(el.id) return '//*[@id=' + xpathLiteral(el.id) + ']';
  const name = el.getAttribute('name');
  if(name) return '//' + tag + '[@name=' + xpathLiteral(name) + ']';
  const label = getLabelFor(el);
  if(label) return '//' + tag + '[contains(normalize-space(.), ' +
xpathLiteral(label.trim()) + ')]';
  const cls = (el.getAttribute('class')||'').trim().split(/\s+/)[0];
  if(cls) return '//' + tag + '[contains(concat(" ", normalize-space(@class), "
"), ' + xpathLiteral(' ' + cls + ' ') + ')]';
  const parent = el.parentElement;
  if(!parent) return '//' + tag + '[1]';
  const same = Array.from(parent.children).filter(x => x.tagName ===
el.tagName);
  const idx = same.indexOf(el) + 1;
  return genBasicXPath(parent) + '/' + tag + '[' + idx + ']';
}
```

Explanation / What it does / Why upgrade:

Explanation: Multiple generator functions give alternative locator strategies; bestXPath chooses the most reliable (Salesforce/LWC, Pega, axes, functions, wildcards, basic). This improves resilience across apps.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 4 — UI bridge: fillLocatorList and interactions

Requirements:

Requirements: Render locator rows, support copy/highlight, dynamic patterns, template toggle.

Previous (short):

```
// Previous (simplified)
function fillLocatorList(){ /* simple table */ }
```

Upgraded (representative code):

```
// Upgraded (representative)
function fillLocatorList(panel = 'basic'){
  const data = Array.isArray(window.CURRENT_LOCATORS) ? window.CURRENT_LOCATORS :
 [];
  // create container and toolbar if needed...
  // build rows with checkboxes, value cell, and single-click copy+highlight
behavior
  // support template toggle and copy/download actions
  // (see in-chat full renderer implementation)
}
```

Explanation / What it does / Why upgrade:

Explanation: The upgraded renderer is resilient to different shapes of CURRENT_LOCATORS, provides UI controls (select all, template toggle, copy/download) and wires per-row click to copy the value and run tryHighlight for xpaths.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 5 — Dynamic XPath picker & patterns

Requirements:

Requirements: On-page picker overlay to hover/click element and generate multiple parameterized xpath templates (%s).

Previous (short):

```
// Previous (simplified)
No picker implemented.
```

Upgraded (representative code):

```
// Upgraded (representative)
(function openDynamicPicker(){
  // overlay with hover highlight and click-to-pick behavior
  // generateTemplatesForElement(el) returns prioritized array of templates:
  // e.g. id-based, data-attr, name, aria-label, exact-text, contains-text,
class-based, positional fallback
})();
```

Explanation / What it does / Why upgrade:

Explanation: Picker lets user pick an element visually and immediately get multiple %s templates to test. It stores selection in toolbar and enables applying templates to selected locator rows.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 6 — Robust highlight across shadow DOM & iframes

Requirements:

Requirements: tryHighlightByXPath that searches document, shadow roots, and frames and transiently highlights element.

Previous (short):

```
// Previous (simplified)
function tryHighlight(xpath){ const node = document.evaluate(xpath,...);
node.style.outline = '3px solid'; }
```

Upgraded (representative code):

```
// Upgraded (representative)
function tryHighlightByXPath(xpath){
  // evaluate in main doc, then find in shadow roots via findInShadowRoots, and
search iframes recursively via findInIframes
  // transientHighlightElement applies boxShadow/outline, scrollIntoView, then
restores styles after timeout
}
```

Explanation / What it does / Why upgrade:

Explanation: Frame and shadow DOM aware highlighter increases chance of highlighting elements inside web components or different frames.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 7 — Action candidate finder (findActionCandidates)

Requirements:

Requirements: Heuristic to find 'action' buttons (close/menu/etc.) and expose them as CURRENT_LOCATORS for quick pickup.

Previous (short):

```
// Previous (simplified)
No dedicated action candidate finder.
```

Upgraded (representative code):

```
// Upgraded (representative)
function findActionCandidates(){
  // scans buttons/a/div/span, heuristics on text/title/aria-label and icon
classes, returns list with basic/sf/pega xpaths and sets window.CURRENT_LOCATORS
}
```

Explanation / What it does / Why upgrade:

Explanation: This is useful for quickly finding close/menu/action controls on pages like modals or dynamic menus.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 8 — Artifact generator (ZIP multi-file exporter)

Requirements:

Requirements: Build a zip of language/framework test scaffolding from selected locators (Java Selenium, Playwright, etc.).

Previous (short):

```
// Previous (simplified)
Manual copy-paste of locators into a project.
```

Upgraded (representative code):

```
// Upgraded (representative)
function buildArtifactsFiles(options){
  // normalize locators, then produce files map for Java (pom.xml, BaseTest.java,
 PageObject with @FindBy xpath fields, tests), JS, Playwright, etc.
}
async function downloadZipFromFiles(filesMap, zipName){
  // uses JSZip to package and trigger download
}
```

Explanation / What it does / Why upgrade:

Explanation: Automates scaffolding; useful for demo tests or quick PoCs. Keep generated files minimal and allow user edits after download.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.

Phase 9 — UI compatibility bridges & safe standalone dumps

Requirements:

Requirements: Provide drop-in fillers (printFillLocatorList, safe fillLocatorList) so the patch works even if original UI functions are absent.

Previous (short):

```
// Previous (simplified)
Application had a single monolithic renderer; failing if missing functions.
```

Upgraded (representative code):

```
// Upgraded (representative)
function printFillLocatorList(panel, containerId){ /* creates a standalone dump
*/ }
function fillLocatorList(panel){ /* robust, works with multiple CURRENT_LOCATORS
shapes */ }
(function uiCompatBridge(){ window.extractAllLocators = function(){ /* uses
collector + generators then sets CURRENT_LOCATORS and calls fillLocatorList
*/ }})();
```

Explanation / What it does / Why upgrade:

Explanation: Compatibility layers make the patch non-invasive — they don't require replacing whole app code and gracefully fall back.

Suggested next improvements:

- Add unit tests for each generator function.
- Add telemetry/logging for failures (XPath eval errors).
- Add UI tests to ensure picker overlay and highlight work across pages with frames.