# Adhyan POM Studio — Phase-by-Phase: Previous vs Upgraded (Expanded)

This document contains the consolidated Phase-by-Phase requirements, previous implementation (short), and the upgraded implementation (full script excerpts) based on the chat. Generated by assistant.

## Phase 1 — Core helpers & label handling

Previous (summary):

Previous: minimal xpathLiteral & simple label resolver.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: robust xpathLiteral and defensive label fetcher  
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;  
}

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 2 — Skip / dedupe logic

Previous (summary):

Previous: basic filtering (visible inputs, buttons).

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: comprehensive shouldSkip + getSmartAnchorTextSafe + collectCandidateElements  
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){  
 const id = el.id;  
 if (el.getAttribute('aria-label')) return true;  
 if (el.getAttribute('placeholder')) return true;  
 if (el.getAttribute('name')) return true;  
 if (id){  
 const sel = `label[for="${id.replace(/([#.;,[\\]()>+~=:\"\*\\\\])/g,'\\\\$1')}"]`;  
 if((window.CURRENT\_DOC||document).querySelector(sel)) return true;  
 }  
 const t = (el.textContent||'').trim();  
 return t.length > 0;  
}  
function isSFNoise(el){ /\* framework heuristics \*/ }  
function isPegaNoise(el){ /\* framework heuristics \*/ }  
  
function shouldSkip(el){  
 if(!el || el.nodeType !== 1) return true;  
 const tag = el.tagName.toLowerCase();  
 if(el.hasAttribute('hidden')) return true;  
 if(el.getAttribute('aria-hidden') === 'true') return true;  
 if(isHiddenByInlineStyle(el)) return true;  
 if(el.hasAttribute('disabled') || el.getAttribute('aria-disabled') === 'true') return true;  
 if(el.hasAttribute('readonly')) return true;  
 if(tag === 'input' && (el.getAttribute('type')||'').toLowerCase() === 'hidden') return true;  
 if(tag === 'a' && (!el.hasAttribute('href') || el.getAttribute('href') === '#')) return true;  
 if(isSFNoise(el)) return true;  
 if(isPegaNoise(el)) return true;  
 if(!hasUsefulLabel(el)) return true;  
 return false;  
}  
  
function getSmartAnchorTextSafe(el){ /\* defensive anchor text \*/ }  
  
function collectCandidateElements(doc){  
 const basics = Array.from(doc.querySelectorAll([  
 'input:not([type="hidden"])','button','a','select','textarea',  
 '[role="button"]','span[role="button"]',  
 '[data-ctl]','[data-qa-locator]','[data-aura-class]','[data-aura-rendered-by]','[data-key]','[data-id]',  
 'lightning-input','lightning-button','lightning-combobox','lightning-textarea',  
 'lightning-record-edit-form','lightning-tab','lightning-input-field','lightning-formatted-text',  
 'force-record-view','force-input','force-button','force-lookup','force-list-view'  
 ].join(',')));  
 let filtered = basics.filter(el => !shouldSkip(el));  
 const seen = new Set();  
 filtered = filtered.filter(el=>{  
 const anchor = getSmartAnchorTextSafe(el);  
 const sig = [  
 el.tagName.toLowerCase(),  
 el.id||'',  
 el.getAttribute('name')||'',  
 el.getAttribute('data-qa-locator')||'',  
 el.getAttribute('data-ctl')||'',  
 el.getAttribute('data-key')||'',  
 el.getAttribute('data-id')||'',  
 anchor  
 ].join('|');  
 if(seen.has(sig)) return false;  
 seen.add(sig);  
 return true;  
 });  
 return filtered;  
}

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 3 — XPath generators

Previous (summary):

Previous: single basic generator.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: multiple generators (basic, wildcard, axes, function, salesforce, pegA)  
function genBasicXPath(el){ /\* id -> name -> label -> class -> position fallback \*/ }  
function genWildcardXPath(el){ /\* similar but uses \* \*/ }  
function genAxesXPath(el){ /\* label[for], aria-labelledby, headings/legend \*/ }  
function genFunctionXPath(el){ /\* placeholder/title/alt \*/ }  
function genSalesforceXPath(el){ /\* data-qa-locator, data-key, data-aura-\* \*/ }  
function genSalesforceLWCXPath(el){ /\* data-id, data-field, lightning-\* handling \*/ }  
function genSalesforceAuraXPath(el){ /\* data-aura-class/rendered-by \*/ }  
function genPegaXPath(el){ /\* data-test-id, data-node-id, data-ctl \*/ }  
  
function bestXPath(el){  
 // prioritized: LWC, Aura, SF generic, Pega, axes, functions, wildcards, basic  
}

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 4 — Locator list UI (renderLocatorList / fillLocatorList)

Previous (summary):

Previous: simple list showing xpaths.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: interactive list with checkboxes, template toggle, copy/download, select-all  
function fillLocatorList(panel = 'basic'){  
 // builds table, toolbar (select all, template toggle, copy, download)  
 // uses \_\_toTemplate helper to convert xpath -> template with %s  
 // ensures safe HTML escaping and reflects active UI chips  
}

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 5 — Dynamic XPath picker & patterns

Previous (summary):

Previous: none / ad-hoc dynamic patterns.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: picker overlay + template generator + toolbar integration  
- openDynamicPicker() shows overlay, hover highlight, click to pick element  
- generateTemplatesForElement(el) returns prioritized array of templates (id, data-\*, name, aria-label, exact/contains text, class-based, positional)  
- ensureDynamicPanel() injects toolbar (dropdown, Open Picker, Apply to selected, preview, copy)  
- picker cleans up listeners on close/cancel

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 6 — Highlighting across frames & shadow DOM

Previous (summary):

Previous: naive scrollIntoView + outline.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: tryHighlightByXPath + findInShadowRoots + findInIframes + transientHighlightElement  
- evaluateXPathInRoot(xpath, rootNode) safe evaluation  
- findInShadowRoots(xpath, startRoot) walks shadowRoots  
- findInIframes(xpath, win) searches nested frames, skips cross-origin safely  
- transientHighlightElement(el, options) applies boxShadow/outline, scrolls into view, restores after timeout  
- tryHighlightAllByXPath(xpath) convenience to highlight many

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 7 — Artifact/ZIP generator (POM/project scaffolds)

Previous (summary):

Previous: manual templates or none.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: buildArtifactsFiles(options) producing files map per fw/lang/runner  
- supports Java + Selenium (Maven + TestNG + Cucumber), JS Selenium, Playwright (JS/TS/PY)  
- creates pom.xml, BaseTest, PageObject, SampleTest, feature files, stepdefs as needed  
- downloadZipFromFiles(filesMap, zipName) uses JSZip in browser to create zip

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 8 — Action candidates helper (findActionCandidates)

Previous (summary):

Previous: manual scanning.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: heuristic-based finder for action-like elements  
- looksLikeAction(el) scores elements using text/title/aria/icon classes/data-action  
- findActionCandidates() returns map with basicXPath, sfXPath, pegaXPath, css  
- exposes highlightCandidate(n) to flash selected candidate  
- populates CURRENT\_LOCATORS shape for reuse by UI

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

## Phase 9 — Integration / UI bridge & safe drop-ins

Previous (summary):

Previous: single monolith patch applied to adhyan HTML.

Upgraded (code & explanation):

**Notes:** Upgraded code includes defensive checks, framework-specific heuristics, shadow/iframe handling, and cleaner UI hooks.

// Upgraded: modular override functions and UI-compatibility bridges  
- safe fallbacks for generateCssSelector, generatePlaywrightSelectors, showToast  
- extractAllLocators() override builds UI-shaped objects and calls fillLocatorList(panel)  
- UI-compat bridge script for older code shapes  
- Post-filtering, de-duplication and stable DOM-order maintained  
- Ensures not to break original adhyan HTML by checking for existing functions before binding

**Checklist:** Review, test on target adhyan HTML, confirm panel integration, run extractAllLocators() and pickers.

Delivery notes:

• This document includes expanded upgraded code snippets for the nine phases discussed.\n• I did not paste every single variant/duplicate; use these as the full replacement patch pieces to assemble into Adhyan HTML.\n• When you are ready I can produce a single consolidated JS file (full patch) and also embed the full scripts into the Adhyan HTML on your command.