Adhyan POM Studio - Consolidated Full Patch Report

This document consolidates all phases of the Adhyan POM Studio patch process. For each phase, it includes:  
 - Requirements  
 - Previous Implementation (code)  
 - Upgraded Implementation (code)  
 - Explanation of Changes (What it does)  
 - Pros, Cons, and Next Actions  
All upgrade requirements are highlighted in text and inline comments.

# Phase 1: Core Helpers (XPath, Label Fetching)

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 2: Skip/Dedupe Logic

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 3: Candidate Collection

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 4: Generators (Basic, Wildcard, Axes, Function-based)

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 5: Salesforce Locators (Generic, LWC, Aura)

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 6: Pega Locators

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 7: Best XPath Chooser

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 8: Extract All Locators (Main function)

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

# Phase 9: UI Bridge & Rendering

## Requirements

[Requirement details for this phase go here]

## Previous Implementation

```javascript  
// Previous implementation code snippet here  
```

## Upgraded Implementation

```javascript  
// Upgraded implementation code snippet here  
// <-- Inline comments highlight upgrades  
```

## Explanation of Changes

[Explanation of what was changed, what it does, and why it is better]

## Pros

• [List of pros]

## Cons

• [List of cons]

## Next Actions

• [Next action items]

## Phase 1 – Core Helpers

\*\*Requirement:\*\*   
Strengthen string literal handling for XPath and improve label fetching reliability.  
  
---  
  
\*\*Previous Implementation (Core Helpers):\*\*  
```javascript  
function xpathLiteral(s){  
 s = String(s||'');  
 if (s.indexOf('"') === -1) return '"' + s + '"';  
 if (s.indexOf("'") === -1) return "'" + s + "'";  
 return 'concat("' + s.replace(/"/g, '",'"',"') + '")';  
}  
  
function getLabelFor(el){  
 const id = el && el.getAttribute ? el.getAttribute('id') : null;  
 if (id) {  
 const lab = document.querySelector('label[for="'+id+'"]');  
 if (lab && lab.textContent) return lab.textContent.trim();  
 }  
 return null;  
}  
```  
  
---  
  
\*\*Upgraded Implementation (Core Helpers):\*\*  
```javascript  
// ✅ Improved handling of quotes in XPath literals  
function xpathLiteral(s){  
 s = String(s);  
 if (s.indexOf('"') === -1) return `"${s}"`; // double-quote safe  
 if (s.indexOf("'") === -1) return `'${s}'`; // single-quote safe  
 // fallback: concat both safely  
 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(',')})`;  
}  
  
// ✅ Defensive label fetching (with bestLabelFor support & parent label)  
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 of Changes:\*\*   
- `xpathLiteral` now supports \*\*mixed quote cases\*\* by splitting on `"` and rebuilding with `concat()` safely.   
- `getLabelFor` now:   
 - Falls back to `bestLabelFor` if available.   
 - Escapes special characters in `label[for]`.   
 - Supports nearest `<label>` parent as fallback.   
  
\*\*Pros:\*\*   
- Handles edge cases in XPath literals.   
- More reliable label resolution (works with custom label functions).   
  
\*\*Cons:\*\*   
- Slightly more verbose.   
- Minor overhead in string handling.   
  
\*\*Next Actions:\*\*   
- Verify against \*\*Salesforce\*\* and \*\*Pega\*\* DOMs for label detection consistency.   
- Extend to handle `aria-labelledby` directly if needed.

## Copy-to-Clipboard Feature for Upgrade Code

To make it easier to copy the \*\*Upgrade Code\*\* directly from this document, we include a small JavaScript snippet that can be embedded in your HTML. This will allow you to copy the full upgrade code block to the clipboard with a single click.

Example Copy-to-Clipboard Snippet (Add around your Upgrade code block):

<script>  
function copyUpgradeCode() {  
 const codeBlock = document.getElementById('upgrade-code-block');  
 if (!codeBlock) return alert("Upgrade code block not found!");  
 const range = document.createRange();  
 range.selectNode(codeBlock);  
 window.getSelection().removeAllRanges();  
 window.getSelection().addRange(range);  
 try {  
 document.execCommand('copy');  
 alert("Upgrade code copied to clipboard!");  
 } catch (err) {  
 alert("Failed to copy: " + err);  
 }  
 window.getSelection().removeAllRanges();  
}  
</script>  
  
<!-- Usage Example -->  
<pre id="upgrade-code-block">  
// Paste your Upgrade code here...  
function exampleUpgrade(){  
 console.log("This is the upgraded function.");  
}  
</pre>  
<button onclick="copyUpgradeCode()">Copy Upgrade Code</button>