

Absolute Autonomous Systems Workflow Description

This client-demonstration pack documents the real implemented workflows in the static website: language settings, search recommendations, voice commands, video commands, automated consequence management, examples, training, shortcuts, contact capture, mobile navigation, and verification handover.

Generated: 2026-08-03T13:06:42.658Z

Workflow images: 12

Source: docs/workflow-description-manifest.json

1. Homepage Command Console Workflow

Show the user the live autonomous status, language layer, command controls, and consequence queue from the first screen.

Workflow 1 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Homepage Command Console Workflow

Show the user the live autonomous status, language layer, command controls, and consequence queue from the first screen.

Trigger / Input

User opens
`http://127.0.0.1:4180/index.html`.

Responsible Component

`index.html` hero section, command bar, and
`app.js` status renderer.

Ordered System Steps

1. Load the static page shell, CSS, app icon, manifest, and JavaScript.
2. Read the saved language preference or default to English.
3. Render the hero, command status, language layer, and consequence summary.
4. Expose Search, Voice, Video, Read page, language, and navigation controls.

Gates / Fallbacks

- All static assets must return HTTP 200.
- No blank hero, missing command status, or fatal console error is allowed.

Expected output

Expected output: the homepage displays the soft 3D console, translated hero text, ready command status, and working controls.

Trigger / Input

User opens
`http://127.0.0.1:4180/index.html`.

Responsible Component

`index.html` hero section, command bar,
and `app.js` status renderer.

Gates

- All static assets must return HTTP 200.
- No blank hero, missing command status, or fatal console error is allowed.

Expected output

Expected output: the homepage displays the soft 3D console, translated hero text, ready command status, and working controls.

2. 11 Language Settings Workflow

Adapt every implemented screen section and dynamic component to the selected South African language.

Workflow 2 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

11 Language Settings Workflow

Adapt every implemented screen section and dynamic component to the selected South African language.

Trigger / Input

User chooses a language from the top navigation selector.

Responsible Component

languageSelect control, translation packs, applyTranslations(), and localStorage.

Ordered System Steps

1. Validate the selected language code against available packs.
2. Store the preference in localStorage.
3. Translate data-i18n text, placeholders, status messages, rules, examples, and training cards.
4. Rebuild search index and visible recommendations in the selected language.

Gates / Fallbacks

- Unsupported language codes are ignored.
- Dynamic content must not keep English fallback text for non-English selections.

Expected output

Expected output: all visible app content follows the selected one of 11 South African languages.

Trigger / Input

User chooses a language from the top navigation selector.

Responsible Component

languageSelect control, translation packs, applyTranslations(), and localStorage.

Gates

- Unsupported language codes are ignored.
- Dynamic content must not keep English fallback text for non-English selections.

Expected output

Expected output: all visible app content follows the selected one of 11 South African languages.

3. Global Search Recommendation Workflow

Recommend searchable subjects from sections, examples, and training content across the whole app.

Workflow 3 of 12 - Absolute Autonomous Systems - Generated 2026-08-03 Recommend searchable subjects from sections, examples, and training content across the whole app.

Global Search Recommendation Workflow

Trigger / Input

User clicks Search, presses Ctrl+K, or uses a search voice command.

Responsible Component

search panel, buildSearchIndex(), updateSearch(), and result navigation

Ordered System Steps

1. Open the search panel and focus the input.
2. Index page sections, 20 examples, and training cards in the current language.
3. Filter recommended subjects as the user types.
4. Open the matching section when a recommendation is selected.

Gates / Fallbacks

- No-result state must remain readable.
- Result buttons must target real implemented sections.

Expected output

Expected output: recommended subjects appear immediately and jump the user to the correct section.

Trigger / Input

User clicks Search, presses Ctrl+K, or uses a search voice command.

Responsible Component

search panel, buildSearchIndex(), updateSearch(), and result navigation handlers.

Gates

- No-result state must remain readable.
- Result buttons must target real implemented sections.

Expected output

Expected output: recommended subjects appear immediately and jump the user to the correct section.

4. Voice Command Workflow

Let a supported browser navigate, search, read, change workflow state, and start video command mode using speech.

Workflow 4 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Voice Command Workflow

Let a supported browser navigate, search, read, change workflow state, and start video command mode using speech.

Trigger / Input

User clicks Voice or the hero Start voice control button.

Responsible Component

startVoice(), SpeechRecognition, handleCommand(), and command status UI.

Ordered System Steps

1. Check browser support for SpeechRecognition or webkitSpeechRecognition.
2. Set a visible listening status before microphone results arrive.
3. Listen in the active language speech locale.
4. Map recognized phrases to search, navigation, video, read page, next, previous, and language actions.

Gates / Fallbacks

- Unsupported browsers must show a readable fallback.
- Unknown phrases must show a no-match state instead of failing silently.

Expected output

Expected output: the command bar shows what happened and the requested action runs or displays a clear fallback.

Trigger / Input

User clicks Voice or the hero Start voice control button.

Responsible Component

startVoice(), SpeechRecognition, handleCommand(), and command status UI.

Gates

- Unsupported browsers must show a readable fallback.
- Unknown phrases must show a no-match state instead of failing silently.

Expected output

Expected output: the command bar shows what happened and the requested action runs or displays a clear fallback.

5. Video Motion Command Workflow

Use camera motion to move between sections or open search without using the keyboard.

Workflow 5 of 12 - Absolute Autonomous Systems - Generated 2026-08-03 Use camera motion to move between sections or open search without using the keyboard.

Video Motion Command Workflow

Trigger / Input

User clicks Video and grants camera access.

Responsible Component

startVideo(), getUserMedia(), motion canvas, and analyzeMotion().

Ordered System Steps

1. Check camera API support and request video permission.
2. Start the preview and set the active camera command overlay.
3. Compare video frames on the hidden motion canvas.
4. Map left motion to previous section, right motion to next section, and central motion to search.

Gates / Fallbacks

- Missing camera or denied permission must show a readable fallback.
- Motion commands are rate-limited to prevent accidental rapid navigation.

Expected output

Expected output: camera command mode becomes active and deliberate motion updates navigation or search state.

Trigger / Input

User clicks Video and grants camera access.

Responsible Component

startVideo(), getUserMedia(), motion canvas, and analyzeMotion().

Gates

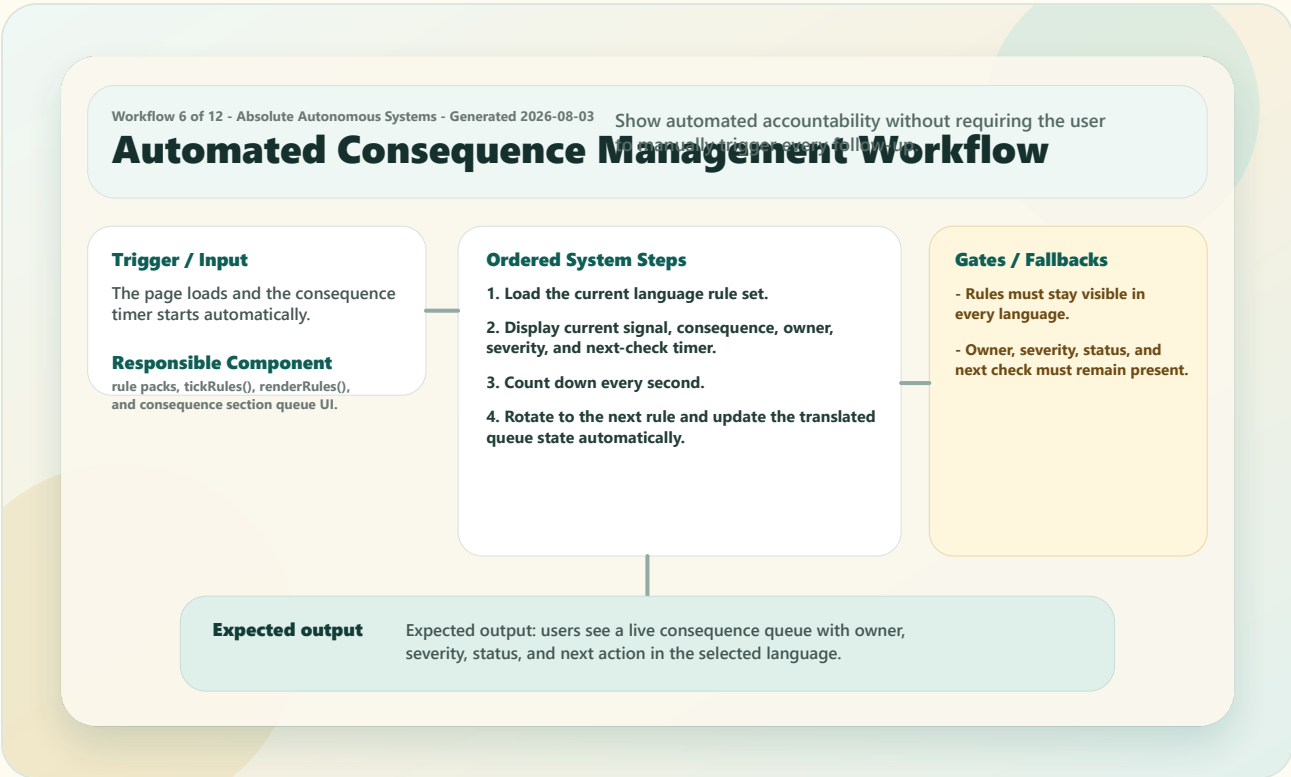
- Missing camera or denied permission must show a readable fallback.
- Motion commands are rate-limited to prevent accidental rapid navigation.

Expected output

Expected output: camera command mode becomes active and deliberate motion updates navigation or search state.

6. Automated Consequence Management Workflow

Show automated accountability without requiring the user to manually trigger every follow-up.



Trigger / Input

The page loads and the consequence timer starts automatically.

Responsible Component

rule packs, tickRules(), renderRules(), and consequence section queue UI.

Gates

- Rules must stay visible in every language.
- Owner, severity, status, and next check must remain present.

Expected output

Expected output: users see a live consequence queue with owner, severity, status, and next action in the selected language.

7. 20 Demonstration Examples Workflow

Give prospective users realistic scenarios that make the system value concrete.

Workflow 7 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Give prospective users realistic scenarios that make the system value concrete

20 Demonstration Examples Workflow

Trigger / Input

User opens Examples or filters the example list.

Responsible Component

demo subject packs, renderExamples(), and examples filter control.

Ordered System Steps

1. Build 20 sector-specific examples for the current language.
2. Display subject, sector, automation summary, and human value line.
3. Filter the examples when the user types a subject.
4. Reset the filter to restore all 20 examples.

Gates / Fallbacks

- Every language must render exactly 20 examples.
- Filtering must never leave the grid in a broken layout state.

Expected output

Expected output: the user can browse or filter 20 realistic examples and see the human value of automation.

Trigger / Input

User opens Examples or filters the example list.

Responsible Component

demo subject packs, renderExamples(), and examples filter control.

Gates

- Every language must render exactly 20 examples.
- Filtering must never leave the grid in a broken layout state.

Expected output

Expected output: the user can browse or filter 20 realistic examples and see the human value of automation.

8. Training, Tutorial, and Manual Workflow

Teach users how to use language settings, search, voice, video, consequence management, and contact flows.

Workflow 8 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Teach users how to use language settings, search, voice, video, consequence management, and contact flows.

Training, Tutorial, and Manual Workflow

Trigger / Input

User opens Training or the full user manual.

Responsible Component

training section, docs/USER_MANUAL.html, screenshot evidence, and workflow PDF.

Ordered System Steps

1. Render translated training cards in the app.
2. Open the full manual in a separate tab when requested.
3. Show screenshots from verified Playwright evidence.
4. Provide multilingual training guidance and workflow description material.

Gates / Fallbacks

- Manual link must resolve to a real file.
- Screenshot paths must point to generated verification images.

Expected output

Expected output: the user has tutorials, screenshots, and multilingual training material for the implemented workflows.

Trigger / Input

User opens Training or the full user manual.

Responsible Component

training section,
docs/USER_MANUAL.html, screenshot
evidence, and workflow PDF.

Gates

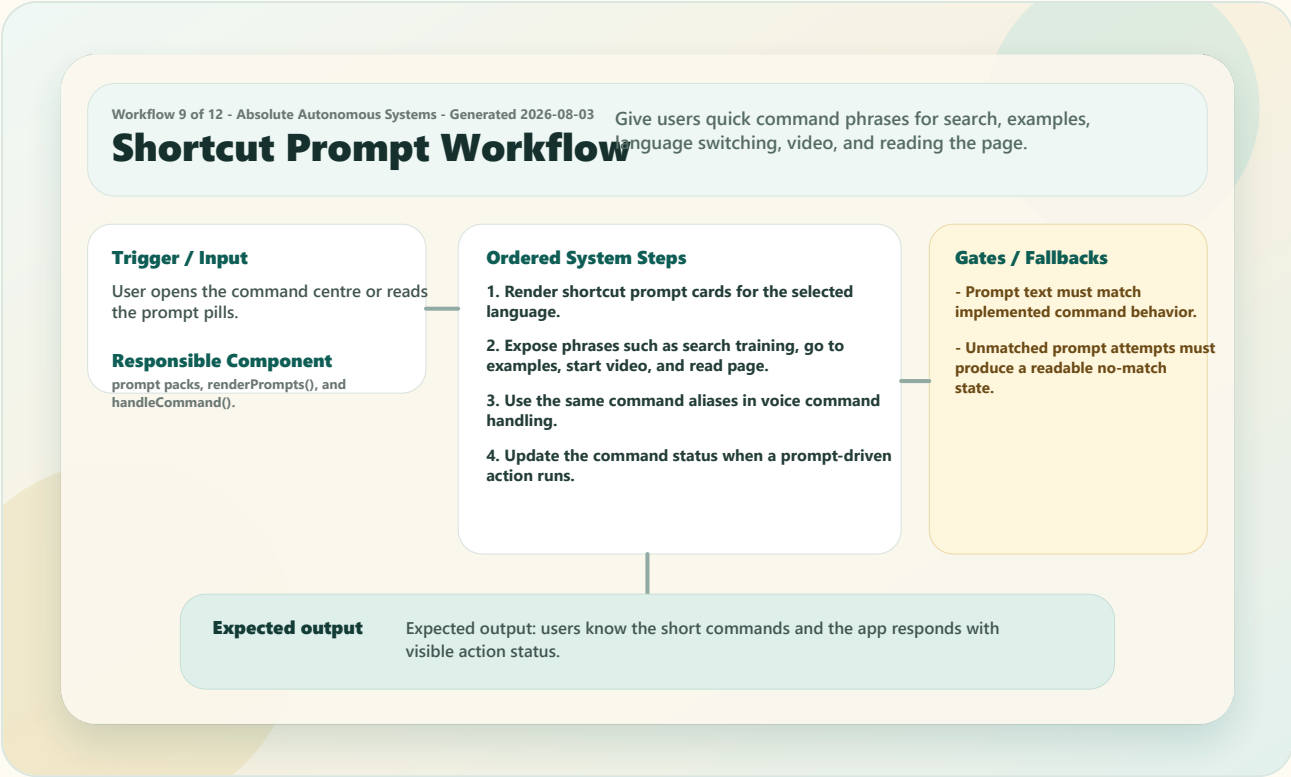
- Manual link must resolve to a real file.
- Screenshot paths must point to generated verification images.

Expected output

Expected output: the user has tutorials, screenshots, and multilingual training material for the implemented workflows.

9. Shortcut Prompt Workflow

Give users quick command phrases for search, examples, language switching, video, and reading the page.



Trigger / Input

User opens the command centre or reads the prompt pills.

Responsible Component

prompt packs, renderPrompts(), and handleCommand().

Gates

- Prompt text must match implemented command behavior.
- Unmatched prompt attempts must produce a readable no-match state.

Expected output

Expected output: users know the short commands and the app responds with visible action status.

10. Contact Request Capture Workflow

Capture a valid local request and give the user immediate confirmation without a backend dependency.

Workflow 10 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Capture a valid local request and give the user immediate confirmation without a backend dependency.

Contact Request Capture Workflow

Trigger / Input

User completes the Contact form and submits it.

Responsible Component

contactForm submit handler and form status UI.

Ordered System Steps

1. Validate required name, email, and message fields in the browser.
2. Prevent default form navigation.
3. Reset the form after valid submission.
4. Display a translated success state and toast.

Gates / Fallbacks

- The form must not navigate away from the static page.
- Success state must be visible to the user.

Expected output

Expected output: the form clears and the user sees that the next response task has been created.

Trigger / Input

User completes the Contact form and submits it.

Responsible Component

contactForm submit handler and form status UI.

Gates

- The form must not navigate away from the static page.
- Success state must be visible to the user.

Expected output

Expected output: the form clears and the user sees that the next response task has been created.

11. Mobile Navigation Workflow

Keep the app usable on narrow screens without clipping or hidden workflows.

Workflow 11 of 12 - Absolute Autonomous Systems - Generated 2026-08-03

Keep the app usable on narrow screens without clipping or hidden workflows.

Mobile Navigation Workflow

Trigger / Input

User opens the site on a mobile viewport and taps the menu button.

Responsible Component

responsive CSS, menuToggle, and navLinks state.

Ordered System Steps

1. Apply responsive layout constraints for the soft 3D UI.
2. Open or close the navigation menu from the menu button.
3. Let navigation links close the menu after selection.
4. Preserve search, language, command, examples, training, and contact access.

Gates / Fallbacks

- No horizontal overflow is allowed.
- Menu state must become visible when opened.

Expected output

Expected output: mobile users can open navigation and access every implemented section without layout overflow.

Trigger / Input

User opens the site on a mobile viewport and taps the menu button.

Responsible Component

responsive CSS, menuToggle, and navLinks state.

Gates

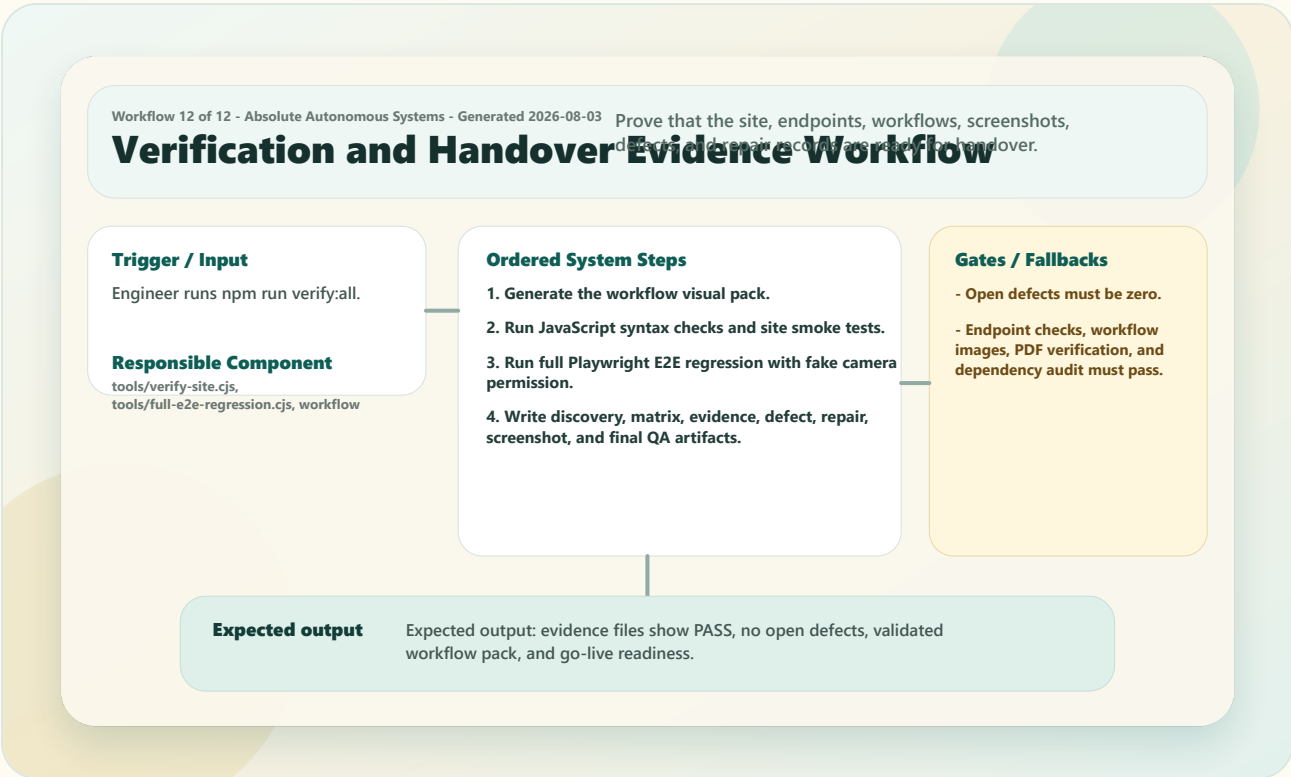
- No horizontal overflow is allowed.
- Menu state must become visible when opened.

Expected output

Expected output: mobile users can open navigation and access every implemented section without layout overflow.

12. Verification and Handover Evidence Workflow

Prove that the site, endpoints, workflows, screenshots, defects, and repair records are ready for handover.



Trigger / Input

Engineer runs npm run verify:all.

Responsible Component

tools/verify-site.cjs, tools/full-e2e-regression.cjs, workflow generator, and output/evidence.

Gates

- Open defects must be zero.
- Endpoint checks, workflow images, PDF verification, and dependency audit must pass.

Expected output

Expected output: evidence files show PASS, no open defects, validated workflow pack, and go-live readiness.