JPE Sims 4 Mod Translation Suite
UI/UX Product Requirements Document — v2
Desktop & iOS

1. Purpose & Scope

This PRD defines the user interface (UI) and user experience (UX) requirements for the JPE Sims 4 Mod Translation Suite, covering:
- Desktop Application ("JPE Studio") for Windows (and optionally macOS/Linux).
- iPhone Application (Swift/SwiftUI client).
- Shared UX for:
  - Reading all supported Sims 4 mod file types.
  - Translating to/from Just Plain English (JPE) and JPE-XML.
  - Compiling back to valid Sims 4 XML tuning.
  - Surfacing structured diagnostics and error reports.
  - Integrating Better Exceptions-style error logs and highlighting problematic mods/files.

This PRD focuses on UI/UX. Core engine, language design, and infrastructure are governed by the master SOP and engine PRDs.

2. Alignment with SOP

The UX must:
- Expose capabilities of the core translation engine and IR (XML → IR → JPE / JPE-XML and back).
- Surface validators (structural, semantic, version-specific) as first-class UI concepts.
- Use the diagnostics layer for human-friendly and machine-readable reports.
- Reflect the project structure and environments (core, languages, desktop, mobile/ios, cloud, plugins).
- Respect operating practices:
  - Never touch user saves.
  - Never overwrite original mods; always write outputs to new folders.
  - Operate only on user-supplied content.
- Follow development phases defined in the SOP:
  - Phase 3 — Desktop UI/UX.
  - Phase 4 — iPhone UI/UX.

3. Target Users & Personas

3.1 Modding Newcomer ("Curious Player")
- Owns Sims 4, uses several mods, has no technical background.
- Wants to understand what a mod does in plain English and perform simple edits (buff values, loot chances).
- Fear: breaking the game or corrupting saves.

UX Needs:
- High-level, friendly language ("What this mod does" summaries).
- Wizards and guardrails for common edits.
- Strong visual safety cues: read-only vs safe-to-edit areas.
- Undo-friendly flows and "Export to New Folder" defaults.

3.2 Intermediate Mod Author ("Tuner Hacker")
- Has edited XML tuning before, maybe used S4S or similar tools.
- Wants a faster workflow: JPE editing, quick roundtrips, and strong diagnostics.

UX Needs:

- Dual-pane XML ↔ JPE/JPE-XML.
- Autocompletion, inline validation, quick navigation to tests, buffs, loot tables.
- Clear build/compile pipeline and diagnostics panel.

3.3 Advanced Creator / Team (“Mod Studio”)
- Maintains multiple larger mods or modpacks.
- Cares about regression testing and compatibility.

UX Needs:
- Project-based workspace.
- Batch analysis and batch translations.
- Diagnostics dashboard, error trend visibility, exportable reports.

3.4 Support / Troubleshooting User
- Using or generating Better Exceptions-style logs.
- Wants to know which mod is breaking, in plain English, and what to do.

UX Needs:
- Easy import of exception logs.
- Visual highlighting of problematic mods/files.
- Suggestions or “next steps” per error.

4. UX Principles

1) Plain English First
- JPE is always one click away from any raw data.
- Critical actions and status messages use human-language labels (not engine jargon).

2) No-Fear Editing
- Defaults: never overwrite original content; always write to a separate output directory.
- Clear banners and icons for original vs generated vs experimental content.

3) Always Explain What’s Happening
- Build/compile processes show progress, context, and next steps.
- Diagnostics show why something failed, not just that it failed.

4) Progressive Disclosure
- Newcomers see simplified controls and wizard flows.
- Advanced users can expand to “Advanced View” (raw XML, IR semantics, version rules).

5) Consistency Across Desktop & iPhone
- Same mental model: Projects → Files → Editor → Build → Diagnostics.
- Shared iconography, labels, and severity colors.

6) Accessibility & Inclusivity
- Keyboard and controller-friendly on desktop (for Steam Deck-style setups later).
- Large tap targets, scalable fonts, and high contrast modes on iPhone.

5. Information Architecture

5.1 Top-Level Structure (Desktop)
- Project Workspace (Recent Projects, Open Folder, New Project Wizard).
- Main Workspace Layout:
  - Left: Project Explorer.
  - Center: Editor Area (Tabs + Split Views).
  - Right: Context Pane (Preview, Metadata, Quick Actions).
  - Bottom: Diagnostics & Build Console.
- Global Sections: Home/Project Switcher, Build & Tools, Diagnostics Center, Settings,

Help & Docs.

5.2 Top-Level Structure (iPhone)
- Tab Bar / Main Navigation:
  - Projects
  - Files
  - Diagnostics
  - Settings
- Screen Hierarchy:
  - Projects list → Project detail → Editor / Diagnostics detail.

6. Core User Flows (Cross-Platform)

6.1 Import Mod(s) and Create Project
- User selects “New Project” on desktop or iPhone.
- Chooses folder with mods, or individual files (.package, XML, STBL, etc.).
- App scans contents, detects supported file types, and offers project name + output directory.
- Project is created with:
  - “Source” tree (original files).
  - “Generated” tree (translations, compiled XML, reports).

6.2 View Mod in JPE / JPE-XML
- User clicks a mod file in the Project Explorer.
- Editor opens with JPE view as default (for supported content) and tabs for JPE-XML and raw XML (if available).
- App converts XML → IR → JPE/JPE-XML.
- User can expand/collapse sections: interactions, buffs, loot, tests, localization.

6.3 Edit and Compile
- User edits JPE or JPE-XML.
- Editor provides real-time validation and inline errors.
- User hits “Build” / “Compile”.
- Engine performs JPE/JPE-XML → IR → XML, runs validators.
- On success: green banner and generated files appear under “Generated”.
- On failure: diagnostics summary and highlighted error lines.

6.4 Diagnostics & Better Exceptions Flow
- User imports a Better Exceptions log or compatible error report.
- App maps stack traces/mod paths to files.
- Diagnostics Center shows grouped errors by mod/file/subsystem.
- Selecting an error opens the relevant file and explains the issue in JPE.
- User can apply suggested fix (if any), mark as “known issue,” and export reports.

7. Desktop Application — Core UI Requirements (JPE Studio)

7.1 Main Layout
- Three-pane layout with bottom diagnostics bar:
  - Left: Project Explorer (source, generated, logs with badges and file-type icons).
  - Center: Editor Tabs (multiple open files, split views, per-tab status).
  - Right: Context Pane (summaries, metadata, quick actions).
  - Bottom: Diagnostics & Build Console (Problems + Build Log).

7.2 Project Explorer UX
- Tree view with file type icons and error badges.
- Filter toggles for files with errors and by type (interactions, traits, buffs, loot, scripts).
- Right-click menu: Open in JPE, Open raw XML, Translate to JPE-XML, Rebuild, Reveal in

OS.

7.3 Editors

7.3.1 JPE Editor
- JPE syntax highlighting and autocompletion for enums, tuning references, and patterns.
- Inline diagnostics with underlines and margin icons.
- Actions: Preview XML, Validate Only, Build & Export, “Explain This Block.”

7.3.2 JPE-XML Editor
- XML-aware editing with indentation guides, tag autocompletion, attribute hinting.
- Hover to see resolved references and related tuning.

7.3.3 Raw XML Viewer
- Read-only by default; explicit opt-in editing.
- Syntax highlighting and folding.
- “Synchronize with JPE” indicator showing when XML diverges from JPE representation.

7.4 Diagnostics Center (Desktop)
- Dockable panel with columns: Severity, Message, File, Line, Category, Source.
- Filters by severity, source (core validator, scripts, exception logs), and category.
- Grouping by file or mod package.
- Better Exceptions integration with import button and last-log metadata.

7.5 Wizards & Templates
- New Project Wizard for root folder, mod paths, output directory, patch version.
- New Mod Pattern Wizard for Interaction, Trait, Buff, Loot templates.
- Fix-It Wizard that responds to common diagnostics by guiding STBL fixes, ranges, enums.

7.6 Settings (Desktop)
- General: theme, language.
- Engine & Versions: current engine/jpe/jpe-xml/diagnostics versions, optional version rollback with warnings.
- Diagnostics: category toggles, export destinations.
- Editor: font size, tab width, autosave, default view (JPE vs JPE-XML).
- Paths & Output: default project locations, output folder patterns, backup behavior.

8. iPhone Application — Core UI Requirements

8.1 Navigation
- Tab bar: Projects, Files, Diagnostics, Settings.

8.2 Projects Tab
- Project cards: name, last opened, location hint, error badges.
- Actions: New Project (import from Files or connect to desktop/cloud), rename, archive, remove.

8.3 Files Tab
- Shows files for the selected project.
- Structured and flat views with search/filter.
- Tapping a file opens JPE editor (or read-only viewer for unsupported/binary types).

8.4 JPE Editor (iOS)
- Focused editing with minimal chrome.
- Toolbar: Save, Validate, Build, View XML (preview).
- Quick insert controls for common patterns.
- Long-press on tokens for enum or reference info.
- Offline-first edits with queued sync and status indicator.

8.5 Diagnostics Tab (iOS)
- Summary of error/warning/info counts.
- List of diagnostics grouped by severity.
- Swipe actions: mark resolved, mark follow-up.
- Import exception logs from Files/clipboard.
- Tap to open related file or JPE explanation view.

8.6 Settings (iOS)
- Sync options, last sync timestamp.
- Editor font size and quick actions config.
- Diagnostics severity filter defaults.
- Storage: clear cache, export project as zip.

9. Accessibility Requirements

- Desktop:
  - Full keyboard navigation and shortcuts.
  - Screen-reader-friendly labels.
  - High contrast mode.
- iOS:
  - Dynamic Type, VoiceOver labels for file types, severities, and actions.
  - Minimum 44x44 pt tap targets.
  - Clear focus states and motion-reduced modes where applicable.

10. Non-Functional UX Requirements

- Performance: responsive UI for medium projects with progress indication during indexing.
- Responsiveness: smooth scrolling while background validation runs.
- Resilience: controlled, human-readable errors for engine failures; retry and export options.
- Safety: no auto-overwrites of source files; confirmation for destructive actions on outputs.

11. Telemetry & Logging (UX Perspective)

- Opt-in telemetry toggle with clear explanation.
- “Export Debug Bundle” action to package logs, diagnostics, and configuration into a zip.

12. Edge Cases & Special Scenarios

- Unsupported file types: clear “limited support” message and raw view.
- Roundtrip mismatches: diff summary, explanation of normalization, inspection option.
- Version mismatches: warnings and suggestions for matching patch/version rulesets.
- Broken or corrupted inputs: human-readable errors and guidance to use backups.

13. Roadmap Notes (UI/UX Priorities)

13.1 Phase 3.1 — Desktop MVP
- Project Explorer, basic JPE Editor, single-file build, basic diagnostics panel.

13.2 Phase 3.2 — Desktop Enhanced
- Split views, context pane, wizards, Better Exceptions integration, batch tools, dashboards.

13.3 Phase 4.1 — iPhone MVP
- Project list, read-only JPE viewer for existing projects, offline sync, basic diagnostics.

13.4 Phase 4.2 — iPhone Editor
- Full JPE editing, exception log import, build triggers (local or via desktop/cloud).

14. Global Navigation & Menus (Desktop)

14.1 Application Menu Bar
- File menu:
  - New Project…, Open Project…, Open Recent (list), Close Project.
  - Import Mods…, Import Exception Log…
  - Export Project as Zip…, Export Diagnostics Report…
  - Exit / Quit.
- Edit menu:
  - Undo, Redo, Cut, Copy, Paste, Select All.
  - Find…, Find in Files…, Replace…, Go To Line….
- View menu:
  - Toggle Project Explorer, Toggle Context Pane, Toggle Diagnostics Panel.
  - Split Editor Horizontal, Split Editor Vertical, Close Split.
  - Zoom In, Zoom Out, Reset Zoom, Toggle Word Wrap.
- Tools menu:
  - Run Full Project Diagnostics.
  - Batch Translate to JPE.
  - Batch Build / Rebuild All.
  - Import Better Exceptions Log.
  - STBL Tools (Key Browser, Missing Key Finder).
- Window menu:
  - Next Editor Tab, Previous Editor Tab, Close Tab, Reopen Closed Tab.
- Help menu:
  - Open Documentation.
  - Quickstart Tutorial.
  - Keyboard Shortcuts.
  - Check for Updates.
  - About JPE Studio.

14.2 Global Toolbar (Desktop)
- Project selector dropdown.
- Buttons (with icons + tooltips):
  - New Project.
  - Open Project.
  - Save (and Save All).
  - Build File.
  - Build Project.
  - Run Diagnostics.
  - Import Exception Log.
  - Toggle Split View.
  - Toggle Diagnostics Panel.
  - Settings (gear icon).
- States:
  - Disabled when context not valid (e.g., Build File when no editor tab is active).
  - Spinner or progress bar when background tasks are running.

15. Toolbars, Buttons & Command Palette (Desktop)

15.1 Editor Toolbar (Per-Editor)
- Buttons:
  - Undo / Redo.
  - Find in File.
  - Toggle Word Wrap.

  - Toggle Line Numbers.
  - Preview XML (split view).
  - Format / Beautify (JPE or XML).
  - Show Invisible Characters (tabs, spaces).
  - Comment / Uncomment selection (for JPE syntax where applicable).
- Status indicators:
  - Current file encoding.
  - Line/Column position.
  - Validation state (OK / Warnings / Errors).

15.2 Command Palette
- Shortcut: Ctrl+Shift+P (Windows/Linux), Cmd+Shift+P (macOS).
- Searchable list of actions:
  - Open File…, Go To Symbol…, Run Diagnostics, Import Log, Toggle Theme, etc.
- Supports fuzzy search and keyboard-only navigation.

15.3 Status Bar
- Left side:
  - Active project name.
  - Active file path (truncated).
- Center:
  - Engine/patch version chip (click opens Version settings).
  - Diagnostics summary chip (Errors, Warnings; click opens Diagnostics Center).
- Right:
  - Background jobs indicator (spinner with count; click opens “Background Tasks” popup).
  - Current theme (icon; click toggles theme options).
  - Profile / account menu for future cloud integration.

16. Search, Filter & Navigation Patterns

16.1 Global Search
- Shortcut: Ctrl+Shift+F.
- Modal or docked panel with:
  - Search field.
  - Filters: file type, directory scope, include/exclude patterns.
  - Results list with file path and snippet.
- Clicking a result opens file and scrolls to match.

16.2 Quick Open
- Shortcut: Ctrl+P / Cmd+P.
- Overlay with:
  - Filename search (fuzzy).
  - Category icons for file types.
- Selecting result opens in editor.

16.3 Symbol Navigation
- For JPE-XML and XML views:
  - Go To Symbol menu listing tuning IDs, traits, interactions, loot actions.
  - Clicking a symbol jumps to its definition.

16.4 In-Editor Navigation Aids
- Breadcrumb bar above editor:
  - Shows hierarchy: Project / Folder / File / Section (e.g., Buffs → Mood Buff).
- Folding controls:
  - Collapse/expand sections in JPE and XML for large tuning files.

17. History, Rollback & Diff UI

17.1 File History Panel
- Access via:
  - Right-click file → View History.
  - Editor toolbar “History” button (clock icon).
- Shows chronological list of saved versions with timestamp and comment (optional).
- Actions per version:
  - View read-only snapshot.
  - Compare to current (diff view).
  - Restore as current (with confirmation dialog).

17.2 Diff Viewer
- Side-by-side view:
  - Left: older version.
  - Right: newer version.
- Highlighted additions, deletions, and changes.
- Navigation controls:
  - Next/Previous change.
  - Sync scroll toggle.

18. Mod Library, Lookup & External Feeds (Phase 5+)

18.1 Mod Library Browser (Optional / Advanced)
- Additional sidebar or tab for “Mod Library”:
  - Local mods (installed in projects).
  - Watched mods from external sources (e.g., ts4rebels, mod sites).
- List items:
  - Mod name, author (if known), version, last updated date.
  - Status badges: up-to-date, update available, issues detected.

18.2 External Feed / Issue Stream
- “Issues Feed” panel aggregating:
  - Diagnostics from local builds.
  - Parsed RSS/JSON feeds from mod sites (if enabled in settings).
- Cards contain:
  - Source (local vs external).
  - Affected mod(s).
  - Summary and link to details.
- Actions:
  - Open in browser.
  - Attach to Project Diagnostic list.
  - Mark as ignored.

19. Notifications, Dialogs & Error Handling

19.1 Notification Toasts
- Non-blocking notifications for:
  - Build success/failure.
  - Diagnostics completed.
  - New feed items detected.
- Behavior:
  - Appear at top-right for a few seconds.
  - Collapse to bell icon history list for review.

19.2 Modal Dialogs
- Confirmations for:
  - Deleting generated outputs.
  - Restoring file versions.
  - Changing engine/patch version.

- Error dialogs:
  - Clear title (“Engine Error”, “Unexpected Crash in Validator”).
  - Explanation in plain English.
  - Buttons: Retry, Export Debug Bundle, Cancel.

19.3 First-Run & Onboarding
- First-launch flow:
  - Short multi-step intro explaining Import → Edit → Build.
  - Option to create a sample project with sample mods.
- Contextual tips:
  - One-time tooltips pointing at key UI regions (Explorer, Editor, Diagnostics Center).

20. Keyboard Shortcuts & Gestures

20.1 Desktop Shortcuts (Examples)
- Project/Files:
  - Ctrl+N: New Project.
  - Ctrl+O: Open Project.
  - Ctrl+S: Save File, Ctrl+Shift+S: Save All.
- Navigation:
  - Ctrl+P: Quick Open.
  - Ctrl+Shift+F: Find in Files.
  - Ctrl+G: Go To Line.
- Editor:
  - Ctrl+F: Find in File.
  - Ctrl+/: Toggle comment.
  - Ctrl+Alt+L: Format / Beautify.
- Diagnostics:
  - F8: Next Problem.
  - Shift+F8: Previous Problem.
  - Ctrl+` (backtick): Toggle Diagnostics Panel.

20.2 iOS Gestures
- Pull to refresh project feed or diagnostics list.
- Swipe left on list rows for quick actions (rename, delete, mark resolved).
- Long-press on text for “Explain Selection” and “Copy JPE Snippet”.

21. Iconography, Status Indicators & Theming

21.1 Icon Set
- Base icon types:
  - Files (XML, JPE, JPE-XML, scripts, logs).
  - Status (info, warning, error, success).
  - Actions (build, run, import, export, history, diff).
- Style:
  - Simple, high-contrast line icons with minimal color accents.

21.2 Status Colors
- Errors: red.
- Warnings: amber/yellow.
- Infos: blue.
- Success: green.
- Respect theme constraints and accessibility contrast ratios.

21.3 Themes
- Dark, Light, and High-Contrast variants.
- Theme selection accessible from Settings and status bar theme chip.
- Optional “Sims-like” accent mode using familiar greens/blues for fun, but always

subordinate to accessibility.

22. Additional iOS UI Elements & Buttons

22.1 Quick Actions
- Home screen quick actions:
  - New Project.
  - Open Recent Project.
  - Import Exception Log.
- In-app floating action button (FAB) on Files and Diagnostics screens:
  - On Files: “Import Mods” or “New File (template)”.
  - On Diagnostics: “Run Diagnostics Now”.

22.2 Local Notifications
- Optional notifications for:
  - Long-running diagnostics completing.
  - New issues discovered while app is in background.
- Tapping notification deep-links into relevant project/diagnostic detail.

23. Wireframes & Screen Blueprints (Summary)

23.1 Desktop Main Workspace
- Global menu + toolbar at top.
- Project Explorer left, Editor center, Context Pane right, Diagnostics bottom.
- Clear empty/project-loaded states and strong visual hierarchy.

23.2 Diagnostics Center
- Filter bar, list with columns, and details pane.
- Row interactions for navigation and copying, detail pane for JPE explanations and actions.

23.3 iOS Project → File → Editor Flow
- Projects screen with cards and FAB for creating/importing.
- Files screen with segmented control (Types / All), search, and swipe actions.
- Editor screen with top navigation, main text area, bottom toolbar buttons, diagnostics markers, and XML preview sheet.

24. Summary

This v2 UI/UX PRD extends the original specification with all major UI elements and buttons required for a mature developer-grade tool: full menu and toolbar structure, command palette, global and in-file search, history and diff views, optional mod library and issue feeds, rich notifications and dialogs, robust keyboard shortcuts, mobile gestures, and detailed iconography and theming. Together, these requirements ensure that the JPE Sims 4 Mod Translation Suite is not only functionally powerful but also discoverable, safe to use, and pleasant for everyone from curious players to professional mod studios.