Design & Engineering Best Practices

AI Retirement Income Planner — Version 7

Technical overview prepared for management review · June 2026

> This application is a single, self-contained web tool that helps people model retirement income across multiple life phases, currencies, and tax regimes. Beyond its features, it embodies a deliberate set of **design and engineering practices** that make it **secure by construction, portable, correct, and trustworthy**. The summary below is organised for both technical and non-technical review.

Zero-dependency · No backend · Privacy-first · Defense-in-depth · Deterministic engine · Backward-compatible · Progressive disclosure · Responsible AI

1 Architecture & delivery

- **Single self-contained file.** The entire application — markup, styles, and logic — ships as one HTML file. No build step, package manager, installer, or server is required. *Anyone can read, audit, archive, or deploy the whole product as a single artifact.*
- **Runs anywhere, including offline.** It opens directly from disk or from any static host and works with no network connection. *Maximum portability and resilience; nothing to provision or keep running.*
- **No runtime third-party fetches.** The charting library is inlined into the file after verifying its published SHA-512 integrity hash, and the fonts are self-hosted (embedded). *Removes supply-chain exposure and external points of failure.*

2 Security & privacy (defense in depth)

- **No server-side attack surface.** There is no backend, database, or user-account system — eliminating an entire class of server-side risk by design.
- **Data stays with the user.** The financial plan lives only in the user's own browser and any file they choose to export; nothing is transmitted to the vendor.
- **Content-Security-Policy.** A restrictive policy constrains where the app may send data, so injected content cannot exfiltrate information to arbitrary destinations.
- **No dynamic code execution.** The code contains no `eval()` or equivalent; imported plans are parsed strictly as data and never executed.
- **Consistent output escaping.** Any text a user types is HTML-escaped before display, preventing cross-site-scripting injection.
- **Input sanitisation.** Imported files are normalised and validated — non-finite numbers are coerced to safe defaults — so malformed or hostile data cannot corrupt the model.
- **Careful secrets handling.** The only credential is the user's own optional AI key, stored locally and sent only to the official endpoint, with a session-only mode for shared devices.

3 State management & data integrity

- **Single source of truth.** All application state lives in one well-defined object, persisted as a unit. *Predictable behaviour and a single place to reason about the plan.*
- **Canonical internal units.** Every monetary value is stored internally in one base currency and converted only at the display/input boundary. *Avoids rounding drift and currency-mixing defects.*
- **Separation of concerns.** State, calculation, and rendering are distinct layers that can be reasoned about and changed independently.

4 Calculation engine design

- **Layered, pure-function pipeline.** The math flows through focused, side-effect-free stages — phase configuration, all-phases, single-phase, then month-by-month simulation. *Each layer is small, testable, and predictable.*
- **One engine, many consumers.** The same deterministic core powers the projection, Monte Carlo simulation, historical backtest, stress test, and scenario comparison. *The math is correct in one place rather than re-implemented per feature.*
- **Explicit, inspectable arithmetic.** Compounding, tax brackets, inflation, and benefit growth are computed step-by-step with documented rules, not opaque shortcuts.

5 Backward compatibility & robustness

- **Forward-safe persistence.** Saved plans are merged over a defaults template on load, so a plan created before a feature existed automatically gains the new fields with sensible values. *Old files never break as the product evolves.*
- **Inert-by-default features.** New capabilities default to "off" or zero, so introducing them produces identical results for existing plans — a built-in regression guarantee.
- **Fallback chains.** Fields resolve through safe fallbacks, tolerating missing or partial data without error.

6 User experience & progressive disclosure

- **Context-aware interface.** Sections that don't apply to the user's situation — by currency, residency, or filing status — are hidden automatically, reducing clutter and the chance of error.
- **Navigation that scales.** As the tool grew, the many settings were organised into sub-section ("pill") navigation so the user sees one coherent group at a time.
- **Education built in.** Inline info popovers, an A–Z glossary, a structured help system, concept primers, and contextual explainer links make a genuinely complex domain approachable.
- **Change transparency.** A "changed since baseline" indicator gives an at-a-glance audit of exactly what the user has edited in a session.
- **Inclusive presentation.** Multi-currency and multi-residency support, plus light/dark and additional visual themes, are driven cleanly by CSS custom properties.

7 Responsible AI integration

- **Opt-in and consent-based.** AI features are optional and key-gated, and the interface is explicit that figures are sent to the provider under the user's own account.
- **Constrained and verifiable.** AI suggestions may only write numeric values to named fields — never execute code — and every proposal is simulated against the real engine and requires explicit user approval, with one-click undo.
- **Always in sync.** An internal fingerprint of the plan ensures the AI reasons over the current numbers rather than stale context.

8 Engineering discipline & maintainability

- **Self-documenting code.** Key functions carry explicit contracts and rationale; naming and data-binding conventions are consistent throughout.
- **Pattern reuse.** Shared UI building blocks (for example, the same sub-navigation component) are reused rather than reinvented.
- **A change-safety checklist.** Every change is held to a documented verification routine before it is considered done.

Verification step	What it confirms
Persistence round-trip	New data survives save/load and merges cleanly into older plans.
Calculation propagation	A new input reaches every layer of the engine that needs it.
AI-context sync	The AI sees the new field and its cache refreshes correctly.
Educational surfaces	Help, glossary, and tooltips are updated to match the change.
Backward-compatibility regression	Existing plans produce identical results after the change.

Prepared as a technical overview of the AI Retirement Income Planner (v7). Describes design and engineering practices embodied in the application as of June 2026. The tool is an educational planning aid, not financial, tax, or legal advice.