Implementation Notes Extraction Protocol (INEP) v0.1

1. Purpose

INEP defines how an assistant interrogates an existing implementation and its relevant development conversation in order to recover a reviewable implementation-notes artifact.

Its job is narrow:

- recover non-governing but materially useful implementation context actually supported by the available evidence
- distinguish retained implementation knowledge from incidental detail
- preserve current successful tactics, local constraints, and learned operational context without promoting them to invariants
- produce a compact artifact that helps future reconstruction, handoff, or contract authoring

2. What INEP Is

INEP is:

- an implementation-context recovery protocol
- a retention-filtering protocol
- a review-oriented extraction protocol
- a bootstrap protocol for retaining non-governing implementation knowledge from an existing system

3. What INEP Is Not

INEP is not:

- a source-of-truth protocol
- a contract
- a runtime gating protocol
- a builder-ready contract by itself
- a chance to invent rationale for local tactics
- a dump of every implementation detail
- a code explanation exercise

4. Inputs

Required:

- target codebase, selected files, or diff
- relevant chat or session history tied to the implementation being analyzed

Optional only when explicitly supplied:

- tests
- bug reports
- tickets
- commit history
- screenshots or UX notes

- existing docs/specs/contracts

This should mirror SOTEP's input discipline so the two protocols compose cleanly.

5. Authority Topology

For INEP:

- the current implementation is the behavioral anchor for what currently works
- conversation history is supplemental evidence of intent, correction, local tradeoffs, debugging history, environment assumptions, and accepted tactics
- INEP must not invent implementation rationale beyond what the available evidence supports
- INEP must not promote retained notes into governing truth merely because they worked once

This mirrors SOTEP's anchor model, but changes the extraction target from governing truth to retained non-governing knowledge.

6. Core Objective

For each meaningful implementation element, ask:

- what does this unit do?
- why does it appear to exist locally?
- what practical problem does it seem to solve?
- is it non-governing but still worth retaining?
- would losing this note likely cause expensive rediscovery?

- is this an environmental constraint, successful tactic, local workaround, operational assumption, or current successful path?

- or is it merely incidental detail not worth preserving?

That is the key shift from SOTEP:

not governing-vs-not, but retain-vs-omit within the non-governing surface.

7. Retention Rule

A candidate note may be included only if all of the following are true:

- it is actually supported by the current inputs

- it is non-governing

- it is semantically meaningful

- it reflects current successful implementation context, local constraint, operational assumption, or learned tactic

- losing it would plausibly increase rediscovery cost in future iteration, handoff, or reconstruction

- it can be stated honestly without overstating scope or certainty

If any of these fail, omit it or mark it ambiguous.

This is the direct analogue of SOTEP's promotion rule, but for retained non-governing material instead of preserved truth.

8. Classification

Each analyzable unit or note candidate should be classified as one of:

- Retain

    - non-governing and worth preserving
- Ambiguous
    - meaningful candidate exists, but support or retention value is too weak to promote confidently
- Incidental
    - non-governing and not worth retention
- Unresolved
    - meaningful local logic exists, but its practical role cannot be reconstructed confidently

This borrows the exercise's classification discipline but swaps in a retention-specific positive class.

9. Preferred Note Types

INEP should give extra attention to things like:

- current successful access/runtime path
- environmental or deployment constraints
- successful local tactics that solved repeated friction
- local workarounds that are replaceable but worth remembering
- operational assumptions
- builder-relevant current behavior
- boundaries on what is intentionally left flexible
- current known-good sequencing or ordering that is non-governing but practically important

INEP should avoid:

- language boilerplate
- arbitrary code organization
- style
- trivial syntax choices
- non-semantic local variation

That follows the exercise's rule to focus on semantically meaningful logic and not waste effort on obvious boilerplate.

10. Required Output

Primary output:

- implementation-notes.md

Optional output:

- implementation-notes-extraction-report.md

Required contents of

implementation-notes.md

For each extracted note, include at minimum:

- Implementation-note ID
- Short title

- Statement
- Scope statement
- Note type: current successful tactic, environment constraint, operational assumption, local workaround, or other bounded category
- Support type: CODE-ENFORCED, CHAT-EXPLICIT, COMBINED-SUPPORT, or AMBIGUOUS
- Explicit or inferred
- Supporting unit IDs
- Source references
- Notes on ambiguity or replaceability
- Recommended status:
    - Preserve as Implementation Note
    - Needs Human Confirmation
    - Omit from contract authoring inputs

Required contents of implementation-notes-extraction-report.md

If requested, include:

- extracted notes
- omitted candidates
- ambiguous candidates
- unresolved local logic

- repeated notes that were merged
- uncertainties likely to affect handoff or reconstruction
- narrow human questions

11. Success Criteria

INEP succeeds when:

- a reviewable implementation-notes.md is produced
- extracted notes are traceable to current input materials
- no governing truth is accidentally invented
- retained notes are clearly non-governing
- incidental detail is filtered out
- ambiguity is surfaced instead of hidden
- prior-context residue does not leak in unless present in current inputs

12. Failure Criteria

INEP fails when:

- unsupported implementation rationale is introduced
- governing truths are silently downgraded into notes
- notes are promoted beyond their supported scope
- incidental noise is dumped as if it were meaningful retained context
- the result becomes a code-tour or prose summary instead of retained implementation knowledge

- prior-example residue leaks into the output

13. Relationship to SOTEP

SOTEP and INEP should be runnable independently.

Normal workflow:

- run SOTEP only

Heavier reconstruction / measurement workflow:

- run SOTEP
- run INEP
- feed both into a contract-authoring brief
- then author a contract

That keeps SOTEP light while still giving you a principled way to recover retained implementation context when you actually need it.