Template Design Documentation

Overview

The refined prompt template is the standardized output format for all prompt refinement operations. This document explains the design rationale and field justification.

Core Principles

1. Transparency over Assumptions
   Never silently fill gaps. All inferences are made explicit with confidence levels.

2. Source Attribution
   Each requirement tracks which modality (text, image, or document) contributed it. This enables traceability and conflict detection across inputs.

3. Conflict Flagging over Auto-Resolution
   Contradictions are flagged with evidence rather than resolved automatically. Final decisions are left to humans with full context.

4. Explicit Risk Assessment
   Assumptions are documented along with the risk if they are incorrect.

Template Structure

Complete Template Schema

```
{
  "refined_prompt": {
    "intent": {
      "purpose": "",
      "problem_being_solved": "",
      "domain": "",
      "confidence": "high | medium | low"
    },
    "requirements": [
      {
        "text": "",
        "status": "confirmed | inferred",
        "source": "text | image | document"
      }
```

```
    ],
    "constraints": [
      {
        "text": "",
        "status": "confirmed | inferred",
        "impact": ""
      }
    ],
    "deliverables": [
      {
        "item": "",
        "confidence": "high | medium | low"
      }
    ],
    "conflicts_and_ambiguities": [
      {
        "issue": "",
        "evidence": {},
        "impact": ""
      }
    ],
    "assumptions": [
      {
        "assumption": "",
        "risk_if_wrong": ""
      }
    ]
  },
  "validation": {
    "is_valid_prompt": true,
    "rejection_reason": null,
    "completeness_score": 0.0
  },
  "processing_metadata": {
    "input_modalities": [],
    "notes": []
  }
}
```

Key Template Fields

1. Intent Section

- purpose (required, minimum 10 characters): What needs to be built
- problem_being_solved (strongly recommended): Why this solution is needed
- domain (optional): Business or technical context
- confidence (high / medium / low): Certainty of intent interpretation

Design choice: Purpose and problem are kept separate because solutions can be clear even when the underlying problem is not explicitly stated.

2. Requirements Section

An array of objects with fields `{text, status, source}` where status is `confirmed` or `inferred`.

- Essential field: At least one requirement must be present
- Accounts for 40% of the completeness score
- Source attribution enables multi-modal conflict detection, such as when text describes one product and an image suggests another
- Missing requirements are surfaced through constraints or assumptions, never as requirements with a “missing” status

3. Constraints Section

An array of objects `{text, status, impact}` representing budget, timeline, or technical limitations.

- Optional field
- Accounts for 15% of the completeness score
- Constraints are often implicit in early prompts and should not block validation

4. Deliverables Section

An array of `{item, confidence}` describing expected outputs or artifacts.

- Optional field
- Accounts for 10% of the completeness score
- Often derivable from the listed requirements

5. Conflicts and Ambiguities Section

An array of `{issue, evidence, impact}` entries.

- Conflicts are flagged, never auto-resolved
- Supports batch processing without user interaction
- An empty conflicts array provides a 5% completeness bonus
- Example: Text requests a movie booking system while an image shows a food delivery interface

6. Assumptions Section

An array of `{assumption, risk_if_wrong}` entries.

- Makes gap-filling explicit
- Documents inferred decisions and their potential impact
- Critical for maintaining transparency

7. Validation Section

- is_valid_prompt: Boolean indicating whether essential checks passed
- rejection_reason: Explanation when validation fails
- completeness_score: Weighted score between 0.0 and 1.0

The score is weighted as follows:

- Requirements: 40%
- Intent: 30%
- Constraints: 15%
- Deliverables: 10%
- No conflicts: 5%

8. Processing Metadata

- input_modalities: List of input types processed
- notes: Processing warnings or special cases

This metadata is used for debugging and analysis and does not affect validation.

Design Alternatives Considered

Two-Stage Pipeline
Interactive questioning to fill gaps after extraction.
Rejected because it breaks batch processing, adds complexity, and increases latency.

Richer Requirement Structure
Adding priority, rationale, and dependencies to each requirement.
Rejected because this information cannot reliably be extracted from initial prompts and would introduce many unknown values.

Hierarchical Requirements
Grouping requirements into functional, non-functional, or security categories.
Rejected because classification is often ambiguous and a flat structure with source attribution is simpler.

Auto-Resolve Conflicts
Using priority rules such as “text overrides image” to choose a winner.
Rejected because it violates the transparency principle and hides important contradictions from users.

Complete Template Schema

```
{
```

```
  "refined_prompt": {
    "intent": {
      "purpose": "",
      "problem_being_solved": "",
      "domain": "",
      "confidence": "high | medium | low"
    },
    "requirements": [
      { "text": "", "status": "confirmed | inferred", "source": "text
| image | document" }
    ],
    "constraints": [
      { "text": "", "status": "confirmed | inferred", "impact": "" }
    ],
    "deliverables": [
      { "item": "", "confidence": "high | medium | low" }
    ],
    "conflicts_and_ambiguities": [
      { "issue": "", "evidence": {}, "impact": "" }
    ],
    "assumptions": [
      { "assumption": "", "risk_if_wrong": "" }
    ]
  },
  "validation": {
    "is_valid_prompt": true,
    "rejection_reason": null,
    "completeness_score": 0.0
  },
  "processing_metadata": {
    "input_modalities": [],
    "notes": []
  }
}
```