Metatron - Prompt Architecture & Tutor Session Design v1.2

# ✅ Prompting System Architecture — Final Structure (V2.1)

## 🔑 1. Key Terms (Refined)

|  |  |
| --- | --- |
| Term | Description |
| Master Prompt | System-admin-controlled, AI-personality-defining base prompt. Defines tutor's tone, behavior, methods, teaching strategy, scope, etc. Controlled by backend or admin console. |
| Goal-Specific Prompt | Optional augmentation prompt per coaching goal. Advanced users can edit this during goal setup. Defaults to none. |
| Session Prompt | Dynamically generated at session start. Combines: Master Prompt + Goal-Specific Prompt + Summarized Goal Description + File Metadata + Optional Progress Snapshot. |
| Live Tutoring Messages | Subsequent user-tutor messages. Tutor responds with awareness of current goal, prior conversation, and session context. |
| Tutoring Cycle Tracker | Core logic engine tracking which instructional stages each topic has passed through: Engage → Explain → Discuss → Practice → Test → Reflect → Repeat. |

## 🔄 2. Prompt Lifecycle Flow (Expanded)

| Step | Description |
| --- | --- |
| 1 | User clicks "Start Session" on a goal |
| 2 | Backend builds Session Prompt: → Master Prompt → Goal Prompt (if any) → Goal Summary → Uploaded file summaries (if parsed) → Milestone plan from Goal Plan Tracker → Prior Progress Snapshot (optional) |
| 3 | Model session is initialized using this Session Prompt |
| 4 | Live conversation begins. All user and AI messages are appended to the chat record |
| 5 | Session Context Buffer: A lightweight, dynamic context summary (e.g., last 3–10 messages or condensed insight log). Stored and served from app backend or local wrapper. Configurable per model strategy. |
| 6 | Asynchronous Chat Summarization Engine: Triggers every N messages or at session end. → Summarizes chat logs → Updates *Progress Summary Engine* → Updates *Goal Plan Tracker* |
| 7 | On next session, inject Progress Snapshot (not full logs) into new Session Prompt. Ensures continuity without bloating the token window. |

## 🧠 Tutor Engine Breakdown (Expanded)

## 🧱 A. Core Engines

| Component | Description |
| --- | --- |
| Chat History DB | Stores raw message logs by session ID and goal. Never sent to model. Used for audit and re-summarization. |
| Progress Summary Engine | Periodically reads chat logs, diagnoses instructional progress, infers mastery per topic using stage-tracking and quiz performance. Writes: insight summaries, recommended focus, flags for remediation. |
| Goal Plan Tracker | Stores canonical milestones from parsed files or manual goal setup. Each milestone has status: Not Started, In Progress, Covered, Requires Repeat. Also includes per-topic stage cycle data. |
| Content Understanding Tracker | Tracks how deep a user has engaged with content: → Initial concept intake → Stages of knowledge transfer → Engagement with high-level concepts and sub-components → Reflection and test readiness. |
| Instructional Stage Tracker (Tutoring Cycle) | For each topic: Engage → Explain → Discuss → Practice → Test → Reflect → Repeat. Tracked per topic in DB and report files. Integrated with summary reports and tutoring logic. |

## 📈 Tracker Architecture

## These are separate but linked engines/tables (can share goal/session ID):

## Goal Plan Tracker (including content to be studied) — What must be taught (curriculum milestones). What content has been parsed, scaffolded, and sequenced from uploaded files or connected DB.

## Progress Summary Engine — What has been achieved, flagged, or missed.

## Tutoring Cycle Tracker — How deeply each topic has been engaged (7 stages).

## These will power:

## Live Chat Context Buffer

## Session Prompt Progress Injection

## Live Status Summaries (“How far am I?”)

## PDF Report Generator

## 💾 Prompt Strategy by Model Type (Refined)

| Model Type | Memory Support | Prompt Strategy |
| --- | --- | --- |
| OpenAI GPT-4-turbo | Yes | One-time system prompt + conversation history window (OpenAI handles memory). Use chat history + summarized insight on new session. |
| Local/Open-Source (Mistral, LLaMa) | No | Re-send: condensed Master Prompt + Session Prompt + [optional last few user/AI messages] + [memory snapshot if needed]. Use backend memory store. |
| Hybrid (Falcon, Claude) | Partial | Use abstracted prompt compiler per session: → Master Prompt + dynamic content + insights → Configurable by token window size and caching rules. |

## 🧠 Important: Strategy is dynamically abstracted. The prompt compiler detects model type and chooses the correct assembly method.

## 🧠 Tutor Memory System Design

• Chat History Log — Stored per goal+session ID+ every prompt & response, full complete log

• Progress Summary Engine — Periodically summarizes chat, extracts insights in one of the three trackers mentioned below

• Goal Plan Tracker — Canonical milestones and progress % linked to conversation and the Goal plan configured by the user)

• **Progress Summary Engine/trakcer** — What has been achieved, flagged, or missed. Tracked against the actual Goal Plan Tracker

**Tutoring Cycle Tracker/Engin** — How deeply each topic has been engaged (7 stages) against Tutor Framework

• Memory Store — Optional vector/text memory store per session. (must still learn or understand how this will be used, it my by model or tutor back end done dynamically not sure yet)

• Long-Term Insights — Fed back into next session's prompt context. (stored in the Progress Summary Engine)

## 📐 Prompt Engineering Best Practices

| Practice | Implementation |
| --- | --- |
| Prompt Length | Master Prompt ≤ 1 page (~750 tokens); Session Prompt ≤ 2 pages (~1500 tokens max) |
| Avoid Redundancy | Inject only required memory or insight on new session |
| Token Efficiency | Use summarized milestones and tutoring stage status, not full message logs |
| Chunking Logic | If needed, send last 2–5 messages OR condensed “progress snapshot” instead of entire history |
| On-Demand Reports | Users can request “Show my progress,” “Give me my report” anytime; GPT uses shared table logic |

## 🧱 What Should Be Built First

|  |  |
| --- | --- |
| ✅ Prompt Compiler | Generates the full prompt for any session given a goal\_id |

|  |  |
| --- | --- |
| ✅ Session Start Handler | Accepts goal ID, returns session prompt, configures model backend |

|  |  |
| --- | --- |
| ✅ Goal Plan Tracker | Stores milestones, parsed scope from uploaded files |

|  |  |
| --- | --- |
| ✅ Tutoring Cycle Tracker | Tracks which instructional stages are complete |

|  |  |
| --- | --- |
| ✅ Progress Summary Engine | Summarizes progress based on quiz/test/reflection |

|  |  |
| --- | --- |
| ✅ Content Parser & Plan Builder | Generates structured scope/milestones from uploaded PDFs or text |

|  |  |
| --- | --- |
| 🧪 Chat Summarizer | Optional MVP component — async background summarization |

|  |  |
| --- | --- |
| 🚀 TutorChat UI | UI component that interacts with backend and manages session memory buffer |

## 📌 Design Philosophy & Summary

 Tutor is **instructional authority**, initialized with pedagogical rules and behavior templates.

 Session prompts are lightweight, smartly constructed, and **model-aware**.

 Progress tracking is layered: *plan*, *cycle*, *performance*, *understanding*.

 User data is not lost between sessions but **condensed intelligently**.

 Architecture must remain **model-agnostic and modular**.