📄 FSD_08 – Real-time Agent Response Logic

🧠 Purpose:

This module defines how the live Audira agent generates and returns responses to end-user queries using:

- Injected prompts (from FSD_06)
- Retrieved memory (from FSD_07)
- Embedded business logic (tag maps, policies, instructions)
- Agent role configuration (from onboarding framework)

It acts as the **final decision engine** at runtime, orchestrating how the LLM behaves, how responses are filtered or guarded, and how multi-turn sessions are handled safely.

🧩 FSD_08 – Section Breakdown

Section	**Description**
1. Scope	What the real-time logic controls and excludes
2. Input Requirements	What is required to generate a live response
3. Runtime Flow & Decision Logic	The step-by-step inference pipeline
4. Guardrails & Safety Filters	Logic for blocking, editing, or rejecting responses
5. Multi-turn Conversation Handling	How session memory and context persistence work
6. Output Formatting	What the final user-facing response includes
7. Integration Modes	Web, API, embedded assistant, mobile
8. Future Enhancements	Voice mode, zero-shot memory patching, fine-tuning control

---

🔷 Section 1: Scope

🎯 Purpose:

This module governs the **final runtime behavior** of an Audira agent — from the moment a user asks a question to the moment a response is delivered.

It controls:

- **What information is retrieved**
- **How prompts are assembled**
- **Which LLM is invoked**
- **How the response is filtered, formatted, and returned**
- **Whether the response is allowed, blocked, or flagged**

✅ Responsibilities:

Function	Description
Runtime Prompt Assembly	Merges user query + compiled prompt (FSD_06) + agent memory (FSD_07)
Model Selection Logic	Chooses which LLM to use (based on use case, context, token limits)
Inference Trigger	Executes the call to the LLM
Guardrail Enforcement	Applies filters for safety, hallucination, sensitivity, or confidence drop
Session Context Linking	Retains state across multi-turn conversations
Live Response Formatting	Adds metadata, sources, disclaimers, or visual enhancements before display

---

❌ Not in Scope:

Excluded	Reason
Prompt generation	Handled by FSD_06 – Prompt Chain Compiler
Memory population or update	Managed by FSD_07 – Agent Memory
Document parsing	Handled by FSD_01 to FSD_03
Launch readiness or qualification scoring	Already validated in FSD_05

📌 **Key Characteristics:**

- **Stateless API Compatible**: Works in single-turn mode for integrations
- **Session-Persistent Capable**: Supports multi-turn, memory-linked logic for web or embedded agents
- **Model-Agnostic**: Supports routing to GPT, Claude, Mixtral, or open models via abstraction layer

---

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines orchestration layers and switching logic
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – determines agent persona, tone, and goal
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – ensures only compliant, verified agents go live
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – defines deployment targets and interface modes

Great — let's continue with **Section 2: Input Requirements** for FSD_08 – **Real-time Agent Response Logic**.

🔷 Section 2: Input Requirements

This section defines the **runtime inputs** needed to generate a safe, context-rich, and accurate agent response.

📦 A. User Query Input

The raw message or question from the end user.

```
{
  "query_text": "How does our pricing model work for enterprise clients?",
  "session_id": "sess_45813",
  "user_id": "USER_908",
  "language": "English"
}
```

📦 B. Agent Prompt Chain

Source: FSD_06
Structured prompt segments that establish context, role, and tone.

```
{
  "prompt_chain": [
   {
    "block_type": "business_summary",
    "content": "The company provides SaaS tools to SMEs with monthly Stripe billing."
   },
   ...
  ]
}
```

📦 C. Agent Memory Snapshot

Source: FSD_07
Knowledge graph of validated facts and preferences available for reference or injection.

```
{
 "pricing_model": {
  "value_text": "Three-tier: Basic, Pro, Enterprise",
  "confidence": 0.94
 },
 "target_customer": {
  "value_text": "SMEs and mid-sized enterprise teams",
  "confidence": 0.89
 }
}
```

📦 D. Agent Configuration Profile

Source: Agent Onboarding Config
Defines intent, personality, tone, LLM selection hints, guardrail levels.

```
{
 "agent_id": "AGENT_101",
 "capabilities": ["Q&A", "Sales Pitch", "Compliance Filter"],
 "preferred_model": "gpt-4",
 "response_tone": "consultative",
 "guardrails": {
  "hallucination_block": true,
  "sensitive_tags": ["compliance", "finance"]
 }
}
```

📦 E. Session Memory

For multi-turn logic — includes conversation history, slot-filling, clarification state.

```
{
 "previous_turns": [
  {"user": "What are our support hours?", "agent": "24/7 via Zendesk"},
  {"user": "Can vendors request manual payouts?", "agent": "Let me check that for you..."}
 ]
}
```

🧩 Summary Input Table:

Input	Description	Required
query_text	User message	✅
prompt_chain	Context blocks	✅
agent_memory	Retrieved knowledge base	✅
agent_profile	Configuration + guardrails	✅
session_memory	Multi-turn state	✅

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – prompt + memory composition path
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – source of tone, intent, persona
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – provides input context to memory recall

- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – used to define capabilities and model hints

---

🔷 Section 3: Runtime Flow & Decision Logic

This section defines the **step-by-step logic pipeline** that powers each real-time agent response — from receiving a user query to returning a filtered, context-aware answer.

🔁 Real-Time Processing Pipeline

🧭 Step 1: Query Intake

- Capture user message
- Sanitize input (e.g., remove excessive punctuation, normalize casing)
- Detect language (if not explicitly passed)

🔎 Step 2: Intent Mapping

- Match query to relevant tags, categories, or agent skills using:
  - LLM classification (zero-shot or few-shot intent mapping)
  - Keyword → tag dictionary matching
  - Memory relevance scoring

```
{
  "matched_tags": ["pricing_model", "enterprise_support"],
  "agent_skill": "Q&A"
}
```

🧠 Step 3: Memory Retrieval

- Pull relevant memory blocks from FSD_07
- Inject confidence scores and timestamps
- Skip memory if confidence < 0.75 unless fallback is enabled

🧱 Step 4: Prompt Construction

- Combine:
    - User query
    - Prompt blocks from FSD_06
    - Memory content from FSD_07
- Inject into LLM template based on use case and agent tone

Example (Q&A-style):

User asked: “How does our pricing model work for enterprise clients?”

Here’s what we know:

- Pricing Model: Three-tier (Basic, Pro, Enterprise)

- Payment method: Stripe, monthly recurring

Based on this, respond with clear, verified information only.

🧪 Step 5: Model Selection

- Choose LLM based on:
    - Agent config preference
    - Token limits from prompt
    - Risk category (e.g., use GPT-4 for financial/compliance logic)
    - Runtime fallback availability

🧠 Step 6: Inference Execution

- Send constructed prompt to selected LLM endpoint
- Record raw output, token usage, latency, model ID

🛡️ Step 7: Guardrail Evaluation

- Run response through:
    - Hallucination detector
    - Sensitive content filter

    - Confidence fallback logic (e.g., “I’m not sure” if weak info)
    - Traceability checker (was the memory used?)

🗣️ Step 8: Response Finalization

- Format with UI metadata:
    - Inline source references (optional)
    - Disclaimers (if compliance required)
    - Language tag + response tone markers

🔁 Pipeline Modes

Mode	Description
Stateless	Used for single-turn calls (API or button-click interface)
Session-Persistent	Includes prior turns, memory deltas, or slot-filling
Simulated Mode	Used by admin testers to simulate agent behavior from memory/prompt versions

---

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines prompt formatting, model routing, and input handoff
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – defines agent tone, language, fallback preferences
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – maps LLM infrastructure (e.g., Claude, GPT, Mixtral)

🔷 Section 4: Guardrails & Safety Filters

This section defines how the system **detects, prevents, or rewrites** unsafe, incorrect, or untrusted responses before final delivery to the user.

🛡️ Guardrail Types

Guardrail	Purpose	Default Behavior
Hallucination Filter	Blocks responses not grounded in memory, prompt, or documents	Replace with fallback message
Sensitive Tag Filter	Prevents unverified responses to regulated or high-risk tags (e.g., finance, HR, legal)	Respond with “Unable to confirm that yet.”
Token Overflow Detector	Detects if prompt + answer exceeded LLM’s safe token range	Retry with truncated prompt
Low Confidence Block	Flags any answer generated from weak tag/memory	Insert soft disclaimer
Disallowed Phrases Filter	Blocks use of terms like “guarantee,” “legally binding,” or unconfirmed claims	Redact or rewrite

---

✅ Enforcement Modes

Mode	Description	Trigger Example
Hard Block	Don’t return the answer at all	LLM responds with hallucinated financial data
Rewrite Mode	Regenerate response with stricter template or fallback phrasing	LLM says “we guarantee returns”
Soft Disclaimer Injection	Keep response, but append warning or citation gaps	“This reflects the latest available info as of June 2025.”
Ask-for-Clarification	Prompt user for more input to refine answer safely	“Can you confirm which country this applies to?”

🧪 Guardrail Evaluation Engine

Every LLM response is passed through a validation pipeline before being shown:

```
{
  "original_response": "We offer tax-free returns to all customers.",
  "flags": ["sensitive_tag:tax_policy", "confidence:low"],
  "action_taken": "blocked_and_rewritten",
  "final_response": "Tax policy may vary by location. Please consult your local advisor."
}
```

🔍 Additional Guardrail Tools (Pluggable)

Tool	Usage
LLMScore	Evaluates factuality and tone
DeepEval	Detects hallucination probability
PromptLayer	Version-tracks prompt and response pairs
TruLens	Measures alignment with known memory blocks

🧰 Customizable Parameters

Parameter	Default	Description
hallucination_block	true	Enforced for all compliance and financial agents
min_confidence_required	0.75	Prevents weak memory use
sensitive_tags	["compliance", "HR", "pricing_strategy"]	Hardcoded or dynamic from tag dictionary
rewrite_on_disallowed_phrases	true	Redacts banned expressions like

		“guaranteed” or “unlimited access”

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines model → output → validator interface
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – defines sensitive and regulated tags
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – sets minimum quality rules for production agents
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – configures tone, trust level, and fallback preferences

---

🔷 Section 5: Multi-turn Conversation Handling

This section defines how the agent manages **context, continuity, and memory** across a live, multi-message user session — enabling human-like dialogue and backtracking.

🧠 Core Multi-turn Capabilities

Capability	Description
Session Persistence	Maintains agent state between turns using a session_id
Context Recall	Automatically references recent answers or memory blocks
Slot-Filling Logic	Captures missing info across multiple messages
Clarification Looping	Allows user to confirm or correct previous agent assumptions
Follow-up Understanding	Detects implicit references like “what about support hours?” after pricing was discussed

---

🧾 Session Memory Object

```
{
  "session_id": "sess_45813",
  "agent_id": "AGENT_101",
  "user_id": "USER_908",
  "turns": [
   {"user": "What do we charge?", "agent": "You offer three pricing tiers: Basic, Pro, and
Enterprise."},
   {"user": "And support?", "agent": "Support is 24/7 via Zendesk for all plans."}
  ],
  "active_slots": {
   "pricing_model": "answered",
   "support_model": "answered",
   "payout_policy": "pending"
  },
  "inferred_intents": ["pricing", "support"]
}
```

🔁 Conversation Loop Handling

User Input	Agent Behavior
“And shipping?”	Detects context → refers to pricing/distribution tag
“That’s not accurate”	Triggers rollback → revisits last tag and asks for clarification
“Update our payout policy”	Opens editable memory block (if allowed) or routes to admin if restricted
“Summarize what we’ve said so far”	Returns a combined summary of tagged responses and agent answers

---

⏳ Memory Duration & Expiry

Type	Lifespan
session_memory	Active until logout or 24h timeout
agent_memory	Persistent and versioned (from FSD_07)
live_delta_memory	Temporary patches stored only for session replay, not persisted unless confirmed

---

🧰 Conflict & Correction Handling

If a new message contradicts earlier info (e.g., “No, we don’t have 24/7 support”), the system:

1. Compares to memory value
2. Flags the mismatch
3. Suggests either:
    - Temporary override in-session
    - Update request (if user is an admin)
    - Re-routing to qualification follow-up (FSD_04)

📎 **References**:

- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – defines slot types, session timeout rules
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – determines how context is injected into each turn
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – links segment-to-turn references when user follows up
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – configures conversation memory depth and personality

---

🔷 Section 6: Output Formatting

This section defines how each response from the agent is **structured, enhanced, and delivered** to ensure it meets Audira's standards for clarity, trust, and usability.

📤 Final Output Object Structure

```
{
  "response_text": "Your pricing model includes Basic, Pro, and Enterprise tiers. Enterprise clients receive custom billing terms.",
  "source_tags": ["pricing_model", "enterprise_terms"],
  "confidence": 0.94,
  "guardrail_flags": [],
  "disclaimer": null,
  "language": "English",
  "tone": "consultative",
  "response_type": "direct_answer",
  "session_id": "sess_45813",
  "timestamp": "2025-06-11T03:18:00Z"
}
```

🧩 Output Components

Field	Description
response_text	Final message shown to user
source_tags	List of tags used to generate the answer
confidence	Weighted score based on memory + LLM output
guardrail_flags	Any filters triggered (e.g., hallucination_blocked)
disclaimer	Optional legal or clarity note

language	Based on input or agent config
tone	Maps to onboarding config (e.g., friendly, expert)
response_type	Used for downstream behavior (direct_answer, clarification, summary, etc.)
session_id	Traces response to a live thread
timestamp	Audit and replay tracking

💬 Tone & Style Modifiers

The agent automatically adjusts output phrasing based on the assigned tone profile from onboarding:

Tone	Behavior
consultative	Adds clarifying phrases, offers options
expert	Confident, technical phrasing
friendly	More casual, “we” language, uses emojis if UI allows
neutral	Minimalist, factual delivery

🧪 Output Enhancements

Feature	Description	Configurable?
Inline Source Highlighting	Annotate phrases linked to specific document segments or memory blocks	✅
Confidence Meter	Visual indicator or icon (e.g., green dot = verified)	✅
Expandable Disclaimers	Click-to-reveal notes for legal or compliance topics	✅
Voice Tag	Used in TTS/voice deployment mode	✅

📎 Display Integration Targets

Channel	Integration Mode
Audira Web Agent	HTML bubble with Markdown and tooltips
Partner Embed	iframe or JS-rendered component
Mobile SDK	JSON-to-UI adapter (light/dark mode supported)
API	Raw JSON, with optional Markdown format preview

📎 **References**:

- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – defines tone, disclaimer needs, and user persona
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – determines how output is shaped based on memory and tags
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – defines formatting compatibility per integration layer
- 📘 *AUDIRA PRODUCT BLUEPRINT* – sets clarity, traceability, and trust design principles

---

🔷 Section 7: Integration Modes

This section outlines how the real-time agent logic is **embedded and consumed** across different deployment channels — ensuring consistent behavior whether used via Audira's UI, partner apps, or external APIs.

🌐 Supported Integration Types

Mode	Description	Use Case
Web Agent (Audira Native)	Default frontend experience with prebuilt chat UI, Markdown formatting, and source tooltips	SMB onboarding, web support

Partner iframe Embed	Lightweight widget loaded into any partner website or dashboard via iframe	CRM, LMS, or white-labeled portals
API Access (Headless)	Pure JSON API with prompt-in → response-out model	Custom frontends, advanced developers
Mobile SDK	Agent response logic wrapped in mobile components	Audira app, mobile partner platforms
Admin Sim Mode	Response engine run from back office for QA, debugging, and simulations	Internal audit, testing, tuning

---

🧩 Web Agent: (UI-first, Audira-hosted)

- Receives formatted response_text
- Supports:
  - Confidence indicators
  - Segment-linked highlights
  - Session-based memory view
  - Emoji/tone rendering based on agent style
- Auto-adapts to dark/light mode

🧰 iframe / JS Embed

<script src="https://agent.audira.ai/embed.js" agent-id="AGENT_123" config="consultative" />

- Fully sandboxed
- Loads real-time agent logic + UI handler
- Receives token-limited prompt chain on startup
- Supports event hooks (onResponse, onEscalate, onClarify)

🔌 API (Stateless & Stateful Modes)

Endpoint	Description
POST /generate_response	Stateless prompt + memory injection
POST /generate_response/session	Maintains turn-by-turn context with session ID
POST /agent_response/simulate	Runs simulated inputs for admin preview
GET /agent/{id}/capabilities	Returns agent prompt/memory/runtime config

---

📱 Mobile SDK (v1.0)

- Wraps real-time agent logic with:
    - Chat interface components
    - Touch-optimized Markdown renderer
    - Light/dark theme sync
- Optional offline fallback for prompt previews (no inference)

🧪 Admin Simulation Mode

- Uses stored prompt + memory + user input to:
    - Reconstruct agent output
    - Log all inference calls
    - Inject edge-case QA (e.g., “Try saying: I need legal advice”)

📎 **References**:

- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – defines endpoints, iframe logic, SDK parameters
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – determines which modes are enabled per agent
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – confirms prompt + memory assembly pre-inference

🔷 Section 8: Future Enhancements

This section outlines forward-looking upgrades to make the agent smarter, safer, faster, and more human-like in real-time interactions — aligned with Audira's long-term product roadmap.

🚀 Planned Features

Feature	Description	Value
Zero-Shot Memory Patching	Inject real-time answers from uploaded documents even if memory hasn't been updated yet	Enables instant intelligence from new data
LLM Strategy Optimizer	Dynamically selects best LLM (e.g., GPT vs. Mixtral vs. Claude) based on query type, tag, tone, or latency	Reduces cost and boosts performance
Tone Style Cloner	Matches response tone to user writing style (formal, playful, concise)	Deep personalization
Real-Time Feedback Buttons	Let users rate or correct answers, triggering micro-updates or flags	Crowdsourced QA and retraining
Voice Interaction Layer	Real-time speech-to-text input and TTS output via WebRTC or mobile mic	Powers kiosk, app, and accessibility agents
Chain-of-Thought Reasoning Prompts	Adds intermediate reasoning steps before final answer for complex logic	Improves explainability and trust
Agent Self-Explanation Mode	Users can ask: "Why did you say that?" → agent returns memory + prompt trace	Enhances transparency
Industry-Aware Answer Templates	Pre-trained response skeletons per sector (e.g., healthcare, logistics, SaaS)	Faster, more relevant replies
Fine-Tune Injection Protocol	Allows snapshot training of mini LLMs on agent memory + segments (for open model agents only)	Ultra-fast responses on edge devices

---

🧰 Compatible OSS Tools

Tool	Purpose
LangGraph	Multi-path agent routing based on memory, fallback, and LLM profile
TruLens / DeepEval	Output validation and QA loop
Unstructured.io + LlamaIndex	Segment-to-answer routing for longform responses
OpenRouter	Unified interface for multi-model LLM orchestration
PromptLayer / PromptTools	Version control and replay analysis of prompt + output pairs

---

📎 **Linked Modules**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines how prompt blocks are merged with user input
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – future tone and user-style cloning features will extend from here
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – voice, mobile, and WebRTC interfaces
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to evaluate output behavior before deploying advanced logic

---