ModelMix Deliberation Control & Consensus Engine

Goal

Enable multi-model / multi-persona task execution with **bounded deliberation**:

- Task-driven (not free chat)
- Short back-and-forth responses
- Minimum **6 rounds** (configurable)
- Explicit **Start / Pause / Resume / Stop** controls
- Deterministic path to **consensus**

This document defines the **control plane**, **state machine**, **API**, **prompt contracts**, and **UI wiring** required to implement this cleanly in a web app.

---

1. Core Mental Model

A deliberation is **not a conversation**. It is a **managed execution loop**.

- The user gives a **task**
- Models discuss for **X fixed rounds**
- Responses are **intentionally short**
- The system controls time, flow, and memory
- Consensus is synthesized at the end (or early if converged)

The browser never orchestrates logic — it only sends **commands**.

---

2. Deliberation State Machine

States

```
idle → running → paused → running → completed
                ↘ stopped
```

State Rules

- `running`: rounds advance automatically
- `paused`: no model calls occur
- `stopped`: deliberation terminates immediately (no synthesis)

- `completed`: synthesis + checkpoint written

---

3. Canonical Deliberation Object (Server-Side)

```
type DeliberationStatus =
  | "idle"
  | "running"
  | "paused"
  | "completed"
  | "stopped"

type Deliberation = {
  id: string
  sessionId: string
  task: string
  agents: AgentConfig[]
  maxRounds: number            // enforce >= 6
  currentRound: number
  status: DeliberationStatus
  rounds: Round[]
  consensus?: ConsensusResult
  createdAt: number
}

type AgentConfig = {
  id: string
  model: string
  systemPrompt: string
  maxResponseTokens: number    // e.g. 80–120
}

type Round = {
  roundNumber: number
  responses: AgentResponse[]
}

type AgentResponse = {
  agentId: string
  content: string
  tokenCount?: number
}

type ConsensusResult = {
  summary: string
  confidence: "low" | "medium" | "high"
  dissentingAgents?: string[]
}
```

Important: This object is the *single source of truth*. The UI reflects it; it never mutates it directly.

---

4. Control API (Minimal but Sufficient)

Create deliberation

```
POST /deliberations
```

```
{
  "sessionId": "...",
  "task": "Evaluate architectural options for X",
  "agents": ["planner", "critic", "implementer"],
  "maxRounds": 6
}
```

Control execution

```
POST /deliberations/:id/start
POST /deliberations/:id/pause
POST /deliberations/:id/resume
POST /deliberations/:id/stop
```

Read state (poll or stream)

```
GET /deliberations/:id
```

---

5. Execution Loop (Server-Side)

High-level loop

```
while (status === "running") {
  if (currentRound >= maxRounds) break

  executeRound()
  if (consensusDetectedEarly()) break

  currentRound++
}

if (status === "running") {
  runConsensusSynthesis()
  status = "completed"
}
```

Pause / Stop Behavior

- **Pause**: loop halts after current model call
- **Resume**: loop continues at same round
- **Stop**: loop terminates immediately, no synthesis

---

6. Round Execution Rules (Critical)

Round Contract

Each agent receives:

- Task
- Prior compressed memory (if any)
- **Only the previous round**, not full history
- Explicit round number

Short-Response Enforcement

Hard constraints:

- Max tokens (80–120)
- No lists > 3 bullets
- No repetition of task

---

7. Agent Prompt Template

```
You are participating in a multi-agent deliberation.

Task:
{{TASK}}

Round {{ROUND}} of {{MAX_ROUNDS}}.
```

```
Previous round summary:
{{PREVIOUS_ROUND_SUMMARY}}

Rules:
- Respond in ≤ {{MAX_TOKENS}} tokens
- Do not repeat the task
- Focus only on advancing consensus
- If you agree with another agent, say so briefly
- If you disagree, state why in one sentence

Your response:
```

This prompt is **non-negotiable** — it prevents verbosity and drift.

---

8. Early Consensus Detection (Optional but Recommended)

After each round, run a lightweight check:

```
function consensusDetectedEarly(round: Round): boolean {
  return similarityScore(round.responses) > THRESHOLD
}
```

If triggered:

- Skip remaining rounds
- Mark `confidence = high`
- Proceed to synthesis

---

9. Consensus Synthesis Step

Executed **once**, at the end.

Synthesis Prompt

```
You are a consensus synthesizer.

Task:
{{TASK}}

Agent conclusions:
{{ALL_AGENT_FINAL_POSITIONS}}
```

```
Produce:
1. Final consensus recommendation (short paragraph)
2. Confidence level (low / medium / high)
3. Any remaining dissent (if applicable)

No prose. No hedging.
```

Output is stored as `ConsensusResult` and written to **compressed session memory**.

---

10. Web UI Wiring

Chatroom-Style Deliberation View (New)

ModelMix must support a **chatroom-style log** for deliberation mode that visually resembles a real-time group chat, while preserving strict orchestration rules.

Core Requirements

- Each agent message appears as a chat bubble
- Messages are ordered by round, then agent
- Persona **display name** is shown (short label), not model name
- Model identity (provider / model ID) is hidden from the UI but preserved in metadata
- Deliberation chat is **read-only** to the user once started

Visual Rules

- Header: `Deliberation Mode · Round X / Y`
- Agent label = persona title (e.g., `Planner`, `Critic`)
- No emojis, no role words like `assistant`, `analyst`, `creative`
- Optional color-coding per persona

After Completion

- Final consensus is rendered as a system message
- A divider indicates: `— Returned to ModelMix Session —`
- Only the consensus output is injected into the main MM chat

Required Controls

- ▶ Start
- ⏸ Pause
- ▶ Resume
- ⏹ Stop

UI Behavior

- Disable Start after running
- Disable Pause when idle
- Show round counter: `Round 3 / 6`
- Stream responses per round (SSE / WS)

Optional Enhancements

- Timeline view (rounds as nodes)
- Agent-color coding
- “Jump to round” for review (read-only)

---

11. Context & Memory Interaction

Deliberation → Main Session Handoff

- Full deliberation transcript is stored separately (audit / review)
- **Only the final consensus output** is appended to the main ModelMix session
- The main chat does **not** replay agent chatter

This preserves normal MM flow while allowing deep side deliberation.

11a. Message Visibility & Privacy Rules (Critical)

Message Types

```
type MessageVisibility = "broadcast" | "private"
```

Broadcast Messages

- Visible to all agents
- Used for round-based deliberation
- Default for agent-to-agent discussion

Private Messages (One-Way)

Requirement: Sending to an individual model must be truly private.

Rules:

- Private messages are delivered to **only the target agent**
- They are **not echoed** to other agents
- They are not summarized into round history

- They may influence the target agent’s next response only

This fixes the current flaw where "private" messages are effectively broadcast.

11b. Explicit Exclusion Handling

- Agents excluded from a deliberation **must not** receive:

- Task updates

- Private messages

- Round summaries

- Orchestrator enforces recipient filtering at dispatch time

---

12. Model Identity & Persona Enforcement (Trae Fix Requirements)

Problem Statement

Current issues:

- Models self-identify as `analyst` / `creative`
- Transcripts show full provider model names
- `@mentions` expose long identifiers and clutter UI

This breaks persona-driven deliberation.

Required Fixes

1. Persona-First Identity (Hard Requirement)

- Every agent has a **persona ID** (short, human-readable)
- Persona ID is the **only visible identifier** in transcripts
- Model name is metadata only

```
type AgentConfig = {
  personaId: string      // e.g. "Planner"
  modelId: string        // e.g. "qwen2.5-32b"
  provider: string       // e.g. "lmstudio"
}
```

LLMs must be instructed:

Never state your role as analyst/assistant/creative. Always speak as your persona.

---

2. Remove @Mentions for Routing

- UI-level @mentions are **not** used for routing
- Routing is explicit via:

```
{ "toAgent": "Planner" }
```

- UI displays only persona label, never full routing string

---

3. Transcript Rendering Rules

- Display name = personaId only
- No model names
- No provider labels
- No role prefixes

Example:

```
[Planner]: We should prioritize latency over cost.
[Critic]: That assumption fails at scale.
```

---

13. Chatroom + Main Session Integration Summary

- Deliberation runs in a **side-channel chatroom**
- User can observe rounds in real time
- User cannot interject mid-round (unless paused)
- Consensus result is returned to the main ModelMix session as a single message
- Main session continues as normal

This makes deliberation a **mode**, not a forked experience.

---

14. Why This Matters

Without these fixes:

- Personas collapse into generic roles
- Privacy semantics are broken
- UI becomes unreadable

- Deliberation loses trust

With them:

- ModelMix feels intentional
- Agents feel distinct
- Consensus is defensible
- The chat log is human-readable and reviewable

This is not cosmetic. It is architectural.

- Full deliberation transcript → stored for audit
- Only **final consensus** → injected into compressed memory
- Round chatter is **never re-prompted** after completion

This keeps ModelMix clean, cheap, and deterministic.

---

12. Why 6 Rounds Works

- Round 1–2: framing & assumptions
- Round 3–4: disagreement & refinement
- Round 5–6: convergence & alignment

Less than 6 → premature consensus More than \~10 → diminishing returns

---

Implementation Status Checklist

-

Bottom Line

This turns ModelMix from a chat system into a **controlled reasoning engine**.

You get:

- Predictable depth
- Token discipline
- Human-readable debate
- Trustworthy consensus

This is the right abstraction. Don't dilute it.