Local AI odMel Architecture (Tier-1 / Tier-2) Complete Report

This is a **descriptive, end-to-end explanation** of the Tier-1 / Tier-2 local-first AI architecture.

---

1. Executive Summary

Objective

Replace a cloud-heavy, agent-driven AI backend with a **local-first architecture** where:

- Core intelligence runs **on the device**
- Cloud is used **only for deterministic tools**
- The mobile app (Flutter) controls orchestration

Core Principle

> **Local = Brain, Cloud = Hands**

- The *brain* (decision-making) must be local for speed, privacy, and reliability
- The *hands* (APIs, payments, search, bookings) can live in the cloud

System Components

- **Tier-1 (always-on, local):** routing & control model
- **Tier-2 (on-demand, local):** deep reasoning model
- **Cloud:** tool execution only (strict JSON)
- **Flutter app:** orchestrator and state manager

---

2. Tier-1: What It Is and Why It Exists

Tier-1 is the **most important component** in the system.

It is **not** a chatbot and **not** a reasoning model.

It is a **deterministic decision engine** whose only job is to understand *what should happen next*.

Tier-1 Responsibilities (Explained)

Tier-1 takes raw user text and produces a **structured control decision**:

- **Intent classification** – what the user wants at a high level
- **Journey / domain selection** – which agent(s) are relevant
- **Tool selection** – whether an external API is needed
- **Argument extraction** – pull structured parameters from text
- **Missing-field detection** – identify what is still required
- **Clarification generation** – ask exactly one focused question if needed
- **Formatting guidance** – how the final answer should look
- **Complexity scoring** – how hard this task is
- **Routing confidence** – how sure the model is about its own decision

What Tier-1 Explicitly Does NOT Do

- It does **not** call APIs
- It does **not** generate long answers
- It does **not** perform multi-step reasoning
- It does **not** hallucinate content

Think of Tier-1 as a **traffic controller**, not a thinker.

---

3. Why `complexity_score` and `routing_confidence` Matter

These two numbers make the entire system stable and scalable.

`complexity_score` (0–100)

This tells Flutter **how much intelligence is required**.

Range	Meaning	Action
0–20	Trivial	Tier-1 only
21–40	Simple tool	Call cloud tool
41–60	Explanation	Tier-2 optional
61–80	Heavy reasoning	Tier-2 required
81–100	Planning / multi-domain	Tier-2 + orchestration

This avoids loading Tier-2 unnecessarily.

`routing_confidence` **(0–1)**

This expresses **how confident Tier-1 is in its own output**.

- High confidence → proceed automatically
- Medium confidence → proceed but be cautious
- Low confidence → clarify or escalate

Both values are **learned from data**, not computed with rules.

---

4. Canonical Tier-1 Output Schema (System Contract)

Every Tier-1 inference **must** return this schema.

```
{
  "intent": "<intent>",
  "primary_journey": "<domain>",
  "journeys": ["<domain>", "<domain2>"],

  "needs_tools": true,
  "tool": "<tool_id_or_null>",
  "arguments": { "key": "value_or_null" },

  "missing_fields": [],
  "needs_clarification": false,
  "clarification": null,

  "complexity_score": 0,
  "routing_confidence": 0.0,

  "formatting_style": "<style>",
  "response_template": "<template_or_null>"
}
```

This schema is the **hard boundary** between model logic and application logic.

---

5. Tier-1 Dataset Design (How the Model Learns)

Dataset Format (Important)

Use **ChatML (``**),** not Alpaca.

```
{
  "messages": [
    {"role":"system","content":"You are a Tier-1 router. Output JSON only."},
    {"role":"user","content":"Book a flight"},
    {"role":"assistant","content":"{ ...STRICT JSON... }"}
  ]
}
```

Why ChatML

- Matches Qwen-2.5 pretraining
- Stronger role separation
- Much better JSON stability
- Safer after quantization and mobile deployment

Scenarios You Must Include

You do **not** need multiple schemas. You need **many situations**:

- Greeting / casual text
- Simple fact (no tool)
- Tool call with all parameters
- Tool call with missing parameters
- Multi-domain request
- Follow-up / continuation
- High-complexity planning

Same schema, different values.

---

6. Tier-2: Reasoning and Generation Layer

Tier-2 is the **intelligence depth layer** of the system. It exists to improve *quality*, *depth*, and *human-likeness* of responses **after Tier-1 has already decided what should happen**.

A useful mental model:

> **Tier-1 decides *what* to do. Tier-2 decides *how well* to do it.**

Tier-2 is intentionally separated so that expensive reasoning is **only used when truly required**.

6.1 Why Tier-2 Is Needed

Not all user queries need deep reasoning.

Examples that **do NOT** need Tier-2:

- "What is the capital of France?"
- "Track my package"
- "Book a flight" (initial step)

Examples that **DO** need Tier-2:

- "Explain quantum computing in simple terms"
- "Plan a 10-day Europe trip with a budget"
- "Compare these two approaches and recommend one"

Without Tier-2, you would either:

- Always run a large model (slow, expensive, battery-heavy), or
- Deliver shallow, low-quality answers

Tier-2 solves this by being **on-demand**.

6.2 What Tier-2 Is Used For (In Practice)

Tier-2 handles **generation-heavy and reasoning-heavy tasks**, such as:

- Long explanations and tutorials
- Step-by-step planning
- Multi-paragraph summaries
- Teaching and tutoring
- Interpreting tool results into natural language

Tier-2 is allowed to:

- Think step-by-step internally
- Generate long text
- Use richer language
- Optimize for helpfulness instead of speed

6.3 What Tier-2 Explicitly Never Does

This boundary is critical.

Tier-2 must **never**:

- Perform intent classification
- Select tools
- Decide which agent/domain to use
- Extract structured arguments

- Ask clarification questions about missing fields

Those responsibilities belong **exclusively to Tier-1**.

If Tier-2 were allowed to do these, routing would become unstable and non-deterministic.

6.4 How Tier-2 Is Triggered

Tier-2 is invoked **only when Tier-1 signals it**.

Common triggers:

- `complexity_score > threshold` (e.g., >60)
- Explicit planning requests
- Multi-domain or open-ended questions

Tier-1 communicates this via structured output, not text.

Flutter then:

1. Loads Tier-2 if not already resident
2. Passes the user request (and optionally context)
3. Receives a natural-language response

6.5 Tier-2 Dataset Design Philosophy

Tier-2 datasets are **fundamentally different** from Tier-1 datasets.

Aspect	Tier-1	Tier-2
Output	Strict JSON	Natural language
Goal	Control	Quality
Determinism	Very high	Moderate
Reasoning	Minimal	Deep

Tier-2 training data should look like **good tutoring and planning examples**, not routing logic.

6.6 Tier-2 Dataset Example

```
{
  "messages": [
    {"role":"system","content":"You are a helpful reasoning assistant."},
    {"role":"user","content":"Explain quantum computing simply."},
    {"role":"assistant","content":"Quantum computing uses quantum bits, or qubits, which can represent both 0 and 1 at the same time. This allows quantum computers to solve certain problems much faster than classical computers..."}
  ]
}
```

Key properties:

- No routing fields
- No tool selection
- No JSON constraints
- Clear, structured explanations

6.7 Operational Characteristics

- Tier-2 is **downloadable**
- Loaded only when needed
- Unloaded on memory pressure
- Optional for low-end devices

This keeps the system fast, battery-efficient, and scalable.

6.8 Summary of Tier-2 Role

> **Tier-2 is not required for correctness. It is required for excellence.**

The system remains functional without Tier-2, but Tier-2 is what makes responses feel intelligent, thoughtful, and premium.

---

7. Model Selection (Practical)

Tier	Model	Reason
Tier-1	Qwen-2.5-1.5B	Best structured output & routing
Tier-2	Qwen-2.5-3B / 7B	Strong reasoning quality

Qwen models are:

- Instruction-aligned
- Quantization-friendly

- Excellent for mobile + ExecuTorch

---

8. Fine-Tuning Reality Check

Tier-1

- Method: LoRA / QLoRA
- Samples: \~2,000 high-quality examples
- Epochs: 2–3
- Time: \~30–90 minutes on a modern GPU

  Training is easy. **Designing the dataset is the real work.**

---

9. Flutter Orchestration (How Everything Runs)

```
Future<Response> handle(String text) async {
  final r = await tier1.run(text);

  if (r.needsClarification) {
    return askUser(r.clarification);
  }

  if (r.complexityScore > THRESHOLD) {
    await tier2.ensureLoaded();
    return tier2.run(text);
  }

  if (r.needsTools) {
    final json = await cloud.run(r.tool, r.arguments);
    return format(json, r);
  }

  return formatLocal(r);
}
```

Flutter is the **brainstem** coordinating models, tools, and UI.

---

10. Architecture & Flow Diagrams

High-Level Control Flow

```
User Input
   │
   ▼
Tier-1 Router (local)
   │
   ├─ clarify?
   ├─ needs tool?
   ├─ complexity high?
   │
   ├─▶ Cloud Tools (JSON)
   ├─▶ Tier-2 (local)
   ▼
Formatter → UI
```

Tier-1 Internal Decision Flow

```
Text
 │
 ▼
Intent + Journey
 │
 ├─ Missing fields → Clarify
 ├─ Tool needed → Tool JSON
 ├─ High complexity → Tier-2
 ▼
Local formatted response
```
