Multilingual Farm Assistant — End-to-End Pipeline Design (MVP → V1)

> Goal: Low-latency, low-cost, multilingual assistant for Indian farmers. Keep processing **offline/by local models** when quality ≥ cloud APIs and latency is acceptable; use APIs only where necessary (external data) or when offline models underperform.

---

0) High-Level Architecture

Client: Android app (WebRTC audio streaming)\ **Edge/API**: ExpressJS server (Node), Tooling microservices\ **Inference**: Local model host (Whisper, XLM-R, IndicTrans2, Mistral/Gemma)\ **Data**: External APIs (Weather, Mandi, Schemes), internal DBs (disease/soil knowledge), feature store, logs/metrics

```
Android (WebRTC) → Express (ingress) →
  1) ASR (Whisper.cpp) → transcript chunks
  2) NLU (XLM-R: intent + keywords)
  3) Normalize (IndicTrans2: intent+keywords → en)
  4) Router (rules + light ML) → select tools
  5) Tools (Weather, Mandi, Disease DB, Schemes, Soil/Fertilizer)
  6) Synthesizer (Mistral/Gemma offline; API LLM fallback) → response (target language)
  7) (Optional) Back-translation if synthesizer outputs in English
```

---

1) Models & Decisions (Best Candidates)

Stage	Task	Default (Offline/Local)	Rationale	Fallback/Alt
1	**ASR**	**Whisper-small (quantized via whisper.cpp)**	Strong Indic ASR, runs on CPU/GPU, real-time capable	Larger Whisper-medium if accuracy needed; cloud ASR if strict latency
2	**Intent Classification**	**XLM-RoBERTa-base** (HF)	Multilingual, fine-tunable, ms-level latency	Distil-XLM-R for speed; API LLM only if label space changes rapidly

Stage	Task	Default (Offline/Local)	Rationale	Fallback/Alt
3	**Keyword Extraction**	**XLM-R (token-labeling head)**	Phrase-sensitive, stays in source language	KeyBERT on XLM-R embeddings as backup
4	**Normalization (→ English)**	**IndicTrans2-small**	Optimized for Indian langs, lighter than NLLB	NLLB distilled; API MT if latency too high
5	**Tool Selection / Routing**	**Rule-based + light classifier (XGBoost)** on features	Transparent, fast, robust	LLM router as last resort
6	**Response Generation**	**Mistral-7B-Instruct (Q4) or Gemma-7B-Instruct (Q4)**	Strong local instruct models; templating for simple answers	API LLM for complex/ unsafe/edge cases
7	**Back-translation**	**IndicTrans2-small** (en → target lang)	Keep stack symmetric	API MT if quality is critical

Rule: Prefer offline if quality ≥ API and latency acceptable; otherwise escalate per fallback tree.

2) External Tools / Data Sources (Top 5 for MVP)

1. **Weather**: Open-Meteo (no key) as default; WeatherAPI/Visual Crossing as secondary.
2. **Mandi Prices**: AGMARKNET (Data.gov.in) APIs; optionally e-NAM dashboard scraping/partner API.
3. **Plant Disease/Symptom/Cure**: Local knowledge base built from PlantVillage/PlantDoc + ICAR advisories (curated table).
4. **Govt Schemes**: myScheme search; curated list for PM-KISAN, PMFBY, PMKSY, PM-Kusum.
5. **Soil/Fertilizer**: Soil Health Card datasets + ICAR crop-soil-fertilizer mappings (rule base).

All tool adapters return **strict JSON**; no free-text inside adapters.

3) Data Contracts (Schemas)

3.1 Streaming ASR (server events)

```
// Client → Server (chunk)
{
  "session_id": "uuid",
  "seq": 17,
  "audio_pcm16": "base64",
  "lang_hint": "hi-IN"
}
```

```
// Server → Client (partial transcript)
{
  "session_id": "uuid",
  "seq": 17,
  "partial_text": "दिल्ली का मौसम…",
  "t0_ms": 5320,
  "t1_ms": 7010,
  "confidence": 0.86
}
```

3.2 NLU Output

```
{
  "lang": "hi",
  "text": "दिल्ली का मौसम कैसा है?",
  "intent": {
    "label": "weather.query",
    "confidence": 0.92
  },
  "keywords": [
    {"span": "दिल्ली", "type": "location"},
    {"span": "मौसम", "type": "topic"}
  ]
}
```

3.3 Normalized Intent (to English)

```
{
  "lang_src": "hi",
  "lang_tgt": "en",
  "intent_en": "weather.query",
  "keywords_en": [
    {"text": "Delhi", "type": "location"},
    {"text": "weather", "type": "topic"}
  ],
  "original": {"text": "दिल्ली का मौसम कैसा है?"}
}
```

3.4 Tool Router Request

```
{
  "intent_en": "weather.query",
  "features": {
    "has_location": true,
    "has_date": false,
    "urgency": "normal"
  },
  "keywords_en": ["Delhi", "weather"],
  "context": {"geo_hint": {"lat": 28.61, "lon": 77.21}}
}
```

3.5 Tool Adapter Response (example: Weather)

```
{
  "tool": "weather.open_meteo",
  "ok": true,
  "data": {
    "location": {"name": "Delhi", "lat": 28.61, "lon": 77.21},
    "forecast": [
      {"date": "2025-08-17", "tmin_c": 28.1, "tmax_c": 36.5, "rain_mm": 4.2},
      {"date": "2025-08-18", "tmin_c": 27.9, "tmax_c": 35.8, "rain_mm": 8.0}
    ]
  }
}
```

3.6 Synthesis Prompt (LLM input)

```
{
  "user_lang": "hi",
  "original_text": "दिल्ली का मौसम कैसा है?",
  "intent_en": "weather.query",
  "tool_payloads": [ { /* weather JSON */ } ],
  "policy": {
    "style": "concise",
    "safety": "agri_advice_v1"
  }
}
```

4) Routing & Fallback Logic

1. **ASR**: If Whisper confidence < 0.7, request repeat or re-record; else continue.
2. **Intent**: If XLM-R confidence < 0.6 → pass transcript to small LLM (Mistral/Gemma) for intent suggestion; else accept.
3. **Normalization**: If IndicTrans2 latency > 200 ms or output is empty → skip normalization and route using source language features.
4. **Tool selection**: Rules first; if no rule match, use classifier. If still no tool → fallback to LLM planner.
5. **Synthesis**: Use local LLM. If safety/policy or uncertainty flags trigger → escalate to API LLM.

Uncertainty signal (to trigger fallbacks): low confidence, conflicting tools, missing required fields (e.g., no location for weather), or safety-sensitive queries.

---

5) Latency Budget (Target, per request)

Stage	Target	Notes
ASR (streaming)	< 300 ms per chunk	Emit partials; finalization < 1 s
Intent+Keywords	30–60 ms	Batchable per turn
Normalization	100–200 ms	Short sequences
Tool Calls	300–800 ms	Depends on remote API
Synthesis (local LLM)	800–1500 ms	7B Q4, 10–20 tok/s
Total P50	**1.8–3.0 s**	End-to-end, after last audio chunk

Continuous UX: start showing intent/tool badges as soon as they are resolved; stream the final answer.

---

6) Services & Deployment Topology

- **Express Gateway**: WebRTC ingress, auth, rate-limit, request assembly.
- **ASR Worker**: whisper.cpp service w/ queue (e.g., BullMQ).
- **NLU Service**: XLM-R model server (Torch/ONNX, CPU/GPU).
- **MT Service**: IndicTrans2 (TorchServe).
- **Router**: Rule engine + XGBoost microservice.
- **Tool Adapters**: Weather, Mandi, Schemes, Disease, Soil; each a stateless microservice.
- **LLM Synth**: Mistral/Gemma server (llama.cpp/text-generation-inference).
- **Observability**: Prometheus + Grafana; structured logs; tracing (OpenTelemetry).
- **Storage**: Redis (features/session), Postgres (logs/metrics), MinIO/S3 (artifacts).
- **Feature Store**: Simple KV (Redis) for per-user preferences/language.

---

7) Knowledge Bases (Local)

- **Disease KB**: table: crop → symptom → probable cause → remedy (dosage, wait period).
- **Soil/Fertilizer KB**: region/soil_type → recommended NPK → crop-specific schedule.
- **Schemes KB**: scheme → eligibility (state, landholding, document list) → application steps → links.

  Start with curated CSVs; expose via read-only REST adapters.

---

8) Safety, Guardrails & Tone

- **Safety policy**: agri_advice_v1 — prohibits dangerous pesticide dosages; always add *consult local agri officer* disclaimer for chemical use.
- **Grounding**: LLM strictly conditions on tool JSON; avoid hallucinated numbers.
- **Citation**: Include tool provenance in response metadata (tool name + timestamp).
- **Tone**: Local language, respectful, concise; add "what else can I help with?" prompt.

---

9) Evaluation Plan

- **ASR**: WER on Hindi/Marathi/Telugu test clips.
- **Intent/Keywords**: F1 / support by intent; exact-match keyword span F1.
- **MT**: BLEU/COMET on short intents/keywords.
- **Synthesis**: Human eval: helpfulness, factuality (uses tool), tone.
- **Latency**: P50/P95 per stage; SLO alerts.

---

10) MVP Endpoint Contracts

- `POST /asr/stream` (WebRTC signaling)
- `POST /nlu` → returns intent+keywords
- `POST /normalize` → returns en intent+keywords
- `POST /route` → returns selected tools
- `POST /tools/:name` → strict JSON
- `POST /synthesize` → streamed text (SSE/WebSocket)
- `GET /healthz` / `GET /metrics`

---

11) Example End-to-End (Hindi → Weather)

1. Audio → transcript: "दिल्ली में अगले हफ्ते बारिश होगी?"
2. Intent: `weather.query` (0.93), Keywords: [दिल्ली(location), अगले हफ्ते(time)]
3. Normalize: `Delhi`, `next week`
4. Router: picks **weather.open_meteo**

5. Tool: forecast JSON (7-day rain)
6. Synthesis: local LLM produces Hindi answer; cites tool+date
7. Return: stream to client + JSON metadata.

---

12) Roadmap (Post-MVP)

- Add **pest/insect** detection & advisory; satellite NDVI (ISRO/NASA POWER).
- Personalization: user's **location, crops, sowing window** memory.
- On-device light NLU (future): intent hints before server round-trip.
- Active learning loop: annotate low-confidence cases to improve XLM-R.

---

13) Ops & Maintenance

- Version models by semantic versioning (`nlu-xlmr:1.2.0`).
- Canary deploys for new models; shadow eval before promote.
- Rotating API keys; per-tool timeouts; circuit breakers.
- Nightly data refresh for mandi/schemes; cache weather by lat/lon/day.

TL;DR

Keep **speech + NLU + routing** offline and fast; use **external tools** for facts; synthesize with a **local 7B instruct** and escalate to cloud only when quality demands it. This maximizes user privacy, reduces cost, and keeps latency low while covering the broad query space.