Tech Stack

1. High-level design mapping

- **Main Hub (MH)** — regional/cluster brain (higher compute, persistent storage, orchestrator).
- **Mini Hub (mH)** — edge gateway (local AP, cache, lightweight inference).
- **End user device** — smartphone (browser/app) connecting to mH AP.
- **Transport layers**: Local access (Wi-Fi / BLE) and Long-range low-power link (LoRa or long-range Wi-Fi).
- **Cloud (optional)** — for model updates, advanced inference, analytics when internet available.

2. Hardware (recommended components + alternatives)

Main Hub (1 per cluster)

- **Primary compute**:
    - Option A (cost/energy balanced): Single-board computer with quad-core+ (e.g., latest Raspberry Pi 4 / 5, 8GB+ RAM).
    - Option B (higher performance): Small form-factor PC (Intel NUC / AMD mini PC) or edge GPU device if local large-model inference is required.
- **Storage**: 256 GB–1 TB NVMe SSD (for model files, capsule DB, logs).
- **Long-range radio**: LoRa transceiver module (SX126x family or equivalent) with external high-gain antenna.
- **Network**: Ethernet, optional SIM/LTE modem (for opportunistic cloud sync), USB Wi-Fi adapter (for directional links).
- **Power**: PoE or stable DC supply + small UPS for graceful shutdown.
- **Optional accelerators**: USB Coral TPU, Edge TPUs, or small GPU if running heavier quantized models.

Mini Hub (per site)

- **Primary compute**:

    - Option A (low-cost): Raspberry Pi Zero 2 W or Raspberry Pi 4 (4GB) depending on local model needs.
    - Option B (more compute for local inference): Pi 4 / 8GB, or small x86 SBC.
- **Wi-Fi/BLE**: Onboard (for AP), optional external CPE Wi-Fi antenna for extended coverage.
- **LoRa**: SX126x / Ebyte LoRa module (with antenna).
- **Storage**: 32–128 GB microSD or USB storage for caches & capsule DB.
- **Power**: PoE (preferred) or solar + battery with controller for remote deployments.
- **Enclosure**: Weatherproof IP65 for outdoor units.
- **Optional**: Microphone + speaker for voice interface.

End-user (phones)

- Any smartphone supporting Wi-Fi; web UI or lightweight app.

Antennas & Networking

- **LoRa**: High-gain omnidirectional for MH; directional or omnidirectional for mHs depending on placement.
- **Wi-Fi CPE**: Outdoor CPEs for 1–2 km coverage where needed; directional bridges for cluster interconnects.

Estimated BOM (per-node rough)

- Mini Hub: SBC + LoRa module + CPE antenna + enclosure + power = **$80–$250** (depends on Pi model and antenna).
- Main Hub: SBC/NUC + LoRa gateway + SSD + UPS = **$300–$1200**.
  (Procurement prices vary by region—adjust accordingly.)

3. Connectivity & Network Layer

Local access channel (user ↔ mini hub)

- Wi-Fi AP hosted by mini hub (802.11n/ac if hardware supports) or BLE for ultra-low bandwidth discovery.
- Local HTTP(S) endpoint (browser UI) for queries.

Long-range channel (mini hub ↔ main hub & peer mH)

- Primary: **LoRa** for low-power long-range text/metadata and compact capsules.
    - Use short packets, chunking & reassembly, ACKs and small sliding windows.

    - Keep routine payloads small (capsule IDs, hashes, compressed embeddings).
- Secondary (opportunistic): Directional Wi-Fi bridges or temporary LTE for bulk syncs (model updates, large payloads).
- Inter-mH mesh: Peer gossip via LoRa or local Wi-Fi links for capsule exchange and redundancy.

Network protocols & patterns

- **Store-and-forward**: For queued queries and delayed synchronization.
- **Gossip protocol**: Controlled probabilistic push/pull to disseminate capsules among mHs.
- **Manifest-based sync**: Small manifest messages listing capsule IDs/versions; selective fetch reduces bandwidth.

4. Software Stack (components & responsibilities)

Note: below lists components and responsibilities — you can map to specific frameworks/libraries based on preference.

System services (on both MH & mH)

- **Service manager** (systemd or container orchestrator) to run processes.
- **Local API server** (lightweight web service) exposes endpoints:
    - /query — receive user queries,
    - /status — health/telemetry,
    - /admin — local management.
- **Packet/forwarder service** — converts HTTP queries to packets for long-range transport; manages chunking, retries, encryption.
- **Cache manager** — stores capsules and handles ANN index lookups and garbage collection.
- **Local inference agent** — runs compact LLM or retrieval-based response generator for fallback.
- **Sync manager** — handles manifest exchange, capsule requests, and reconciliation.
- **Security/key manager** — holds node keypair, handles session key derivation.
- **Telemetry agent** — batched metrics/logs for when uplink available.

Datastore model (abstract)

- **Capsule DB** (local): stores capsule metadata, compressed embedding, answer text, source, signature, timestamp.
- **Index**: approximate nearest neighbor (ANN) index for embeddings (compact, memory-frugal).
- **Queue**: pending outgoing queries / chunks awaiting ACK.

Dev/Operations

- **OTA / update manager** — secure incremental updates (signed updates).
- **Monitoring** — lightweight metrics (heartbeats, cache hit/miss, storage usage).
- **Logging** — local log rotation; batch export when uplink available.

5. Models, embeddings & inference (recommendations)

Model strategy (tiered)

- **Main Hub**: larger models for authoritative responses / capsule generation (quantized models for local CPU inference or cloud if available).
- **Mini Hub**: compact quantized models for fallback answers; rely primarily on capsule retrieval for accuracy.

Model formats & optimization

- **Quantized model formats** (Q4/Q8, GGUF, or ONNX quantized) to reduce footprint.
- **Inference runtimes**: efficient CPU runtimes (ggml/llama.cpp, ONNX Runtime with quantization) or hardware accelerators when available.

Embeddings & retrieval

- **Embeddings**: low-dimension, quantized/compressed vectors stored with each capsule.
- **ANN index**: small memory HNSW-like index (nmslib, hnswlib, or embedded approximate index) tailored to edge memory limits.
- **Similarity threshold policy**: configurable threshold to determine cache hit vs forward.

6. APIs & External Interfaces

Internal APIs (between services)

- **mH ↔ MH transport API** (packetized, encrypted): carries query packets, response packets, capsule manifests, and control messages (ACK, SYNC_REQ, SYNC_ACK).

- **Capsule manifest API**: list of capsule IDs + versions for selective fetch.

Optional cloud APIs (used only when internet available)

- **Model hosting / ML inference APIs**: for heavy or specialized queries (cloud fallback).
- **Telemetry / analytics**: aggregated logs/metrics for centralized analysis.
- **OTA update API**: secure update distribution.

End-user interface

- **Web UI**: simple form for questions + display answers and confidence.
- **Optional mobile app**: for offline caching, voice-recording (upload when connected), and richer UX.

7. Security & Key Management

Core primitives

- **Node identities**: asymmetric keypairs per node (Ed25519 recommended) — used for capsule signing and verifying provenance.
- **Session keys**: ephemeral key exchange (X25519) to derive per-transport symmetric keys.
- **Payload encryption**: authenticated encryption (AES-GCM or an authenticated AEAD) for query/response payloads.
- **At-rest encryption**: encrypted local storage for sensitive data; encryption keys managed by local key manager and protected by hardware where possible.

Trust model

- **Bootstrap**: MH acts as cluster root/trust anchor during provisioning; MH signs cluster certificates.
- **Revocation**: maintain revocation lists for compromised nodes (distributed and periodically synced).
- **Verification**: mH verifies capsule signatures and rejects unauthenticated capsules.

Privacy controls

- **PII redaction**: optional pre-capsule filtering to remove sensitive identifiers.
- **Retention policies**: configurable TTL for capsules and query logs to comply with privacy/regulatory constraints.
- **Access control**: local admin controls for who can query or access logs.

8. Data formats & packet design (abstract schemas)

Query packet (conceptual fields)

- pkt_id, src_id, dst_id, timestamp, payload_enc, meta (q_hash, emb_hint, priority), chunk_index (if chunked), total_chunks.

Capsule (conceptual)

- capsule_id, question_hash, answer_text, compressed_embedding, source_id, timestamp, signature.

Design choices:

- Keep capsules compact — compress answer text, quantize embeddings.
- Use deterministic question hashing for dedupe & conflict resolution.

9. Storage, indexing & caching policies

Storage

- Local DB (lightweight, ACID): used for capsule storage plus metadata.
- Periodic compaction and backup to attached SSD on MH.

Indexing & cache

- ANN index kept in memory for fast lookups; persistent snapshot on disk.
- Cache policies:
    - **LRU + heuristics** (favor high-frequency, high-value capsules).
    - TTL and priority tags (sensitive / high-value / stale).
- **Cache decision**:
    - Compute embedding of incoming query → query ANN → if sim >= Tsim and age <= Tmax then local hit; else forward.

10. Operations, provisioning & deployment

Provisioning workflow

- Pre-provision node identities and initial config for each device (node id, frequency, cluster id, MH public key).
- Securely ship devices, power up and verify trust handshake with MH.

Monitoring & maintenance

- Lightweight telemetry (heartbeats, storage percentage, cache hit/miss, battery status).
- Batch telemetry upload when uplink available; alerting for critical events.

OTA & updates

- Signed update packages; atomic update mechanism and rollback capability.

11. Observability & evaluation tools

- Local metrics: cache-hit ratio, avg response latency (local vs upstream), capsule propagation time, storage usage, error/retry rates.
- On MH: aggregate metrics, trend analysis, and deployment analytics.
- Test harness: simulated packet loss, delayed sync scenarios, stress tests (many simultaneous queries), and battery/power loss simulations.

12. Redundancy, resilience & failure modes

- **Local resilience**: mH continues serving from cache and light LLM when MH unreachable.
- **Redundancy**: multiple mHs per cluster; peer exchange reduces single-point-of-failure.
- **Recovery**: queued queries reconciled and authoritative capsules reissued on reconnection.
- **Power resilience**: solar + battery sizing, graceful shutdown, low-power sleep schedules.

13. Optional / advanced components (for later phases)

- Hardware accelerators on MH (edge GPUs) for higher-quality local models.
- Federated learning loop for capsule weighting or model finetuning (privacy-aware).
- Voice interface: offline speech-to-text and TTS on mH.
- Incentive mechanisms for cross-organization capsule sharing (reputation tokens).

14. Implementation priorities & tradeoffs

1. **Start minimal & robust**: prioritize capsule format, encryption, packet reliability, and a simple retrieval-based cache on edge.

2. **Add local inference later**: once capsule retrieval works reliably, introduce small quantized LLM on mH.
3. **Optimize bandwidth**: compress & quantize embeddings, send manifests, selective fetch to minimize long-range use.
4. **Balance quality vs footprint**: higher-quality models require more compute; rely on MH for authoritative resolution initially.

15. Quick recommended mapping to concrete (if you want specifics later)

I kept the above technology-agnostic where useful. If you want, I can map each component to specific software libraries, runtimes, and exact hardware SKUs (e.g., which SBC, which LoRa module, exact model formats and versions) and provide:

- A Bill-of-Materials with cost estimates, and
- A minimal reproducible Stage-1 software scaffold (local simulator + packet & crypto modules).