Chatbi: Vision & Roadmap

# 1. Vision (North Star)

A production-grade, multilingual hotel assistant that:  
- answers FAQs with citations,  
- searches live room inventory/prices,  
- remembers context (city, dates, budget),  
- routes to the right skill (rooms, policies, payments, support),  
- is observable, testable, and safe.

# 2. System Architecture

API layer: FastAPI  
Orchestrator: LangGraph (graph state machine)  
LLM layer: local (Ollama/vLLM) or cloud; small router → large generator  
Retrieval: vector DB + hybrid keyword search  
Business data: Postgres (hotels, rates, policies)  
Cache & state: Redis (short-term memory, rate limiting)  
Monitoring & eval: tracing + RAG eval + offline tests

# 3. Key Libraries

## Orchestration & API

* - LangGraph – dialogue graph
* - FastAPI – serving endpoints
* - Pydantic – typed state and schemas

## LLMs

* - Transformers + vLLM / Ollama – local or hosted LLMs
* - Small router model for intent classification
* - Larger model for generation

## Retrieval (RAG)

* - Langchain text splitters
* - Sentence-transformers (bge, gte, e5)
* - Qdrant / FAISS / pgvector – vector DBs
* - Reranking with bge-reranker
* - Hybrid search (BM25 + vector)

## Data & Tools

* - Postgres + SQLAlchemy/SQLModel
* - Redis – memory and caching
* - Pandas / Polars – data transforms

## Safety & Guardrails

* - Microsoft Presidio – PII scrub
* - NeMo Guardrails or Llama Guard – policy checks
* - Pydantic validation – tool constraints
* - Tenacity – retries and timeouts

## Observability & Evaluation

* - Langfuse / LangSmith – tracing and eval
* - OpenTelemetry – distributed tracing
* - Prometheus + Grafana – metrics
* - RAGAS, DeepEval – evaluation
* - Giskard – regression and bias tests

## Dev & CI/CD

* - pre-commit (ruff, black, mypy)
* - Docker Compose
* - GitHub Actions CI

# 4. Data Model

Postgres tables:  
- hotels(hotel\_id, name, city, country, stars, lat, lon, amenities\_json)  
- room\_rates(hotel\_id, room\_type, occupancy, currency, base\_rate, refundable, breakfast\_included)  
- policies(hotel\_id, key, value, updated\_at)  
  
Vector docs (JSONL):  
- faqs: {id, hotel\_id, city, lang, question, answer, tags, chunks, embedding}  
- also policies and amenities for richer answers

# 5. Dialogue Graph (LangGraph nodes)

1. Router – classify intent and extract slots  
2. Rooms – SQL query on Postgres  
3. FAQ – RAG pipeline with embeddings + citations  
4. Booking – stub, validate and create provisional hold  
5. Safety – input/output checks and fallback  
6. Memory – Redis short-term, optional long-term in Postgres

# 6. Roadmap

## Phase 1 – Foundations (Week 1–2)

1. Environment setup: Python venv, install deps, Docker Compose skeleton (App + Postgres + Qdrant).
2. Data prep: Convert CSV → JSON/JSONL, load hotels/rates into Postgres, index FAQs/policies into Qdrant.
3. MVP Graph: LangGraph with router, rooms, faq, fallback; simple intent classification.

## Phase 2 – MVP Bot (Week 3–4)

1. Rooms skill: SQL filters (city, budget, occupancy), return top-k.
2. FAQ skill (RAG v1): Embed FAQs, query Qdrant, return top docs.
3. FastAPI /chat: run graph, return JSON reply.
4. Basic tests: pytest golden paths (rooms, faq, unknown, error).

## Phase 3 – Production Pilot (Month 2)

1. Memory: Redis for slots (city, budget), continue context.
2. Safety & guardrails: Presidio scrub, Pydantic validation, fallback node.
3. Observability: Langfuse tracing, Prometheus + Grafana metrics.
4. RAG improvements: stronger embeddings, reranker, hybrid search.

## Phase 4 – Full V1 Launch (Month 3–4)

1. Booking skill: mock API for reservations.
2. Multilingual support: language detection, multilingual embeddings.
3. vLLM / Ollama integration: stronger LLM for generation.
4. CI/CD pipeline: GitHub Actions for lint, tests, RAG eval.

## Phase 5 – Scale & Features (Beyond)

1. Real booking/payment APIs integration.
2. Preference learning and reranking from feedback.
3. Fine-tuned router model on logs.
4. Continuous evaluation with synthetic and real conversations.
5. Expand domains: flights, packages, restaurants.