1. Problem Description

General-purpose AI models often struggle with the strict referencing required in religious contexts. They may invent verses, misattribute Hadiths, or fail to distinguish between authentic (*Sahih*) and weak (*Da'if*) narrations. This solution mitigates these risks by decoupling knowledge retrieval (Database/Search) from answer synthesis (LLM).

2. System Architecture

The system follows a modular microservices approach suitable for rapid hackathon development:

1. **User Interface:** A lightweight frontend to accept natural language queries.
2. **Orchestration Layer:** A middleware that processes text, manages state, and routes queries between the database and the web.
3. **Knowledge Base (Vector Database):** A specialized SQL database storing texts and their high-dimensional vector representations for semantic similarity search.
4. **External Search Module:** An API integration to fetch real-time information for questions not covered by the static corpus (e.g., modern ethical issues).
5. **Generative Engine:** A constrained LLM environment that synthesizes the final answer using *only* retrieved contexts.

3. Database Schema Design

The database is designed to be **normalized** to ensure data integrity, with separate concerns for metadata, textual content, and vector embeddings.

3.1. Quranic Corpus Tables

These tables are based on the general schema, it would be changed as per the data provided from external apis.

These tables store the structure and text of the Holy Quran.

- **Surahs (Metadata)**
    - Stores metadata about the 114 chapters.
    - *Schema:* (id, surah_number, name_arabic, name_english, revelation_place, total_verses)
- **Quran_Verses (Content)**

- Stores the actual text in Arabic and English translations.
- *Schema:* (id, surah_id_FK, verse_number, text_uthmani, text_translation_eng)

- **Quran_Vectors (Embeddings)**
  - Stores the vector representations of the English translation for semantic search.
  - *Schema:* (id, verse_id_FK, embedding_vector)

3.2. Hadith Corpus Tables

These tables handle the complexity of various books, narrators, and authenticity grades.

- **Hadith_Collections (Metadata)**
  - Stores information about the source books (e.g., Sahih Bukhari, Sahih Muslim).
  - *Schema:* (id, collection_name, author_name, total_hadiths)
- **Hadith_Chapters (Structure)**
  - Organizes Hadiths into thematic chapters (e.g., Book of Prayer, Book of Zakat).
  - *Schema:* (id, collection_id_FK, chapter_title_eng, chapter_title_ara)
- **Hadith_Narrations (Content)**
  - Stores the narration text and its specific metadata.
  - *Schema:* (id, chapter_id_FK, hadith_number, text_arabic, text_english, grade, narrator_chain)
- **Hadith_Vectors (Embeddings)**
  - Stores the vector representations of the Hadith text.
  - *Schema:* (id, narration_id_FK, embedding_vector)

4. Functional Workflow

The application logic follows a strict step-by-step flow to ensure validity:

1. **Query Analysis:** The user's input is cleaned and converted into a vector embedding.
2. **Primary Retrieval (Database):**
   - The system queries Quran_Vectors and Hadith_Vectors simultaneously.
   - It retrieves the top 'N' matches based on cosine similarity.

    - *Validation:* If the similarity score is below a defined threshold (e.g., 75%), the system marks the database search as "insufficient."

3. **Secondary Retrieval (Web Fallback):**
    - If the database yields low confidence, the system triggers the Search API (Serper) to find verified articles from a whitelist of domains.
4. **Context Assembly:** The relevant database rows (Verse/Hadith text) or web snippets are compiled into a text block.
5. **Response Generation:** The LLM receives the user question and the assembled context with a strict instruction set to cite the specific Surah:Verse or Book:Hadith Number provided in the context.

5. Technology Stack

- **Backend Framework:** FastAPI (Python) – Chosen for high-performance asynchronous processing.
- **Database:** Supabase (PostgreSQL) – chosen for its native pgvector extension support.
- **Orchestration:** LangChain – For managing the prompt templates and retrieval chains.
- **AI Model:** Google Gemini 1.5 Flash – Selected for its large context window and cost-efficiency (Free Tier/Low Cost).
- **Web Search:** Serper Dev – A Google Search wrapper optimized for AI data ingestion.
- **Frontend:** Streamlit – For rapid prototyping of the chat interface.

6. Limitations & Future Scope

- **Current Limitation:** The system relies on English translations for semantic search. Nuances in the original Arabic might be missed during the retrieval phase.
- **Future Scope:**
    - Implementation of multilingual embeddings.
    - Integration of *Tafseer* (Exegesis) tables to provide deeper explanations for Quranic verses.
    - Voice-to-Text input for accessibility.

7. External API

- Hadiths https://github.com/fawazahmed0/hadith-api?tab=readme-ov-file

- Hadiths https://hadithapi.com/docs/chapters
- Quran The Noble Quran - Quran.com
-

Supabase Vector: Supabase Docs