A Multi-Agent Architecture for Grounded Islamic QA

Anonymous ACL submission

Abstract

Large language models (LLMs) can answer religious knowledge queries fluently, yet they often hallucinate and misattribute sources, which is especially consequential in Islamic settings where users expect grounding in canonical texts (Qur'an and Hadith) and jurisprudential (fiqh) nuance. Retrieval-augmented generation (RAG) reduces some of these limitations by grounding generation in external evidence. However, a single "retrieve-then-generate" pipeline is limited to deal with the diversity of Islamic queries. Users may request verbatim scripture, fatwa-style guidance with citations or rule-constrained computations such as zakat and inheritance that require strict arithmetic and legal invariants. We present a bilingual (Arabic/English) *multi-agent system* that routes queries to specialized modules within an agentic, tool-using architecture. The system supports intent-aware routing, retrieval-grounded fiqh answers with deterministic citation normalization and verification traces, exact verse lookup with quotation validation, and deterministic calculators for Sunni zakat and inheritance with madhhab-sensitive branching. We evaluate the complete end-to-end system on public Islamic QA benchmarks and demonstrate effectiveness and efficiency. Our system is currently publicly and freely accessible through API and a Web application, and has been accessed ≈1.9M times in less than a year.[1]

1 Introduction

Recent advances in large language models have enabled conversational assistants that can handle knowledge-intensive question answering (QA) across many domains. Despite these gains, hallucination and source-attribution errors remain common, particularly when users expect answers grounded in authoritative references rather than plausible-sounding narrative. In religious applications, these failures carry higher stakes. Fabricating a Quranic verse, misattributing a Hadith, or presenting a jurisprudential position subject to scholarly disagreement without stating relevant conditions can mislead users. This motivates Islamic QA systems that not only answer correctly, but also provide clear grounding, stable citations, and explicit handling of cases where the system should abstain or surface scholarly disagreement.

The community has begun formalizing these reliability requirements through benchmarks and shared tasks. QuranQA (Malhas et al., 2023) has established standardized evaluation for Quranic passage retrieval and reading comprehension. IslamicEval (Mubarak et al., 2025) further emphasizes grounded Quran/Hadith QA and includes tasks for detecting and correcting Quranic hallucinations, reflecting real user risks when quotations are fabricated or corrupted. For structured religious reasoning, QIAS focuses on Islamic inheritance (aka faraid in Islamic Jurisprudence), a domain where correct answers require rule-based computation and legal constraints (Bouchekif et al., 2025a).

Islamic QA systems should *(i)* ground responses in canonical sources, *(ii)* provide transparent citations, and *(iii)* properly handle domain-specific reasoning tasks that exceed the reliability of free-form generation. RAG combines parametric generation with non-parametric retrieval to ground responses in external documents (Lewis et al., 2020); dense retrievers such as DPR (Karpukhin et al., 2020) and generative readers such as Fusion-in-Decoder (FiD) (Izacard and Grave, 2021) (as well as retrieval-in-the-loop models such as RETRO (Borgeaud et al., 2022) and Atlas (Izacard et al., 2023)) show retrieval can improve factual QA and knowledge updating. Nevertheless, a fixed retrieve-then-generate pipeline is often not a good match for real-world Islamic queries; some are best satisfied by *exact lookup*, e.g., "What does verse 2:255 say?", others require *rule-constrained computation*, e.g., zakat, inheritance, and others require

[1] anonymous.com

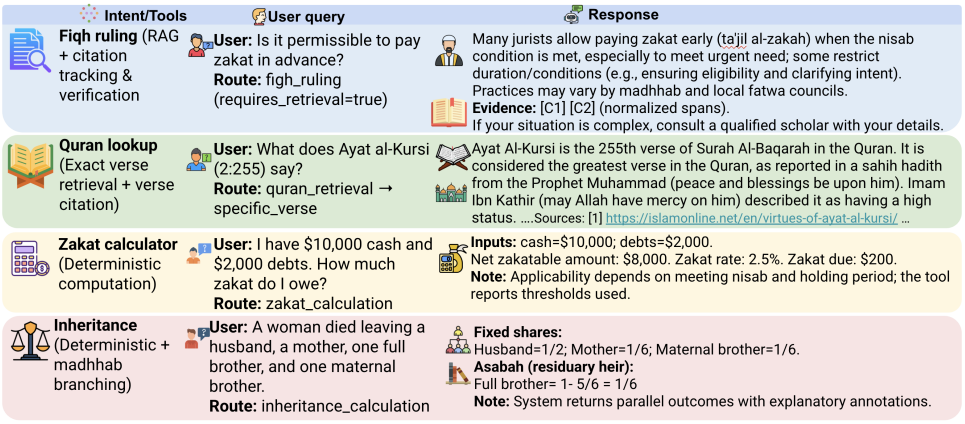

Figure 1: Illustrative end-to-end examples showing intent routing to specialized tools, traceable citations for fiqh QA, exact Quranic verse handling, deterministic zakat computation, and explicit madhhab-sensitive branching for disputed inheritance cases. Citation tags [C*] denote normalized evidence spans: [Q*] denotes verse-level citations.

jurisprudential reasoning with evidence presentation, e.g., fatwa-style questions with stated assumptions, conditions, and madhhab sensitivity. Thus, treating heterogeneous intents uniformly can degrade correctness and user experience. Tool-using approaches suggest a path beyond rigid pipelines. ReAct (Yao et al., 2023) interleaves reasoning with actions for iterative retrieval/verification, while Toolformer (Schick et al., 2023) shows models can learn when to call tools, e.g., calculators, motivating an *architecture* that *selects among multiple execution modes* based on query intent rather than forcing every question through the same pipeline.

In this paper, we present a bilingual ***multi-agent*** Islamic QA system built around an agentic, multi-tool architecture (see Figure 2). It is explicitly designed for the heterogeneity emphasized by contemporary Islamic QA benchmarks. At a high level, the system *(a)* classifies incoming queries into fine-grained Islamic intent types (few examples are provided in Figure 1), *(b)* routes each query to an appropriate specialized module, and *(c)* enforces transparency and reliability via citation tracking and post-generation verification.

The contributions of this work include:

- A *multi-agent architecture* for Islamic QA that goes beyond fixed RAG by routing queries to specialized tools and integrating evidence tracking and verification.
- A comprehensive evaluation spanning multiple public benchmarks covering both generative and multiple-choice Islamic QA.
- Our findings show that tool- and evidence-routed execution improves faithfulness, vital for Islamic QA, while remaining competitive on broader Islamic knowledge benchmarks.

2 System Architecture

In Figure 2, we present our *multi-agent end-to-end architecture*. The system is designed for heterogeneous Islamic QA, spanning rule-heavy obligations best handled with symbolic computation and canonical text retrieval where verbatim accuracy is essential. User queries fall into three broad classes: *(i)* text-grounded questions (Quran/Hadith/fiqh/general Islamic knowledge), *(ii)* rule- and arithmetic-constrained questions (zakat and inheritance), and *(iii)* symbolic time/geo questions (Hijri calendar and prayer times). Treating all of these intents as a single "retrieve-then-generate" task leads to predictable failure modes, including misquoted verses, weak or missing sourcing for jurisprudential claims, and numerically inconsistent zakat or inheritance outputs. To contextualize these design choices, in Table 1, we compare our system with prior agentic and Islamic QA systems, highlighting why Islamic QA benefits from specialized modules for computation and scripture handling.

Most existing Islamic QA systems implement a text-only retrieval–generation workflow, sometimes enhanced with reranking (e.g., AFTINA) or iterative retrieval refinement (e.g., FARSIQA). However, they still treat heterogeneous Islamic queries as a single "retrieve evidence then generate" task (Mohammed et al., 2025; Asl and Bidgoli, 2025). Other systems, such as MufassirQAS, similarly focus on vector-database RAG with transparent citations (Alan et al.). Agentic RAG approaches introduce structured tool calls for iterative evidence seeking and answer revision, improving generative faithfulness. However, they primarily extend retrieval behavior rather than integrating deterministic jurisprudential calculators

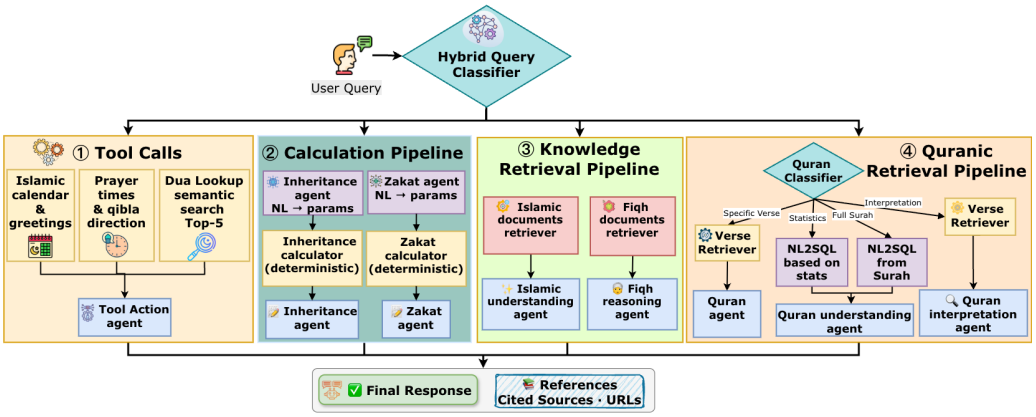

Figure 2: Out ***multi-agent architecture***. A hybrid query classifier selects among *(i)* tool calls, *(ii)* deterministic calculation, *(iii)* document-grounded retrieval QA, and *(iv)* Quranic retrieval routes, before assembling the final response with references.

Work	Calc.	Quran	NL2SQL	Evidence	Tools
Ours (Fig. 2)	✓	✓	✓	✓	✓
Karpas et al. (2022) (MRKL)	✗	✗	✗	✓	✓
Yao et al. (2023) (ReAct)	✗	✗	✗	✓	✓
Schick et al. (2023) (Toolformer)	✗	✗	✗	✓	✓
Asai et al. (2024) (Self-RAG)	✗	✗	✗	✓	✗
Yan et al. (2024) (CRAG)	✗	✗	✗	✓	✗
Al-Azani et al. (2025)	✗	✓	✗	✓	✗
Bhatia et al. (2026)	✗	✓	✗	✓	✓
Omayrah et al. (2025)	✗	✓	✗	✓	✗
AL-Smadi (2025)	✗	✗	✗	✓	✗
Alowaidi (2025)	✗	✗	✗	✓	✗

Table 1: Comparison against widely-recognized tool/agent/RAG architectures. **Calc.** refers to explicit rule engines. **Quran** denotes verse-anchored retrieval distinct from generic document search.

and symbolic geo-temporal tools under a unified router (Bhatia et al., 2026). In contrast, **our multi-agent architecture** routes each query to a heterogeneous tool suite, including deterministic zakat and inheritance calculators (Figure 3 in Appendix), canonical verse lookup, and rule-based calendar and prayer-time computation. We further apply citation normalization and post-generation verification to reduce Quran/Hadith misquotation and support rule-intensive inheritance reasoning. In our multi-agent system, we use Fanar (Team et al., 2025) as the LLM agent. We chose it because it offers free API access[2] and has demonstrated strong performance across diverse benchmarks.

2.1 Hybrid Query Classifier

As shown in Figure 2, we implement a hybrid routing classifier to predict the query type and select the execution route as the system entry point. The primary classifier is an LLM prompted to output an intent label, a confidence score, a short rationale, optional decomposition subquestions, and a retrieval flag indicating whether evidence retrieval is required. We define nine intent classes aligned with the system tools: *(i)* fiqh rulings, *(ii)* Qur'an retrieval, *(iii)* general Islamic knowledge, *(iv)* greetings/chitchat, *(v)* zakat calculation, *(vi)* inheritance calculation, *(vii)* du'a (or supplication) lookup, *(viii)* Islamic calendar, and *(ix)* prayer times (see more details in Appendix (App.) D.1).

Our nine intent classes are motivated by established query-intent and dialogue-act views that separate (a) social acts (e.g., greetings), (b) information-seeking retrieval (e.g., Quran text and dua lookup), and (c) transactional/computational requests (e.g., zakat, inheritance, calendar, and prayer-time queries), where each intent implies a different execution strategy and error profile (Broder, 2002). We further ground the schema in canonical subdomains of Islamic knowledge: jurisprudential queries require fiqh-oriented reasoning (Hallaq, 2009), while zakat and inheritance follow structured rule systems amenable to calculator-style tools (al Qaradawi, 2000). Calendar conversion and prayer-time/Qibla requests are naturally modeled as spatiotemporal computations (Reingold and Dershowitz, 2018), yielding a taxonomy that is both operationally tool-aligned and semantically coherent. We developed a manually annotated dataset of 700 queries to evaluate the hybrid classifier, which achieves 90.1% accuracy. More details on the dataset development process and comparative results are provided in App. D.1.3.

2.2 Tool Calls

Queries requiring utility lookups are routed to the *Tool Action Agent*, which orchestrates deterministic modules. A lightweight **Greeting Tool** handles culturally appropriate greetings and pleasantries (App. D.2). The **Islamic Calendar** tool performs

[2] https://api.fanar.qa/docs

rule-based time reasoning for Hijri date queries, Gregorian–Hijri conversion, and event lookups using multilingual intent cues, `hijridate` conversions, and a curated bilingual event ontology with explicit year-rollover logic, with a controlled fallback that warns about local moon-sighting variance (App. D.3). The **Prayer Times and Qibla** tool resolves locations to coordinates via a curated city database with a rate-limited geocoding fallback, computes prayer timetables using `pyIslam` with method parameters, and computes great-circle distance and bearing to Makkah for Qibla requests while logging trace metadata for interpretability (App. D.4). Finally, the **Dua Lookup** tool provides high-recall, deterministic retrieval by selecting top-k occasions via semantic search over precomputed embeddings, then using a lightweight LLM selector to map the best match to canonical `page_title` keys, returning the supplication verbatim (Arabic, translation, and reference) from a structured store to avoid rewriting (App. D.5).

2.3 Calculation Pipeline (Deterministic)

Rule-heavy financial and legal questions are routed to the *Calculation Pipeline* to eliminate arithmetic drift and enforce jurisprudential constraints.

Zakat Calculator. Zakat is a Shariah-mandated almsgiving governed by established juristic rules (Al-Qaradawi, 1999). Our Zakat agent extracts structured parameters such as asset classes, amounts, and debts, then passes them to a deterministic module. The calculator computes the *nisab* or minimum threshold based on precious-metal prices and applies category-specific logic.[3] For agriculture, differentiated rates are applied based on irrigation methods. For livestock, Hadith-based schedules are used for camels, cattle, and sheep. For assets, rates are applied to cash, gold, business assets, and investments after deducting eligible debts. The output is a structured breakdown of inputs, deductions, and totals, formatted into a user-facing explanation with citations (App. D.6).

Inheritance Calculator. The inheritance calculator (Figure 3) is a deterministic Sunni module that computes estate distribution while explicitly handling madhhab-specific differences. The workflow proceeds in three phases. First, fixed shares (*fard*) are assigned to eligible heirs after validating kinship and removing impeded individuals. Second, the remaining estate is allocated via a priority chain of paternal-line relatives known as residuaries (*'asaba*). Third, the module enforces arithmetic consistency by applying *'awl* (proportional reduction) if shares exceed the estate, or *radd* (return of remainder) if a surplus exists. Crucially, jurisprudentially disputed cases trigger a policy selector that returns parallel distributions, such as Hanafi versus Jumhur (majority opinion), rather than a single collapsed ruling (App. D.7).

[3] `https://sunnah.com/bukhari:1483`

2.4 Knowledge Retrieval Pipeline

Informational queries are routed to retrieval-augmented QA workflows that instantiate *usul al-fiqh* reasoning patterns through evidence linkage.

Fiqh Rulings. This module uses a *Fiqh Documents Retriever* followed by a reasoning agent. The agent is prompted to state ruling scope and assumptions, separate rulings from evidence, and assign deterministic citation tags to evidence spans. The system supports retrieval-time source normalization to ensure every claim maps to a stable source text. If a ruling relies on exact scriptural wording, the agent can invoke the Quranic tool to prevent paraphrase drift (App. D.8).

General Islamic Understanding. For general inquiries, the system retrieves candidate documents and normalizes them into a bounded context. An *Islamic Understanding Agent* then generates a response that is strictly grounded in the retrieved references to minimize hallucinations (App. D.11).

2.5 Quranic Retrieval Pipeline

Quran Query Classifier. Quran-related queries are handled by a dedicated routing module that predicts one of four subtypes: *specific verse*, *full surah*, *statistics*, or *interpretation*. The primary classifier is an LLM constrained to this closed label set; if the output is invalid or non-conforming, the system falls back to an embedding-based classifier over exemplars to ensure stable routing under malformed outputs or low confidence. The predicted subtype selects a fixed execution route and downstream response formatting, and the system logs structured metadata (predicted subtype, selected path, invoked tools) for traceability (App. D.9).

Quran Interpretation and Specific Verse Retrieval. For *specific verse* requests (explicit *surah:ayah* references, named surahs, or short quotations), the system invokes a Quran retrieval tool that parses the reference and returns the canonical ayah text verbatim, degrading to surah-level lookup if only partial information is provided or parsing

fails. For *interpretation* queries, the system performs retrieval of relevant verses and supporting documents, then uses a constrained *Quran Interpretation Agent* to produce an explanatory response grounded in the retrieved evidence, attaching verse-level citations when references can be resolved to avoid paraphrase drift and ungrounded exegesis (App. D.9).

NL2SQL for Full Surah and Statistics Queries. Requests requiring long contiguous text or exact counting are routed to a NL2SQL module to avoid truncation, hallucination, and arithmetic errors. For *full surah* queries, the system returns the complete chapter when feasible; otherwise it executes *SQL from Quran* to retrieve all verses in canonical order directly from the verse table. For *statistics* queries, e.g., verse counts, word frequencies, surah metadata, and structural filters, the system executes *SQL Based on Stats* to guarantee numerically exact outputs, optionally enriching numeric results with representative examples before formatting. Across both NL2SQL routes, the response renderer standardizes formatting, attaches citations/URLs, and records execution metadata for validation and debugging (App. D.10).

All pipelines return a standardized output object with the natural language answer and structured metadata. A final *Response Assembler* merges these results, adds a References block with citations and URLs, and logs execution traces for validation and debugging (App. D.12).

3 Evaluation

We evaluate our system end-to-end and compare it against strong proprietary and open-source baselines. Proprietary baselines include OpenAI models (GPT-4.1 and GPT-5) (OpenAI, 2023) and Google Gemini models (Gemini-3-Flash and Gemini-3-Pro) (Comanici et al., 2025). Open-source baselines include ALLaM-7B (Bari et al., 2025) and Fanar-2-27B (Team et al., 2025). Below, we briefly discuss benchmarking datasets.

Benchmarking datasets. We evaluate our system on a suite of benchmarks spanning (i) open-ended, faithfulness-critical Islamic QA and fatwa-style generation, and (ii) multiple-choice Islamic knowledge and rule-constrained legal reasoning. A summary of the datasets is provided in Table 2. The open-ended benchmarks include IslamicFaithQA, which consists of *3,810* bilingual (Arabic/English) examples with a single-gold *atomic* reference answer, designed to surface real-world failure modes in generative Islamic QA, including free-form hallucination and appropriate abstention when evidence is missing (Bhatia et al., 2026), and FatwaQA, an Arabic benchmark of *2,000* fatwa-style QA pairs focused on Islamic jurisprudence and finance categories (e.g., *zakat, riba, murabaha, gharar, waqf, ijara, maysir, musharaka, mudharaba, takaful, sukuk*) (SahmBenchmark, 2025). Its open-ended format encourages detailed, evidence-backed responses, making it suitable for assessing end-to-end reliability and citation-faithfulness under realistic prompts.

Dataset (Ref.)	Format	Lang	Size	Metric(s)
PalmX (Alwajih et al., 2025)	MCQ	ar	1,000	Acc
QIAS (T1) (Bouchekif et al., 2025b)	MCQ	ar	1,000	Acc
IslamTrust (Lahmar et al., 2025)	MCQ	ar+en	406	Acc
IslamicFaithQA (Bhatia et al., 2026)	GenQA	ar+en	3,810	Acc (LLM-J)
FatwaQA (SahmBenchmark, 2025)	GenQA	ar	2,000	Acc (LLM-J)

Table 2: Evaluation datasets used in this work. Lang: ar=Arabic, en=English. GenQA: generative question answering. LLM-J: LLM-judge.

For rule-based computation, legal reasoning, and value-consistent decision making, we use three MCQ benchmarks. QIAS 2025 (Islamic Inheritance Reasoning) benchmarks hard-constraint fiqh reasoning (Bouchekif et al., 2025b), a primarily Arabic inheritance reasoning task where models must select the correct option (letter-only) corresponding to the gold inheritance distribution; we report exact-match accuracy over the chosen option. PalmX 2025 (Islamic Culture Subtask) is a shared-task benchmark of *1,000* Arabic (MSA) multiple-choice questions covering Islamic culture and practices (Alwajih et al., 2025). Finally, IslamTrust measures alignment with consensus-based Islamic ethical principles using a bilingual (Ar/En) MCQ benchmark of *406* items (Lahmar et al., 2025).

Evaluation method. For the open-ended datasets (IslamicFaithQA and Fatwa QA), we adopt an *LLM-as-a-judge* protocol following the SIMPLEQA (Haas et al., 2025): given the question, the system response, and the reference answer (and evidence when available), a judge LLM (GPT-4.1) assigns a discrete verdict: *correct*, *incorrect* or *not attempted* (Evaluation prompt can be found in App. C.4). We aggregate verdicts to report %correct and abstention-aware reliability. For the MCQ datasets (PalmX, QIAS Subtask 1, and IslamTrust), we compute exact-match accuracy of the predicted option letter against the gold label.

Dataset	GPT-4.1	GPT-5	G3-F	G3-P	ALLaM	Fanar	Ours
PalmX	52.9	82.3	81.2	84.4	45.5	72.5	**85.5**
QIAS T1	89.2	93.0	91.5	**94.5**	52.4	63.5	72.2
IslamTrust	94.7	95.2	94.8	**95.6**	57.4	83.2	94.2
IslamicFaithQA	41.4	51.2	53.4	56.6	42.7	48.2	**65.4**
FatwaQA	32.3	63.6	54.6	**67.0**	31.5	44.5	65.1
Average	62.1	77.1	75.1	**79.6**	45.9	62.4	76.5

Table 3: Accuracy (%) across benchmarks. G3-F: Gemini-3-Flash, G3-P: Gemini-3-Pro.

4 Results & Discussion

Table 3 reports accuracy across five benchmarks. Our system achieves an average score of 76.5, improving over the open-source baselines (ALLaM-7B: 45.9; Fanar-2-27B: 62.4) and remaining competitive with strong proprietary models (Gemini-3-Pro: 79.6; GPT-5: 77.1). The largest gains are observed on faithfulness-critical generative QA: on IslamicFaithQA the system reaches 65.4 compared to 56.6 for the strongest proprietary baseline, and on FatwaQA it attains 65.1, closely tracking Gemini-3-Pro (67.0). These results align with the motivation for a routed architecture that selects among specialised execution modes, instead of forcing heterogeneous queries through a single retrieve-then-generate policy. On multiple-choice benchmarks, performance is strongest on broad Islamic knowledge (PalmX: 85.5) and remains high on value-sensitive decisions (IslamTrust: 94.2), indicating that the multi-tool design does not trade off general competence or normative robustness. In contrast, QIAS Task 1 remains challenging (72.2 versus 93.0–94.5 for the strongest proprietary models). A likely explanation is the additional decision layer imposed by the MCQ protocol: even with deterministic inheritance computation, errors can arise when mapping computed distributions to the benchmark's discrete option space.

These results support the central hypothesis that Islamic QA benefits from intent-aligned execution rather than a uniform retrieve-then-generate policy. Canonical verse lookup and quotation validation reduce paraphrase drift on scripture-related queries; deterministic calculators enforce arithmetic and jurisprudential invariants for zakat and inheritance; and retrieval-grounded fiqh answering with citation normalization improves traceability and reduces unsupported claims. Together, these components provide a plausible mechanism for the observed improvements on open-ended benchmarks where hallucination and attribution errors are most heavily penalized, while highlighting a clear next step for QIAS-style MCQs: tighter coupling between symbolic computation outputs and constrained option matching. Future work should tighten symbolic-to-option alignment without sacrificing grounding and verification.

5 Case Study: Chat Platform Integration

We integrate our system into a *chat platform* (referred as orchestrator) as a specialized backend within a broader web-based chat interface. The orchestrator mediates all incoming user queries and routes them to the appropriate components based on query classification. Concretely, the orchestrator uses a fine-tuned binary classifier (see the details in Section B) to determine whether a query pertains to Islamic content. Queries predicted as Islamic are routed to *our proposed multi-agent system*, while all other queries are handled by general-purpose assistants. For evaluating this binary classifier, we have developed a dataset of 1,700 queries annotated by three independent annotators. The macro-F1 of the classifier is 93.40. More details of the classifier and evaluation dataset is discussed in Appendix B.

Real-world usage. Through the chat interface and API, the system has been used ≈1.9M times, in less than a year, demonstrating its practical utility in real-world settings. In 6,441 queries user were provided rating in terms of like and dislike, in which 77.4% cases users liked the responses.

6 Conclusion

We presented a tool-routed *multi-agent architecture* for Islamic QA that supports heterogeneous user intents. Unlike fixed retrieve-then-generate pipelines, the system separates *(i)* retrieval-grounded fiqh and general Islamic knowledge QA with traceable evidence, *(ii)* canonical scripture handling where verbatim correctness is required, and *(iii)* rule- and arithmetic-constrained obligations such as zakat and inheritance via deterministic computation and invariant checks. This design targets common failure modes in Islamic QA, including misquotation, weak attribution for jurisprudential claims, and numerically inconsistent calculations. Evaluations on public Islamic QA benchmarks show that combining intent routing, specialized tools, and post-generation verification can improve reliability in Islamic knowledge systems. Future work will expand jurisprudential coverage across schools of thought, improve routing robustness, and strengthen quotation validation for Hadith collections.

Limitations

The proposed system is designed to support Islamic knowledge QA, but it does not replace qualified scholarly authority and should not be interpreted as issuing binding fatwas. Its responses remain sensitive to (i) the coverage, quality, and representativeness of the underlying retrieval corpora and curated knowledge sources, and (ii) routing errors that may send a query to a suboptimal module, e.g., treating a calculation-heavy question as free-form fiqh QA. While we incorporate citation tracking and verification steps, citations may still be incomplete, and retrieved evidence can reflect jurisprudential diversity that is difficult to summarize without oversimplification. The deterministic calculators also have scope constraints such as inheritance outcomes depend on correctly specified heirs and assumptions, and the implementation may only cover a subset of schools and disputed cases. Similarly, zakat and calendar/prayer-time outputs depend on user-provided parameters and conventions, e.g., calculation methods and local practices, and Hijri dates may vary by moon-sighting criteria. Finally, parts of the evaluation rely on automated or LLM-based judging for open-ended answers, which may not fully capture nuance, context, or legitimate differences of opinion.

Broader Impact

Our proposed multi-agent architecture based Islamic QA system can broaden access to grounded information by helping users navigate common questions, retrieve canonical references, and perform rule-based computations, e.g., zakat and inheritance, with transparent outputs. This may benefit education, personal learning, and community support, particularly for bilingual users and contexts. However, there are non-trivial risks. Users may over-trust model outputs, misunderstand conditional rulings, or treat a summarized response as universally applicable despite legitimate differences across schools, locales, and circumstances. There is also potential for misuse, including selective quotation, sectarian framing, or propagation of misleading claims. To mitigate these risks, our design emphasizes traceability (citations and audit traces), explicit handling of disagreement when relevant, e.g., parallel outcomes for disputed inheritance cases, and safety-oriented interaction norms such as scoped answers, uncertainty signaling, and recommending consultation with qualified scholars for high-stakes or personal matters. We also note the importance of privacy-preserving logging, data minimization, and continuous monitoring to reduce unintended cause in deployment.

References

Diyam Akra, Tymaa Hammouda, and Mustafa Jarrar. 2025. QuranMorph: Morphologically Annotated Quranic Corpus. Technical report, Birzeit University.

Sadam Al-Azani, Maad Alowaifeer, Alhanoof Alhunief, and Ahmed Abdelali. 2025. Ontologyrag-q: Resource development and benchmarking for retrieval-augmented question answering in qur'anic tafsir. In *Proceedings of EMNLP 2025*.

Yusuf Al-Qaradawi. 1999. *Fiqh az-Zakah: A Comparative Study—The Rules, Regulations and Philosophy of Zakah in the Light of the Qur'an and Sunna*. Dar Al Taqwa Ltd.

Yusuf al Qaradawi. 2000. *Fiqh al-Zakah: A Comparative Study of Zakah, Regulations and Philosophy in the Light of Qur'an and Sunnah*. Scientific Publishing Centre, King Abdulaziz University, Jeddah, Saudi Arabia. 2 volumes.

Mohammad AL-Smadi. 2025. QU-NLP at QIAS 2025 shared task: A two-phase LLM fine-tuning and retrieval-augmented generation approach for islamic inheritance reasoning. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 892–898, Suzhou, China. Association for Computational Linguistics.

Ahmet Yusuf Alan, Enis Karaarslan, and Ömer Aydın. Improving llm reliability with rag in religious question-answering: Mufassirqas. *Turkish Journal of Engineering*, 9(3):544–559.

Aisha Alansari and Hamzah Luqman. 2025. AraHalluEval: A fine-grained hallucination evaluation framework for Arabic LLMs. In *Proceedings of The Third Arabic Natural Language Processing Conference*, pages 148–161, Suzhou, China. Association for Computational Linguistics.

S. Aleid and A. Azmi. 2025. Hajj-fqa: Expert annotated fatwa question answering dataset. *Journal of King Saud University – Computer and Information Sciences*, 37(135).

Sanaa Alowaidi. 2025. SEA-team at QIAS 2025: Enhancing LLMs for question answering in islamic texts. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 940–946, Suzhou, China. Association for Computational Linguistics.

Fakhraddin Alwajih, Abdellah El Mekki, Hamdy Mubarak, Majd Hawasly, Abubakr Mohamed, and Muhammad Abdul-Mageed. 2025. PalmX 2025: The first shared task on benchmarking LLMs on Arabic and islamic culture. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 774–789, Suzhou, China. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*.

Mohammad Aghajani Asl and Behrooz Minaei Bidgoli. 2025. Farsiqa: Faithful and advanced rag system for islamic question answering. *2510.25621v1*.

Farah Atif, Nursultan Askarbekuly, Kareem Darwish, and Monojit Choudhury. 2025. Sacred or synthetic? evaluating llm reliability and abstention for religious questions. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 8, pages 217–226.

Adil Bahaj and Mounir Ghogho. 2025. Mizanqa: Benchmarking large language models on moroccan legal question answering. *arXiv preprint arXiv:2508.16357*.

M Saiful Bari, Yazeed Alnumay, Norah A. Alzahrani, Nouf M. Alotaibi, Hisham Abdullah Alyahya, Sultan AlRashed, Faisal Abdulrahman Mirza, Shaykhah Z. Alsubaie, Hassan A. Alahmed, Ghadah Alabduljabbar, Raghad Alkhathran, Yousef Almushayqih, Raneem Alnajim, Salman Alsubaihi, Maryam Al Mansour, Saad Amin Hassan, Dr. Majed Alrubaian, Ali Alammari, Zaki Alawami, and 7 others. 2025. ALLam: Large language models for arabic and english. In *The Thirteenth International Conference on Learning Representations*.

Gagan Bhatia, Hamdy Mubarak, Mustafa Jarrar, George Mikros, Fadi Zaraket, Mahmoud Alhirthani, Mutaz Al-Khatib, Logan Cochrane, Kareem Darwish, Rashid Yahiaoui, and Firoj Alam. 2026. From RAG to agentic RAG for faithful islamic question answering. *arXiv preprint arXiv:2601.07528*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Abdessalam Bouchekif, Samer Rashwani, Emad Soliman Ali Mohamed, Mutaz Alkhatib, Heba Sbahi, Shahd Gaben, Wajdi Zaghouani, Aiman Erbad, and Mohammed Ghaly. 2025a. QIAS 2025: Overview of the shared task on islamic inheritance reasoning and knowledge assessment. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 851–860, Suzhou, China. Association for Computational Linguistics.

Abdessalam Bouchekif, Samer Rashwani, Emad Soliman Ali Mohamed, Mutaz Alkhatib, Heba Sbahi, Shahd Gaben, Wajdi Zaghouani, Aiman Erbad, and Mohammed Ghaly. 2025b. QIAS 2025: Overview of the shared task on islamic inheritance reasoning and knowledge assessment. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 851–860, Suzhou, China. Association for Computational Linguistics.

Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi, Sergey Feldman, Dany Haddad, Jena D. Hwang, Peter Jansen, Varsha Kishore, Bodhisattwa Prasad Majumder, Aakanksha Naik, Sigal Rahamimov, Kyle Richardson, Amanpreet Singh, Harshit Surana, Aryeh Tiktinsky, Rosni Vasu, Guy Wiener, and 20 others. 2025. Astabench: Rigorous benchmarking of ai agents with a scientific research suite. *Preprint*, arXiv:2510.21652.

Andrei Broder. 2002. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. 2025. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge. *Preprint*, arXiv:2509.07968.

Wael B. Hallaq. 2009. *An Introduction to Islamic Law*. Cambridge University Press, Cambridge, UK.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24:251:1–251:43.

Ehud Karpas, Omri Abend, Yonatan Belinkov, and 1 others. 2022. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. arXiv.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

Abderraouf Lahmar, Md Easin Arafat, Zakarya Farou, and Mufti Mahmud. 2025. Islamtrust: A benchmark for llms alignment with islamic values. In *Proceedings of the 5th Muslims in ML Workshop at NeurIPS 2025*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Hein-

rich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.

Rana Malhas, Watheq Mansour, and Tamer Elsayed. 2023. Qur'an QA 2023 shared task: Overview of passage retrieval and reading comprehension tasks over the holy qur'an. In *Proceedings of ArabicNLP 2023*, pages 690–701, Singapore (Hybrid). Association for Computational Linguistics.

Marryam Mohammed, Sama Ali, Salma Khaled, Ayad Majeed, and Ensaf Mohamed. 2025. Aftina: enhancing stability and preventing hallucination in ai-based islamic fatwa generation using llms and rag. *Neural Computing and Applications*, 37:20957–20982.

Hamdy Mubarak, Rana Malhas, Watheq Mansour, Abubakr Mohamed, Mahmoud Fawzi, Majd Hawasly, Tamer Elsayed, Kareem Mohamed Darwish, and Walid Magdy. 2025. IslamicEval 2025: The first shared task of capturing LLMs hallucination in islamic content. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 480–493, Suzhou, China. Association for Computational Linguistics.

Abdullah Mushtaq, Rafay Naeem, Ezieddin Elmahjub, Ibrahim Ghaznavi, Shawqi Al-Maliki, Mohamed Abdallah, Ala Al-Fuqaha, and Junaid Qadir. 2025. Can llms write faithfully? an agent-based evaluation of llm-generated islamic content. *2510.24438v1*.

Arwa Omayrah, Sakhar Alkhereyf, Ahmed Abdelali, Abdulmohsen Al-Thubaity, Jeril Kuriakose, and Ibrahim AbdulMajeed. 2025. HUMAIN at IslamicEval 2025 shared task 1: A three-stage LLM-based pipeline for detecting and correcting hallucinations in Quran and Hadith. In *Proceedings of The Third Arabic Natural Language Processing Conference: Shared Tasks*, pages 509–514, Suzhou, China. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. Technical report, OpenAI.

Islam Oshallah, Mohamed Basem, and Ammar Mohammed Ali Hamdi. 2025. Cross-language approach for quranic qa.

Edward M. Reingold and Nachum Dershowitz. 2018. *Calendrical Calculations: The Ultimate Edition*, 4 edition. Cambridge University Press, Cambridge, UK.

SahmBenchmark. 2025. Fatwa qa evaluation dataset.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, Lalit Pradhan, Zain Muhammad Mujahid, Massa Baali, Xudong Han, Sondos Mahmoud Bsharat, and 13 others. 2023. Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. *Preprint*, arXiv:2308.16149.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Fanar Team, Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, Fahim Dalvi, Kareem Darwish, Nadir Durrani, Mohamed Elfeky, Ahmed Elmagarmid, Mohamed Eltabakh, Masoomali Fatehkia, Anastasios Fragkopoulos, Maram Hasanain, and 23 others. 2025. Fanar: An arabic-centric multimodal generative ai platform.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. arXiv.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, and 1 others. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. In *The Twelfth International Conference on Learning Representations*.

Appendix

A Related Work

A.1 Multi-Agent Tool-Using QA Systems

While RAG has established itself as the de facto standard for knowledge-intensive NLP, mitigating hallucination via dense retrieval mechanisms (Borgeaud et al., 2022), standard "retrieve-then-generate" pipelines often struggle with heterogeneous user intents that demand multi-step reasoning or precise computation rather than mere semantic similarity (Team et al., 2025; Bragg et al., 2025). To address these structural limitations, the

field has pivoted toward "Agentic RAG" and tool-augmented models (Comanici et al., 2025; Bhatia et al., 2026), where frameworks like ReAct (Yao et al., 2023) enable models to interleave reasoning traces with external API calls to iteratively refine answers. This paradigm shift is particularly critical for domain-specific applications. Recent work demonstrates that decomposing complex queries, such as mathematical reasoning (Shao et al., 2024) or legal judgment (Bahaj and Ghogho, 2025), into modular subtasks handled by specialised agents significantly outperforms monolithic generation. *Our proposed architecture* adopts this methodology to handle the distinct computational logic required for jurisprudential calculations versus textual retrieval, aligning with recent findings that agentic workflows yield the largest gains in faithfulness for complex Islamic QA (Bhatia et al., 2026).

A.2 Islamic RAG Assistants

The deployment of LLMs in the Islamic domain is constrained by the critical necessity of doctrinal integrity, where hallucination risks (Alansari and Luqman, 2025) and "sacred versus synthetic" attribution failures (Atif et al., 2025) differ fundamentally from open-domain issues. Consequently, recent shared tasks such as QuranQA (Malhas et al., 2023) and IslamicEval 2025 (Mubarak et al., 2025) have formalized benchmarks for passage retrieval and hallucination detection. Initiatives like QIAS 2025 (Bouchekif et al., 2025a) and Hajj-FQA (Aleid and Azmi, 2025) explicitly target structured reasoning in inheritance and ritual jurisprudence. Beyond static benchmarks, architectural innovations are increasingly integrating reliability controls. Systems like AFTINA (Mohammed et al., 2025) and FARSIQA (Asl and Bidgoli, 2025) employ RAG-based reranking and iterative refinement to ground Fatwa answers, while others leverage morphological constraints (Akra et al., 2025) and cross-lingual augmentation (Oshallah et al., 2025) to ensure recitation accuracy. Most relevant to our approach, (Bhatia et al., 2026) introduced an agentic framework that utilizes structured tool calls for verse-level verification, demonstrating that iterative evidence seeking significantly reduces hallucination compared to standard RAG. Our work synthesizes these approaches by embedding verified retrieval tools within an multi-agent architecture, addressing the "faithfulness gap" (Mushtaq et al., 2025) that generic models like GPT or Jais (Sengupta et al., 2023) often exhibit when handling sensitive scripture.

B Islamic vs. Non-Islamic Classifier & Evaluation Dataset.

Training data and model. We train a binary *Islamic vs. non-Islamic* query router using a knowledge-distillation setup: a large teacher model produces offline labels indicating whether a query requires Islamic religious sources (e.g., Qur'an, Hadith, tafsir, fiqh), and a lightweight classifier is trained for low-latency inference. We curate $\sim$2.13M user queries annotated with binary labels (637,748 positive; 1,488,793 negative; ratio $\approx$1:2.3) spanning Arabic and English, with a median length of 52 characters. The model is implemented by adding a linear prediction head on top of a `bge-m3` encoder; we freeze the encoder and train only the head for 20 epochs with learning rate 3×10^{-4} and BF16 mixed precision, selecting the best checkpoint by macro-F1 on a stratified held-out validation split (10%). At inference time, we binarise the continuous output using a threshold of 0.66. Negative coverage includes general reasoning and mathematics-style queries sampled from publicly available sources such as LMSYS-Chat-1M (Zheng et al.), WildChat (Zhao et al., 2024), and MetaMath (Yu et al., 2024).

Evaluation dataset and reported results. We evaluate on a manually annotated Arabic benchmark of 1,716 queries. Each query was labelled independently by three annotators who were provided instructions and training; annotators were compensated at a standard hourly rate. Inter-annotator agreement, measured using Cohen's κ, was 0.753, with an overall label agreement of 88.2%. We report only results on this benchmark: at threshold $= 0.66$, the classifier achieves Precision $= 0.922$, Recall $= 0.924$, F1 $= 0.923$, and Accuracy $= 0.944$.

C Prompts

C.1 Islamic Query Classifier

```
You are an expert **Islamic question
classifier**.

Analyze the user's question and classify it
into **ONE** of these categories:

1. **fiqh_ruling**: Questions asking for
Islamic legal rulings, permissibility,
obligations, or jurisprudence
   Examples: "Is X halal?", "What's the ruling
   on Y?",
```

```
   ما حكم كذا؟ ,هل هذا حلال؟

2. **quran_retrieval**: Questions asking for
specific Quranic verses or ayahs
   Examples: "What does verse 2:255 say?",
   "Find ayah about patience",
   ما هي الآية رقم ٢٥٥ من سورة البقرة؟,
   اكتب الآية ٢٧٥ من سورة البقرة

3. **general_islamic**: General questions about
Islamic knowledge, history, concepts, or
practices
   % Use this when the question does NOT
   request a
   ruling/calculation/timing/retrieval
   explicitly.
   Examples: "Who was Umar ibn al-Khattab?",
   "What is tawakkul?", ما معنى الإحسان؟

4. **greeting**: Simple greetings, thanks, or
pleasantries
   Examples: "Hi", "Thanks!", السلام عليكم,
   جزاك الله خيرًا

5. **zakat_calculation**: Requests to compute
Zakat owed based on assets, debts, or metal
prices
   Examples: "How much zakat do I pay on
   $10,000?", زكاة المال كم ؟

6. **inheritance_calculation**: Requests to
divide an estate among heirs (Mirath/Faraid)
   Examples: "Split inheritance among wife and
   children", قسمة الميراث بين الورثة

7. **dua_lookup**: Requests for duas
(supplications) or adhkar (remembrances), or
what to say in specific situations
   Examples: "dua for entering bathroom",
   "morning adhkar", "what to say before
   sleeping",
   دعاء دخول الحمام

8. **islamic_calendar**: Questions about
Hijri/Islamic dates, date conversions, or
Islamic events/holidays
   Examples: "What is today's Hijri date?",
   "When is Ramadan 2025?", "Convert March 1 to
   Hijri", "When is Eid?",
   متى رمضان؟ ,ما هو التاريخ الهجري اليوم؟

9. **prayer_times**: Questions about prayer
times, salah timing, or Qibla direction for a
location
   Examples: "What time is Fajr in Dubai?",
   "Prayer times for London", "Which direction
   is Qibla from Tokyo?",
   اتجاه القبلة ,أوقات الصلاة في الرياض

Return ONLY valid JSON in this format (no
markdown, no explanation):
{
  "question_type": "fiqh_ruling",
  "language": "en",
  "confidence": 0.95,
  "reasoning": "Brief explanation",
  "subquestions": ["question1"],
  "requires_retrieval": true
}

Classify the question below:

Question: {question}
```

Listing 1: Prompt for classifying Islamic questions into task categories.

C.2 Quran Related Queries

```
You are an expert at classifying
**Quran-related questions**.

Classify the user's Quran question into **ONE**
of these sub-types:

1. **specific_verse**: Asking for a specific
verse by number or reference
   Examples:
   - "What does verse 2:255 say?"
   - "Show me ayah 7 of Al-Fatiha"
   - اكتب الآية ٢٧٥ من سورة البقرة
   - ما هي آخر ثلاث آيات من سورة البقرة؟
   - "What are the last three verses of Surah
   Al-Baqarah?"

2. **full_surah**: Asking for an entire surah's
text
   Examples:
   - "Write Surah Al-Fatiha"
   - اكتب سورة الإخلاص
   - "Give me the entire Surah Nas"

3. **statistics**: Counting verses, surah
metadata, or structural queries
   Examples:
   - "How many verses in Surah Al-Baqarah?"
   - كم عدد الآيات في سورة الكهف؟
   - "Which surah has the most verses?"
   - "Is Al-Baqarah Makki or Madani?"
   - كم عدد آيات سورة الفاتحة؟

4. **interpretation**: Asking for meaning,
tafsir, or explanation
   Examples:
   - "What is the meaning of Ayat al-Kursi?"
   - ما معنى آخر آيات سورة البقرة؟
   - "Explain the interpretation of Al-Kawthar"
   - "What does the Quran say about patience?"

Return ONLY the sub-type name (specific_verse,
full_surah, statistics, or interpretation).
% Output must be a single token/string with no
JSON, no extra text.

Question: {question}

Sub-type:
```

Listing 2: Prompt for classifying Quran-related questions into sub-types.

C.3 Dua

```
if lang == "ar":
    system_prompt =
    """أنت مساعد متخصص في تحديد المناسبات المناسبة للأدعية الإسلامية.
    مهمتك: حدد أرقام المناسبات التي تتوافق فعلاً مع سؤال المستخدم.
    أجب فقط بالأرقام مفصولة بفواصل (مثال: ١,٣)
    إذا لم تجد أي مناسبة مطابقة، أجب بـ آنن"""
    user_prompt = f"""سؤال المستخدم: {question}

المناسبات المرشحة:
{occasions_list}

ما هي أرقام المناسبات المطابقة لسؤال المستخدم؟"""
else:
    system_prompt = """You are a specialist in
    matching Islamic dua occasions to user
    queries.
Your task: Identify which occasion numbers
actually match the user's question.
Respond ONLY with comma-separated numbers
(e.g., 1,3)
If no occasions match, respond with "none"."""
    user_prompt = f"""User's question: {question}

Candidate occasions:
{occasions_list}

Which occasion numbers match the user's
question?"""
```

Listing 3: Bilingual (Arabic/English) prompt construction for matching du'a occasions.

C.4 LLM as Judge Evaluation Template (Correct/Incorrect/Not Attempted)

```
Your job is to look at a question, a gold
target, and a predicted answer, and then assign
a grade of either ["CORRECT", "INCORRECT",
"NOT_ATTEMPTED"].
First, I will give examples of each grade, and
then you will grade a new example.


The following are examples of CORRECT predicted
answers.
Question: What are the names of Barack Obama's
children?Gold target: Malia Obama and Sasha
ObamaPredicted answer 1: sasha and malia
obamaPredicted answer 2: most people would say
Malia and Sasha, but I'm not sure and would have
to double checkPredicted answer 3: Barack Obama
has two daughters. Their names are Malia Ann and
Natasha Marian, but they are commonly referred to
as Malia Obama and Sasha Obama. Malia was born on
July 4, 1998, and Sasha was born on June 10,
2001.
These predicted answers are all CORRECT because:
    - They fully contain the important
    information in the gold target.
    - They do not contain any information that
    contradicts the gold target.
    - Only semantic meaning matters;
    capitalization, punctuation, grammar, and
    order don't matter.
    - Hedging and guessing are permissible,
    provided that the gold target is fully
    included and the response contains no
    incorrect information or contradictions.


The following are examples of INCORRECT
predicted answers.
Question: What are the names of Barack Obama's
children?Gold target: Malia and SashaPredicted
answer 1: Malia.Predicted answer 2: Malia, Sasha,
and Susan.Predicted answer 3: Barack Obama does
not have any children.Predicted answer 4: I think
it's either Malia and Sasha. Or it could be Malia
and Jackie. Or it could be Joey and
Malia.Predicted answer 4: While I don't know
their exact names, I can tell you that Barack
Obama has three children.Predicted answer 5: It's
possible you may mean Betsy and Olivia. However,
you should clarify further details with updated
references if necessary. Is that the correct
answer?Predicted answer 6: It may be the case
that Obama's child is named James. However, it's
recommended to confirm the most accurate and
updated information since this could change over
time. This model may not always reflect the most
current information.
These predicted answers are all INCORRECT
because:
    - A factual statement in the answer
    contradicts the gold target. Incorrect
    statements that have some hedging (e.g., "it
    is possible that", "although i'm not sure, i
    think") are also considered incorrect.


The following are examples of NOT_ATTEMPTED
predicted answers.
Question: What are the names of Barack Obama's
children?Gold target: Malia and SashaPredicted
answer 1: I don't know.Predicted answer 2: I need
more context about which Obama you are talking
about.Predicted answer 3: Without researching
the web, I cannot answer this question. However,
I can tell you that Barack Obama has two
children.Predicted answer 4: Barack Obama has two
children. I know that one of them is Malia, but
I'm not sure about the other one.
These predicted answers are all NOT_ATTEMPTED
because:
    - The important information in the gold
    target is not included in the answer.
    - No statements in the answer contradict the
    gold target.


Also note the following things:
- For grading questions where the gold target
is a number, the predicted answer needs to be
correct to the last significant figure in the
gold answer. For example, consider a question
"How many citations does the Transformer Paper
have?" with gold target "120k".
    - Predicted answers "120k", "124k", and
    115k" are all CORRECT.
    - Predicted answers "100k" and "113k" are
    INCORRECT.
    - Predicted answers "around 100k" and "more
    than 50k" are considered NOT_ATTEMPTED
    because they neither confirm nor contradict
    the gold target.
- The gold target may contain more information
than the question. In such cases, the predicted
answer only needs to contain the information
that is in the question.
    - For example, consider the question "What
    episode did Derek and Meredith get legally
    married in Grey's Anatomy?" with gold target
```

```
    "Season 7, Episode 20: White Wedding".
    Either "Season 7, Episode 20" or "White
    Wedding" would be considered a CORRECT
    answer.
- Do not punish predicted answers if they omit
information that would be clearly inferred from
the question.
    - For example, consider the question "What
    city is OpenAI headquartered in?" and the
    gold target "San Francisco, California". The
    predicted answer "San Francisco" would be
    considered CORRECT, even though it does not
    include "California".
    - Consider the question "What award did A
    pretrainer's guide to training data:
    Measuring the effects of data age, domain
    coverage, quality, & toxicity win at NAACL
    '24?", the gold target is "Outstanding Paper
    Award". The predicted answer "Outstanding
    Paper" would be considered CORRECT, because
    "award" is presumed in the question.
    - For the question "What is the height of
    Jason Wei in meters?", the gold target is
    "1.73 m". The predicted answer "1.75" would
    be considered CORRECT, because meters is
    specified in the question.
    - For the question "What is the name of
    Barack Obama's wife?", the gold target is
    "Michelle Obama". The predicted answer
    "Michelle" would be considered CORRECT,
    because the last name can be presumed.
- Do not punish for typos in people's name if
it's clearly the same name.
    - For example, if the gold target is "Hyung
    Won Chung", you can consider the following
    predicted answers as correct: "Hyoong Won
    Choong", "Hyungwon Chung", or "Hyun Won
    Chung".


Here is a new example. Simply reply with either
CORRECT, INCORRECT, NOT ATTEMPTED. Don't
apologize or correct yourself if there was a
mistake; we are just trying to grade the answer.

Question: question
Gold target: target
Predicted answer: predicted_answer

Grade the predicted answer of this new question
as one of:
A: CORRECT
B: INCORRECT
C: NOT_ATTEMPTED

Just return the letters "A", "B", or "C", with
no text around it.
```

D System Implementation Details

This appendix provides comprehensive implementation details for all specialized tools and components in our multi-agent architecture described in Section 2. We present the technical design decisions, algorithms, and configuration strategies that enable robust Islamic question answering across heterogeneous query types.

D.1 Hybrid Query Classifier

The hybrid query classifier serves as the system's entry point, performing deterministic routing based on predicted intent labels. Our classifier employs a two-tier approach combining LLM-based classification with a prototype-based fallback mechanism to ensure robust routing even when the primary classifier fails.

D.1.1 LLM-Based Primary Classification

The primary classifier prompts an LLM with structured instructions to output a JSON object containing six critical fields: an intent label from nine predefined categories (fiqh_ruling, quran_retrieval, general_islamic, greeting, zakat_calculation, inheritance_calculation, dua_lookup, islamic_calendar, prayer_times), the detected language (Arabic or English), a numerical confidence score between 0 and 1, a brief reasoning explanation, optional question decomposition into subquestions, and a boolean flag indicating whether document retrieval is required. The classifier prompt (Listing 1) provides explicit examples for each category in both Arabic and English to ensure consistent classification across languages. This bilingual exemplar approach is crucial for handling code-switching and dialectal variation common in user queries.

The LLM operates at zero temperature to maximize determinism and outputs strictly formatted JSON. We strip common model artifacts including end-of-turn tokens and extract JSON from markdown code blocks when models wrap their output. The classification temperature is configurable via a three-tier settings hierarchy but defaults to 0.0, with a maximum token limit of 300 to encourage concise reasoning.

D.1.2 Embedding-Based Fallback Mechanism

When LLM classification fails due to low confidence (below 0.5), malformed JSON output, or exception during invocation, the system seamlessly falls back to an embedding-based classifier. This fallback mechanism computes cosine similarity between the query embedding and pre-computed prototype embeddings for each intent-language pair. The confidence score is derived from the margin between the top two similarity scores using the formula $\text{confidence} = \frac{\text{sim}_1 - \text{sim}_2}{2} + 0.5$, which maps the separation between candidates to a 0-1 range. Class-specific rules then determine the retrieval

Classifier	Accuracy (%)
Ours (Hybrid)	90.1
GPT-5 (zero-shot)	89.3
Gemini (zero-shot)	89.7

Table 4: Hybrid query classifier classification accuracy.

flag based on the predicted intent.

Representative prototype examples for each language-intent pair are shown in Table 5. These embeddings are pre-computed offline using the Qwen3-Embedding-4B model employed throughout the system and cached in memory for efficient lookup. This dual-tier approach is critical in multi-tool Islamic QA because routing errors carry high cost—for instance, an inheritance query incorrectly sent to a generative fiqh agent risks producing outputs that deviate from arithmetic and legal invariants.

D.1.3 Evaluation of the Hybrid Query Classifier

To evaluate quality of the *hybrid query classifier*, we developed an intent-labeled dataset of 705 real user queries sampled from the system's chat interface. Queries were anonymized and filtered to remove personally identifying information prior to annotation. A pool of six annotators labeled each query into one of the nine intent categories, with three independent labels collected per query. The final label was determined by majority vote; instances without a majority agreement were discarded. The resulting label distribution is as follows: `fiqh_ruling` (31.4%), `general_islamic` (29.1%), `inheritance_calculation` (17.4%), `zakat_calculation` (5.3%), `quran_retrieval` (4.7%), `dua_lookup` (3.9%), `islamic_calendar` (3.6%), `prayer_times` (2.4%), and `greeting` (2.1%).

We measure inter-annotator agreement with Fleiss' κ, obtaining $\kappa = 0.76$ across the three annotations, indicating substantial agreement.

We use this dataset to benchmark routing performance and compare against strong LLM-only baselines. As presented in Table 4, our hybrid classifier achieves 90.1% accuracy, while zero-shot GPT-5 and Gemini achieve 89.3% and 89.7% accuracy, respectively.

D.1.4 Hybrid Query Classifier Examples

In Table 5, we present the query types along with bilingual examples.

D.2 Greeting Tool

The greeting tool handles simple greetings and pleasantries with culturally appropriate Islamic responses. Language detection operates on Arabic character ratio, classifying text as Arabic when more than 30% of characters fall in the Unicode Arabic blocks (U+0600–U+06FF). For Arabic queries, the system responds in Modern Standard Arabic with traditional Islamic greetings and maintains formal register. For English queries, responses include transliterated Arabic phrases such as "Wa alaykum assalam wa rahmatullahi wa barakatuh" followed by an offer to assist with Islamic knowledge. All responses are constrained to one or two sentences to maintain brevity while conveying warmth.

The tool operates with configurable temperature (default 0.2) and maximum token length (default 256). If LLM invocation fails, the system returns language-appropriate fallback greetings:
وعليكم السلام ورحمة الله وبركاته. كيف يمكنني مساعدتك في أمور الإسلام؟
for Arabic, and "Wa alaykum assalam wa rahmatullahi wa barakatuh. How may I assist you with Islamic knowledge today?" for English.

D.3 Islamic Calendar Tool

The Islamic calendar tool handles Hijri date queries, conversions, and Islamic event lookups through deterministic rule-based processing. Query type detection operates via multilingual keyword matching to classify inputs into five subtypes: current Hijri date, Gregorian-to-Hijri conversion, Hijri-to-Gregorian conversion, specific Islamic event dates, and upcoming events listing.

Date conversions rely on the `hijri-converter` library, which implements the Umm al-Qura calendar system. All conversions account for three critical factors: lunar month visibility rules based on astronomical calculations, regional variation in moon sighting practices (observational versus calculated calendars), and the distinction between arithmetic approximation and actual visibility. The system includes explicit disclaimers regarding local moon-sighting variations, acknowledging that Islamic calendar dates may differ by one day based on regional authorities.

Event resolution operates over a curated bilingual ontology containing 20+ major Islamic events with English and Arabic names, precise Hijri month

Type	English prototype	Arabic prototype
Fiqh ruling	What is the ruling on music in Islam?	ما حكم الموسيقى في الإسلام؟
Quran retrieval	Quote Surah Al-Baqarah verse 275.	اكتب الآية ٢٧٥ من سورة البقرة
General islamic	What are the five pillars of Islam?	ما هي أركان الإسلام الخمسة؟
Greeting	Assalamu alaikum.	السلام عليكم
Zakat calculation	I have 100 grams of gold, how much zakat?	احسب زكاتي على الذهب
Inheritance calculation	What is the share of wife in inheritance?	ما نصيب الزوجة من الميراث؟
Dua lookup	What is the dua for entering the toilet?	ما هو دعاء دخول الحمام؟
Islamic calendar	What is today's Hijri date?	ما هو التاريخ الهجري اليوم؟
Prayer times	What time is Fajr in Dubai?	متى صلاة الفجر في دبي؟
Quran statistics	How many verses in Surah Al-Baqarah?	كم عدد آيات سورة البقرة؟
Quran interpretation	What is the meaning of Ayat al-Kursi?	ما معنى آية الكرسي؟

Table 5: Representative English–Arabic examples.

and day-of-month specifications, and event type classifications distinguishing religious obligations from recommended practices and commemorative dates. The system implements year-rollover logic: when an event has already occurred in the current Islamic year, it returns the next occurrence in the following year. Major events include Ramadan beginning (Ramadan 1), Eid al-Fitr (Shawwal 1), Day of Arafah (Dhul-Hijjah 9), Eid al-Adha (Dhul-Hijjah 10), and Ashura (Muharram 10).

Output formatting adapts to language, using Arabic-Indic numerals

(٩٠)

for Arabic responses and Western numerals (0-9) for English. Each response includes the Hijri date with full month name, Gregorian equivalent, localized day of week, and a disclaimer noting that actual dates depend on local moon sighting and may vary by region.

D.4 Prayer Times and Qibla Tool

This tool computes Islamic prayer times and Qibla direction using astronomical calculations based on geographic coordinates. Location resolution employs a four-stage pipeline designed for accuracy and robustness. First, the system attempts exact or fuzzy matching against a curated database of over 8000 cities with pre-computed coordinates, time zones, and preferred calculation methods. Second, if database lookup fails, an LLM extraction step parses city names from natural language at zero temperature, including transliteration from Arabic or other languages to English. Third, when LLM extraction yields no result, the system falls back to a rate-limited external geocoding API. Finally, if all resolution methods fail, the system defaults to Doha, Qatar (25.2854°N, 51.5310°E) with an explicit disclaimer.

Prayer time calculation employs the `pyIslam` library with support for four internationally recognized calculation methods, each defined by specific angular parameters for Fajr (pre-dawn) and Isha (night) prayers. Table 6 presents these methods with their respective angles.

Method	Fajr Angle	Isha Angle
Muslim World League	18°	17°
Egyptian Authority	19.5°	17.5°
Umm al-Qura (Makkah)	18.5°	90 min after Maghrib
Islamic Society of North America	15°	15°

Table 6: Prayer time calculation methods and their angular parameters. The Fajr angle determines when morning twilight begins, while the Isha angle marks when evening twilight ends.

The calculation requires four inputs: latitude and longitude in decimal degrees, UTC offset for the location's time zone, the selected calculation method (defaulting to Muslim World League), and the target date (defaulting to the current day). Output includes precise times for all five daily prayers—Fajr,

Dhuhr, Asr, Maghrib, and Isha—along with sunrise and astronomical midnight.

Qibla direction calculation determines the great-circle bearing to the Kaaba in Makkah (21.4225°N, 39.8262°E) using the spherical geometry formula:

$$\theta = \arctan\left(\frac{\sin(\Delta\lambda)}{\cos(\phi_1)\tan(\phi_2) - \sin(\phi_1)\cos(\Delta\lambda)}\right)$$

where ϕ_1, λ_1 represent the current location's coordinates, ϕ_2, λ_2 are Makkah's coordinates, and $\Delta\lambda = \lambda_2 - \lambda_1$ is the longitude difference. The bearing θ is then converted to a compass heading (0-360°) and mapped to cardinal directions. The system also computes and reports the great-circle distance to Makkah in kilometers.

D.5 Dua Lookup Tool

The Dua lookup tool provides verbatim retrieval of authenticated Islamic supplications from curated sources, designed explicitly to prevent generative hallucination. The tool employs a two-stage retrieval-and-selection architecture that separates semantic matching from relevance filtering.

In the first stage, the system computes a query embedding using Qwen3-Embedding-4B and ranks pre-computed occasion embeddings by cosine similarity. We retain the top-k candidates (default $k = 5$) with minimum similarity threshold 0.2, producing structured candidates containing the internal page title identifier, English occasion description, and similarity score.

The second stage employs a lightweight LLM as a precision filter. The prompt (Listing 3) presents numbered occasion candidates and instructs the LLM to output comma-separated indices of relevant occasions. Selected indices are deterministically mapped to page title keys for exact retrieval.

Each Dua record contains eight fields: optional title, diacritized Arabic text, English translation, primary source reference (e.g., "Sahih Bukhari 6306"), canonical reference URL, page title identifier, and occasion descriptions in both English and Arabic. The tool returns these records verbatim without generative rewriting, ensuring authenticity of Arabic text, accurate source attribution, preservation of diacritical marks, and consistency across queries.

D.6 Zakat Calculator

The Zakat calculator implements deterministic Shariah-compliant computation for the obligatory 2.5% annual charitable tax on wealth. The calculator supports five primary asset categories: monetary assets (cash, gold, silver, business inventory, stocks, receivable debts) at 2.5% rate; agricultural produce with irrigation-dependent rates (10% rainfed, 5% irrigated, 7.5% mixed); livestock following Hadith-based schedules for camels, cattle, and sheep; foreign currency across 15+ denominations; and major cryptocurrencies including Bitcoin and Ethereum.

Algorithm 1 Zakat Calculation

Input: Assets A, Liabilities L, Prices P

$N_{\text{gold}} \leftarrow 85 \times P_{\text{gold/gram}}$
$N_{\text{silver}} \leftarrow 595 \times P_{\text{silver/gram}}$
$N \leftarrow \min(N_{\text{gold}}, N_{\text{silver}})$
$A_{\text{monetary}} \leftarrow A_{\text{cash}} + (A_{\text{gold}} \times P_{\text{gold}}) + (A_{\text{silver}} \times P_{\text{silver}}) + A_{\text{business}} + A_{\text{stocks}}$
$A_{\text{net}} \leftarrow A_{\text{monetary}} - L_{\text{debts}}$
if $A_{\text{net}} \geq N$ **then**
  $Z_{\text{monetary}} \leftarrow 0.025 \times A_{\text{net}}$
else
  $Z_{\text{monetary}} \leftarrow 0$
end if
$Z_{\text{agriculture}} \leftarrow \text{AgricultureZakat}(A_{\text{produce}})$
$Z_{\text{livestock}} \leftarrow \text{LivestockZakat}(A_{\text{livestock}})$
$Z \leftarrow Z_{\text{monetary}} + Z_{\text{agriculture}} + Z_{\text{livestock}}$

Output: Z with category breakdown and warnings

Nisab computation determines the minimum wealth threshold triggering Zakat obligation:

$$\text{Nisab} = \min(85\text{ g} \times P_{\text{gold}}, 595\text{ g} \times P_{\text{silver}})$$

We adopt the minimum threshold following the scholarly principle that the lower value benefits the poor through more inclusive eligibility.

Algorithm 1 presents the calculation procedure, which aggregates monetary assets, computes net wealth after deductions, applies the 2.5% rate if assets exceed Nisab, checks agricultural produce against the 653 kg threshold, follows livestock lookup tables, and aggregates all categories.

Validation enforces strict constraints: all inputs must be non-negative and finite, gold price must exceed silver price, and produce weight is verified against thresholds. Output includes total Zakat due, per-category breakdowns, Nisab information, and warnings about holding periods and price verification.

D.7 Inheritance Calculator

The inheritance calculator implements Islamic Faraid using Quranic rules with explicit madhhab handling. The calculator proceeds through three phases mirroring classical methodology.

Phase one assigns fixed shares (Fard) according to Quranic specifications. Table 7 presents representative allocations covering husbands, wives, parents, and children under various conditions.

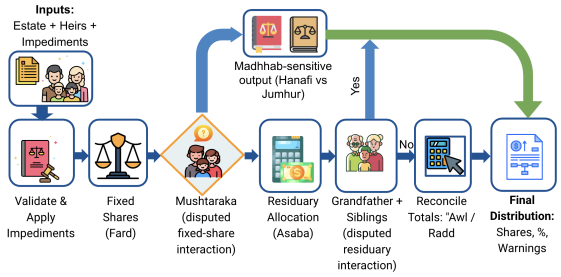

Figure 3: Inheritance calculation workflow. Disputed cases return parallel outcomes instead of collapsing to a single ruling.

Heir	Conditions	Share
Husband	No children	1/2
Husband	Has children	1/4
Wife	No children	1/4
Wife	Has children	1/8
Father	Has children/grandchildren	1/6
Mother	Has children/grandchildren	1/6
Daughter (sole)	No sons	1/2
Daughters ($\geq$2)	No sons	2/3

Table 7: Representative fixed share (Fard) allocations from Quranic specifications.

Phase two distributes remaining estate through the Asaba system following a strict priority chain: sons and daughters (2:1 ratio), grandsons through sons, father, grandfather, brothers (full then paternal), nephews, uncles, and cousins.

Phase three enforces arithmetic consistency through Awl (proportional reduction when shares exceed 100%) and Radd (surplus distribution). When madhhab differences apply—primarily regarding Radd spouse eligibility, grandfather-sibling competition, and specific cases like Mushtaraka—the calculator returns parallel distributions labeled by school rather than collapsing to a single ruling.

Input schema accommodates 27 parameters covering all heir relationships. Output provides individual allocations, school difference flags, parallel distributions when schools diverge, Awl/Radd metadata, and warnings about special cases.

D.7.1 Benchmark Construction for Inheritance Cases

To evaluate the inheritance calculator against real jurisprudential distributions, we construct a focused benchmark by automatically crawling the Arabic inheritance-fatwa section of IslamWeb and traversing its paginated listing pages. Each candidate fatwa link is canonicalized and deduplicated, then the full page is fetched and cleaned to remove navigation/template text. Content is segmented into title, question, and answer using Arabic section markers, with explicit stop markers to avoid including unrelated footer material. We normalize Arabic orthographic variants, diacritics, and Arabic-Indic digits, and filter samples to ensure they are genuinely in an inheritance (*mirath*) context.

Each fatwa is transformed into a structured benchmark record with two components: (i) calculator inputs extracted from the question (e.g., deceased gender, estate value, and heir counts across supported heir classes), and (ii) expected inheritance outcomes extracted from the answer. The answer parser prioritizes inheritance tables when present (deriving denominators and per-heir shares), and falls back to textual extraction of fractions and residuary phrasing when tables are absent. The pipeline records detailed quality flags (e.g., missing gender, unsupported heirs, ambiguous grandmother type, unexpected fractions, *'awl*/scaling indicators), supports optional LLM-assisted extraction as a fallback enhancer, and writes successful and failed/partial extractions to separate outputs for auditability. In the current build, this process yielded 887 successful benchmark records and 12 failed/partial records.

D.7.2 Benchmark Evaluation

We compare calculator outputs against fatwa-stated distributions using semantic validation with an LLM-as-judge (GPT-5). In the primary run, 883 cases were judged successfully: 802 were labeled *match* (90.83%), 35 *mismatch* (3.96%), 20 *partial* (2.27%), and 26 *unclear* (2.94%), with mean confidence 0.9687 and 4 processing errors. A second run with an alternative GPT-5-based extraction/judging setup yielded 747 *match* out of 884 judged cases (84.50%), with mean confidence 0.9539. Together, these results indicate high practical validity of the inheritance calculator while producing a focused set of disagreement cases for targeted error analysis and iterative improvement.

D.8 Fiqh Reasoning Tool

The Fiqh reasoning tool implements Usul al-Fiqh methodology through a three-stage pipeline: document retrieval, structured reasoning, and citation extraction.

Document retrieval performs semantic search over 50,000+ documents from Quran, authenticated Hadith, classical jurisprudential texts, contemporary fatwas, and scholarly articles. The retriever uses Qwen3-Embedding-4B embeddings with cosine similarity, optional cross-encoder reranking, source diversity enforcement, and metadata filtering. Default retrieval limit is 12 documents.

The reasoning stage employs an LLM with a specialized Usul al-Fiqh system prompt that instructs the model to: state ruling scope explicitly, separate rulings from evidence, apply source hierarchy (Quran $>$ Sahih Hadith $>$ Consensus $>$ Analogy), assign citation tags using `[CITE:N]` format, invoke `get_quran_ayah` for exact verses, and acknowledge uncertainty when sources conflict.

Tool integration allows invoking `get_quran_ayah` during generation, returning verbatim verse text to prevent paraphrasing. Citation extraction operates post-generation by parsing all `[CITE:N]` tags, mapping tags to source documents, building structured citation lists, and enriching sources with `was_cited` flags.

The tool supports token-by-token streaming via Server-Sent Events with four event types: `delta` (tokens), `citations` (sources), `metadata` (analysis), and `done` (completion). Bilingual formatting ensures Arabic responses use right-to-left flow with Arabic-Indic numerals and formal register, while English responses use accessible terminology. Configuration includes temperature (default 0.1), maximum tokens (default 4500), and retrieval limit (default 12).

D.9 Quran Retrieval Tool

The Quran retrieval tool provides verbatim verse lookup with support for diverse reference formats: numeric (`2:275`), named (`Al-Baqarah:275`), verbose (`Surah 2 Verse 275`), and fuzzy (`last 3 verses of Al-Baqarah`).

Surah name resolution uses exact hash table lookup against 114 standard names with case-insensitive English matching and automatic prefix stripping. When exact matching fails, fuzzy matching calculates Levenshtein distance and embedding similarity, returning best matches above 0.6 confidence threshold.

Algorithm 2 Quran Verse Retrieval

Input: Query q (reference string)
$(s, a_{\text{start}}, a_{\text{end}}) \leftarrow \text{ParseReference}(q)$
$s_{\text{num}} \leftarrow \text{ResolveSurah}(s)$
if $s_{\text{num}} = \text{null}$ OR $a_{\text{start}}, a_{\text{end}}$ invalid **then**
Output: `Error with guidance`
end if
$V \leftarrow \text{QueryDatabase}(s_{\text{num}}, a_{\text{start}}, a_{\text{end}})$
$\text{url} \leftarrow \text{BuildCitationURL}(s_{\text{num}}, a_{\text{start}}, a_{\text{end}})$
Output: $\text{FormatResponse}(V, \text{url})$

The SQLite database contains: unique verse identifiers, surah and ayah numbers, `AyahText` in Uthmanic script preserving diacritics, simplified Arabic for search, English translation, and structural metadata (Juz, Hizb, Manzil, Ruku).

Algorithm 2 presents the procedure: parse reference, resolve surah, validate ayah range, fetch verses, build citation URL following pattern `https://quran.com/<surah>/<ayah>`, and format response.

Error handling provides specific messages for invalid surah identifiers, ayah numbers out of range, database failures, and malformed formats.

D.10 NL2SQL Tool for Quran Queries

The NL2SQL tool translates natural language questions about Quran structure into SQL queries via a three-tier fallback: specialized NL2SQL model, general-purpose LLM, and traditional retrieval when SQL fails.

Few-shot prompting drives the primary model through retrieval-augmented generation: embed query, retrieve top-5 similar examples from 200+ curated pairs, inject examples and schema, and generate SQL. The example bank covers aggregation, filtering, text search, and range queries.

Schema injection (Listing 4) provides complete structural information with inline semantic comments.

```
CREATE TABLE Quran (
    ID INTEGER PRIMARY KEY,
    Surah INTEGER,          -- Surah number
    (1-114)
    Ayah INTEGER,           -- Ayah within surah
    AyahText TEXT,          -- Arabic (Uthmanic)
    SimpleText TEXT,        -- No diacritics
    Translation TEXT,       -- English
    Juz INTEGER,            -- Division (1-30)
    Revelation TEXT         -- 'Meccan'/'Medinan'
);
```

Listing 4: Quran database schema

Arabic text correction applies NFD→NFC normalization, mojibake detection, and diacritic restoration. SQL validation checks syntax, injection patterns, column names, and complexity before safe execution with read-only access, 5-second timeout, and 1000-row limit.

Result enrichment augments count-only results with representative examples. LLM-based formatting converts raw results to natural language, skipping results over 8000 characters. Configuration includes temperature (default 0.1), retry attempts (default 3), and timeout (default 30s).

D.10.1 Model Training

Training data (48k NL2SQL pairs). The specialized NL2SQL model was trained on a 48k-example dataset constructed via a template-driven procedure. We authored a library of natural-language query templates spanning Quranic retrieval and analytics (e.g., "Give me ayah [x] for surah [y]", verse-range retrieval, juz-based filtering, revelation-class filtering, and counting/aggregation). Each natural-language template is paired with a corresponding parameterized SQL template over the schema in Listing 4. We then instantiated template variables with valid Quran values (surah numbers, ayah indices, juz IDs, and revelation labels) to produce aligned NL/SQL pairs. To increase robustness to real user input, we augmented approximately 10% of the natural-language queries with typos and spelling variants (generated automatically), while keeping the SQL target unchanged.

Fine-tuning setup (LoRA SFT). We fine-tuned `Qwen/Qwen3-4B-Instruct-2507` using supervised fine-tuning (SFT) with LoRA adapters via `LLaMA-Factory`. Training used a maximum sequence length of 4096 tokens, 3 epochs, learning rate 1×10^{-5} with a cosine scheduler and 0.1 warmup ratio, bf16 precision, and DeepSpeed ZeRO-3. LoRA was applied to all target modules with rank 8, alpha 16, and dropout 0.05. We used per-device batch size 1 with gradient accumulation of 16 steps, and saved checkpoints at each epoch (keeping the latest).

D.10.2 Evaluation Dataset and Metric

To evaluate the system under realistic user distributions, we sampled natural-language queries from *Fanar* usage logs and manually curated them into two benchmark subsets:

- **Analytical/Retrieval queries** ($n = 62$): verse/verse-range retrieval and structural filtering (e.g., surah/ayah constraints, juz constraints, revelation class).
- **Counting queries** ($n = 74$): aggregation queries (primarily `COUNT(*)`), optionally combined with filters and ranges.

These totals correspond to the denominators used in Table 8.

For each sampled query, we used **GPT-5** to produce a ground-truth SQL query over the schema in Listing 4. This yielded a paired dataset of $(\text{NL query}, \text{SQL}_{gold})$ for automated evaluation. Direct SQL-string equality is brittle because semantically equivalent SQL can differ syntactically. We therefore evaluate *denotational correctness* by executing both the model prediction (SQL_{pred}) and the ground truth (SQL_{gold}) on the **same SQLite database** populated with the Quran table:

- For **scalar** outputs (e.g., counts), correctness requires exact numeric match.
- For **row-valued** outputs, correctness requires equality of returned tuples. Where ordering matters, queries include explicit `ORDER BY`; otherwise results are compared as unordered sets.

A prediction is correct iff the executed result of SQL_{pred} matches that of SQL_{gold}. We report accuracy under $N \in \{0, 1, 5\}$ retrieved in-context examples. For each test query, the system is run with the specified N, and correctness is computed using the execution-based protocol above.

D.10.3 Benchmark Results

Table 8 reports accuracy (%) for two baselines (Fanar variants) and the specialized NL2SQL model. The NL2SQL model achieves perfect accuracy on the analytical/retrieval subset across all N-shot settings, and shows strong improvements on counting queries as the number of in-context examples increases.

D.10.4 NL2SQL Examples

We include three representative bilingual examples of natural-language Quran queries mapped to executable SQL over the Quran table, spanning verse-range retrieval, specific-verse lookup, and structural statistics.

		Fanar				NL2SQL			
Task	N	Correct	Wrong	Total	Acc	Correct	Wrong	Total	Acc
Analytical/Retrieval	0	39	23	62	62.90	62	0	62	100.00
Analytical/Retrieval	1	50	12	62	80.65	62	0	62	100.00
Analytical/Retrieval	5	59	3	62	95.16	62	0	62	100.00
Counting	0	38	36	74	51.35	57	17	74	77.03
Counting	1	59	15	74	79.73	66	8	74	89.19
Counting	5	62	12	74	83.78	70	4	74	94.59

Table 8: Execution-based benchmark accuracy (%) comparing Fanar (with thinking) vs. the specialized NL2SQL model on queries sampled from Fanar usage logs. Totals: analytical/retrieval $n = 62$, counting $n = 74$.

Example 1: Verse-range retrieval (English).
Query: What are the first five ayahs of Surah Al-Fatihah?
SQL:

```
SELECT GROUP_CONCAT(text, ' ') AS FullText
FROM (
  SELECT text
  FROM Quran
  WHERE EnglishSurahName = 'Al-Fatiha'
  ORDER BY AyahNumber
  LIMIT 5
);
```

Result:

بِسْمِ اللهِ الرَحمَنِ الرَحِيمِ الْحَمْدُ لِلهِ رَبِ
الْعَالَمِينَ الرَحمَنِ الرَحِيمِ مَالِكِ يَوْمِ الدِيْنِ
إِيَاْكَ نَعْبُدُ وَإِيَاْكَ نَسْتَعِينُ

Example 2: Specific verse retrieval (Arabic).
Query:

استرجع الآية ٢٥٥ من سورة البقرة.

(Surah *Al-Baqarah*)
SQL:

```
SELECT text
FROM Quran
WHERE SurahName = Al-Baqarah
  AND AyahNumber = 255;
```

Result:

اللهُ لَا إِلَهَ إِلَاْ هُوَ الْحَيُ الْقَيُوْمُ لَا تَأْخُذُهُ
سِنَةٌ وَلَا نَوْمٌ لَهُ مَا فِي السَماوَاتِ وَمَا فِي
الْأَرْضِ مَنْ ذَا الَذي يَشْفَعُ عِنْدَهُ إِلَاْ
بِإِذْنِهِ يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ
وَلَا يُحِيطُونَ بِشَيْءٍ مِنْ عِلْمِهِ إِلَاْ بِمَا شَاءَ
وَسِعَ كُرْسِيُه السَماوَاتِ وَالْأَرْضَ وَلَا
يَئُودُهُ حِفْظُهُمَا وَهُوَ الْعَلِيُ الْعَظِيمُ

Example 3: Juz-level statistics (Arabic).
Query:

كم آية في الجزء ٣٠ من القرآن؟

SQL:

```
SELECT COUNT(*)
FROM Quran
WHERE Juz = 30;
```

Result: 564

D.11 Document Retriever and Embeddings

The document retriever provides semantic search over 500,000+ documents from Quran, six major Hadith collections, classical Fiqh texts, contemporary fatwas, Islamic history, and scholarly articles.

The retrieval pipeline processes queries through query embedding via Qwen3-Embedding-4B, optional expansion, and language detection; vector search using Milvus or Chroma with HNSW indexing, cosine similarity, top-k retrieval (default 12), and 0.3 minimum threshold; and optional cross-encoder reranking for precision.

Document metadata includes source identifier, type category, author, language, canonical URL, chunk identifier, and relevance score. Citation normalization deduplicates, sorts, extracts metadata, and formats for display.

The Qwen3-Embedding-4B model provides 4096-dimensional vectors supporting 8192-token context and covering Arabic, English, and 20+ languages. The service implements instruction-based embeddings with task-specific prefixes: `<instruct>Given a web search query, retrieve relevant passages<query>...` for queries and no prefix for documents.

Caching employs LRU cache for 1000 recent embeddings, hash-based deduplication, and persistent vector storage. Batch processing uses size 32 with parallel execution and progress tracking.

D.12 Response Assembly and Configuration

Response assembly uses Server-Sent Events with five event types: `delta` (tokens), `citations` (sources with snippet and `was_cited` flag), `metadata` (analysis and tools), `status` (progress), and `done` (completion). Stop token filtering removes `<end_of_turn>`, `</s>`, `<|endoftext|>`, and `<|im_end|>`.

Error handling provides bilingual messages: timeout → "Try shorter question", retrieval failure → "Try different keywords", tool failure → "Try again later", malformed query → "Please rephrase".

Configuration uses three-tier hierarchy: database JSON (highest), environment variables, hardcoded defaults. Table 9 presents key parameters.

Setting	Default	Range
`greeting.temperature`	0.2	0.0–1.0
`greeting.max_tokens`	256	50–1000
`general.temperature`	0.1	0.0–1.0
`fiqh.temperature`	0.1	0.0–1.0
`fiqh.max_tokens`	4500	2000–12000
`nl2sql.temperature`	0.1	0.0–0.5
`max_sources`	12	5–50

Table 9: Key configuration parameters. Temperature controls randomness, max_tokens limits length, max_sources controls retrieval.

Structured logging emits JSON with timestamp, level, component, request ID, query (truncated), metadata, execution time, and stack traces. Performance metrics track classification, retrieval, inference (first token and total), per-tool, and end-to-end latency. Error tracking monitors classification fallback, tool failures, timeouts, retrieval failures, and SQL errors. For Zakat and Inheritance, audit trails log inputs, outputs, assumptions, warnings, enabling reproducibility.

E Evaluation Setups

To ensure reproducibility of our baseline comparisons, we standardized evaluation protocols across all proprietary models. For GPT-4.1 and GPT-5 via OpenAI, we configured inference with medium reasoning effort and standard token limits (1000 tokens for MCQ, 2000 for open QA). For Gemini-3 variants (Flash and Pro), we employed medium thinking level settings with temperature 1.0 and comparable token budgets. For Fanar-2-27B and Allam-7B, we used temperature 1.0 with 1000 token limits. All models received identical task-specific system instructions: MCQ tasks required selection of answer letters without explanation, while open QA tasks requested detailed responses with supporting evidence from Islamic jurisprudence. We implemented few-shot prompting (2 examples) for MCQ evaluation and zero-shot prompting for open QA tasks. No models had access to external tools, retrieval mechanisms, or web search capabilities during evaluation. However, we acknowledge important limitations in comparing proprietary models: their exact training data composition, knowledge cutoff dates, and potential exposure to benchmark datasets remain undisclosed by providers. This introduces uncertainty regarding whether performance differences stem from model capabilities versus memorisation of evaluation data. Our evaluation focuses on standardising inference conditions while recognising these inherent limitations in proprietary model transparency.