

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology



أكاديمية كاوست
KAUST ACADEMY

SAFAA

A Real-Time Edge AI Voice Assistant for Pilgrims using Retrieval Augmented Generation (RAG)

Abrar Alkhamjani
Abrar.alkamajani@gmail.com

Alhanouf Alqahtani
AlhanoufAbdulraqib@gmail.com

Bashayer Abdullah
Bashalqah@gmail.com

Ghaida Alshehri
ghaidaSaadAlshehri@gmail.com

Norah Alshahrani
noraalshahranisa@gmail.com

Mentor: Mojtaba AlShams
mojtaba.alshams@kaust.edu.sa

August 27, 2025

Abstract

Millions of pilgrims and visitors arrive annually at the holy sites, creating one of the most crowded and complex religious environments worldwide. Despite the efforts of guides and organizers, the diversity of languages and the high proportion of elderly and less tech-savvy pilgrims make it impossible to meet all needs instantly and accurately. In such conditions, even a minor error in guidance may cause confusion, highlighting the need for immediate, reliable, and user-friendly assistance. Existing solutions, including smartphone applications and voice assistants such as Siri or Alexa, remain static and general, lacking credibility from official Islamic sources. Similarly, the "Nusuk" application provides valuable services but falls short in delivering real-time, natural, voice-based guidance in crowded settings. To bridge these gaps, we developed **SAFAA**, structured around three foundational pillars. The first is a complete voice interaction pipeline enabling seamless end-to-end assistance. The second is deployment on a Raspberry Pi edge-based system ensuring on device processing. The third is a powerful RAG framework, based on authenticated Islamic sources, which guarantees accuracy, contextual relevance, and trustworthiness of responses. From a 90-second baseline, the system achieved end-to-end responses in less than 10 seconds, confirming real-time applicability.

Contents

Abstract **1**

1 Introduction **3**

2 System Overview **3**

3 Methodology **5**
3.1 System Setup and Environment 5
3.2 Data Preparation 5
3.3 Wake Word Detection 5
3.4 Pipeline Design 6
3.5 Constraints and Trade-offs 6
3.6 Evaluation and Benchmarking 7
3.7 Summary 7

4 Results and Discussion **8**
4.1 Wake-word Detection 8
4.2 Speech-to-Text (STT) 8
4.3 Retrieval and Response Generation 8
4.4 TTS and End-to-End Latency 10
4.5 Resource Usage 10
4.6 Summary and Observations 10

5 Conclusion **10**

1 Introduction

The Hajj pilgrimage represents one of the largest annual gatherings in the world, with millions of pilgrims and visitors arriving at the holy sites. This unique setting transforms the sacred areas into one of the most crowded and logistically complex religious environments globally. While teams of guides and organizers are deployed to assist pilgrims, the overwhelming diversity of languages and the high proportion of elderly and less technologically experienced participants make it nearly impossible to meet all needs instantly and accurately. Under such conditions, even a minor misguidance can result in confusion, repeated rituals, or potential safety risks. These challenges highlight the urgent need for immediate, reliable, and user-friendly guidance systems that are tailored to the specific context of the pilgrimage.

Current digital solutions have proven insufficient. General-purpose smartphone applications and commercial voice assistants such as Siri [1] or Alexa [2] remain static, delivering either superficial answers or responses that lack credibility from official Islamic sources [3]. Similarly, the "Nusuk" application provides valuable services but is limited by its static content, reliance on manual interaction, and inability to offer real-time, natural voice-based guidance in critical moments of crowding. At the same time, while edge devices present a promising alternative by enabling local, low-latency processing, they still face constraints in terms of computational power, energy efficiency, and scalability.

In light of these limitations, this project introduces SAFAA, a voice assistant specifically designed to address the unique demands of the pilgrimage. SAFAA is structured around three foundational pillars: a complete voice interaction pipeline that delivers end-to-end assistance, deployment on a Raspberry Pi [4] with edge-based processing for timely and scalable performance, and a robust RAG framework based on multilingual and authenticated Islamic sources to ensure trustworthy, context- aware responses. By integrating these pillars, SAFAA seeks to bridge existing gaps and demonstrate the potential of edge-based Artificial Intelligence (AI) systems in enhancing both safety and the overall pilgrimage experience. This work assumes that a RAG-powered, voice-to-text pipeline-based system will provide accurate and practical guidance to pilgrims with low latency despite the Raspberry Pi's computational limitations.

2 System Overview

The proposed system, SAFAA, is a RAG-based edge voice assistant developed for deployment on the Raspberry Pi 5. Its primary objective is to support pilgrims by transforming spoken queries into accurate and contextually relevant answers. The workflow begins with converting the pilgrim's speech into text through a Speech-to-Text (STT) engine. This text is then passed to a retrieval module, which searches across reliable and verified Islamic knowledge sources. Once the relevant information is retrieved, the system generates a coherent and contextually grounded response. Finally, the text is delivered back to the user through a Text-to-Speech (TTS) mechanism, ensuring smooth and natural interaction. By combining retrieval accuracy with generative flexibility, SAFAA provides real-time, trustworthy, and accessible guidance in crowded pilgrimage environments.

The overall architecture, illustrated in Figure1, follows a modular pipeline that integrates speech recognition, retrieval, generation, and speech synthesis. While certain components are deployed on the Raspberry Pi 5 to ensure edge processing, others are

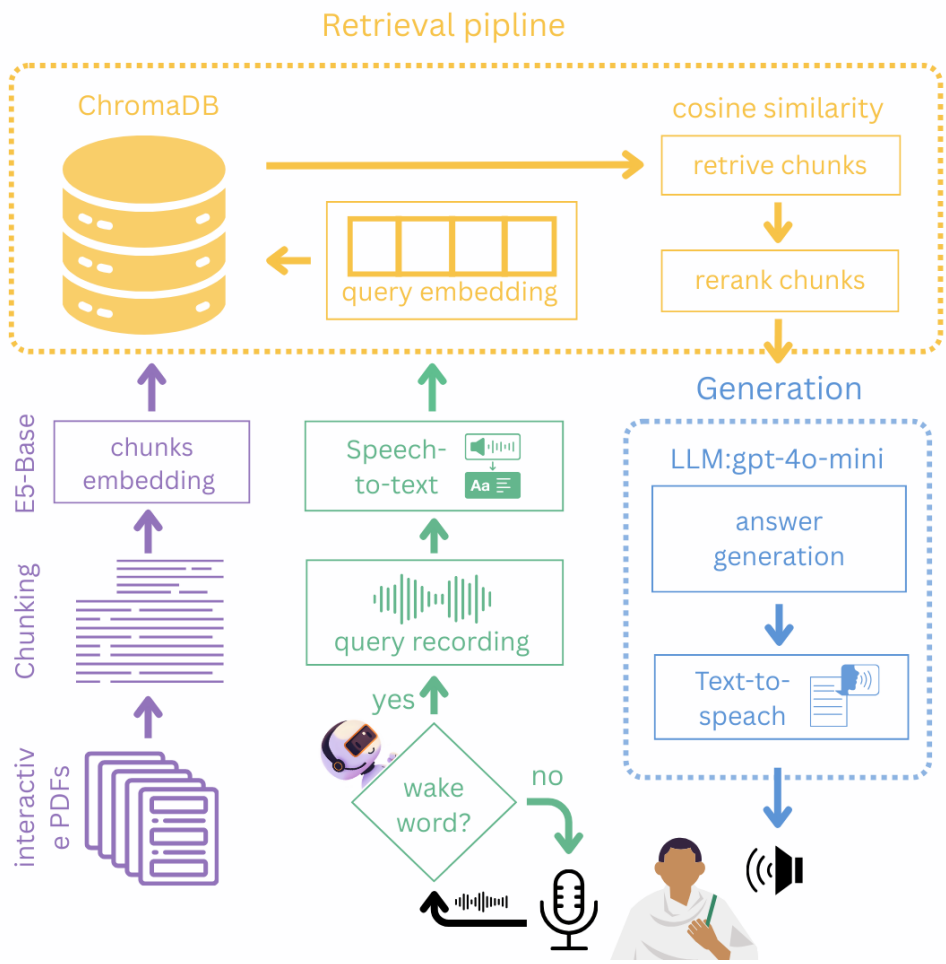


Figure 1: SAFAA's architecture.

executed on cloud services to leverage advanced models such as OpenAI's STT, TTS, and Large Language Model (LLM). The process begins when the assistant is activated with the wake word "SAFAA" allowing the Raspberry Pi to capture and record the spoken query. On the device, the recorded audio is processed into vector embeddings, which are compared against a local ChromaDB index using Hierarchical Navigable Small World (HNSW) search, built from chunked Islamic references such as ritual guides, safety instructions, and emergency procedures. Using cosine similarity and lexical re-ranking, the system identifies the most relevant knowledge chunks. These selected passages are then forwarded to cloud-based modules, where OpenAI's STT transcribes the audio into text, GPT-4o-mini generates a concise and contextually accurate answer, and TTS converts the result into natural speech. This hybrid design leverages both edge processing for efficient retrieval and cloud services for advanced models, providing pilgrims with an immediate and easily understandable response.

This sequential flow ensures low-latency, reliable, and contextually precise interaction, while maintaining the efficiency of edge deployment. It also minimizes unnecessary processing and preserves user privacy by relying on wake-word activation rather than continuous listening.

SAFAA integrates several unique features that distinguish it from generic voice assistants. It is designed as a RAG system, ensuring that every response is grounded in verified

Islamic references rather than uncontrolled generative output. It is deployed on a Raspberry Pi 5, making it portable, low-cost, and scalable for large deployments in crowded environments. It also employs wake-word activation "SAFAA", balancing usability with privacy and resource efficiency. Furthermore, the pipeline combines STT, semantic retrieval, generative answering, and TTS synthesis, forming a complete interaction loop tailored specifically for pilgrims.

3 Methodology

3.1 System Setup and Environment

SAFAA was implemented and deployed on a Raspberry Pi 5, selected for its affordability, portability, and capacity to run lightweight AI applications in resource-constrained environments. Running the system directly on the device allowed it to function as an edge assistant, guaranteeing privacy, reduced dependency on cloud services, and low-latency interactions. The software stack was based on Python 3.11 and integrated several core components: ChromaDB for vector storage and retrieval, the intfloat/e5-base-v2 model [5] for semantic embeddings, the GPT-4o-mini [6] API for answer generation, gpt-4o-transcribe for STT, lightweight TTS modules [7], and OpenWakeWord [8] for wake-word detection. Early trials with llama-cpp-python and quantized local models such as Mistral-7B [9] and LLaMA-2.5 [10] were abandoned due to extreme latency, confirming the need for a hybrid design in which retrieval was local while generation could be offloaded to GPT-4o-mini when internet was available.

3.2 Data Preparation

The assistant's knowledge base was constructed from official Hajj and Umrah guides, safety manuals, and emergency procedure documents [11]. These resources were primarily provided as PDFs. Early experiments revealed major issues: interactive PDFs contained hidden elements such as links and buttons, polluting the dataset and leading to irrelevant retrieval. In addition, flat chunking strategies resulted in poor retrieval quality, with cosine distances often exceeding 700.

To resolve these problems, all PDFs were converted to clean plain text and segmented using a hierarchical and overlapping chunking strategy that preserved semantic continuity across multiple levels (document $\rightarrow$ section $\rightarrow$ paragraph $\rightarrow$ sentence). Embeddings were then generated using E5-base-v2, with the required prefixes (passage: and query:) and normalization to unit length. The dataset was indexed in ChromaDB with HNSW, enabling efficient cosine similarity search even on Raspberry Pi hardware. After these improvements, retrieval quality improved dramatically, with the average cosine distance for correct matches reduced to 0.174, ensuring that queries aligned with the most relevant references.

3.3 Wake Word Detection

Voice interaction was enabled through a wake-word system to avoid continuous listening, which would have consumed unnecessary resources and compromised user privacy. The team initially experimented with Picovoice [12], but the customized wake-word models

lacked sufficient accuracy and consistency. To address this, the design shifted to OpenWakeWord, where a custom model was trained on the keyword "Safa" (صَفا).

The word was chosen for both its cultural relevance—as it is tied to the context of Hajj and Umrah—and its phonetic simplicity, making it easy to pronounce across languages. Multiple testing loops and scenarios were implemented to simulate realistic usage, ensuring robust activation under different conditions. This approach achieved an accuracy between 0.85 and 0.9, producing a detector that was both reliable and user-friendly while maintaining system efficiency and privacy.

3.4 Pipeline Design

The pipeline was structured as a modular sequence. Once the wake word "Safa" was detected, the user's voice query was captured and transcribed into text using gpt-4o-transcribe. The text was then embedded with E5-base-v2 and compared against the ChromaDB HNSW index, which stored chunked Islamic references including ritual guides, safety protocols, and emergency instructions. The top retrieved passages were re-ranked and compiled into a structured prompt.

This prompt was passed to the GPT-4o-mini generation model, which produced a coherent and contextually grounded response. Finally, the response was synthesized into natural speech by a lightweight TTS module, closing the interaction loop.

During development, attempts to keep the pipeline fully offline with models such as Mistral and LLaMA-2.5 proved infeasible: response times exceeded 90 seconds, with generation speeds as low as 3 tokens per second. The final design therefore adopted a hybrid trade-off: all retrieval and embedding remained local, while the generation step was delegated to GPT-4o-mini through the API. This reduced total response time to under 4 seconds, achieving real-time usability while preserving accuracy.

3.5 Constraints and Trade-offs

Several constraints shaped the final design of SAFA. Hardware limitations on the Raspberry Pi 5 restricted CPU and RAM usage, requiring careful optimization of embeddings and retrieval. Latency was a central issue: local LLMs were too slow for practical use, leading to the hybrid solution of cloud-assisted generation. Privacy concerns led to the rejection of continuous listening in favor of a wake-word system. TTS naturalness had to be balanced with speed, resulting in the choice of a lightweight engine that provided clear but fast responses. Data quality also posed challenges, as polluted PDFs required a thorough preprocessing pipeline, while multilingual support was enabled through the API rather than offline models limited to English. These trade-offs ensured that the system achieved both feasibility and reliability under real-world conditions.

3.6 Evaluation and Benchmarking

To rigorously assess the performance of SAFA, a set of well-established evaluation metrics was employed, covering accuracy, latency, efficiency, and usability. These metrics are widely used in the evaluation of speech-driven and retrieval-augmented systems, and they were carefully selected to reflect the practical requirements of pilgrims in real-world conditions.

Word Error Rate (WER) and **Character Error Rate (CER)** were used to evaluate the speech-to-text (STT) component. WER measures the proportion of incorrectly recognized words, while CER captures finer-grained character-level errors. Together, they provide a reliable indication of transcription quality, which is critical since even small recognition errors may alter religious instructions or safety guidelines.

Latency was another core metric, defined as the total time from the spoken query to the generated voice response. Both module-level latency (STT, retrieval, generation, and TTS) and end-to-end latency were considered. This measure is crucial for usability, as delays in response can reduce trust and hinder the flow of rituals that often require immediate guidance.

Retrieval Quality was measured through cosine similarity between query embeddings and knowledge base passages. High similarity indicates that the system is selecting contextually relevant references, which is essential to ensure accuracy and trustworthiness when answering from authenticated Islamic sources.

Resource Usage, including CPU and RAM consumption, was monitored to evaluate the feasibility of deploying the system on a resource-constrained device such as the Raspberry Pi 5. This metric reflects the efficiency and sustainability of the implementation under edge-computing conditions.

Wake-word Detection Accuracy assessed the reliability of the activation mechanism. A high detection rate with negligible false positives is essential for maintaining seamless and private interaction, ensuring that the system activates only when explicitly invoked by the user.

Finally, **Token Throughput (tokens/sec)** was considered for the language generation step. This metric reflects the efficiency of the underlying language model in producing responses and guided the design decision to shift from slower local models to a hybrid architecture leveraging cloud-based generation for practical, real-time interaction.

In summary, these metrics collectively provide a comprehensive framework for evaluating SAFA. They cover linguistic accuracy, system responsiveness, computational efficiency, and user interaction reliability.

3.7 Summary

The methodology demonstrates how SAFAA was engineered through careful system setup, structured data preparation, customized wake-word modeling, and hybrid pipeline integration. Constraints such as hardware limitations, latency, privacy, and data quality were systematically addressed through trade-offs that balanced feasibility with performance. By combining optimized local retrieval with API-assisted generation and validating the system through real-world benchmarks, SAFAA achieved its goals of accuracy, efficiency, and practical usability as a RAG-based edge device designed to guide pilgrims during Hajj and Umrah. The detailed numerical results and their implications are discussed in the following section.

4 Results and Discussion

This section evaluates the performance of SAFAA and compares it against a local baseline system. The baseline consists of a local LLM combined with traditional chunking methods, providing a reference for latency, LLM speed, and retrieval efficiency. While functional, the baseline exhibits high end-to-end latency (≈90 s), slow token processing (≈3 tokens/s), and imprecise retrieval (cosine distances ≈740). As seen in Table 1 where SAFAA aims to improve on these aspects with an integrated retrieval and edge LLM pipeline.

Label	SAFAA	Baseline
End-to-End Latency(s)	9.83	90
LLM Speed (tokens/s)	26.5	3
Retrieval Distance	0.17	704

Table 1: High-level comparison of SAFAA with the local baseline

4.1 Wake-word Detection

Wake-word detection is essential for energy efficiency and privacy. SAFAA achieves an accuracy between 0.66 and 0.79 (Table 2), with no false activations observed. However, performance declines under crowded conditions, where accuracy drops to 0.58 compared to the quiet setting (0.79), indicating sensitivity to extreme noise.

Component	Metric	Performance Metric
Wake-word detection	Accuracy	0.66–0.79
STT	Word Error Rate (WER)	0.333–0.502
STT	Character Error Rate (CER)	0.103–0.154
Retrieval	Cosine similarity (top 3)	0.1–0.18

Table 2: SAFAA System Accuracy Metrics

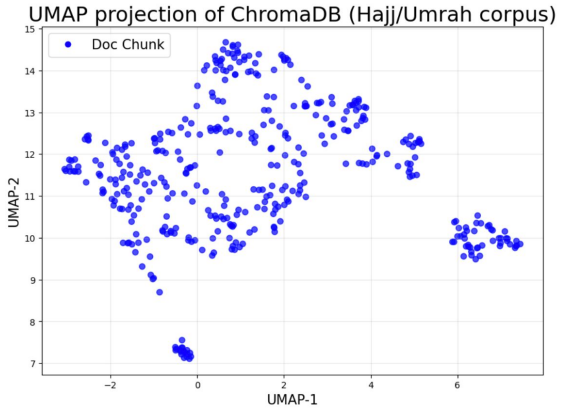
4.2 Speech-to-Text (STT)

The STT module converts spoken input into text. SAFAA achieves a Word Error Rate (WER) of 33–50% and a Character Error Rate (CER) of 10–15%(Table 2). While still below cloud-based STT systems (WER <10% in clean conditions [13]), this is a significant improvement over the local baseline, particularly in noisy or multilingual environments.

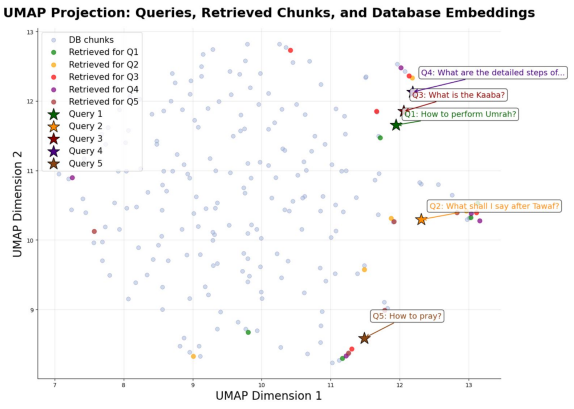
4.3 Retrieval and Response Generation

SAFAA uses an optimized retrieval pipeline, achieving top-3 cosine distances of 0.108–0.181 (Table 2) compared to 740 for the local baseline [5]. UMAP visualizations confirm semantically coherent clustering and proper alignment between queries and retrieved chunks (Figures 2a,2b), while the distribution of cosine distances (Figure 2d) highlights consistent retrieval accuracy per query. Response generation latency is significantly improved: GPT-4o-mini completes responses in 2.57–3.28 s (Table 3), whereas local models such as

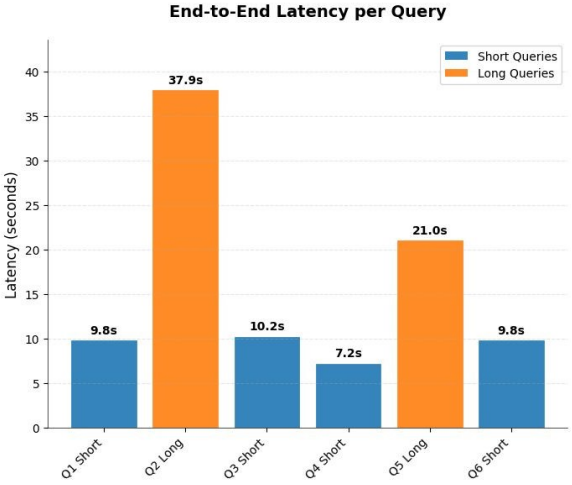
LLaMA and Mistral on a Raspberry Pi exceed 95 s. Switching from a Faiss Flat Index to an HNSW index reduces computational load while maintaining retrieval quality.



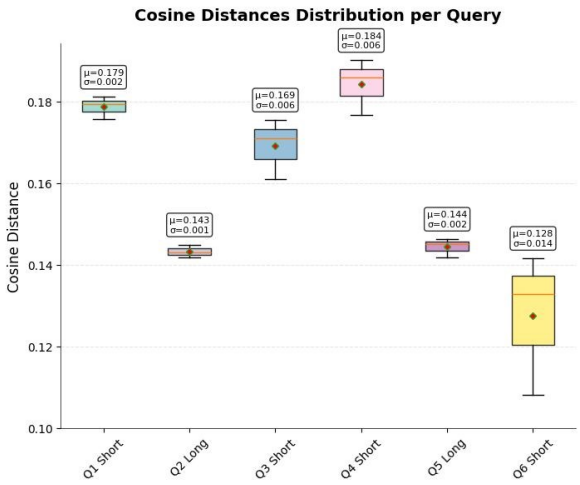
(a) UMAP projection of ChromaDB embeddings



(b) UMAP projection of queries and retrieved chunks



(c) End-to-end latency per query



(d) Cosine distances distribution per query

Figure 2: System evaluation visualizations showing (a) embedding distributions, (b) query–retrieval alignment, (c) end-to-end latency differences between short and long queries, and (d) cosine distance distributions highlighting retrieval accuracy per query.

Component	Performance Metric
STT	2.5–3.6 s
RAG + LLM	2.5–3.2 s
TTS	3.6–4.1 s
End-to-End	9.8–37 s

Table 3: SAFAA System Latency Metrics

4.4 TTS and End-to-End Latency

Text-to-speech (TTS) processing averages 3.6–4.0 s per query(Table 3). Combined with retrieval and STT, total end-to-end latency ranges from 7–10 s for short queries, up to 21–38 s for longer ones(Figure 2c). This is a major improvement over the baseline (≈90 s), demonstrating SAFAA's suitability for real-time interaction on edge devices.

4.5 Resource Usage

SAFAA maintains low resource consumption on edge hardware: CPU usage ≤11% and RAM ≤820 MB (Table 4), indicating feasibility for deployment on constrained devices.

Component	Performance Metric
CPU usage	≤11%
RAM usage	≤820 MB

Table 4: SAFAA System Resource Usage

4.6 Summary and Observations

Compared to the local baseline, SAFAA demonstrates:

- Accurate retrieval (cosine distance 0.17 vs 740)
- Much lower latency (≈10 s vs 90 s)
- Faster LLM token processing (26.5 tokens/s vs 3 tokens/s)
- Improved wake-word detection and STT performance

Challenges remain for long queries, noisy multilingual conditions, and extreme crowd noise. Deploying large LLMs on very constrained devices remains challenging due to hardware limitations and limited optimization time.

5 Conclusion

This project introduced SAFAA, an interactive, real-time voice assistant designed to support pilgrims during Hajj and Umrah through a RAG framework deployed on a Raspberry Pi 5. By combining STT transcription, semantic retrieval from authenticated Islamic sources, generative response synthesis, and TTS output, SAFAA provides accurate, trustworthy, and user-friendly guidance in highly crowded and multilingual environments. Evaluation results demonstrated reliable wake-word detection, acceptable accuracy under noisy conditions, and efficient resource usage on constrained edge hardware. Although end-to-end latency averaged around 9–10 seconds, the system consistently delivered contextually relevant answers and outperformed generic assistants such as Siri, Alexa, and Gemini in providing concise, speech-friendly responses tailored to pilgrimage needs.

References

[1] Apple. *Siri: The Intelligent Assistant*. https://support.apple.com/siri. 2023.

[2] Amazon. *Alexa Voice Service Documentation*. https://developer.amazon.com/en-US/alexa. 2023.

[3] Ministry of Hajj and Umrah. *Official Website*. https://www.haj.gov.sa/Home. 2024.

[4] Raspberry Pi Foundation. *Raspberry Pi 5 Documentation*. https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html. 2024.

[5] Kexin Wang et al. "E5: A new family of open-source embeddings". In: *arXiv preprint arXiv:2304.06720* (2023).

[6] OpenAI. *GPT-4o mini*. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. 2024.

[7] OpenAI. *GPT-4o: Next-Generation Audio Models for Speech-to-Text and Text-to-Speech*. https://openai.com/index/introducing-our-next-generation-audio-models/. 2025.

[8] David Scripka (dscripka). *openWakeWord*. https://github.com/dscripka/openWakeWord. An open-source wake-word detection library with pre-trained models for real-world performance. 2023.

[9] Albert Q. Jiang et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).

[10] Hugo Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models". In: *arXiv preprint arXiv:2307.09288* (2023).

[11] Ministry of Hajj and Umrah. https://www.haj.gov.sa/Guides. Includes comprehensive guides for pilgrims to facilitate performing Hajj and Umrah. 2025.

[12] Picovoice Inc. *Picovoice: On-Device Voice AI Local LLM Platform*. https://picovoice.ai/. Provides lightweight, accurate, private voice AI engines (ASR, TTS, wake-word, NLU) that run entirely on-device with zero latency. 2025.

[13] Vassil Panayotov et al. "Librispeech: An ASR corpus based on public domain audio books". In: *ICASSP*. 2015.