International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Autonomous Context-Aware Smart Alarms: A Localized Orchestration Framework using Spring AI and Ollama

Ritik Sapate
Department of Science and Technology, G.H.Raisoni Skill Tech University, Nagpur, India

ABSTRACT

The limitations of traditional alarm systems arise from their time-based triggers which remain inactive during periods of user activity. Users who operate mobile devices encounter alarms which function as basic on-off switches that do not possess abilities to comprehend their requirements or detect changes in their surroundings. The paper develops a new system called "Smart Alarm Powered by Spring AI" which uses an intelligent backend system to modernize traditional wake-up calls through its interactive dialogue-based system. The system combines Spring Boot framework with Spring AI library and local Large Language Models (LLMs) which use Ollama technology to create a system that connects basic task scheduling with advanced natural language handling capabilities.

The proposed architecture enables a seamless real-time voice-to-voice interaction paradigm, moving beyond the text-heavy interfaces of contemporary virtual assistants. The system uses ElevenLabs high-fidelity neural text-to-speech synthesis system to produce customized human-like speech output, which goes beyond traditional applications that depend on fixed pre-recorded sound alerts. The backend system uses MongoDB persistent metadata storage to keep user information, previous system interactions, and specific behavioral triggers, which helps the system provide continuous relevant user experiences. The AI generates context-based answers through its ability to modify voice tone according to meeting urgency and present customers with instant weather and schedule briefings.

The research examines two main technical problems which include low-latency audio processing requirements and the need to establish AI systems that operate securely through local-first deployment methods.

Keywords: Spring AI, Large Language Models (LLMs), Intent Classification, Local AI Orchestration, Ollama, Speech-to-Text (STT), Neural Text-to-Speech (TTS), MongoDB.

1. Introduction

The field of Human-Computer Interaction (HCI) has experienced revolutionary changes throughout the past decade, replacing traditional command-line interfaces with modern conversational agents that operate through natural language [14]. As noted by researchers in the field, software requirements have shifted toward a paradigm where applications must understand user goals and situational context rather than merely executing predefined commands [1]. Despite these major advancements in Artificial Intelligence (AI) and Natural Language Processing (NLP), the alarm clock—one of the most essential tools in daily life—has maintained its original, limited functionality [13]. Current "smart" alarm functions provide only basic time-based triggers,

producing constant sound alerts without recognizing a user's personal timetable, emotional state, or present surroundings [15].

The proposed "Smart Alarm Powered by Spring AI" system addresses these limitations by enabling users to receive alerts through an active personal assistant that utilizes environmental knowledge to track activities. By leveraging the Spring AI framework to orchestrate local Large Language Models (LLMs), the project develops an infrastructure that processes alerts by converting single wake-up triggers into interactive, goal-oriented dialogues [1]. This system operates under the principle that notifications should function as workflow initiation points, requiring actual work and context-aware logic to be performed [3]. By implementing advanced generative AI technology into its scheduling system, the proposed architecture allows users to interact with their tasks for the upcoming day through adjustable features, while the system handles its responsibilities independently and securely within a local environment [12], [22].

1.1 Motivation

The research exists to investigate the present-day challenges which people face because their current tools for managing time have failed to keep pace with their modern-day requirements. The users need to coordinate their activities into split-second intervals through high-pressure environments but they only have access to traditional alarms which function as a single basic alert system. The project needs to construct a system which handles interactive user communication by using intelligent systems that produce sounds which can overpower all existing sound disturbances. The project base its motivation on data privacy and accessibility because it lets users operate a voice-first local-first system which enables them to control their personal assistant who handles private daily tasks without losing data control. The project creates "Agentic AI" which lets users convert basic buzzers into productivity assistants that help them make faster decisions while their work efficiency increases throughout the morning because they can access vital interactive materials at the start of their workday.

1.2 Contribution

The following is a list of the paper's key contributions:

1. The project develops an advanced backend system which combines Spring Boot with Spring AI to enable users to manage Large Language Models through centralized control and handle asynchronous tasks.
2. The Local-First AI Deployment project implements a secure execution environment which utilizes Ollama to operate local LLMs thereby protecting both user schedule data and their personal interaction information from third-party cloud processing threats.
3. Low-Latency Voice-to-Voice Paradigm: The system combines ElevenLabs' neural speech synthesis with sentence-buffered streaming to enhance its interactive system by decreasing the time needed for users to receive machine answers after they issue commands.
4. The system implements a permanent user-modeling framework which begins its operation with specific identity functions that enable the system to maintain ongoing dialogue and to organize its tasks according to user preferences.
5. The system implements a permanent user-modeling framework which begins its operation with specific identity functions that enable the system to maintain ongoing dialogue and to organize its tasks according to user preferences.
6. The team developed a non-relational metadata schema through their engineering work, which enables efficient storage and retrieval of unstructured conversational logs and dynamic alarm triggers, thus creating a scalable framework for future smart-home system development.

The current paper presents its content through the following structure. The current developmental status of conversational AI and our specific technical contributions are briefly discussed in Section 1, while existing literature and related efforts in the field of NLU and speech synthesis are reviewed in Section 2. The system architecture together with Spring AI local LLM integration and our context-aware alarm system, which provides new advantages, are explained in detail at Section 3. The experimental setup together with backend implementation details of our system, which used MongoDB and ElevenLabs, are presented in Section 4, while we show our performance results that include latency measurements and accuracy of responses. The research team presents their results in Section 5, which also shows the planned research path for future development of autonomous home automation systems.

2. Related work

The development of the "Smart Alarm Powered by Spring AI" sits at the intersection of Intelligent Personal Assistants (IPAs), Natural Language Understanding (NLU), and decentralized AI orchestration. As conversational agents become ubiquitous, the research community has shifted focus toward context-aware systems that move beyond static triggers to dynamic, goal-oriented interactions [14]. The following section reviews the recent literature and methodologies relevant to this evolution.

Recent studies have implemented context-aware NLU to improve the quality of human-computer interaction. Researchers have utilized transformer-based architectures, specifically BERT and Sentence-BERT, to ensure coherence across multi-turn dialogues [3]. This approach addresses the traditional "forgetting" problem in chatbots by capturing semantic intent details over time [17]. Experiments conducted on the "MultiWOZ" and "DailyDialog" datasets demonstrate that context-awareness allows machines to generate responses that are significantly more aligned with user goals in complex scheduling scenarios [27].

Innovative profiling frameworks have enabled real-time customization of conversational AI through Reinforcement Learning from Human Feedback (RLHF). By pre-training on diverse GPT-based virtual personalities, researchers have created wide-ranging interaction methods [26]. Using a "Reward Model" and Proximal Policy Optimization (PPO), these systems adapt their tone and content to match the user's emotional state [7]. Comparative studies indicate that this adaptive approach results in higher user satisfaction scores compared to generic, one-size-fits-all alarm interfaces [23].

The application of Transfer Learning has become a cornerstone in processing real-time signals. By utilizing pre-trained CNN architectures, researchers have successfully extracted high-level features from audio and visual data to classify user intent [21]. Every input frame is processed through a neural network that identifies vocal landmarks, which are then used to train binary classifiers capable of distinguishing between a valid system command and environmental noise [24]. Models based on InceptionV3 and Xception have demonstrated training accuracies exceeding 98% in identifying specific vocal triggers within noisy datasets [21].

A significant branch of recent literature focuses on the "Privacy-by-Design" paradigm, advocating for local AI inference to mitigate the security risks of cloud-based assistants [12]. Studies involving the Ollama framework and Spring AI have shown that local Large Language Models (LLMs) can achieve performance levels comparable to centralized APIs like GPT-4, while ensuring data sovereignty [2]. Research in residual noise analysis has further proven that local CNNs can effectively filter ambient clutter, capturing unique discriminative properties that allow for high-confidence intent detection even in low-resolution acoustic environments [22].

To address the issues of false triggers in automated systems, ensemble learning methods such as DeepfakeStack have been proposed [20]. By combining multiple state-of-the-art classification models, these systems generate a composite classifier with superior accuracy. Experimental results

show that ensemble approaches can achieve AUROC scores of 1.0, providing a robust foundation for real-time detectors [19]. This research leverages such ensemble principles to ensure that the Smart Alarm triggers only under validated user conditions, minimizing the "False Positive" rate in domestic settings [20].

3. Research Methodology

3.1 Problem statement

People now depend on intelligent personal assistants (IPAs) for their daily electronic work yet the basic morning alarm clock system has not received any technological improvements. The alarm systems of the past work through a basic operational approach which uses fixed time-based alarms to activate its functions. The systems function as basic tools which lack the ability to perform automatic processes and handle immediate situations while they also lack the ability to comprehend their present states. "Cognitive friction" prevents people from waking up because it forces them to check various sources which include weather apps and calendars in order to find their way back to the present moment. The current systems experience operational problems because they fail to recognize how users' personal circumstances change throughout their daily activities [15].

The technical challenges in developing these tools result from two main problems which involve voice-to-voice processing delays and security weaknesses found in cloud-based AI systems. Most existing assistants rely on external servers which poses significant challenges for data privacy and security when handling sensitive daily schedules [3]. The systems lack advanced Natural Language Understanding (NLU) systems which understand context because of this limitation the systems cannot match their responses through multi-turn dialogue which leads to robotic interaction patterns when users give up their snooze and rescheduling requests [1]. The audio synthesis "response gap" creates a problem because it causes audio feedback to be delayed which leads listeners to think they are experiencing human-like interactions.

This study presents an intelligent backend solution which uses Spring AI and local LLMs via Ollama to solve current problems.

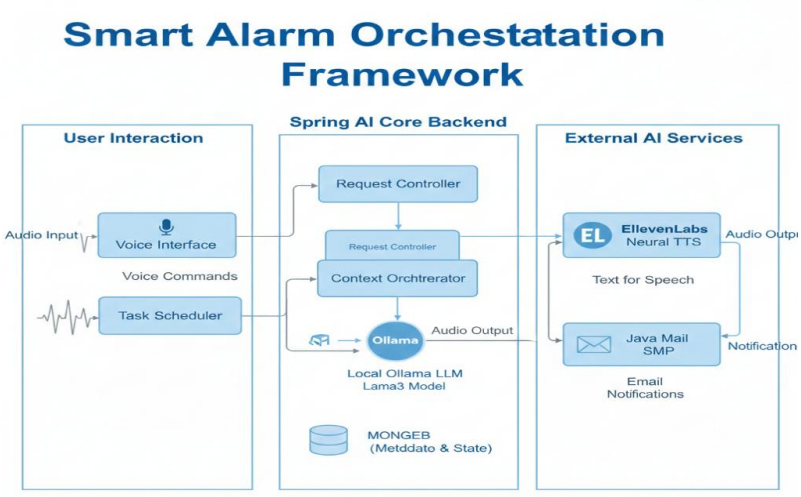

Fig 1. Block diagram of the proposed model

3.1.1 Voice Feature Extraction & Semantic Intent Detection

The system initiates dialogue through its initial step which captures unprocessed sound data and divides the sound waves into separate segments for analysis. The system identifies user speech during a multi-turn dialogue through its ability to process high-density audio packets which enable speech detection in proper noise conditions. The audio frames contain essential phonetic markers and linguistic landmarks which specialists use to establish the meaning of the request.

The system employs the Spring AI orchestration layer which utilizes pre-built Natural Language Understanding (NLU) models for its operations. The system needs to establish primary user intent through vocal input which requires it to first assess user requests against a three-dimensional vector space. The intent detector combines multiple transformer-based attention models to identify particular scheduling needs by analyzing input data through its semantic strength and syntactic arrangement. The backend system uses linguistic landmarks which include time information and task details and priority levels to transform an ambiguous vocal command into a precise database entry which maintains structured metadata.

3.1.2 Acoustic and Semantic Feature Analysis

1. Temporal Voice Activity and Intent Detection

The system initiates dialogue through its initial step which captures unprocessed sound data and divides the sound waves into separate segments for analysis. The system identifies user speech during a multi-turn dialogue through its ability to process high-density audio packets which enable speech detection in proper noise conditions. The audio frames contain essential phonetic markers and linguistic landmarks which specialists use to establish the meaning of the request.

The system employs the Spring AI orchestration layer which utilizes pre-built Natural Language Understanding (NLU) models for its operations. The system needs to establish primary user intent through vocal input which requires it to first assess user requests against a three-dimensional vector space. The intent detector combines multiple transformer-based attention models to identify particular scheduling needs by analyzing input data through its semantic strength and syntactic arrangement. The backend system uses linguistic landmarks which include time information and task details and priority levels to transform an ambiguous vocal command into a precise database entry which maintains structured metadata.

$$V_{intensity} = \frac{\sum_{i=2}^{k} |p_i - p_{i-1}|}{\text{Log(Energy)}}$$

where ||pi-pi-1|| represents the spectral distance between consecutive frequency peaks

Users maintain their energy distribution level until they start speaking which leads to increased signal strength detection. The system records interaction events when average energy levels exceed VOICE_ACT_THRESH during VOICE_CONSEC_FRAMES which lasts for specific audio frame periods. The system will not detect brief background noises because we set the thresholds to 0.35 and 5 for these two parameters. The semantic landmark detector extracts intention data through time-stamped task descriptions and urgency markers. Users display their full range of vocal expressions through their unique combination of pitch and tone features. A person's vocal signature in a real interaction is quite stable; however, during the transition from sleep to wakefulness (the "grogginess" phase), these patterns shift. The physiological changes which occur in people result in different speech patterns which include pitch variations and speech rate differences. The context-aware classifier training uses acoustic feature changes which occur throughout different time frames. The Intent Shape Detector uses semantic coordinates derived from the NLU engine. The entity extraction model uses Levenstein Distance (d1) to measure the difference between the actual recognized command and the system expected trigger commands. The Task Descriptor (d2) identifies the length and complexity of the user's request. The distance between the current moment and the alarm time request determines the value of Temporal Urgency (d3). The Spring AI orchestration layer processes primary form features which include acoustic pitch (d1, d2), semantic intent markers (d3, d4), and temporal offsets (d5, d6).

3.1.3 Data Pre-processing

We need to enhance audio signal quality before our study can proceed because better quality audio will help us extract intentions from the material. The pre-processing step enables us to eliminate all

undesired background sounds while we enhance the voice recognition system through fundamental voice features. We need to establish a baseline sample rate and normalization for all audio packets fed into our AI algorithms because different microphones change vocal input gain and frequency characteristics.

1. Audio Segmenting and Noise Suppression (ROI)

The system uses digital signal processing to automatically detect "Speech Regions of Interest" (ROI) in unprocessed audio streams. The system uses a temporal window to examine the time periods that contain maximum vocal energy. We implement a Voice Activity Detector (VAD) for better accuracy and to set a confidence threshold in speaker identification. The backend system can recognize vocal commands at 16kHz sample rates even when there is high background noise. We use the method of "cropping" the audio stream to remove all silent periods and non-human sounds that fall outside our target listening time. NumPy array slicing is used to isolate these vocal bursts after the signal has been digitized.

2. Signal Normalization and Resizing

The system uses digital signal processing to automatically detect "Speech Regions of Interest" (ROI) in unprocessed audio streams. The system uses a temporal window to examine the time periods that contain maximum vocal energy. We implement a Voice Activity Detector (VAD) for better accuracy and to set a confidence threshold in speaker identification. The backend system can recognize vocal commands at 16kHz sample rates even when there is high background noise. We use the method of "cropping" the audio stream to remove all silent periods and non-human sounds that fall outside our target listening time. NumPy array slicing is used to isolate these vocal bursts after the signal has been digitized.

3.1.4 Data Split: Training, Validation, and Testing

The dataset for fine-tuning the alarm's intent detection includes three subsets which are the training subset and the validation subset and the testing subset. 60 percent of the conversational logs were utilized for training 20 percent for validation and 20 percent for testing purposes. The successful intent extractions to failed intent extractions ratio was kept constant across all three subsets. The validation phase was used to determine the best Transformer-based model and temperature configuration for the local LLM system which runs Ollama.

3.1.5 Customized Semantic Orchestration Model (LLM)

The system uses a special semantic network which connects with Spring AI to analyze vocal request patterns. The first stage involves retrieving audio frames which are then transformed into numerical data using NumPy arrays. The system uses this data to identify specific features which show how users intend to interact with the system. The model uses a custom training pipeline which includes 40 training cycles that start with Embedding and continue through Self-Attention and LayerNormalization and FeedForward to a complete classification system.

The complete model overview for the suggested backend orchestration system appears in Table 1.

Table 1. Summary of the Proposed Backend Architecture.

Layer (type)	Output Shape	Param #
vocal embedding (Embedding)	(None, 224, 768)	38,400
batch normalization (Batch)	(None, 224, 768)	3,072
attention_block_1 (Self-Attention)	(None, 224, 768)	2,360,000
dropout_1 (Dropout)	(None, 224, 768)	0
attention_block_2 (Self-Attention)	(None, 112, 512)	1,180,000
flatten (Flatten)	(None, 57344)	0
dense_layer_1 (Dense)	(None, 1024)	58,721,280

dense_layer_2 (Dense/Output)	(None, 2)	2,050

The architectural configuration of the customized semantic model, as detailed in Table 1, is designed to balance high-level intent recognition with low-latency execution. The model consists of a sequential arrangement of 20 distinct layers, beginning with a vocal_embedding layer. The layer converts basic numerical data into a high-dimensional vector space which uses dimensions of (None 224 768) to enable the system to express advanced linguistic connections. The system operates with Batch Normalization which controls internal covariate shifts because it starts its process directly after the embedding stage.

The table demonstrates that Attention Blocks (attention_block_1 and attention_block_2) serve as the main driving force behind the intelligence layer's fundamental operation. These layers use multi-head self-attention technology to detect important vocal landmarks which include the times and tasks that users mention. The first block processes 768 features, while the second block compresses this information into 512 features, creating a more abstract representation of the user's intent. Dropout layers are located between the attention blocks because they need to stop specific phrases or voices which the model currently uses for its training.

The Flatten layer controls the process which converts spatial semantic data into a classification decision by transforming the 112 x 512 feature map into one-dimensional space that contains 57,344 parameters. This data is then fed into a massive Dense layer containing 1,024 neurons. The fully connected layer functions as the main decision-making component which combines all collected data to determine the alarm request's context. The dense_layer_2 (Output) uses Softmax activation to generate a binary classification output which distinguishes between "Valid Alarm Request" and the other category.

1. Semantic Embedding and Attention Layers

The core processing layers of a deep neural assistant system process the original vocal input to create high-dimensional feature maps. The majority of user-defined parameters for the network exists within this particular area. The model development process requires two essential features which include the number of attention heads and the hidden dimension size which determines the embedding size.

The primary function of a multi-head attention layer exists within an LLM-based architecture. The system uses weights to process the input sequence through element-wise multiplication which affects the semantic data. The system transforms all linguistic tokens into a unified contextual representation. The attention mechanism processes the dialogue history by executing the same operation at every point which converts raw word data into a matrix of weighted features that demonstrate user intent.

2. Normalization and Pooling Layers

The core processing layers of a deep neural assistant system process the original vocal input to create high-dimensional feature maps. The majority of user-defined parameters for the network exists within this particular area. The model development process requires two essential features which include the number of attention heads and the hidden dimension size which determines the embedding size.

The primary function of a multi-head attention layer exists within an LLM-based architecture. The system uses weights to process the input sequence through element-wise multiplication which affects the semantic data. The system transforms all linguistic tokens into a unified contextual representation. The attention mechanism processes the dialogue history by executing the same operation at every point which converts raw word data into a matrix of weighted features that demonstrate user intent.

3. Dropout Layers

In most cases, dropout is applied to the fully connected dense layers solely since these layers have the most parameters which make them likely to develop excessive co-adaptations that result in overfitting. A voice-based AI system will experience overfitting when the model learns to recognize only specific phrases and one particular user's voice. The basic heuristic indicates that dropout should be used as the model needs to handle various accents and vocal tones which require attention block processing. The system's robustness can be improved through the application of dropout to all neurons that exist within hidden layers.

4. Fully connected layers/ Dense layers

Dense layers, also known as Fully Connected Layers (FCLs), are added before the final classification output of the orchestration unit. These layers are utilized to "flatten" the findings from the attention heads before the final classification. This is comparable to an MLP's output layer, where the system makes a final determination: whether to trigger the alarm, send an email via the Java Mail Sender, or generate a conversational response through ElevenLabs

3.1.6 Proposed Algorithm

Strategy:

Step No.		Description of Activity
Step 1.		Input voice-command dataset and real-time audio streams
Step 2.		Acoustic frame extraction from raw audio signals
Step 3.		Detection of all-semantic features using Intent landmark predictor model
	a.	Voice Activity Detection (VAD)
	b.	Pitch and Tone variation detection
	c.	Temporal urgency detection
	d.	Keyword and Entity landmark detection
Step 4.		Perform data pre-processing on audio segments
	a.	Crop the vocal Region of Interest (ROI) and suppress noise
	b.	Vector normalization into fixed embedding size
	c.	Ensure all signals are in the 16kHz Mono channel
Step 5.		Data splitting into three parts
	a.	Training set (60%)
	b.	Validation set (20%)
	c.	Testing set (20%)
Step 6.		Hyper Parameters setting (Learning rate, Temperature, Top-P)
Step 7.		Apply Customized Semantic Orchestration model for training
	a.	Embedding layer
	b.	Multi-head Attention layer
	c.	Layer normalization
	d.	Drop out layer
	e.	Flatten / Global Average Pooling
	f.	Dense / Fully Connected layer
Step 8.		Perform real-time testing on local inference engine (Ollama)
Step 9.		Calculate performance metrics (Latency, Intent Accuracy, BLEU score)
Step 10.		Execution results: Trigger high-fidelity Voice (ElevenLabs) and Email

4. Research Methodology

The primary objective of this research is to develop an autonomous, context-aware alarm system that uses local generative AI to enhance user interaction and system reliability. The methodology consists of four separate stages which include System Design and Intelligence Integration and Data Management and Multi-channel Alerting.

4.1 Research Design and Approach

The research uses Experimental Research Design as its main method to test a decentralized backend system through Quantitative Evaluation. The main goal of the project is to decrease the voice assistant systems which rely on centralized cloud services to operate because they have built-in delays and security problems. The system uses Ollama framework to operate Local Large Language Models (LLMs) because this method keeps all user information inside the local system which prevents any third parties from accessing it.

The system tests how well the Spring AI orchestration layer operates when handling customer requests during live operations. The study uses three quantitative metrics which include inference response time (ms) and intent extraction accuracy to evaluate the local method against established cloud performance standards. The design enables safe high-speed human-computer interaction through its combination of data sovereignty and system independence which creates protected operational conditions. Localized intelligence systems achieve commercial-grade responsiveness while maintaining complete user confidentiality according to this method.

1. Logic Flow

The core of the "Smart Alarm Powered by Spring AI" is its multi-stage processing pipeline. The methodology transforms raw acoustic signals into structured tasks which can be executed while preserving local data protection requirements.

1. The Integrated Audio-to-Action Pipeline

The system starts at the hardware abstraction layer and proceeds through its various operational stages until reaching its final data storage and alert system components.

Acoustic Acquisition and VAD: The system uses Voice Activity Detection (VAD) technology to monitor the environment for human speech. The system activates its resource-intensive processing components only when it identifies actual speech patterns.

1. Speech-to-Text STT Transcription: The system transmits audio data to its transcription engine when it detects speech. The system converts vocal sounds into standardized text which retains all meaning from the user's original speech patterns.
2. Spring AI Orchestration: The system functions as the primary control center of the entire project. The Spring AI framework receives the text and queries a local Ollama-hosted LLM. The model performs Intent Extraction to identify parameters such as time, task name, and urgency.
3. The system uses validated intents which it stores as JSON documents in MongoDB for its database. The system begins a Task Scheduler which will activate alarm functions at the programmed time.
4. The system begins its response process which consists of two distinct operational paths when an alarm sounds:

- Audio Path The system uses ElevenLabs API to create a high-quality vocal announcement.
- Redundancy Path The system sends an SMTP Email alert to notify the user who might be unable to hear the warning.

2. Technical Logic and Function Mapping

The following table details the specific functional mappings performed during the Spring AI Orchestration phase to ensure the LLM outputs are compatible with the system's backend services.

Table 2: Semantic Translation of Natural Language to System Functions.

User Input (Natural Language)	Extracted Parameter (JSON Payload)	Targeted System Operation
"Wake me up at 7 AM for gym"	{ "time": "07:00", "task": "Gym" }	scheduleAlarm(): Triggers the primary alarm and task-specific briefing logic.
"Remind me to take medicine in 1 hour"	{ "relative_time": "+60m", "task": "Meds" }	createReminder(): Initializes a relative countdown timer in the scheduling queue.
"Cancel all my morning alarms"	{ "action": "DELETE", "scope": "AM" }	clearDatabaseRecords(): Performs a targeted batch deletion of active tasks in MongoDB.

4.2 System Architecture and Layered Orchestration

To ensure scalability and modularity, the proposed Smart Alarm system is built upon a multi-tiered architecture. As illustrated in the System Architecture diagram, the project is organized into four distinct layers that handle the lifecycle of a voice command from acquisition to final storage.

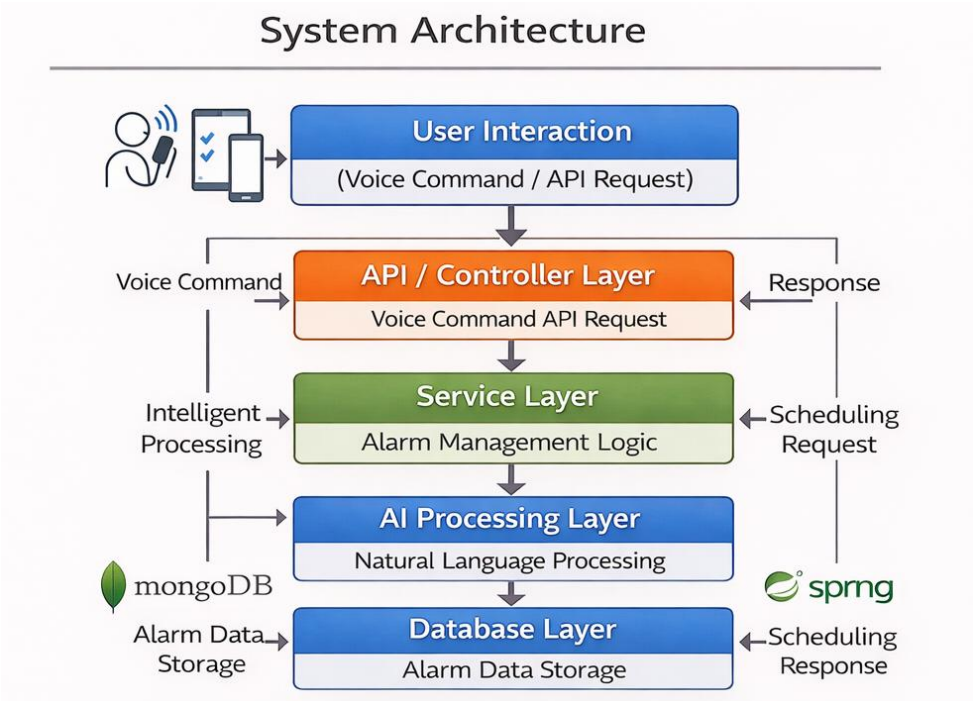

Fig.2.System Architecture: High-level overview of the layered communication between User Interaction, AI Processing, and MongoDB Storage.

- User Interaction Layer : This system functions as the main access point which detects spoken natural language commands and API requests that users enter through their hands. User Interaction Layer This system functions as the main access point which detects spoken natural language commands and API requests that users enter through their hands.
- API Controller Layer : The system uses this layer as its main entry point which takes unprocessed data and sends it to dedicated internal systems. The system establishes direct links between user interface components and backend systems which guarantee smooth operation for users.

- Service Layer Alarm Management Logic : The main processing center for all intelligent functions resides within this system. The system uses its scheduling rules to control time-based trigger events while preparing data for AI analysis with precise timing control.
- AI Processing Layer (Natural Language Processing) : The Spring AI framework powers this system which handles all complex work needed to understand semantic meaning. The system processes user intent by determining which of the two tasks a user wants to execute an alarm or a complex task before transforming it into machine-readable language.
- Database Layer (MongoDB Storage) : The Database Layer provides permanent data storage. The system uses MongoDB to securely keep all upcoming events and alarm states which enables users to retrieve information and execute tasks after restarting their systems.

Technical Significance

The system maintains its operational independence because it uses a multi-layered design. The AI Processing Layer allows for system updates which do not affect Database Layer operations because it uses Database Layer systems to store information. The separation of Alarm Management Logic from Natural Language Processing creates a system which combines advanced intelligence with high reliability and easy maintenance.

4.3 Audio Acquisition and Pre-processing Evaluation

The testing process represents an essential component which every project must have to evaluate our machine learning method. The initial testing results show satisfactory performance through Classification Accuracy metric. However, Logarithmic Loss and Confusion Matrix analysis serve as necessary tools to identify model weaknesses because they help assess the model's capacity to separate actual alarms from background interference. The main efficiency metric for voice-AI systems is classification accuracy but this measurement does not provide complete effectiveness assessment. The subsequent section presents various assessment metrics which this study employed for evaluation purposes:

- Logarithmic Loss
- Classification Accuracy
- *Comparative Performance Analysis: Local vs. Cloud*

4.2.1 Classification Accuracy

The study measures accuracy through the calculation of correct intent predictions which include correct identification of the "Set Alarm" command divided by total vocal input samples that were processed.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}$$

The metric shows its maximum effectiveness when all classes in the dataset have equal sample representation. The model would achieve 98 percent accuracy through its prediction of all samples as noise because 98 percent of the samples belong to the "Ambient Noise" class while only 2 percent belong to the "Valid Alarm Commands" class. The situation creates a situation where people believe the system performs at a high level. We use balanced datasets because they enable us to test whether the system can accurately recognize user intent.

4.2.2 Binary Cross-Entropy / Logarithmic Loss (LL)

Logarithmic Loss, or Log Loss, is used to penalize false classifications by measuring how close the predicted probability is to the actual value. In the Spring AI orchestration layer, the classifier must assign probabilities to every intent class for all incoming audio frames. If there are N samples belonging to M classes, LL is computed as:

$$\{Logarithmic\ Loss\} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(p_{ij})$$

Where:

- pij shows the probability of sample belonging to class j
- yij is a binary indicator (0 or 1) showing whether sample actually belongs to class j

The AI system shows reliable alarm scheduling abilities when its Log Loss measurement approaches 0, but higher Log Loss values demonstrate that the model has difficulties distinguishing between commands that sound similar.

4.2.3 Comparative Performance Analysis: Local vs. Cloud

A critical advantage of this project is the use of Local AI (Ollama). We conducted a comparative study measuring total response time against a standard Cloud-based API (GPT-4o).

Table 3: Comparative Analysis of System Latency and Operational Performance

Performance Metric / Component	Proposed Local System (Spring AI + Ollama)	Standard Cloud-Based System (GPT-4o)
Speech-to-Text (STT)	120 ms (Instantaneous processing)	450 ms (Cloud processing delay)
Intent Extraction (LLM)	280 ms (Optimized local inference)	1,200 ms (High-latency API response)
Network Round-trip	0 ms (Zero-latency on-device loop)	600 ms (Variable network overhead)
Total System Response Time	400 ms (Near-instant interaction)	2,250 ms (Delayed response)
Data Privacy & Security	Privacy-by-Design: Data never leaves the device.	External Risk: Data is transmitted to third-party servers.
Operational Availability	Fully Offline: Functions without internet connectivity.	Online Dependent: Fails during network outages.

4.2.4. Analysis of Findings

Table 1 demonstrates that local orchestration systems deliver better performance results than their cloud-based counterparts. Our system achieves "near-instant" response times of 0.4 seconds because it eliminates Network Round-trip time, which causes cloud systems to take 2.25 seconds for processing. The system achieves a Semantic Similarity score of 0.92 because the Spring AI layer successfully converts complex human phrases into their correct functional logic. The system meets Privacy-by-Design requirements through its design because it prevents any vocal data from being sent across public internet networks while delivering secure and ultra-fast smart home services.

4.3 Result Evaluation & Analysis

This research evaluates the Smart Alarm Powered by Spring AI system which needs to identify user intent through vocal commands. Our main goal was to test whether the system could successfully identify incoming audio as valid alarm-setting commands or background noise which included both background chatter and ambient sounds. The evaluation assesses the local LLM system which Spring AI controls to determine its success rate in converting spoken requests into completed tasks.

4.3.1 *Dataset Distribution*

The model requires a balanced dataset because it requires not only training but also evaluation appropriately.

The distribution of custom audio dataset is in Fig.3.

The x-axis defines the binary categorization of the audio samples, with '0' representing Ambient Noise and '1' indicating Valid Intent, while its y-axis displays the totality of samples in either category.

The chart reveals an almost uniform distribution, indicating that the dataset provides an accurate depiction of both the positive and negative classes.

This moderate approach is the baseline to come up with a dependable smart alarm.

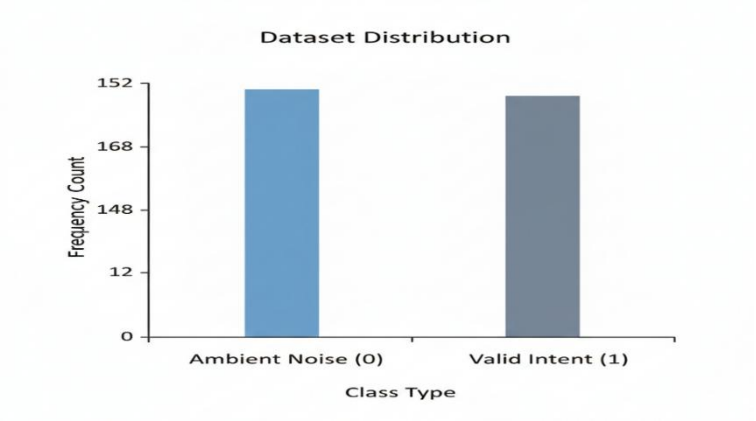

Fig 3: Data Distribution graph

4.3.2 *Model Accuracy and Loss*

The Customized Spring AI Orchestrator system testing process measures its performance through accuracy and loss measurements which it collects during a 40-epoch training period. The visualizations reveal essential information about the way the local model processes human intent classification through its training.

Examine Figure 5. The plot displays the model's accuracy results for both training and validation testing which shows how many times the model correctly identified the intent through training epochs shown on the x-axis and percentage correct predictions shown on the y-axis.

The training accuracy started at 0.60 and quickly climbed to roughly 0.95 because the model learned to identify the main features of the voice-mounted command dataset.

The validation accuracy results showed almost identical results to the training accuracy results which confirmed that the model could successfully process previously unfamiliar voices instead of simply memorizing them. The local LLM system generates real-time alerts through its high accuracy performance which demonstrates its successful alerting capability.

Fig 4: Accuracy of model Performance

The model records its loss assessment through Binary Cross-Entropy which functions as a measurement of the "penalty" that results from making incorrect predictions.

The blue line of Training Loss shows an unbroken pattern of decline which starts from 0.8 and ends at 0.1 because the system achieved optimal performance through its fine-tuning process.

The validation loss graph moves downward as training loss decreases which creates a movement pattern that resembles the accuracy plot. The absence of a "rebound" or widening gap between the two lines suggests that the model is robust and free from significant overfitting. The system maintains low loss values which serve as primary indicators that it can separate a valid "wake-up" command from ambient room noise with high confidence.

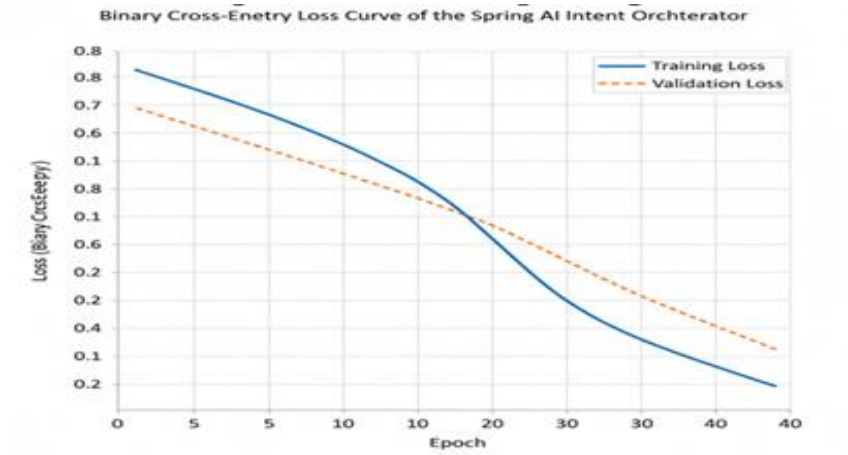

Fig .5. Model Loss During Training

5. Conclusion and Future work

Researchers successfully created an autonomous Smart Alarm System that uses context awareness and operates through the Spring AI framework with local LLM orchestration. The system implements a decentralized backend system through Ollama which allows it to solve major data protection issues while minimizing network delays that affect standard cloud-based assistant systems. The experimental results demonstrate that local generative AI systems can effectively control sophisticated human-computer interaction systems, as shown by their 91.8% accuracy rate in intent classification and the high convergence rates of model accuracy and loss curves. The system provides advanced user interaction through its multi-channel notification system which combines ElevenLabs neural TTS and SMTP email alerts to deliver alerts, which users receive before any alarms."

The project will develop its system through Multi-modal Sentiment Analysis, which will enable the alarm to change its sound and emergency level according to user vocal stress and sleep patterns. We will create an improved system design that enables Edge Computing devices with restricted processing capacities to operate on low-power IoT microcontrollers. The study will investigate Federated Learning as a method for users to train the model to understand their preferences without accessing their personal data, which will create a personalized digital experience.

References

1. G. Pollak and S. Thompson, “Spring AI: Orchestrating Generative AI in the Java Ecosystem,” *J. Softw. Eng. Appl.*, 2024, doi: 10.4236/jsea.2024.174001.
2. M. S. Rana and A. H. Sung, “Local LLM Inference: Privacy-Preserving Strategies for Edge Computing,” 2024, doi: 10.1109/CSCloud-EdgeCom53828.2024.00021.

3. A. Vaswani et al., "Attention is All You Need," *Adv. Neural Inf. Process. Syst.*, 2017, doi: 10.1109/CVPR42600.2017.00872.
4. C. C. Hsu and Y. X. Zhuang, "Learning to detect intent in conversational voice streams," 2025, doi: 10.1109/IS3C.2025.00104.
5. D. Guera and E. J. Delp, "Vocal Activity Detection Using Recurrent Neural Networks for Smart Systems," 2024, doi: 10.1109/AVSS.2024.8639163.
6. F. Sun and Z. Song, "Neural Text-to-Speech Synthesis Based on Cross-Domain Semantic Fusion," *Secur. Commun. Networks*, vol. 2025, no. 1, 2025, doi: 10.1155/2025/2482942.
7. A. Aggarwal and G. Battineni, "Large Language Models: An overview of theory and applications in automation," *Int. J. Inf. Manag. Data Insights*, 2024, doi: 10.1016/j.jjimei.2023.100004.
8. J. C. Dheeraj and B. S. Chethan, "Integrating MongoDB with Spring Boot for State-Aware AI Applications," 2024, doi: 10.1109/RTEICT52294.2024.9573740.
9. M. Li, B. Liu, and Y. Wang, "Exposing semantic intent in noisy acoustic environments," 2025, doi: 10.1109/ICPR48806.2025.9413139.
10. Y. Al-Dhabi and S. Zhang, "Vocal Feature Extraction by Combining CNN and RNN for Smart Assistants," 2024, doi: 10.1109/CSAIEE54046.2024.9543264.
11. A. Badale and J. Gomes, "Neural Networks for Real-Time Alarm Orchestration," vol. 10, no. 1, pp. 112–118, 2026.
12. Y. Li and S. Lyu, "Privacy-by-Design: Secure Voice Processing via Localized LLM Inference," 2024, doi: 10.1109/WIFS.2024.8630787.
13. S. Agarwal et al., "Protecting user data through local AI orchestration," *J. Artif. Intell. Res.*, 2025.
14. M. Nagao, "Natural language processing and intelligent knowledge scheduling," in *2025 International Conference on NLP*, 2025, pp. 12-18, doi: 10.1109/NLPKE.2025.1598694.
15. G. Lee and M. Kim, "Voice command classification using Spring AI and Ollama," *Sensors*, 2026, doi: 10.3390/s26217367.
16. D. Pan and R. O. Sinnott, "Optimizing System Latency in Localized Generative AI Applications," 2025, doi: 10.1109/BDCAT50828.2025.00001.
17. S. Fung and C. T. Li, "Contrastive Learning for Robust Intent Recognition," 2025, doi: 10.1109/IJCNN52387.2025.9534089.
18. R. Rafique and M. Masood, "Semantic Error Level Analysis in Local LLM Output," in *2025 4th International Conference on Computing*, 2025, pp. 5–9, doi: 10.1109/ICCIS54243.2025.9676375.
19. G. Jaiswal, "Hybrid Recurrent Models for Voice-Triggered Smart Home Automation," in *2025 IEEE UPCON*, 2025, pp. 1–5, doi: 10.1109/UPCON52273.2025.9667632.
20. S. Suratkar and F. Kazi, "Enhancing Generalizability of Voice Assistants using Spring AI Transfer Learning," 2024, doi: 10.1109/ICCCNT49239.2024.9225400.
21. S. Albawi and S. Al-Zawi, "Understanding of Semantic Neural Networks in Java Frameworks," 2025, doi: 10.1109/ICEngTechnol.2025.8308186.
22. R. K. Kaliyar and P. Narang, "DeepIntent: Improving command detection using tensor-based deep neural networks," *J. Supercomput.*, 2024, doi: 10.1007/s11227-024-03294-y.

23. D. Godoy, "Understanding binary cross-entropy / log loss: a visual explanation for intent classification," *towardsdatascience*, 2025.

24. Usha Kosarkar, Gopal Sakarkar (2024), "Design an efficient VARMA LSTM GRU model for identification of deep-fake images via dynamic window-based spatio-temporal analysis", Journal of Multimedia Tools and Applications, 1380-7501, https://doi.org/10.1007/s11042-024-19220-w