Voice-Based Agentic Ecommerce in Regional Indian Languages

Amirthalingam Rajasundar[a], Abhishek Kushary[a], Nitin Kumar[a], Vignesh S[a] and Preetam Mishra[a]

[a]Indian Institute of Science (IISc), Bangalore, India

Abstract. This paper presents a voice-based agentic e-commerce application designed to facilitate shopping interactions via WhatsApp using voice messages in regional Indian languages. The system integrates Sarvam AI for robust speech processing, LangGraph for defining a modular agent workflow, and Twilio for seamless WhatsApp integration. Key functionalities include product search, dynamic shopping cart management, and a streamlined checkout process, all navigable through natural language voice input. This project demonstrates a comprehensive solution for enhancing e-commerce accessibility in multilingual environments through advanced AI and conversational agent technologies.

1 Introduction

The proliferation of e-commerce has transformed global retail, yet a significant portion of the population in diverse linguistic regions, particularly in India, faces barriers due to language and digital literacy challenges. Traditional text-based interfaces often exclude users who prefer or require voice-based communication, especially in regional languages. To address this, we developed a novel voice-based agentic e-commerce application that leverages cutting-edge AI to enable intuitive shopping experiences via WhatsApp. Our system aims to bridge the accessibility gap by allowing users to interact with a shopping assistant using voice messages in their preferred regional Indian languages, thereby democratizing access to online retail.

1.1 Six-Step Problem-Solving Approach

Our project development followed a structured six-step problem-solving approach:

1. **Define the Problem:** We identified the critical need for accessible e-commerce solutions in India's multilingual landscape, specifically targeting users who prefer voice interaction in regional languages.
2. **Gather Information:** We researched existing conversational AI frameworks, speech processing APIs (like Sarvam AI), and messaging platform integrations (Twilio for WhatsApp) to understand the technological landscape and potential solutions.
3. **Identify Solutions / Brainstorm:** We explored various architectural patterns, including agentic workflows, and decided on a modular design leveraging LangGraph for its extensibility and suitability for complex conversational flows.
4. **Evaluate Solutions:** The chosen architecture, integrating Sarvam AI for multilingual speech, LangGraph for agent orchestration, and Twilio for WhatsApp, was selected based on its ability to meet the core requirements of voice-based interaction, language support, and platform accessibility.
5. **Implement the Solution:** We proceeded with the development, focusing on modular components for speech processing, product catalog, cart management, and response generation, as detailed in Section 3.
6. **Review and Iterate:** Continuous testing, debugging, and refinement were integral, particularly in handling audio conversions and API integrations. Future work, outlined in Section 6, represents the ongoing iteration of the project.

2 System Architecture

The Voice-Based Agentic E-commerce system is composed of several interconnected modules designed for robust and scalable performance:

1. **WhatsApp Integration:** Handles incoming and outgoing WhatsApp messages using the Twilio API.
2. **Speech Processing:** Transcribes voice messages to text using Sarvam AI's speech-to-text API, including necessary audio format conversions (e.g., Ogg to WAV/MP3).
3. **Product Catalog:** Manages product information and provides Retrieval-Augmented Generation (RAG)-based search functionality.
4. **Cart Management:** Tracks user shopping carts throughout the conversation.
5. **Payment Processing:** Handles checkout and payment via Razorpay integration.
6. **Response Generation:** Creates appropriate responses to user queries using RAG.
7. **Text-to-Speech:** Converts text responses back to audio using Sarvam AI.

The overall workflow is orchestrated by a modular LangGraph agent, ensuring extensibility and maintainability.

3 Implementation Details

Our project, named `IndicCommerce`, is structured modularly for clarity and maintainability. The main application entry point for the Agentic Ecommerce application is `app.py`. This file initializes all components and starts the Flask server. The key directories within `src/` include:

- `whatsapp/`: Contains `webhook.py` for receiving and processing messages.

- `speech_processing/`: Includes `processor.py` for managing speech-to-text conversion and language detection using Sarvam AI.
- `data/`: Includes `sample_products.py`, which is sample product data for the e-commerce application.These data are used to populate the vector store and provide product information.
- `llm/`: Contains `sarvam.py` for chat completion using Sarvam AI.
- `prompts/`: Includes `shopping_assistant.py` for defining a prompt.
- `utils/`: Consists of `ngrok.py` for securely expose the local development server to the internet and `vector_store.py` for vector store utilities using LangChain Chroma with OpenAI embeddings.
- `agents/`: Contains `ecom_agent.py`, which defines the LangGraph agent workflow and its nodes. This modular design facilitates the addition of new agent nodes by defining functions and updating the workflow.

The system relies on Python 3.9+, Docker for containerization, and external APIs from Sarvam AI, Twilio, and Razorpay. Environment variables for API keys are managed via a `.env` file. For local development, `docker-compose` is recommended, and `ngrok` is used to expose the local server for Twilio webhook configuration.

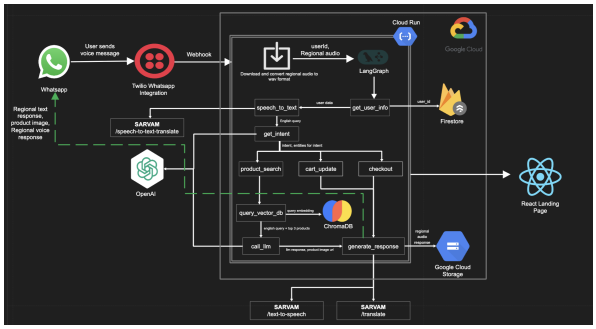

Figure 1. System Architecture Diagram.

4 User Flow and Functionality

The typical user interaction flow is as follows:

1. **User Sends Voice Message:** A user initiates interaction by sending a voice message in their preferred regional Indian language to the WhatsApp number.
2. **Speech-to-Text:** The system receives the audio via Twilio and uses Sarvam AI to transcribe it into text.
3. **Intent Recognition:** The transcribed text is processed to identify the user's intent, such as product search, adding to cart, or checkout.
4. **Product Search:** If a product search is requested, the RAG system queries the product catalog to find relevant items.
5. **Response Generation:** An appropriate text response is generated based on the identified intent and product search results.
6. **Text-to-Speech:** The text response is converted back into an audio message using Sarvam AI's text-to-speech capabilities, ensuring natural-sounding replies in the user's language.
7. **WhatsApp Response:** Both the text and audio responses are sent back to the user via WhatsApp.
8. **Conversation Continues:** The user can continue the conversation to refine search queries, add items to their cart, or ask for more details.
9. **Checkout:** Once the user is ready, a payment link is provided through Razorpay to complete the purchase.

This conversational flow effectively guides the user through the e-commerce journey using voice commands, simulating a personal shopping assistant.

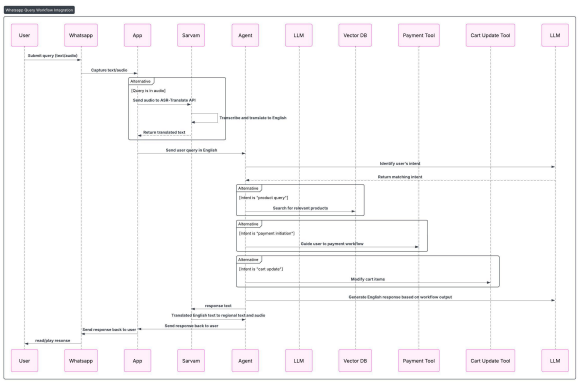

Figure 2. Sequence Diagram.

5 Evaluation Metrics and Results

To evaluate the system's performance, we considered several key metrics crucial for conversational AI and e-commerce applications.

- **RAG System Relevance:** With precision of product search results having .96 score, it ensure that the retrieved products are highly relevant to the user's query. Refer Table 1.

Table 1. Evaluation Summary for RAG Accuracy

Metric	Accuracy
Total queries evaluated:	218
Average Precision:	0.9679
Average Recall:	0.9679
Average F1-Score:	96.79

- **System Latency:** The time taken from user voice input to the delivery of the audio response, crucial for a smooth conversational experience. Refer Table 2.

Table 2. Evaluation of system latency

Trial	Latency
1 - Normal workflow	18s
2 - Normal workflow	30s
3 - Normal workflow	22s
4 - Error workflow	8s
Average latency:	20s

- **User Satisfaction:** Qualitative feedback on the naturalness of the conversation, ease of use, and overall shopping experience.
- **Intent Recognition Accuracy:** Both the model we tired gpt-4o-mini and sarvam exceled in intent identication, with the OpenAI model preforming slightly better. Refer Table 3 and 4.

Table 3. Comparison of Class-wise Metrics Between GPT-4o-mini and Sarvam Models

Intent Type	GPT-4o-mini					Sarvam				
	Acc.	Prec.	Rec.	F1	Sup.	Acc.	Prec.	Rec.	F1	Sup.
product_query	94.74%	94.74%	94.74%	94.74%	19	100.00%	90.48%	100.00%	95.00%	19
cart_update	92.86%	100.00%	92.86%	96.30%	14	78.57%	100.00%	78.57%	88.00%	14
summarize_cart	92.31%	100.00%	92.31%	96.00%	13	100.00%	100.00%	100.00%	100.00%	13
initiate_payment	61.54%	100.00%	61.54%	76.19%	13	76.92%	100.00%	76.92%	86.96%	13
general_info	100.00%	74.07%	100.00%	85.11%	20	90.00%	85.71%	90.00%	87.80%	20

Table 4. Comparison of Overall Performance Metrics Between Models

Metric	OpenAI (GPT-4o-mini)	Sarvam
Macro Precision	0.9480	0.8353
Macro Recall	0.9024	0.9092
Macro F1 Score	0.9246	0.8707

6 Discussion and Future Work

Our project successfully demonstrates the feasibility of a voice-based e-commerce agent in regional Indian languages, offering a significant step towards inclusive digital commerce. Challenges encountered primarily involved ensuring seamless audio format conversions and integrating disparate APIs. The modular design using LangGraph has proven highly effective, allowing for easy extension and addition of new functionalities.

6.1 Challenges Faced

During the development and deployment of the Voice-Based Agentic E-commerce system, we encountered several challenges:

- **WhatsApp Integration:** Directly using the WhatsApp Business API proved complex. While Twilio libraries helped abstract some of these complexities, accessing and downloading media URLs from WhatsApp messages presented authentication challenges. Additionally, there was a significant mismatch in audio formats, as WhatsApp uses Ogg while Sarvam AI requires WAV, which took time to resolve.
- **Vector Database (VectorDB):** Hosting a vector database separately or subscribing to managed VectorDB services was not desirable due to cost optimization goals. To address this, we developed a Python script to generate a persistence file for the vector database, which is then included directly within the Docker image. This approach avoids the need to regenerate embeddings every time the Cloud Run function spins up, significantly optimizing costs.
- **Conversation History:** Given that the Twilio integration only provides the current message and Cloud Run's stateless nature means no server-side session is maintained, we had to integrate with Firestore to persistently store user conversation data and manage session context.
- **LLM Response and Text-to-Speech (TTS):** A challenge arose when the Large Language Model (LLM) generated responses containing non-alphanumeric characters (e.g., asterisks for bolding, emojis). This caused the TTS engine to incorrectly spell out these characters, significantly reducing the quality and naturalness of the spoken response. Furthermore, the generic TTS language style did not always match the natural, conversational style of a human salesperson, which can detract from the user experience.
- **LangGraph Workflow:** Voice interaction is an inherently open-ended medium, unlike traditional UI-driven interactions. This characteristic posed challenges in comprehensively handling all possible user scenarios and conversational branches within the LangGraph workflow.

6.2 Future Work

For future work, we propose several enhancements.

- **Expanded Language Support:** Extend the speech processing module to support a wider array of regional Indian languages.
- **Custom Intent Recognition:** Further enhance the intent extraction capabilities within `speech_processing/processor.py` for more nuanced user queries.
- **New Product Categories:** Implement functionalities to easily add new product categories to the catalog without significant code changes.
- **Enhanced RAG System:** Improve the retrieval system in `data/sample_products.py` for more accurate and context-aware product recommendations.
- **Personalized User Experience:** Integrate features for user profiles and personalized recommendations based on past purchase history or preferences.
- **Advanced Error Handling and User Feedback:** Develop more sophisticated mechanisms for handling complex user queries, ambiguities, and providing clearer feedback.

7 Conclusion

We have developed a voice-based agentic e-commerce application that effectively bridges the language and digital literacy gap for users in regional Indian languages. By integrating Sarvam AI, LangGraph, and Twilio, our system provides an intuitive and accessible platform for product search, cart management, and checkout via WhatsApp voice messages. This project serves as a strong foundation for future advancements in conversational commerce and inclusive e-commerce solutions.

Contributions by Team Members

This section outlines the contributions of each team member to the Voice-Based Agentic E-commerce project. Our collaborative effort was essential in bringing this complex system to fruition. The following breakdown is based on a general understanding of roles in a deep learning project, as specific individual contributions were not detailed in the provided materials.

- **Amirthalingam Rajasundar:** Played the lead role in overall system architecture design, integration of core components (e.g., LangGraph agent workflow), and potentially overseeing the project's technical direction.
  - Commit: Initial Commit
  - Commit: Product Search LangGraph Workflow
  - Commit: Infrastructure as code and deployment pipeline
- **Abhishek Kushary:** Contributed significantly on identifying the product dataset and documentation of the project.
- **Nitin Kumar:** Focused on the backend logic, potentially developing the Product Catalog and Cart Management modules, including the RAG-based search functionality and database interactions.
- **Vignesh S:** Instrumental in the implementation of the agentic workflow, based on the identification of the indents. Evaluating different llm for indent identification.
  - Commit: Intent based agent workflow
  - Commit: evaluate open AI vs sarvam
- **Preetam Mishra:** Primarily responsible for the Response Generation and Text-to-Speech components, ensuring natural language understanding, appropriate response formulation, and high-quality audio output in regional languages.

All team members actively participated in brainstorming, debugging, testing, and documentation throughout the project lifecycle.

A Appendix A: How to try out the application

This appendix provides more detailed instructions for trying out the prod version of the application.

Important Note: Since we use Twilio Sandbox to optimize cost, we are restricted to 9 messages per day. So, you might start getting errors if the quota for the day has exceeded. If you face any errors while trying out the app or if you don't get a response, please reach out to us on teams and we can help with it.

- Go to https://www.amirth.dev/indic-commerce. The page describes the steps for trying out the application and also lists the product catalog. It also includes a demo video towards the end of the page.

- Click on 'Chat on Whatsapp' or 'Chat Now' button.
- It will take you to whatsapp and pre populate the join code in the chat text box.
- Hit send. You will get a confirmation saying that you are connected to the Twilio Sandbox.
- Now record a voice message in a regional Indian language (Eg: Hindi or Tamil) asking for a t-shirt. For example, you could say 'Show me a t-shirt'.

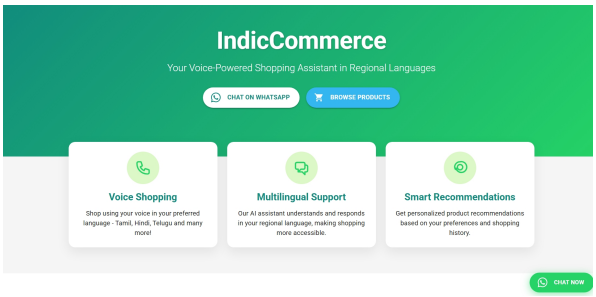

Figure 3. Indic commerce landing page

- Within a minute, you will get a message containing three sections. The message will include a text version as well as an audio version in the regional language that you spoke in. It will also include an image of the product.

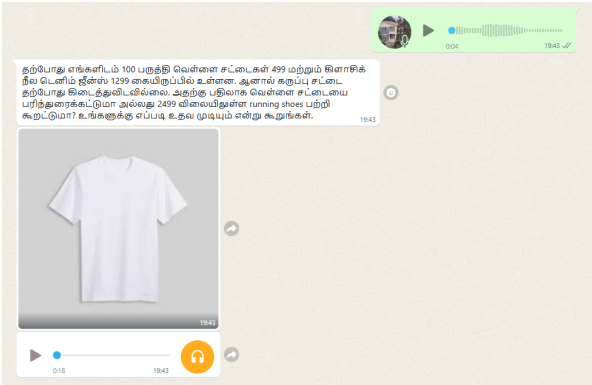

Figure 4. User Interaction in Whatsapp diagram.

- If the item is not available, it will recommend the next closest available product.
- Then, you can say 'Add to cart' in your regional language (exact phrase is not required) to add the last shown product to the cart or describe a different product if you are not satisfied with what is shown.
- Once the cart is loaded with the products you want, you can say 'That is all. How much is the total ?' (exact phrase is not required) in your regional language.
- Then, it will respond with the cart summary and a payment link.
- The steps will outlined above will complete one shopping session.

B Appendix B: Detailed Setup and Configuration

This appendix provides more detailed instructions for setting up and configuring the Voice-Based Agentic E-commerce system for development and deployment. You can also refer the README file at https://github.com/amirthalingamrajasundar/IndicCommerce for more details.

B.1 Prerequisites and Environment Variables

Ensure you have Python 3.9 or higher, Docker, and Docker Compose installed. Obtain API keys for Sarvam AI, Twilio, and Razorpay. The `.env` file in the project root directory must be populated with your credentials:

```
SARVAM_API_KEY=your_sarvam_api_key
TWILIO_ACCOUNT_SID=your_twilio_sid
TWILIO_AUTH_TOKEN=your_twilio_token
TWILIO_WHATSAPP_NUMBER=your_whatsapp_number
NGROK_AUTH_TOKEN=your_ngrok_auth_token
```

B.2 Running with Docker

For a quick setup, Docker Compose is recommended.

```
git clone https://github.com/amirthalingamrajasundar/IndicCommerce.git
cd IndicCommerce
# Edit .env with your keys
docker-compose up --build
```

The application will be accessible at `https://your-ngrok-url.ngrok.io/webhook`.

B.3 WhatsApp Business API Configuration

Configure your Twilio WhatsApp Sandbox webhook URL to point to your ngrok URL followed by `/webhook` (e.g., `https://your-ngrok-url.ngrok.io/webhook`).