Pizza Chatbot Project Report

1. Overview

Author: Suleiman Sultanov

Date: 1 August 2025

This project is a voice- and text-driven pizza ordering chatbot built using a local language model, to run this project you need to download: huggingface-cli download bartowski/Mistral-Nemo-Instruct-2407-GGUF Mistral-Nemo-Instruct-2407-Q5_K_M.gguf --local-dir . --local-dir-use-symlinks False, the following command included in the requirements.txt

(Mistral-Nemo-Instruct-2407) running with llama.cpp. It allows users to place orders using natural language

through either typing or speaking. The backend is built using FastAPI, and includes components for real-time

voice control, intent classification (ML + rule-based), dialog management, and prompt generation.

Technologies Used:

- Python, FastAPI

- Llama.cpp with Mistral-Nemo-Instruct-2407

- Whisper for speech-to-text

- Porcupine for wake-word detection

- scikit-learn for ML classification

- React, html, javascript, css

Following prompt has been used:

“You are an AI assistant for a pizza delivery service.
Here is the menu:
Pizzas: {', '.join(menu['pizzas'])}
Toppings: {', '.join(menu['toppings'])}
Extras: {', '.join(menu['extras'])}

Current Order: {order}
Customer said: "{user_input}"

Based on this, either extract pizza order details, delivery address, allergies, or ask follow-up questions to complete the order.
Always respond concisely and helpfully”

2. Technical Components

- Intent Classification: A hybrid system using TF-IDF + Random Forest (trained on labeled phrases) along

with rule-based checks against the menu ensures accurate interpretation of user messages.

- Dialog Manager: Maintains state and structured order data (pizzas, toppings, extras, notes, address).

Updates are triggered based on identified intent and LLM response.

- LLM Agent: Mistral-Nemo-Instruct model is prompted with structured order data and menu to generate

relevant and context-aware replies. It also classifies intent in fallback scenarios.

- Real-Time Voice Interface: Wake word detection via Porcupine ('Jarvis') and transcription via Whisper.

The bot responds using macOS speech synthesis and updates the order structure accordingly.

- FastAPI Backend: Exposes the `/message` endpoint for frontend integration and runs the voice assistant in a background thread.

-Web interface implemented as web widget for the website, interface includes input text as dialog with chatbot

Example of Final Order JSON:

```
{
   "pizzas": ["Margherita"],
   "toppings": ["Onions", "Chili"],
   "extras": ["Cola"],
   "notes": "very spicy, vegan",
   "address": "Reuterstraße 49, Berlin"
 }
```

3. Flow diagram

Dialog Flow Overview
Here's a simplified logic before turning it into a flowchart:
Start / Wake Word Detected (if voice)
⬇
User Input Received (text or voice)
⬇
Intent Classification
via combined_classify() (ML + rules + LLM fallback)
⬇
Prompt Generation & LLM Response
via generate_prompt() → get_response()
⬇
Dialog Manager Updates Order Based on Intent
"pizza", "topping", "extra", "note", "address"
⬇
Check if Order is Complete (has address)
If yes → confirmation
If no → ask for missing info
⬇
Bot Responds / Speaks Reply

Flow diagram: