Project Title: SupportBot AI

Subtitle: An Autonomous Voice-Enabled Support Ecosystem with Real-Time Knowledge Retrieval.

1. Project Overview:

SupportBot is an End-to-End Artificial Intelligence solution designed to automate corporate customer support. Unlike traditional bots that use rigid buttons, SupportBot understands Natural Human Speech (including low voices/whispers) and provides spoken responses in a natural Indian Accent.

It uses a RAG (Retrieval Augmented Generation) architecture, meaning it doesn't "guess" answers; it searches a verified company database (MongoDB) to provide 100% accurate information.

2. The Problem it Solves:

1. Human Scalability: Companies cannot hire enough humans to answer 24/7.
2. Robotic UX: Most bots sound like machines. SupportBot uses Neural TTS to sound human.
3. Data Accuracy: Standard AI (like ChatGPT) often "hallucinates." Our bot is restricted to official company CSV/JSON data.
4. Privacy: By using Ollama, the AI processing happens locally on the company's server, not in the cloud.

3. Technical Architecture

Explain that the project is built using a Microservices Architecture, separating the "Senses" from the "Brain."

Frontend: React.js (Vite) with Framer Motion for premium animations and Glassmorphism UI.

Backend: FastAPI (Python 3.13) handling high-speed asynchronous requests.

Hearing (STT): Faster-Whisper (OpenAI's Whisper optimized for local C++ execution).

The Brain (LLM): Ollama (Qwen2 / Llama 3.2) for logic and reasoning.

The Mouth (TTS): Edge-TTS (Microsoft Neural Voices) for a high-quality Indian Female accent.

MongoDB: Stores user tickets, chat history, and binary audio logs (GridFS).

Fuzzy Matching: RapidFuzz logic for 100% accurate data retrieval.

4. Step-by-Step Working Flow (The Journey)

Step 1: Entry & Authentication

The user enters the portal. They are greeted by a premium splash screen. To ensure every query is tracked, the user must enter their Name and Email. The system generates a Unique Ticket ID (e.g., TECH-A1B2C3) and stores it in MongoDB.

Step 2: Interaction Choice

The user chooses between Text Chat or Voice Chat.

Step 3: The Voice Loop (Processing)

1. Capture: The browser records the user's voice and sends a WebM blob to the API.
2. Normalization: The API uses FFmpeg to convert the audio into a clean 16kHz WAV file.
3. Transcription: Faster-Whisper converts the audio to text in milliseconds.
4. Retrieval (RAG): The system takes the user's text and performs a Fuzzy Search against the MongoDB Knowledge Base. It finds the exact answer provided by the company.
5. Reasoning: The text and the database evidence are sent to Ollama. The AI confirms the answer and removes any robotic phrasing.6. Synthesis: Edge-TTS converts the final answer into an MP3 file with an Indian accent.

Step 4: Output

The React UI receives the audio and Real-Time Captions. The audio plays automatically, and the captions appear on the screen for accessibility.

Step 5: Human Handover

If the user says keywords like "Human," "Manager," or "Operator," the AI recognizes the intent, provides the support number, and marks the ticket for Escalation in the Admin Panel.

5. Key Features to Highlight during Demo

Speed: Mention the optimization from 10 seconds down to 2-3 seconds using `tiny.en` models and `qwen:0.5b`.

Accuracy: Show how the bot gives the exact same answer that is in your CSV file.

Logging: Show your MongoDB Compass. Explain that every voice the user speaks is saved as a physical file in the database for the company to review later.

Self-Learning: Mention that an Admin can upload a new CSV, and the bot "learns" the new data instantly without restarting.

Workflow:

The 5-Step Working Flow of SupportBots

1. User Entry & Ticket Generation

Action: The user opens the app and enters their Name and Email.

Result: The system instantly generates a Unique Ticket ID and stores the user profile in MongoDB. This ensures every conversation is tracked from the start.

2. Multimodal Input (Voice or Chat)

Action: The user chooses how to communicate.

Chat: User types a query.

Voice: User speaks into the microphone (including low voices/whispers).

Result: The React frontend captures the input and sends it to the FastAPI Backend.

3. Intelligence Layer (The Brain)

Hearing (STT): OpenAI Whisper converts the user's voice into text.

Searching (RAG): The system performs a Fuzzy Search in the MongoDB Knowledge Base to find the exact company answer.

Thinking (LLM): Ollama processes the data to ensure the response is professional and accurate.

4. Human-Like Response (The Voice)

Speaking (TTS): The answer is converted into speech using Edge-TTS with a natural Indian Female Accent.

Captions: The AI text is displayed as Live Captions on the screen.

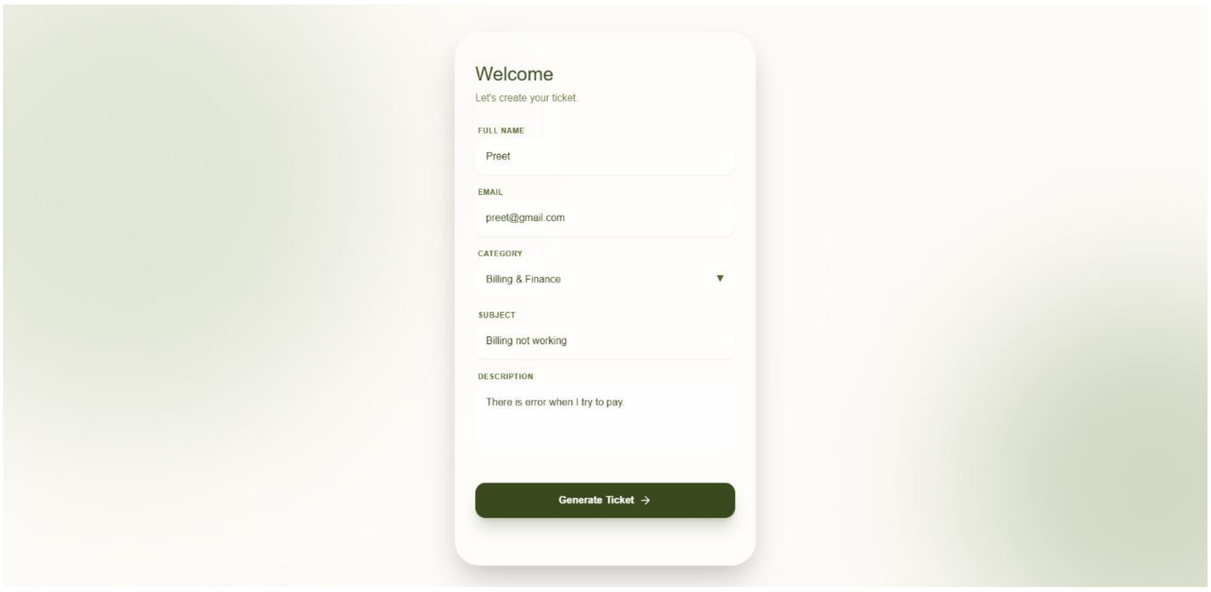
Result: The user hears the AI speak and sees the text simultaneously.

5. Permanent Logging & Handover

Storage: Every word spoken and every audio file generated is permanently saved in MongoDB for the company to review.

Escalation: If the AI cannot solve the issue, it automatically provides the Human Support Number to the user.

Images:



Welcome

Let's create your ticket.

FULL NAME

Preet

EMAIL

preet@gmail.com

CATEGORY

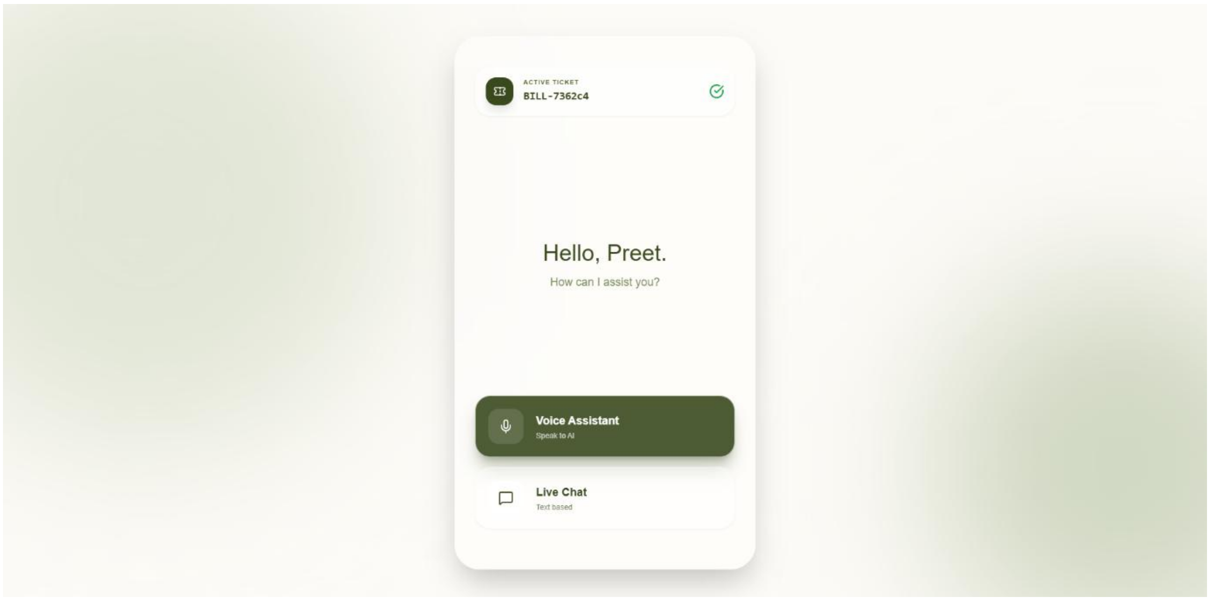
Billing & Finance ▼

SUBJECT

Billing not working

DESCRIPTION

There is error when I try to pay.

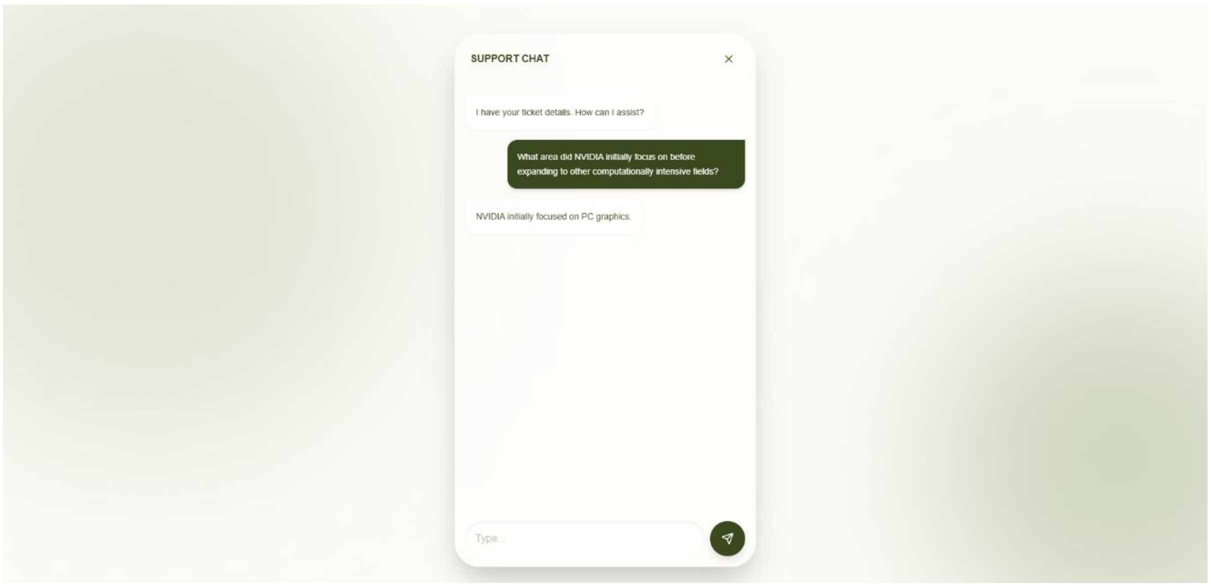


ACTIVE TICKET

BILL-7362c4

Hello, Preet.

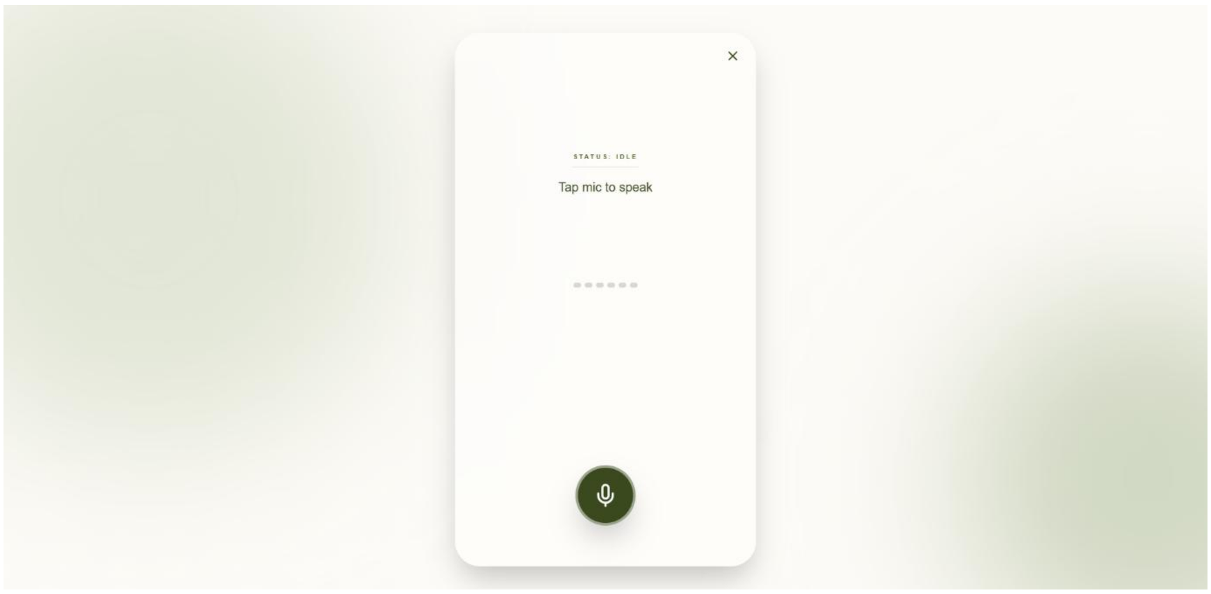
How can I assist you?

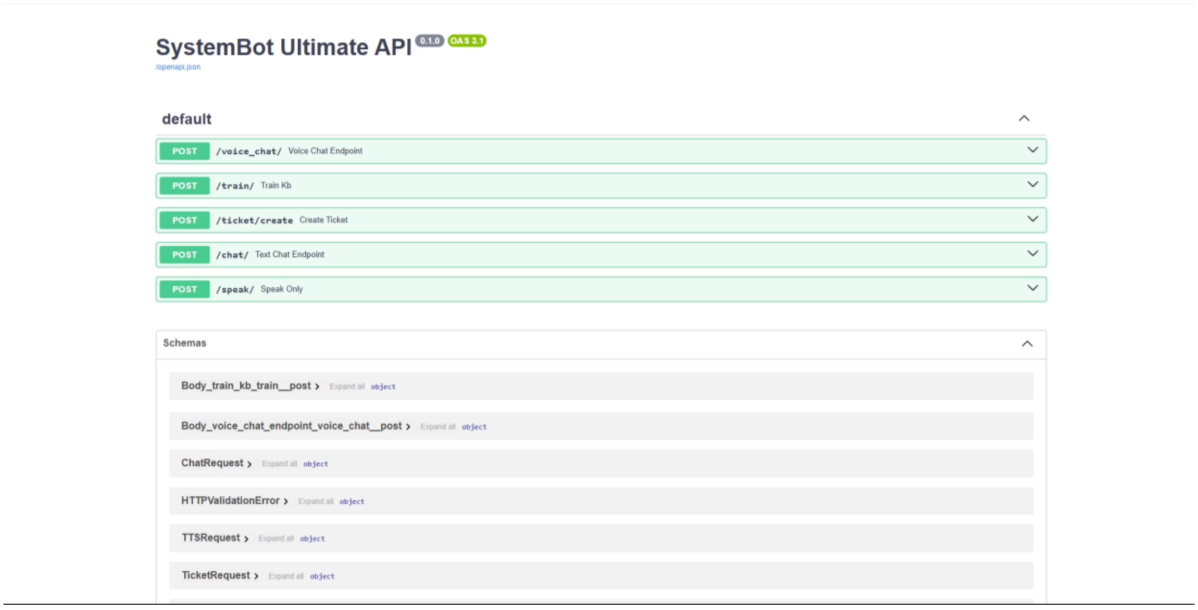

SUPPORT CHAT

I have your ticket details. How can I assist?

What area did NVIDIA initially focus on before expanding to other computationally intensive fields?

NVIDIA initially focused on PC graphics.




SystemBot Ultimate API 0.1.0 OAS 3.1

/openapi.json

default

POST `/voice_chat/` Voice Chat Endpoint

POST `/train/` Train Kb

POST `/ticket/create` Create Ticket

POST `/chat/` Text Chat Endpoint

POST `/speak/` Speak Only

Schemas

Body_train_kb_train__post › Expand all `object`

Body_voice_chat_endpoint_voice_chat__post › Expand all `object`

ChatRequest › Expand all `object`

HTTPValidationError › Expand all `object`

TTSRequest › Expand all `object`

TicketRequest › Expand all `object`