System Architecture Documentation

Project Title: Voice Tutor AI — Real-Time Agentic Chatbot

Problem Statement:

Build a real-time agentic chatbot with:

- LangGraph/Agno for orchestration
- STT, VAD, TTS modules
- LLM APIs (OpenAI, Anthropic)
- Scalable architecture, efficient memory/state management, and multi-user support

Submitted by: Akleema Fatima

Date: 10/07/2025

Team member - One

1. Solution Overview

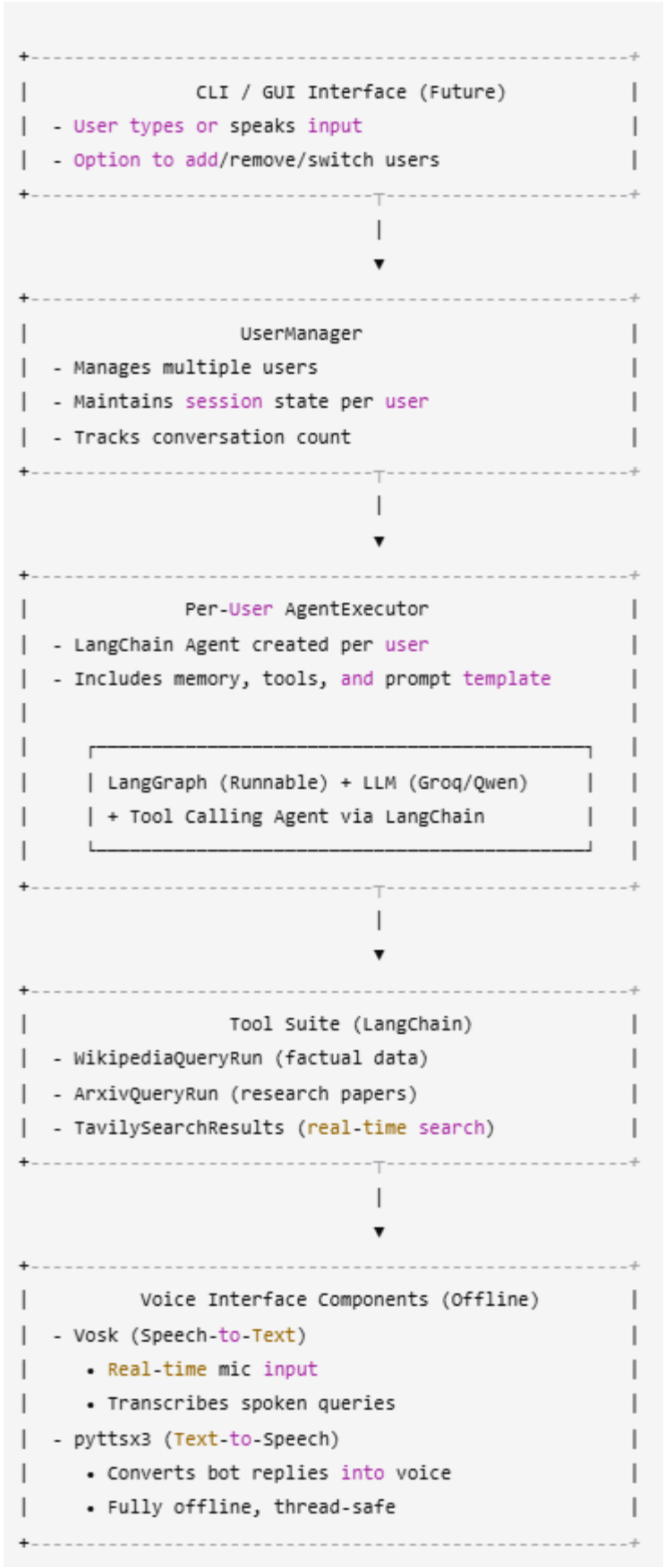
Our project, **Voice Tutor AI – Multi User Edition**, is a **real-time, agentic chatbot** built with LangChain's **LangGraph**, integrated with **offline speech capabilities**, and capable of supporting **multiple users simultaneously**. It is designed to be extensible, memory-efficient, and deployable in both CLI and GUI environments.

It empowers users to interact naturally via **voice** or **text**, with **per-user memory**, and leverages intelligent tool usage through **LangChain Agents**, orchestrated using **LangGraph** and **Runnable interfaces**.

2. How It Meets the Problem Statement

Requirement	Our Implementation
LangGraph / Agno	LangGraph used for orchestrating state transitions using `Runnable` agents
STT (Speech-to-Text)	Implemented using **Vosk** (offline, real-time microphone input)
VAD (Voice Activity Detection)	Simulated with Vosk's built-in real-time silence detection (low-latency)
TTS (Text-to-Speech)	Integrated using **pyttsx3** (offline, fast response)
LLM APIs	Using **Groq (Qwen)**; compatible with OpenAI or Anthropic (swappable backends)
Scalable Architecture	`UserManager` dynamically spawns isolated agent/memory instances per user
Efficient Memory / State Management	`ConversationBufferMemory` scoped per user; LangGraph's `Runnable` for orchestration
Multi-User Support	Fully supports `add`, `remove`, `switch`, and per-user context history

🏗️ 3. Architecture Diagram (Text View)


CLI / GUI Interface (Future)
- User types or speaks input
- Option to add/remove/switch users
UserManager
- Manages multiple users
- Maintains session state per user
- Tracks conversation count
Per-User AgentExecutor
- LangChain Agent created per user
- Includes memory, tools, and prompt template
LangGraph (Runnable) + LLM (Groq/Qwen)
+ Tool Calling Agent via LangChain
Tool Suite (LangChain)
- WikipediaQueryRun (factual data)
- ArxivQueryRun (research papers)
- TavilySearchResults (real-time search)
Voice Interface Components (Offline)
- Vosk (Speech-to-Text)
• Real-time mic input
• Transcribes spoken queries
- pyttsx3 (Text-to-Speech)
• Converts bot replies into voice
• Fully offline, thread-safe

4. System Components

🔹 A. LangGraph-Orchestrated Agent

- Built via `Runnable` interface to enable state-machine-based control
- Returns `chat_history` in state dict
- Easy to expand to complex conversational graphs

🔹 B. Voice Pipeline

- **STT**: Real-time transcription via Vosk with 16kHz sampling
- **VAD**: Real-time detection via waveform acceptance (`AcceptWaveform`)
- **TTS**: pyttsx3 reads cleaned LLM output asynchronously in a thread

🔹 C. LLM Integration

- **Groq (`qwen-qwq-32b`)** as current LLM (faster inference)
- Easily swappable with OpenAI (`gpt-4o`) or Anthropic (`claude-3`) via LangChain interface
- Prompt-template driven behavior, including memory injection

🔹 D. Multi-User Management

- Each user has:
    - A unique `ConversationBufferMemory`
    - Independent `AgentExecutor`
    - Usage tracking (conversation count)
- Commands:

- `add <user>`, `switch <user>`, `remove <user>`, `users`

5. Tools Used

Category	Tool/Library	Role
Orchestration	LangGraph, Runnable	Agent flow
LLM + Tools	LangChain + ChatGroq	Q&A, tool calls
STT	Vosk	Real-time voice-to-text
TTS	pyttsx3	Text-to-speech
Web Retrieval Tools	Wikipedia, Arxiv, Tavily	Enriched information
Memory Management	ConversationBufferMemory	Per-user context
Config & Env	python-dotenv	API key injection

6. Conversation Lifecycle

1. **User Login**
    - CLI prompts for user switch, creation, or input mode
2. **Voice/Text Input**
    - `Vosk` records and transcribes OR user types manually
3. **Agent Execution**
    - LangChain agent processes prompt using memory and tool APIs
4. **Response Handling**
    - Response is parsed, cleaned, and truncated
5. **Speech Output**

- Final response is played aloud using `pyttsx3`

7. Security and Offline Support

Aspect	Design Approach
Audio Privacy	No external STT/TTS API; Vosk and pyttsx3 run locally
API Safety	`.env` file used to store all keys
Multi-user Isolation	Session data and memory not shared

8. Deployment Instructions

Requirements:

- Python 3.11+
- `vosk`, `sounddevice`, `langchain`, `pyttsx3`, `groq`, `python-dotenv`
- Vosk model (downloaded and stored locally)

Run Command:

researchbot.ipynb

9. Future Enhancements

Feature	Description
Streamlit / Web UI	Friendly multi-user GUI
ChromaDB + RAG	Offline educational data retrieval
WebRTC Mic Input	Browser-based STT
LangGraph Full Graph	Complex, reactive state transitions
Whisper / Silero STT	Optional alternate STT backend
Claude / GPT-4o LLM	Swap-in for higher quality responses

10. Conclusion

This system provides a **real-time, modular, agentic AI assistant** that supports **voice-first learning** with **multi-user memory**, **tool augmentation**, and **scalable orchestration**. It aligns closely with the hackathon problem statement and offers a strong base for further development into a full-fledged AI tutor or customer assistant.