AgenTeam Content:

AgenTeam – AI-Powered Communication Agents Platform
Full-Stack System Design, Backend, Frontend, Real-Time Infrastructure & AI Integration

Project Overview

AgenTeam is a full-stack platform designed to create, manage, and operate AI-powered communication agents that handle real phone calls, conversations, and business workflows automatically.
The system enables businesses to deploy intelligent agents for sales, support, appointment scheduling, attendance confirmations, and information delivery — without human operators.
This project demonstrates end-to-end product development, including authentication, dashboards, real-time communication, AI orchestration, billing logic, and administrative tooling.

Core Features & Capabilities

1. User Authentication & Authorization
Secure user authentication system
Support for email/password login and OAuth (Google Sign-In)
Token-based authentication (JWT)
Role-based access control
Company-level authorization (users can belong to and manage multiple companies)
Protected routes on both frontend and backend

2. Company & User Management

Users can create and manage multiple companies
Each company has its own:
Agents
Call jobs
Usage statistics
Credits balance
Clear separation of data between companies (multi-tenant architecture)

3. AI Agent System

The platform includes a highly configurable AI agent system designed for real-world voice interactions and business automation.
Throughout development, I worked extensively with a wide range of AI model types — including Hugging Face open-source models, local and self-hosted models (via Ollama), and cloud-based API models — testing and comparing them across multiple scenarios to achieve the best balance of latency, accuracy, naturalness, and cost.
This iterative experimentation allowed me to fine-tune prompts, context handling, and response strategies for production-grade performance in live phone calls.
Each AI agent is fully configurable and supports:
Clearly defined roles (sales, customer support, appointment scheduling, attendance confirmation, and more)
Behavioral rules and response constraints tailored to business goals
Conversation guidelines optimized specifically for natural, human-like phone speech
Dynamic reasoning without rigid or pre-written scripts
The agent runtime is capable of advanced conversational control, including:
Turn-based conversation flow management
Handling user interruptions during speech
Clarification and disambiguation logic
Context-aware decision making across the entire call lifecycle
In addition, agents support MCP-style actions (tool execution), enabling them to actively interact with external systems and backend services, such as:
Managing and booking calendar appointments
Updating and modifying database fields in real time
Recording structured call outcomes (e.g., confirmations, counts, notes)
Triggering CRM-like actions and custom business workflows
This architecture allows AI agents to move beyond passive conversation and function as autonomous, action-capable systems that both communicate naturally and execute meaningful business operations.

4. Real-Time Voice Calls & Streaming
Full real-time phone call handling
Integration with telephony providers (e.g., Twilio-style architecture)
Live WebSocket streaming between:
Phone audio
Speech-to-Text (STT)
AI inference
Text-to-Speech (TTS)
Voice Activity Detection (VAD) and silence handling
Accurate call state management (start, interruption, resume, end)
Call recording support and post-call processing

5. Speech & AI Pipeline
Speech-to-Text (STT) support:
Whisper-based models
Streaming and batch transcription
Text-to-Speech (TTS) support:
Multiple providers (Azure / Google / ElevenLabs / OpenAI-style APIs)
Optimized phrasing for natural spoken output
AI inference orchestration:
System prompts
Context compilation per call
Memory handling
Tool invocation rules

6. Call Jobs & Automation
Call jobs system for:
Bulk calling
Scheduled calls
Retry logic
Each call job tracks:
Status
Results
Agent decisions
Structured outcomes (yes/no/maybe, counts, notes, etc.)
Automatic post-call summarization and data updates

7. User Dashboard (Frontend App)
Modern dashboard UI for users to:
View companies and agents
Create and edit agents
Launch call jobs
Monitor live and historical calls
View usage statistics
Responsive design
Internationalization support (LTR / RTL, including Hebrew)
Clear UX separation between business logic and presentation

8. Billing & Credit System
Token-based usage system
Custom billing logic:
Fixed cost per call
Credit consumption tracking

9. Admin Dashboard
Internal admin interface for:
Monitoring users and companies
Viewing system-wide usage
Managing credits and limits

Reviewing call logs and recordings
Tools for debugging, auditing, and platform oversight

10. App Review & Quality Controls
Call review system:
Transcripts
Audio recordings
AI decision traces
Error handling and fallback mechanisms
Rate limiting and abuse prevention
Logging and monitoring hooks

Technical Architecture

Backend
Node.js / TypeScript
Modular service-based architecture
REST APIs + WebSocket real-time servers
Authentication middleware
Rate limiting and security layers
Clear separation of:
Runtime logic
AI orchestration
Call lifecycle management
Shared types package for frontend/backend consistency

Frontend
React + TypeScript
Component-based UI architecture
Dashboard-oriented design
State management for auth, companies, agents, and calls
Localization (i18n) with RTL/LTR support

Infrastructure Experiments & Engineering Challenges
During development, I explored running large language models locally as part of the AI agent pipeline. While this approach provided strong control and privacy, it introduced challenges around GPU availability, memory constraints, cold starts, and inference latency when scaling beyond local environments.
To address these limitations, I experimented with deploying LLM workloads on cloud-based GPU instances, running models inside containerized pods backed by dedicated graphics cards. This allowed me to compare local inference against cloud-hosted GPU execution in terms of performance, cost, reliability, and operational complexity.
These experiments informed architectural decisions around:
When to rely on self-hosted or local models versus managed API-based models
How to design the agent runtime to remain model-agnostic
Balancing inference latency with scalability and operational overhead
This process reinforced a flexible, provider-agnostic architecture that allows the system to evolve as model capabilities, infrastructure costs, and deployment requirements change.

Project Intent & Value
AgenTeam was built to solve real operational problems in businesses that rely on phone communication — replacing repetitive human calls with intelligent, reliable AI agents.
From a technical perspective, this project showcases:
Full-stack ownership
Real-time systems design
AI integration in production workflows
Complex state management
Scalable, multi-tenant architecture