The Project ConvAi

**🔍 1. Vision and Scope**

**Purpose**: Automate human-to-human interactions in HR and healthcare scenarios using voice-based conversational AI that can extract structured data and intelligently respond or evaluate.

**Problems Solved**:

* Replaces manual form-filling with intelligent, conversational voice input.
* Enhances efficiency in HR interviews and hospital patient intake.
* Reduces human bias and inconsistencies.

**Target Domains**: Human Resources (interviews), Healthcare (patient intake forms).

**Out of Scope** (for now):

* Deep medical diagnostics
* Non-verbal communication (e.g., facial recognition)

**🧑‍💼 2. Target Users and Use Cases**

**Users**:

* HR professionals
* Doctors/nurses/junior staff
* Job seekers
* Patients

**Use Cases**:

1. HR Interview:
   * AI greets the candidate.
   * Asks “Tell me about yourself.”
   * Extracts key data (name, background, experience, skills).
   * Rates and evaluates based on completeness and relevance.
2. Hospital Intake:
   * AI greets the patient.
   * Asks "What happened?"
   * Captures symptoms, history, identifiers.
   * Detects missing fields and asks context-specific follow-ups.

**🧠 3. Functionality Requirements**

* Conversational voice-based interface.
* Speech-to-text transcription.
* NLP engine to extract structured data.
* Form population system.
* Evaluation module (HR-specific).
* Dynamic question generator.

**🗣️ 4. Conversational Design**

* **Tone**: Empathetic, calm, professional.
* **Style**: Natural, human-like, not chatbot-like.
* **Logic**:
  + Initial open-ended question → analyze response → detect gaps → generate follow-up.
* Handle unclear responses with polite re-asks.
* Maximum conversation time: 1–2 minutes.

**🏗️ 5. Technical Architecture**

**Components**:

* Voice Input Handler
* Speech-to-Text Engine (e.g., Whisper)
* NLP Extractor (e.g., OpenAI or custom model)
* Dynamic Form Filler
* Evaluation Logic (HR Module)
* Frontend: Web or Mobile UI
* Backend: Node/Python Server + Database

**Processing**: Real-time **Deployment**: Cloud-based, API accessible

**🛠️ 6. Tech Stack Choices**

* **Speech Recognition**: OpenAI Whisper, Google STT
* **NLP**: OpenAI GPT-4 Turbo, spaCy for custom rules
* **Frontend**: React or Flutter
* **Backend**: FastAPI / Node.js
* **Database**: PostgreSQL / MongoDB

**📑 7. Data Considerations**

* Need voice/text samples for training/refining.
* Store anonymized transcripts and form data.
* HIPAA/GDPR compliance in healthcare.
* Consent required before recording.

**📏 8. Evaluation and Accuracy**

* **HR**:
  + Completeness of response
  + Relevance of details
  + Confidence score
* **Healthcare**:
  + % of fields auto-filled correctly
  + Time taken to complete form

**🔁 9. Interaction Flow Mapping**

**HR Flow**:

* Greet → Ask intro → Extract info → Score → Thank you

**Medical Flow**:

* Greet → Ask open-ended Q → Fill partial → Ask targeted Qs → Finalize form

**🔧 10. Error Handling and Fallbacks**

* Retry on unclear input ("I didn't catch that. Could you repeat?")
* Ask clarifying question if vague
* Log failures for improvement

**🧪 11. Testing and Iteration**

* Pilot tests with simulated users
* Manual review of form filling
* A/B testing different question orders
* User feedback collection loop

**🔐 12. Security and Ethics**

* End-to-end encryption for data storage/transmission
* Avoid leading questions or biased prompts
* Clear opt-in and data usage disclosure
* Delete recordings post-processing (optional)

**📊 13. Analytics and Monitoring**

* Form completion rate
* Average conversation duration
* User satisfaction rating
* Error rate in extraction
* Dashboard for usage metrics

**📁 14. Documentation Components**

* Project Overview
* Use Case Scenarios
* Technical Blueprint
* Sample Dialogues
* Evaluation Rubrics
* Compliance Checklist
* Development Timeline

**Guide – A to Z Setup**

**Step 1: Define Use Case and Scope**

* Choose target (HR or healthcare first)
* Identify exact form fields to be captured
* Determine evaluation criteria (if HR)

**Step 2: Design Form Templates**

* Collect common intake forms/interview rubrics
* Map mandatory and optional fields
* Identify critical fields for prioritization

**Step 3: Set Up Development Environment**

* Install Python, Node.js, VS Code
* Create GitHub repo and enable version control
* Setup cloud environment (e.g., AWS, GCP)

**Step 4: Speech-to-Text Module**

* Use Whisper or Google STT
* Preprocess audio input
* Output clean, punctuated transcripts

**Step 5: NLP Extraction**

* Use GPT-4 or spaCy to extract entities from text
* Match values to form fields
* Handle synonyms and common language variations

**Step 6: Build Dynamic Form Filler**

* Create field mapping system
* Populate forms dynamically as conversation progresses

**Step 7: Conversation Engine**

* Initial question bank → Predefined + Follow-ups
* Use prompt chaining or FSM for follow-up logic
* Track what’s missing and generate relevant queries

**Step 8: Evaluation Logic (HR)**

* Define scoring model: relevance, completeness, confidence
* Use rule-based + ML approach to assign rating

**Step 9: Web App Frontend**

* Build with React/Flutter
* Microphone input → text display → form preview
* UX: Friendly, modern, accessible

**Step 10: Backend Services**

* Use FastAPI / Node.js for APIs
* Store form data in PostgreSQL or MongoDB
* Auth, logging, and failover systems

**Step 11: Consent & Security**

* Checkbox for user consent (MVP)
* Encrypt data in transit + storage

**Step 12: Deployment**

* Host on Vercel/Render/Netlify (frontend)
* Use AWS/GCP backend and DB
* Setup CI/CD pipelines

**Step 13: Testing**

* Simulate interviews/intake sessions
* Test NLP output mapping and accuracy
* Validate form field population and scoring
* TBD: Extreme stress tests and fail-safe scenarios

**Step 14: Feedback Loop**

* Build dashboard to monitor logs/errors
* Collect feedback from users
* Improve AI prompts and scoring models

**📦 Final Output**

A web-based AI assistant that:

* Uses voice input from users
* Converses naturally to extract details
* Fills structured forms in real-time
* Evaluates candidates (HR) or captures health data (medical)
* Continuously improves from real-world usage