# Technical Document

## Voice Assistant Application

**1. Introduction**

This document provides a step-by-step guide to building a **voice assistant application** that converts spoken commands into text, processes them using AI, and returns responses. Designed for beginners, it explains both frontend (Angular) and backend (Spring Boot) components in simple terms.

**2. System Architecture**

The application has two main parts:

1. **Frontend (Angular)**: Handles audio recording, resampling, and user interface.
2. **Backend (Spring Boot)**: Processes audio using Google Cloud APIs and generates AI responses.

┌───────────────┐ Audio (WAV) ┌───────────────┐

│ Angular │ ----------------------> │ Spring Boot │

│ (Frontend) │ <---------------------- │ (Backend) │

└───────────────┘ Text Response └───────────────┘

▲

│

▼

┌──────────────────┐

│ Google Cloud APIs│

│ - Speech-to-Text │

│ - Gemini AI │

└──────────────────┘

A diagram of a software system

AI-generated content may be incorrect.

**3. Frontend Setup (Angular)**

**3.1 Prerequisites**

* Install Node.js and npm: [Node.js Download](https://nodejs.org/)
* Install Angular CLI:

bash

npm install -g @angular/cli

**3.2 Key Components**

1. **Audio Recording**:
   * Uses browser microphone access.
   * Default sample rate: 44.1kHz.
2. **Resampling**:
   * Converts audio to 16kHz using Resampler.js.
3. **WAV Conversion**:
   * Adds WAV file headers for Google API compatibility.

**3.3 Implementation Steps**

1. **Create Angular Project**:

bash

ng new voice-assistant

cd voice-assistant

1. **Add Audio Worklet Processor**:
   * Create src/assets/recorderWorkletProcessor.js (code below).
2. **Modify app.component.ts**:

typescript

**import** { Component, ViewChild, ElementRef } **from** '@angular/core';

**import** { SpeechService } **from** './speech.service';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

**export** **class** AppComponent {

isRecording = false;

messages: string[] = [];

constructor(**private** speechService: SpeechService) {}

**async** startRecording() { */\* Recording logic \*/* }

**async** stopRecording() { */\* Stop logic \*/* }

}

**4. Backend Setup (Spring Boot)**

**4.1 Prerequisites**

* Java 21 JDK
* Maven
* Google Cloud account

**4.2 Key Components**

1. **Speech-to-Text API**: Converts audio to text.
2. **Gemini AI**: Generates responses.

**4.3 Implementation Steps**

1. **Create Spring Boot Project**:  
   Use [Spring Initializr](https://start.spring.io/) with:
   * **Dependencies**: Spring Web, Google Cloud Vertex AI
2. **Configure Google Cloud**:
   * Enable **Speech-to-Text API** and **Vertex AI API**.
   * Create service account key: [Google Cloud Console](https://console.cloud.google.com/)
3. **Create Configuration Class**:

java

*// File: src/main/java/com/example/config/GeminiConfig.java*

@Configuration

**public** **class** GeminiConfig {

@Bean

**public** VertexAI vertexAI() {

**return** **new** VertexAI("your-project-id", "us-central1");

}

}

1. **Create Controller**:

java

*// File: src/main/java/com/example/controller/SpeechController.java*

@PostMapping("/transcribe")

**public** ResponseEntity<Map<String, String>> transcribeAudio(@RequestBody **byte**[] audioBytes) {

*// Audio processing logic*

}

**5. Audio Processing Workflow**

1. **Frontend**:
   * User speaks → Angular records at 44.1kHz.
   * Resample to 16kHz → Convert to WAV format.
2. **Backend**:
   * Google Speech-to-Text converts WAV to text.
   * Gemini AI generates response from text.

**6. Error Handling Guide**

| **Error** | **Cause** | **Solution** |
| --- | --- | --- |
| Unreachable catch block | Unnecessary exception handling | Remove try-catch blocks |
| PERMISSION\_DENIED | Invalid Google Cloud credentials | Check API keys & enable services |
| HTTP 500 | Invalid audio format | Verify WAV headers & sample rate |

**7. Deployment Instructions**

**Frontend:**

bash

ng build --prod

*# Deploy dist/ folder to hosting service*

**Backend:**

bash

export GOOGLE\_APPLICATION\_CREDENTIALS="path/to/credentials.json"

./mvnw spring-boot:run

**8. Security Best Practices**

1. **Never commit** \*.json credential files.
2. Use .gitignore:

text

\*.json

/target/

1. Store secrets in environment variables.

**9. Troubleshooting Checklist**

1. **Microphone Issues**:
   * Check browser permissions (Chrome → Settings → Privacy).
2. **API Errors**:
   * Verify Google Cloud project billing status.
3. **Empty Responses**:
   * Test Gemini API key validity.

**10. Glossary**

| **Term** | **Definition** |
| --- | --- |
| WAV Header | Metadata describing audio format |
| Resampling | Converting audio from 44.1kHz to 16kHz |
| Service Account | Google Cloud credentials for API access |

**Appendix: Full Code Samples**

**Frontend Worklet Processor**

javascript

*// File: src/assets/recorderWorkletProcessor.js*

**class** RecorderWorkletProcessor **extends** AudioWorkletProcessor {

process(inputs) { */\* Audio chunk processing \*/* }

}

registerProcessor('recorder-worklet-processor', RecorderWorkletProcessor);

**Backend Service Class**

java

*// File: src/main/java/com/example/service/GeminiService.java*

@Service

**public** **class** GeminiService {

**private** **final** GenerativeModel model;

**public** GeminiService(VertexAI vertexAI) {

**this**.model = **new** GenerativeModel("gemini-pro", vertexAI);

}

}