Conversational IVR Modernization Framework

Week 1 Report — Legacy System Analysis

Submitted by:

Manoj Pemmadi

Infosys Internship Project

Date: *October 10, 2025*

1. *Project Overview*

This project focuses on ***modernizing legacy IVR systems*** by integrating them with Conversational AI platforms like **ACS** and **BAP**. Traditional IVR systems, built on *VoiceXML (VXML)*, are rigid and menu-driven. By adding an AI layer, we aim to enable natural, conversational interactions while preserving existing infrastructure.

Week 1 Objectives:

- Review existing IVR architecture and capabilities
- Document integration approach using a real-world use case
- Identify technical challenges and compatibility gaps

2. *Understanding Traditional IVR Systems*

2.1. *What is an IVR System?*

An ***Interactive Voice Response (IVR)*** system automates telephone interactions using voice prompts and keypad input (DTMF). Callers navigate through predefined menus to access information or services. Most legacy IVRs use **VoiceXML**, an XML-based language for defining voice dialogues.

2.2. *Core Components*

- **Telephony Interface:** Connects phone networks (PSTN/SIP) to the IVR system
- **IVR Application Server:** Executes VoiceXML scripts and manages dialogue flow
- **Speech Recognition (ASR):** Converts voice to text using limited grammars
- **Text-to-Speech (TTS):** Converts text responses into spoken audio
- **Backend Integration:** Connects to databases and APIs for information retrieval

2.3. *How It Works*

When a call arrives, the system plays a greeting and presents menu options. Users respond via keypad or simple voice commands. The IVR navigates through scripts, queries backend systems, and delivers responses using TTS or recorded audio.

2.4. Strengths and Limitations

Strengths:

- Handles high call volumes efficiently
- Reliable and low-latency performance
- Integrates with enterprise systems (CRM, databases)

Limitations:

- Rigid menu structures frustrate users
- No natural language understanding or context awareness
- Cannot handle complex, unstructured queries

3. Use Case: Flight Customer Support

3.1. Why This Use Case?

Airlines rely heavily on IVR for customer queries. A ***Flight Customer Support*** system demonstrates both the limitations of legacy IVR and the potential of conversational AI.

3.2. Current System Workflow

Callers use DTMF tones to:

- Check flight status
- Retrieve booking information
- Inquire about schedules

Example: "Press 1 for flight status, Press 2 for bookings..."

3.3. Proposed Conversational Workflow

With AI integration, users can speak naturally:
"What's the status of flight AI-203 from Delhi to Mumbai?"

The system:

1. Understands intent using Natural Language Processing (NLP)
2. Retrieves real-time flight data from backend APIs

3. Responds conversationally with relevant information

Benefits:

- Faster query resolution
- Natural, user-friendly interactions
- Reduced frustration and improved satisfaction

4. Integration Strategy with ACS/BAP

4.1. Proposed Architecture

We introduce a ***Middleware Layer*** between the legacy IVR and the AI platform:

Caller → *Telephony* → **ACS/BAP AI Engine** → *Middleware* → *Legacy IVR/APIs* → *Response*

4.2. How It Works

1. User speaks a query (natural language)
2. **ACS/BAP** extracts intent and entities using NLP
3. Middleware translates AI output to IVR-compatible requests
4. Backend systems process the request
5. AI generates a conversational response

4.3. Key Advantages

- *Preserves existing IVR logic and infrastructure*
- Adds conversational capabilities without full system replacement
- Enables gradual modernization with minimal disruption

5. Technical Challenges and Solutions

Challenge	**Description**	**Proposed Solution**
VoiceXML Rigidity	Cannot handle dynamic conversations	Middleware maps AI intents to VXML scripts dynamically
API Compatibility	Legacy systems lack modern REST APIs	Develop API wrappers for ACS/BAP communication
Voice Latency	Real-time audio processing delays	Optimize audio pipeline with buffering and streaming
Limited ASR	Grammar-based recognition fails with natural speech	Use AI-powered ASR in ACS/BAP
No Context Memory	IVR forgets previous interactions	Middleware manages session context
Cloud Integration	On-premise systems difficult to connect	Use secure API gateways or hybrid architecture
Data Security	Handling sensitive passenger information	Encrypt all communications; comply with regulations
Error Handling	AI failures or unrecognized input	Fallback to traditional IVR menus

6. Functional Requirements

The modernized system must support:

- ***Natural language input*** for flight-related queries
- *Real-time backend integration* with flight databases
- **AI-generated voice responses** for dynamic replies
- *Automatic fallback* to legacy IVR if AI fails
- *Analytics and monitoring* for performance tracking

7. Conclusion and Next Steps

7.1. Key Findings

- Legacy IVR systems are robust but limited in flexibility
- A middleware approach enables AI integration without replacing infrastructure
- Flight support is an ideal use case demonstrating modernization benefits
- Technical challenges are addressable through architectural solutions

7.2. Week 2 and Beyond

1. Design detailed middleware architecture
2. Develop API connectors for ACS/BAP integration
3. Build conversational flows for flight support
4. Implement fallback mechanisms and testing framework

This approach is *scalable* and can be extended to railway booking, mobile service providers, banking, and other domains requiring conversational interfaces.