AI Enabled Conversational IVR Modernization Framework

Project Architecture & Integration Plan

Submitted by

NAME:MOHAMMAD MUSARRAT NAZIA

Mail:musarratnazia02@gmail.com

Infosys Springboard Intern (AI Track)

Index (Table of Contents)

1. Introduction: The Mandate for IVR Modernization

1.1 Context of Legacy IVR Systems

- Interactive Voice Response (**IVR**) systems have long been central to automation in Mobile Service Providers (MSPs).
- Traditionally, they were built using **Voice XML (VXML)** a markup language designed to create static, rule-based voice menus where users interact through keypad (DTMF) or limited voice commands.
- These systems were effective when automation first began but now struggle to meet customer expectations for fast, intelligent, and conversational experiences.

1.2 The Need for Modernization

- Today's users expect **context-aware, human-like** interactions that understand natural language, adapt in real-time, and integrate seamlessly with digital services.
- Traditional VXML IVRs cannot support these due to rigid architectures, limited data access, and high maintenance costs.

1.3 Project Objective

- This report assesses existing VXML-based IVRs, documents their architecture and limitations, and proposes a **modern, AI-driven IVR framework** built on **Azure Communication Services (ACS)** and **Business Application Platform(BAP)**.
- The goal is to achieve intelligent, scalable, and integrated customer support for mobile service providers.

2. Traditional (VXML) IVR Architecture, Capabilities, and Limitations

2.1 Architecture Overview

Legacy IVR systems use fixed logic, hosted on-premises, and controlled by telephony infrastructure.

Component	Function	Type
PSTN/Telephony Network	Handles inbound/outbound calls.	Hardware
VXML Gateway	Interprets VXML scripts; manages voice sessions.	Middleware
Application Server	Executes pre-coded menu trees.	Software
CRM/Database	Retrieves customer records via legacy interfaces.	Backend

2.2 Capabilities of Traditional IVR

Despite their rigidity, legacy systems provided several baseline automation features:

- **Menu Navigation:** Users interact via keypad inputs or basic voice responses.
- **Information Retrieval:** Fetch static data (e.g., account balance, plan details).
- **Call Routing:** Directs calls to the correct department or agent.
- **DTMF Input Recognition:** Accurately interprets numeric inputs.
- **Basic Call Recording:** Stores limited interaction logs for audits.

2.3 Limitations

	Limitation	Impact
1.	**Rigid Flow Structure**	Users must follow preset paths with no conversational flexibility.
2.	**No Context Awareness**	System cannot identify returning customers or recall preferences.
3.	**PoorSpeech Recognition**	Limited grammar leads to misinterpretations of natural speech.
4.	**Scalability Constraints**	Hardware-bound; cannot handle call surges efficiently.
5.	**Maintenance Overhead**	Any menu update requires code redeployment.

2.4 Architecture Plan of Traditional IVR (VXML-Based)

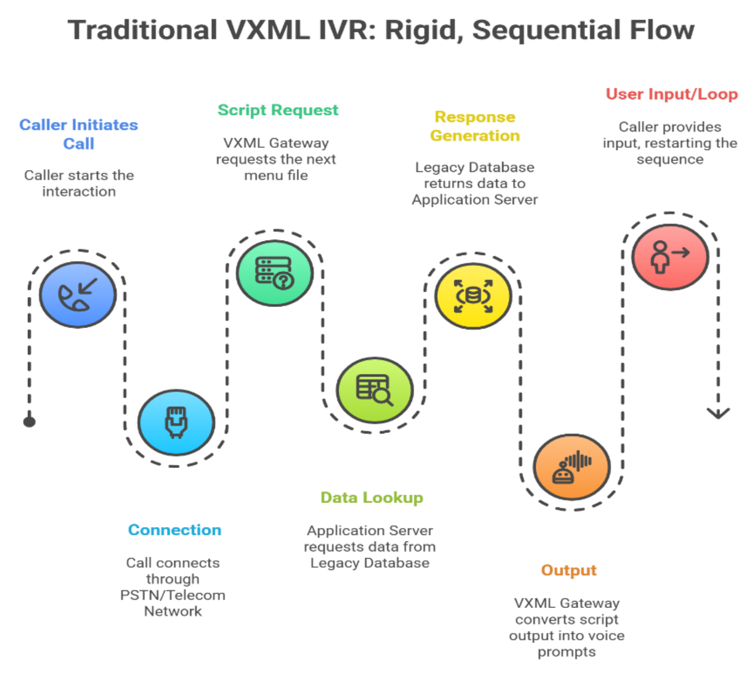

Figure1.Traditional IVR Architecture Plan

3. Middleware/API Layer for Legacy IVR Integration

3.1 Objective

- Build a middleware/API layer to connect legacy VXML IVRs to the modern Conversational AI stack (ACS & BAP).
- Enable real-time session handling, customer context management, and secure communication between legacy systems and AI-driven services.
- Demonstrate feasibility with a working prototype using FastAPI and Twilio.

3.2 Scope

1. Design and implement connectors or APIs to enable seamless communication between VXML IVR systems and ACS/BAP.
2. Ensure real-time data handling and system compatibility, including DTMF input, voice recordings, and session state management.
3. Validate the integration layer with sample transaction flows and call interactions using the FastAPI + Twilio prototype.
    - Includes dynamic menu navigation, balance inquiry, recharge processing, and voice-based issue reporting.

3.3API & Middleware Implementation

Due to not having access to Azure Communication Services (ACS), the FastAPI + Twilio IVR demo was implemented as a functional prototype to validate dynamic call handling, session management, and menu navigation.

Technology Used - Python, FastAPI, Twilio, Pydantic, Twilio VoiceResponse

Key Features Implemented

1. **Start Call Endpoint** – Initializes a customer session and presents the main menu.

2. **Handle Input Endpoint** – Processes user DTMF input and routes to IVR steps (balance inquiry, recharge, report issue).
3. **Twilio Integration** – Supports voice calls, dynamic menu navigation, and recording of issues.
4. **Session Management** – Maintains call context across multiple steps using in-memory storage (call_sessions).
5. **Error Handling** – Provides clear feedback if the customer is not found or session is invalid.
6. **Prototype Demonstration** – FastAPI + Twilio demo validates the integration layer with live call handling and menu flows**.**

3.4Sample Outputs

- **Start Call -** Welcomes the customer and presents menu options.
- **Check Balance** - Returns the customer's current data balance.
- **Recharge Plan** - Accepts numeric input and confirms recharge.
- **Report Issue -** Records a voice message and stores it in the session

4. Modern AI IVR Architecture, Capabilities, and Integration (ACS & BAP)

Modern IVRs adopt **cloud-native, AI-integrated architectures** for flexibility, intelligence, and real-time service delivery.

4.1Capabilities

- **Conversational Understanding (NLU)**:Understands intent regardless of phrasing, accent, or language.
- **Dynamic Call Flow** - Adapts routes based on detected intent and context.
- **Cloud Scalability** - Uses Azure's elastic infrastructure to handle variable loads.
- **Omnichannel Integration** - Extends support to WhatsApp, SMS, and chatbots.
- **Personalization** - Uses ANI/CRM data to provide tailored responses.
- **Analytics &** -Tracks intent success rate and call satisfaction for continuous learning**.**

4.2 Modern IVR Architecture

Layer	Technology	Description
Layer(ACS)	Azure Communication Services	Manages call connectivity, call automation, and session handling.
Telephony Intelligence Layer (Azure Bot Service)	NLU, STT, TTS	Converts speech to text, interprets intent, and manages conversation.
Integration Layer (BAP)	API Gateway & Middleware	Connects the IVR with backend systems using REST APIs and JSON/XML translators.

4.3Key Advantages Over VXML IVR

- **No rigid menus:** Dynamic, AI-driven interactions instead of fixed trees.

- **Reduced wait time:** Contextual responses replace long DTMF flows.
- **Easy integration:** REST APIs replace static backend calls.
- **Continuous improvement:** Models learn from feedback and retraining.

4.4Architecture Plan of Modern IVR

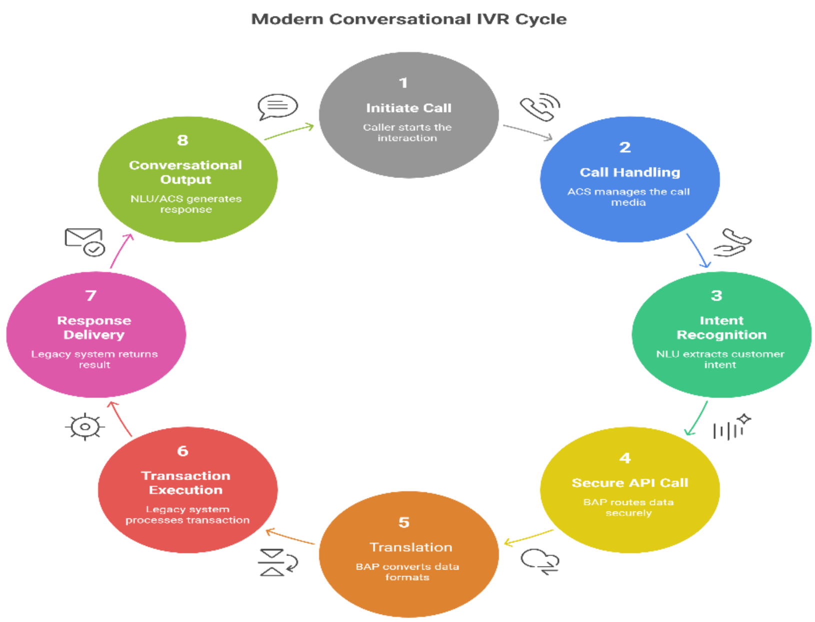

Figure 2.Modern IVR Architecture

5. Technical Challenges, Constraints & Compatibility Gaps

Challenge / Gap	Description	Mitigation
Integration Latency	Legacy systems respond slowly.	Use BAP caching for non-critical data.
Data Format Gap(XML vs JSON)	Modern APIs use JSON; legacy systems use XML.	Build middleware translator via Azure Functions.
Security Protocol Gap	Legacy lacks OAuth2 or token auth.	Add secure API Gateway in BAP.
NLU Regional Accuracy	Struggles with dialects.	Train region-specific language models.
Legacy Dependency	Not all services easily migrated.	Phased co-existence with fallback routing.

6. Conclusion

- Transitioning from **legacy VXML IVRs** to **cloud-native, AI-driven IVRs** using **ACS** and **BAP** is essential for modern customer engagement.
- The new architecture enables intelligent, context-aware, and cost-efficient service automation while integrating securely with existing telecom systems.