AI Enabled Conversational IVR Modernization Framework

Objective: Assess current VXML-based systems and define technical and functional integration requirements.

Task 1- IVR System Architecture – Key Components

1. **VoiceXML Interpreter (Voice Browser)**
    - Executes VXML scripts that define call flow logic (menus, prompts, user input handling).
    - Works like a web browser for voice.
2. **Telephony Interface / Call Control System**
    - Handles inbound/outbound calls via **PSTN**, **SIP**, or **VoIP**.
    - Responsible for call setup, teardown, routing, conferencing, and transfers.
3. **Media Server**
    - Plays audio prompts, collects DTMF tones, and streams audio between caller and system.
    - May also handle voice recording.
4. **ASR (Automatic Speech Recognition)**
    - Converts spoken input into text.
    - Integrated via **MRCP (Media Resource Control Protocol)** or similar standards.
5. **TTS (Text-to-Speech)**
    - Converts dynamic text responses into audio.
    - Enables real-time interaction without needing pre-recorded audio.
6. **Dialog Manager / Application Server**
    - Orchestrates the call flow.
    - Retrieves VXML documents dynamically from application logic based on user input and context.
7. **Backend Integration Layer**
    - Connects the IVR system to databases, CRMs, ERPs, ticketing, payment gateways, etc.
    - Uses REST, SOAP, JDBC, or proprietary APIs to fetch or push data.
8. **Prompt Repository / Audio File Store**
    - Stores pre-recorded audio prompts (e.g., welcome messages, system prompts).
    - May include multi-language or seasonal prompt variations.
9. **Logging & Monitoring Subsystems**
    - Track call events, user input, recognition errors, and performance metrics.
    - Used for troubleshooting and reporting.
10. **Admin Console / IVR Designer**
    - GUI-based tools for building call flows, editing prompts, setting timeouts, and error handling.
    - May allow testing and simulation of call flows.

2. IVR Capabilities – What Existing Systems Can Do

1. **Dual Input Modes**
    - Accept **DTMF (keypad)** and **voice input (speech recognition)**.
    - Fallback between modes is often supported.
2. **Multi-Language Support**
    - Prompts and recognition grammars can be customized per language.
    - Dynamic language switching based on user input or profile.
3. **Dynamic Call Flows**
    - IVR can retrieve real-time data to alter dialog flow (e.g., booking status, account info).
    - VXML supports conditional logic and variables.
4. **Personalization**
    - IVR can greet customers by name, offer services based on profile or history (requires backend integration).
    - Uses caller ID, customer ID, or token for lookup.
5. **Natural Language Processing (NLP)** (in some advanced systems)
    - Goes beyond menu navigation, allowing open-ended queries like "I want to change my flight."
    - Requires integration with NLU (Natural Language Understanding) engines.
6. **Agent Handoff / Call Transfer**
    - Transfers call to a live agent when needed.
    - Can include **screen pop** or context-passing (e.g., booking ID, selections made in IVR).
7. **Self-Service Transactions**
    - Common use cases: checking balance, booking status, payment, recharges, ticket cancellations.
    - Secure IVR implementations can even handle card inputs (PCI-DSS compliant).
8. **Time-Based Routing & Scheduling**
    - IVR flows adapt based on time of day, holidays, or service availability.
    - Supports routing to specific departments based on input or schedules.
9. **Call Recording & Analytics**
    - Voice interactions may be recorded (with consent) for quality and compliance.
    - Usage analytics help optimize flows and identify drop-off points.

Task 2- Use Case: Flights Customer Support IVR System

This IVR allows passengers to interact with an airline via voice or keypad to:

- Check flight status
- Modify bookings
- Request cancellations/refunds
- Get baggage or check-in info
- Speak to an agent

Align Modern IVR with ACS & BAP Platforms

A. Define Key IVR Functionalities (User Intent)

The IVR should support:

- **User identification** via phone number, frequent flyer ID, or booking reference
- **Booking lookup** and flight details (status, gate, delay info)
- **Booking modification or cancellation**
- **Payment/refund handling** (via secure voice inputs or SMS links)
- **Agent escalation** with context transfer
- **Multi-language support**
- **24x7 availability and real-time updates**

B. Identify Data and API Requirements

To integrate with **ACS and BAP**, IVR must interact with these APIs:

- **ACS APIs**:
  - GET /customer-profile: retrieve user details using phone number or ID
  - POST /verify-user: OTP-based or voice authentication
  - GET /language-preference: to set IVR language dynamically
- **BAP APIs**:
  - GET /booking-status: flight and booking information
  - POST /cancel-booking: handle cancellations and initiate refund
  - POST /update-booking: change seats, meals, or travel dates
  - POST /payment-request: initiate or confirm payment/refund

C. Design Integration Architecture

- IVR calls a **middleware layer** (API gateway or integration service) that:
  - Authenticates securely with ACS/BAP
  - Translates IVR inputs into API calls
  - Handles response parsing and error management
- Use **VXML-based dialogs** that call backend APIs in real-time
- Implement **TTS** for dynamic data like flight number, timings
- Use **ASR** to capture spoken inputs (e.g., “Cancel my booking”)

D. Enable User Authentication

- IVR asks for phone number + OTP
- IVR calls ACS to verify user and retrieve profile
- Based on profile, the IVR:
  - Greets user by name
  - Offers options based on current bookings

E. Booking Interaction Logic

Example flow:

1. "Press 1 or say *Flight Status*"
2. IVR: "Please say your flight number or booking reference"
3. IVR sends request to **BAP** → retrieves flight info
4. IVR responds with TTS: "Flight AI-123 from Delhi to Mumbai is on time…"
5. IVR asks: "Do you want to make changes to this booking?"

F. Payment & Refund Integration

- IVR confirms cancellation request
- Calls BAP /cancel-booking and retrieves fee/refund info
- Initiates secure flow (voice credit card input or SMS payment link)
- Confirms transaction and provides reference number

G. Agent Escalation (Optional Step)

- If user says "Speak to agent" or reaches a dead-end:
  - IVR passes context (user name, booking ID, issue type)
  - Transfers call to agent or schedules callback
  - Uses **SIP/VoIP routing** and **screen pop** on agent desktop

H. Technical Requirements for Alignment

- IVR must support:
  - **REST API integration** (ACS/BAP should expose RESTful services)
  - **JSON/XML parsing** in VXML or via middleware
  - **Secure communication**: TLS, OAuth2 tokens
  - **Session management** to track progress across IVR steps
  - **Logging**: log every interaction and backend call for traceability
  - **Multilingual support** with dynamic language switching

I. Optional Enhancements

- **Voice biometrics** for user authentication via ACS
- **AI-powered NLU** to understand user intents more naturally
- **SMS/email alerts** sent post-call (e.g. booking changes, receipts)
- **Chatbot or app integration** for omnichannel handoff

J. Testing and Monitoring

- Simulate flight scenarios in test ACS/BAP environments
- Conduct **end-to-end UAT**: IVR → ACS → BAP → IVR response
- Use **analytics dashboards** to monitor IVR usage, success rate, drop-offs
- Regularly test:
    - Speech recognition accuracy
    - API response times
    - Payment/refund reliability
    - Agent transfer success rate

Task 3- *Identify all the technical challenges, constraints, and compatibility gaps*

Technical Challenges:

- **Speech recognition errors** – especially with accents, noise, or flight numbers.
- **Slow backend response** – APIs may delay IVR flow.
- **Poor internet/voice quality** – affects cloud-based IVR performance.
- **Real-time data sync** – difficult to keep flight/booking info updated instantly.
- **Secure payment handling** – requires PCI-DSS compliance and encryption.
- **Session handling** – tracking users across multiple steps is complex.

Constraints:

- **Legacy systems** – older IVRs may not support APIs or dynamic flows.
- **Limited scalability** – on-prem IVRs may not handle peak call loads.
- **Language limitations** – not all prompts or ASR support multi-language well.
- **Budget and licensing** – ASR/TTS engines and API usage may be costly.
- **Regulatory compliance** – must meet GDPR, PCI-DSS, etc.

Compatibility Gaps:

- **Different data formats** – ACS/BAP may use formats not easily handled by IVR.
- **Auth/token issues** – mismatched security methods between systems.
- **No API standards** – some systems use REST, others use SOAP or proprietary APIs.
- **Media protocol mismatch** – IVR may not support required MRCP/SIP codecs.
- **Voice vs keypad flows** – voice-only features may not work with DTMF paths.

- Ashish Bishoyi