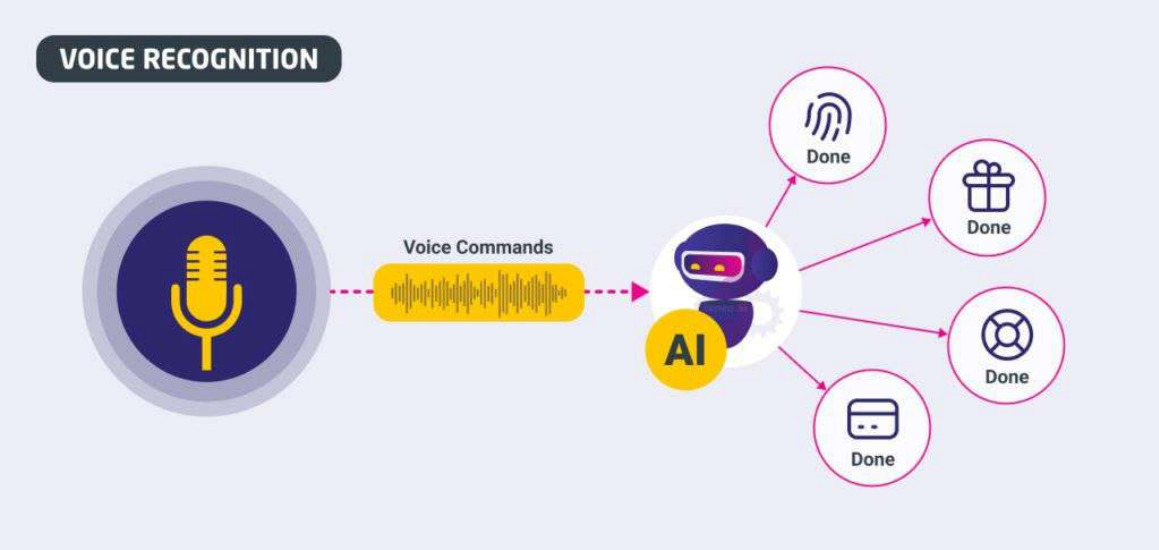
Conversational IVR Modernization Framework

Milestone 2: Delivery of functional integration layer between VXML systems and ACS/BAP



By
Team A

Table of Contents

Milestone 2 Documentation

1. Title & Team Details

- Project Name: Conversational IVR Modernization framework
- Team Name: Team-A

Team Breakdown – Roles & Contributions

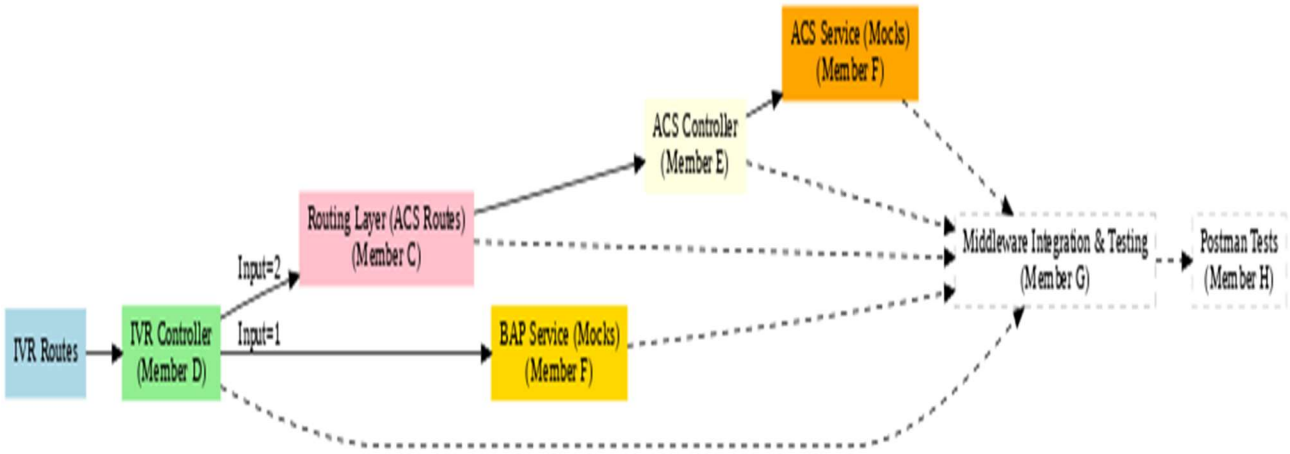
1. Bhargav Molleti – Team Lead
Handled repo setup, CI/CD pipeline, and GitHub workflow management.
2. Animesh Mondal – IVR Routes
Developed /ivr route with request validation and integration to controller logic.
3. Prachi Saxena – ACS Routes
Implemented /acs routes for start, stop, and DTMF APIs with proper validations.
4. Karnika – IVR Controller
Designed IVR controller logic to route inputs toward BAP or ACS services.
5. Vinay Kumar – ACS Controller
Connected ACS routes with mock ACS services and returned structured responses.
6. Shifa – Services (ACS + BAP Mocks)
Built mock ACS and BAP services for balance check and agent transfer handling.
7. Amita Singh – Middleware Integration & Testing
Integrated routes, controllers, and services while debugging middleware flows.
8. Amshula – Postman Collection & Test Cases
Created Postman test collection with positive and negative scenarios.
9. Sumedha Kar – API Documentation
Prepared structured API.md with endpoints, examples, and error handling details.
10. Nithya – Flow Documentation
Created sequence diagrams, setup guides, and troubleshooting documentation.
11. Abilash – Presentation & Aggregation
Prepared the final presentation document and consolidated all modules.

2. Objective

In this milestone, we developed a middleware layer to connect legacy VXML IVR systems with Conversational AI (ACS/BAP). The goal was to create APIs and services that allow seamless communication between the two systems.

- Ensure smooth routing of IVR inputs to either ACS or BAP services.
- Provide mock services for testing call flows without impacting live systems.
- Enable easier integration, debugging, and validation through standardized APIs.
- Support scalability for future enhancements and real system integration.

3. Architecture & Flow Design



- **IVR Routes** capture user inputs from the legacy VXML system.
- **IVR Controller** decides the flow:

  Input=1 → forwards to **BAP Service**.

  Input=2 → forwards to **ACS Service** via Routing Layer.

- **Routing Layer (ACS Routes)** handles / acs endpoints (start, stop, sendDTMF) and validates inputs.

- **ACS Controller** connects the routing layer with **ACS Service**, returning mock responses.
- **Service's Layer**:

  BAP Service → provides balance or transfer-to-agent responses.

  ACS Service → simulates start/stop/sendDTMF actions.

- **Middleware Integration & Testing** connects all layers, ensures data flow, and validates responses with dummy payloads.
- **Postman Collection & Test Cases** automate validations with positive and negative test scenarios

4. System Setup & Configuration

Here's a detailed explanation of the system setup steps for running a Node.js + Express middleware server, suitable for documentation:

4.1 Install Dependencies

Navigate to your project directory in the terminal. Run the command npm install to install all dependencies listed in your package.json file. This ensures that all required packages for your Node.js and Express application are available. If you want your server to automatically restart when you make code changes (useful during development), you can also install nodemon by running npm install --save-dev nodemon.

4.2 Start the Server

To start the middleware server, use the command node index.js if you want to run the server normally. If you installed nodemon, you can use npx nodemon index.js instead; this will watch for file changes and automatically restart the server when necessary.

4.3 Access the Middleware API

Once the server is running, open your web browser and go to http://localhost:3000/api/ivr. This address points to the IVR API endpoint

provided by your middleware, which should now be responding to requests as defined in your Express application.

Key Notes:

- Make sure Node.js and npm are installed on your system before beginning.
- You do not install index.js—it is your main server file, not a package.
- The default server port is 3000, but you can change this in your code if needed.
- For development, nodemon helps by restarting your server automatically when you change code.
- For production, use node index.js for a stable server process.

5. API Reference

5.1 POST /ivr/input

Purpose:

This endpoint accepts user input from the IVR (Interactive Voice Response) system. It is designed to receive data such as speech or DTMF (touch-tone) inputs from users interacting with the IVR.

Request Format:

Send a JSON object in your POST request body with the following fields:

JSON

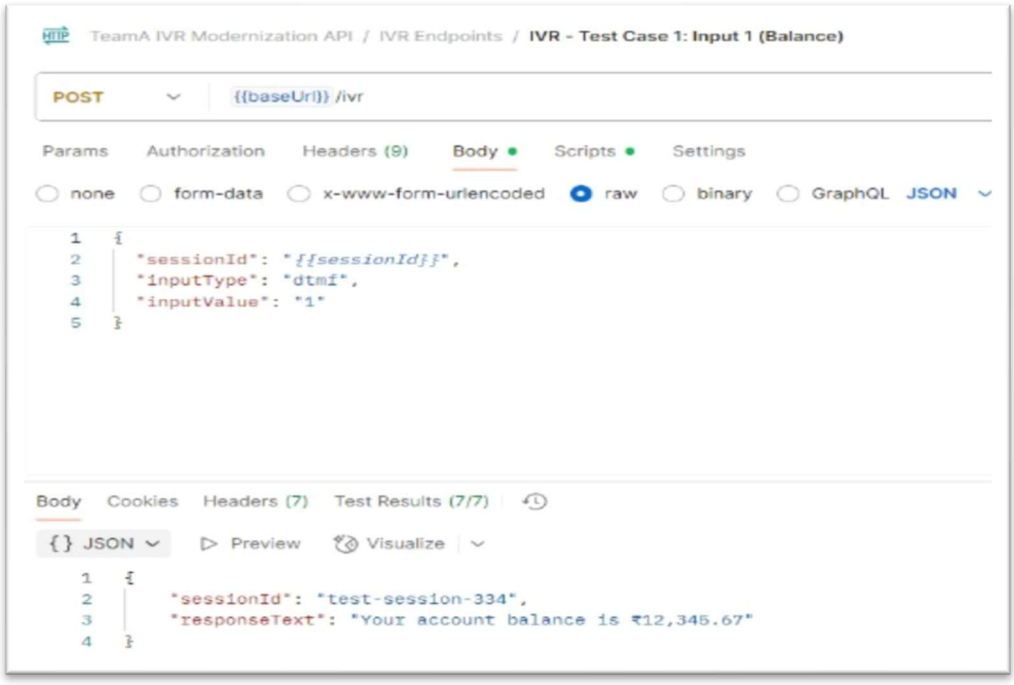
```
{
  "sessionId": "12345",
  "inputType": "speech",
  "inputValue": "Hello"
}
```

sessionId: A unique identifier for the IVR session (string).

inputType: The type of input received. Common values include "speech" or "dtmf" (string).

inputValue: The actual input from the user, such as a spoken phrase or DTMF digits (string).

Usage Example:



You might use this endpoint after capturing user input in your IVR system to send it to the backend for processing.

5.2 GET /acs/response

Purpose:

This endpoint returns the AI-generated response based on the most recent user input provided through the IVR. It is typically called after a user has interacted with the IVR and a response is ready to be delivered.

Response Format:

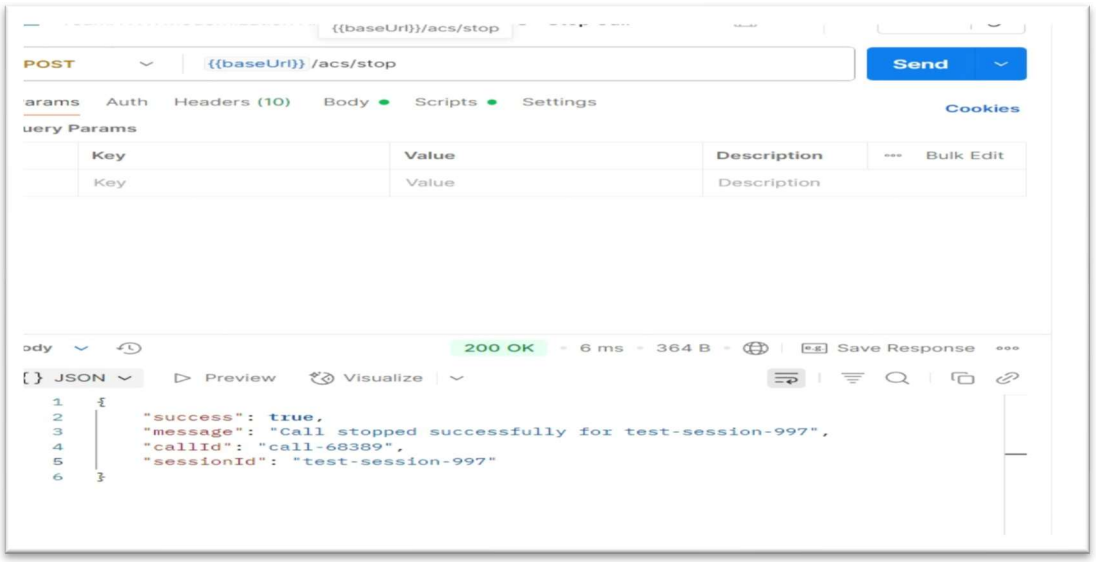
A successful request will return a JSON object like the following:

JSON

```
{
        "success": true,
        "message": "Call stopped successfully  for test-session-997",
        "callId": "call-68389",
        "sessionId": "test-session-997"
}
```

response: The AI-generated message intended for the user (string).

Usage Example:



After sending user input to the system, call this endpoint to fetch the response that should be played or displayed to the user.

Summary of Flow:

- The IVR system collects user input (via speech or DTMF).
- The input is sent to the backend using the POST /ivr/input endpoint.
- The backend processes the input and generates an AI response.
- The client requests the AI response using the GET /acs/response endpoint.
- The response retrieved can be played back to the user.

6. Challenges & Learnings

6.1 Challenges Faced

1. **Express.js setup issues:** Faced dependency errors and local server restarts with nodemon.
2. **JSON parsing errors:** Request bodies returned undefined due to misconfigured middleware.
3. **Route integration:** Struggled to structure /ivr, /acs, /bap routes cleanly.
4. **Error handling:** Difficulty handling missing fields and designing consistent 400 responses.
5. **Collaboration & Git:** Faced merge conflicts and confusion with Git remotes/branches.
6. **Time management:** Balancing code updates and documentation within sprint deadlines.

6.2 Learnings

1. **Middleware development**: Learned to use Express.js middleware and centralized error handling.
2. **API design:** Practiced clean endpoints, HTTP methods, and QA-style error responses.
3. **Debugging**: Improved debugging with Postman, console logs, and input validation.
4. **Teamwork:** Coordinated with peers on test cases, flows, and aligned docs with code.
5. **Git discipline:** Understood branching, commits, and syncing docs with implementation.
6. **Documentation:** Gained experience writing structured, professional API documentation.

7. Conclusion & Next Steps

The middleware successfully simulates integration between legacy IVR and modern AI platforms, setting the base for conversational flow integration in Milestone 3. It demonstrates how /ivr, /acs, and /bap endpoints can work together to mimic call handling, intent recognition, and session management, while also providing QA-ready test coverage and clear documentation. This milestone bridges the gap between traditional IVR systems and modern conversational frameworks, laying a strong technical and collaborative foundation for future enhancements.

Next Steps — Milestone 3: Conversational AI Interface Development

Objective: Introduce natural language capabilities to the IVR system via conversational flows.

Planned Tasks:

- Develop conversational dialogue flows that map to existing IVR logic.
- Integrate conversational flows into the legacy system architecture.
- Enable real-time voice input/output handling via Conversational AI.

Timeline: Weeks 5–6 — Development and successful integration of conversational AI flows.

Overall Impact: This milestone not only advanced the project technically but also strengthened my skills in APIs, debugging, documentation, and teamwork, building confidence for future contributions.