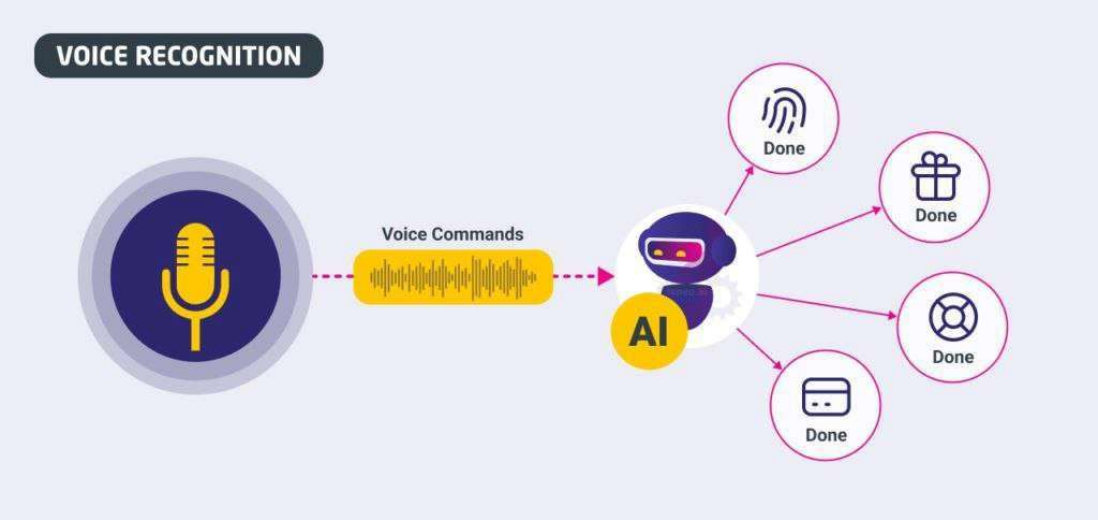
Conversational IVR Modernization Framework

Milestone 3: Conversational

AI Interface Development



By

Team A

Table of Contents

Step 1 – Plan Conversational Flows

1. Introduction

In Milestone 3, the primary goal is to extend the middleware with conversational capabilities. This allows free-text or voice-transcribed queries to be routed intelligently to ACS (Account Customer Services) or BAP (Banking Application Platform) mock services. The solution ensures that users can access advanced services like checking balance, requesting mini-statements, or reporting lost cards in a more natural way.

2. Objectives

The key objectives of Milestone 3 are:

i. Extend IVR routes to include a new /ivr/conversation API endpoint.
ii. Add controller logic to handle natural language queries.
iii. Implement intent detection (NLU simulation via keyword matching).
iv. Route queries to appropriate ACS or BAP mock services.
v. Standardize responses for consistent client and Postman testing.
vi. Validate functionality with positive, negative, and unknown intent cases.

3. Selected Use Cases and Justification

Service	Use Case	Justification
ACS	Balance Inquiry	Users frequently check their remaining account balance. This is a simple retrieval task best suited for ACS.
	Recharge Account	Recharge/top-up directly updates account balance, hence routed to ACS for secure handling.
	Report Lost Card	Security-critical task. ACS immediately blocks the card and ensures customer safety.
	Activate New Card	Card activation is an account operation, requiring validation and controlled by ACS.

	Update Contact Details	Updating mobile or contact details is a customer data management task managed by ACS.
	Report Suspicious Transaction	Fraud/suspicious activity reports need immediate action; ACS initiates protective measures.
BAP	Agent Request	Escalates user requests beyond automation. BAP connects the user with a live agent.
	Mini Statement	Fetches last few transactions, part of BAP's customer support features.
	Pay Utility Bill	Bill payments are external service transactions routed via BAP.
	Loan Details Inquiry	Loan EMI and outstanding balance queries are managed by BAP.
	E-Statement Request	BAP generates and delivers e-statements to user's registered email.

4. System Architecture

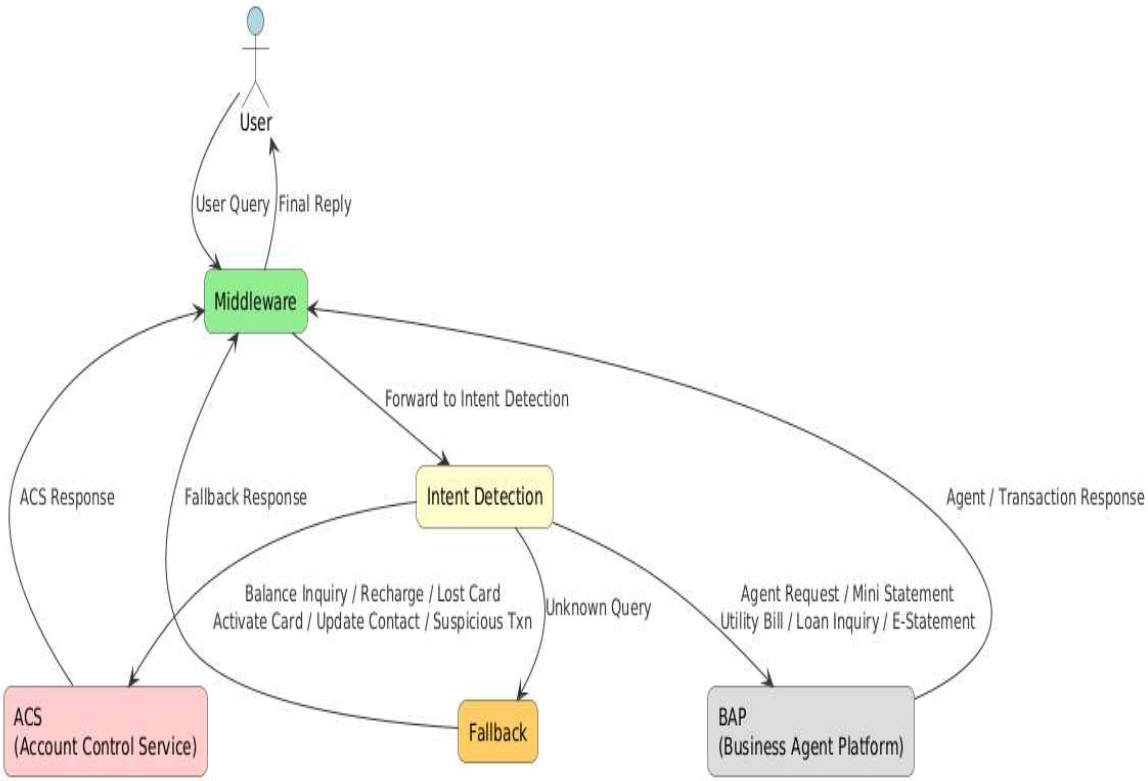


Figure 1: Middleware, Intent Detection, and Service Flow

5. Intent Mapping Table

Intent	Example User Utterances	Mapped Service	Response Type	Response Template
Balance Inquiry	"Check my balance", "How much money do I have?"	ACS	Returns current account balance	"Your current account balance is ₹[balance]."
Recharge Account	"Recharge my account", "Top up ₹100"	ACS	Processes recharge & confirms success	"Your account has been recharged with ₹[amount]. New balance is ₹[balance]."
Report Lost Card	"I lost my card", "Block my card immediately"	ACS	Blocks the card and confirms security action	"Your card ending with [xxxx] has been blocked. For assistance, contact support."
Activate New Card	"Activate my new card", "Enable card 1234"	ACS	Activates the card after validation	"Your card ending with [xxxx] has been successfully activated."
Update Contact Details	"Update my phone number", "Change my email"	ACS	Updates user contact information and confirms	"Your contact details have been updated successfully."
Report Suspicious Transaction	"I see a suspicious charge", "Fraudulent transaction alert"	ACS	Flags transaction and initiates protective measures	"We have flagged the transaction for review and secured your account."
Agent Request	"Talk to agent", "Connect me to support"	BAP	Connects user to a live agent	"Connecting you to a live agent. Please wait..."
Mini Statement	"Show my last transactions", "Mini statement please"	BAP	Fetches and displays recent transactions	"Here are your last [n] transactions: [transaction list]."

Pay Utility Bill	"Pay my electricity bill", "Pay ₹500 for water"	BAP	Processes payment and confirms success	"Your payment of ₹[amount] for [bill type] has been successfully processed."
Loan Details Inquiry	"Check my loan balance", "What's my EMI?"	BAP	Provides loan details and EMI information	"Your outstanding loan balance is ₹[balance]. Your next EMI of ₹[EMI amount] is due on [date]."
E-Statement Request	"Send my e-statement", "Email my statement"	BAP	Generates e-statement and sends to registered email	"Your e-statement has been sent to your registered email address."
Unknown Query (Fallback)	"What's the weather?", "Play a song"	Fallback Handler	Responds: "Sorry, I cannot process that request right now."	"Sorry, I cannot process that request right now. Please try something else."

6. Design Considerations and Assumptions

While preparing these conversational flows, the following assumptions and design principles were applied:

1. Keyword-based intent detection will be used for this milestone instead of advanced NLP/ML models.
2. Mock ACS and BAP services will be used, returning predefined JSON responses.
3. Recharge intent assumes a fixed recharge amount for now (like ₹100), but future versions can support dynamic inputs.
4. Agent request does not handle actual queue management; it simply returns a connection response.
5. Any unsupported or unknown queries will be routed to a fallback:
    - Response: *"Sorry, I cannot process that request right now."*

Step 2 – Middleware Extension

1. Project Structure

```
middleware-project/
├── package.json
├── index.js                    # Main entry point
├── /routes
│   └── ivrRoutes.js            # IVR + conversation routes
│   └── acsRoutes.js            # ACS endpoints (start, stop, DTMF)
├── /controllers
│   └── ivrController.js        # IVR + conversation handling
│   └── acsController.js        # ACS route handling
├── /services
│   └── acsService.js           # ACS mock services (balance, recharge, lost card, etc.)
│   └── bapService.js           # BAP mock services (mini statement, loans, etc.)
├── /handlers
│   └── intentHandler.js        # Intent detection + routing
├── /utils
│   └── logger.js               # Logging utility
├── /docs
│   └── API.md                  # API documentation
│   └── architecture.png        # Sequence diagram
├── README.md
```

2. Objective

The goal of this step is to extend the middleware by adding a new API route `/ivr/conversation` to handle IVR queries. The middleware receives user queries, detects the intent, routes the request to the appropriate service (ACS or BAP), and returns a response.

3. API Route Details

- **Endpoint:** `/ivr/conversation`
- **HTTP Method:** `POST`
- **Request JSON Format:**

```
{
  "sessionId": "101",
  "query": "check balance"
```

}

- **Response JSON Format:**

{

"sessionId": "101",

"response": "Your account balance is ₹500."

}

4. Workflow / Logic Flow

a) Middleware receives JSON input with sessionId and query.

b) Query is passed to detectIntent(query) to identify the intent.

c) Based on the detected intent:

- ACS handles account-related queries: Balance Inquiry, Recharge, Report Lost Card, Activate New Card, Update Contact Details, Report Suspicious Transaction.
- BAP handles support and transactional queries: Agent Request, Mini Statement, Pay Utility Bill, Loan Details Inquiry, E-Statement Request.
- Unknown or unsupported queries are routed to Fallback Handler.

d) The corresponding mock service (ACS/BAP) is called to process the query.

e) Middleware returns JSON response with sessionId and the service response.

5. Features Implemented

- New API route `/ivr/conversation` to accept user queries.
- Integration with detectIntent.js for intent detection.
- Routing logic to ACS and BAP services based on intent.

- Support for all mapped use cases in ACS and BAP.
- Fallback handler for unknown queries.
- JSON response generation with session tracking.
- Endpoint tested successfully in Postman for all intents.

6. Integration Details

- Updated files:
    - ivrController.js – handles API logic.
    - ivrRoutes.js – defines the `/ivr/conversation` route.
    - Mock ACS and BAP services – updated to support all relevant use cases.
- Middleware is connected to detectIntent module to map user queries to the correct service.

Step 3 – Create Intent Handler

1. Objective

Our goal was to build a simple keyword-based Intent Handler, which acts as the Natural Language Understanding (NLU) module of our IVR system.
This component is responsible for analyzing the user's query and mapping it to a specific intent, such as:

- ACS → Account-related services (e.g., check balance, recharge)
- BAP → Agent-related requests (e.g., talk to agent, connect to support)
- UNKNOWN → Any query that doesn't match predefined keywords

The Intent Handler ensures that user inputs like "Check my balance" or "I need an agent" are correctly routed to the appropriate service.

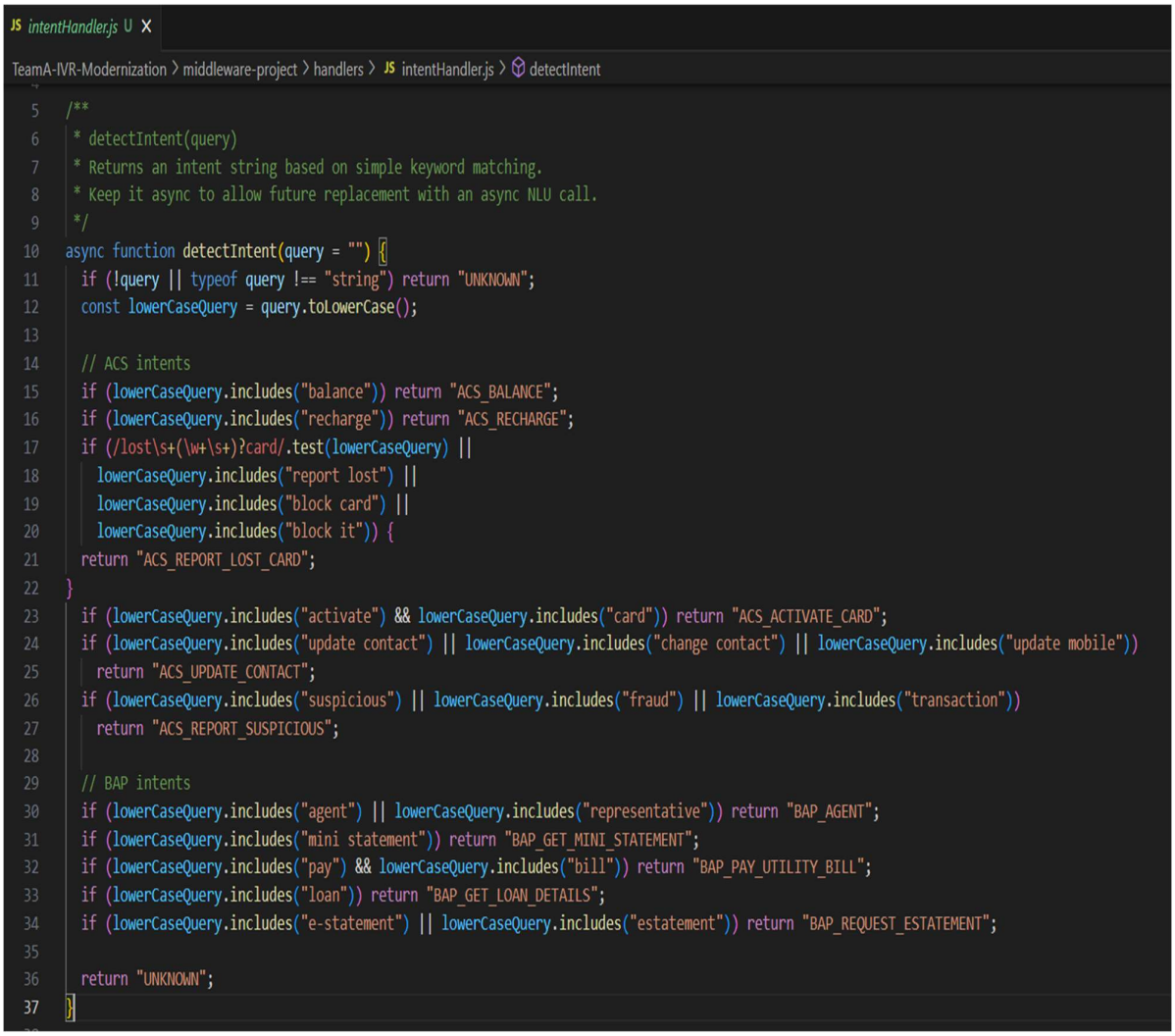
Keywords:

ACS: ["balance", "check balance", "recharge", "top up", "bill"]

BAP: ["agent", "support", "representative", "human"].

If no keyword match- Return error.

2. Implementation (intentHandler.js):



3. Workflow

1. User submits a query via `/ivr/conversation`.
2. The query is passed to the Intent Handler.
3. Intent is detected based on keywords.
4. The IVR system routes the query to:

- **ACS** for account-related tasks
- **BAP** for agent-related tasks
- **Fallback handler** for unknown queries

5. The system returns a response JSON with the session ID and message.

4. Example Queries and Responses

Input Query	Intent Detected	Response
"Check my balance"	ACS	"Your account balance is ₹500."
"Recharge my account"	ACS	"Your recharge of ₹199 is successful."
"I need an agent"	BAP	"Connecting you to an agent..."
"Can you connect me to support?"	BAP	"Connecting you to an agent..."
"Tell me weather"	UNKNOWN	"Sorry, I did not understand that."
"Random text"	UNKNOWN	"Sorry, I did not understand that."

Step 4 – Update ACS & BAP Mock Services

1. Objective

Extend the **ACS** and **BAP** mock services to handle all conversational intents in the IVR system.

- **ACS (Automated Communication System):** Handles account-related tasks such as balance inquiry, recharge, card management, and suspicious transaction reporting.
- **BAP (Business Agent Platform):** Handles agent-related requests and transactional support, such as connecting users to a live agent, mini statements, bill payments, loan details, and e-statement requests.

Goal: Provide **realistic JSON responses** for each intent to enable testing and demonstration.

2. ACS Mock Service

Overview:
ACS simulates **call-related actions** and **account-based conversational intents**. It supports both **DTMF inputs** and **natural language queries**.

Location: `services/acsService.js`

Features Implemented:

1. **Call Management**
    - startCall(sessionId) → Starts a mock call session.
    - stopCall(sessionId) → Stops an active call session.
    - sendDTMF(sessionId, dtmf) → Handles keypress input during calls.
2. **Conversational Intents**
    - getBalance(sessionId) → Returns mock account balance.
    - recharge(sessionId, amount) → Simulates a recharge.
    - reportLostCard(sessionId) → Blocks the card and sends confirmation.
    - activateNewCard(sessionId) → Guides the user through card activation.
    - updateContactDetails(sessionId) → Sends a secure link for updating details.
    - reportSuspiciousTransaction(sessionId) → Places a temporary hold and escalates to fraud team.
3. **Query Handling**
    - handleQuery(sessionId, queryOrIntent) → Main function that maps free-text queries or explicit intent strings to the corresponding service function.

- Includes fallback for unrecognized queries.

Example Requests & Responses:

- **Check Balance**

POST `/ivr/conversation`

{

"sessionId": "101", "query": "Check my balance"

}

Response:

{

"success": true, "sessionId": "101", "message": "Your account balance is ₹500."
}

- **Recharge**

POST `/ivr/conversation`

{

"sessionId": "102", "query": "Recharge my account"

}

Response:

{

"success": true, "sessionId": "102", "message": "Recharge successful for ₹100."
}

- **Report Lost Card**

POST `/ivr/conversation`

{

"sessionId": "202", "query": "I lost my card"

}

Response:

{

"success": true, "sessionId": "202", "message": "Your card has been blocked immediately. A customer service agent will call you shortly."

}

Key Improvements:

- Modular intent handling improves maintainability.
- Supports both **DTMF** and **natural language queries**.
- Provides realistic, mock JSON responses for all ACS scenarios.

3. BAP Mock Service

Overview:
BAP simulates **agent-related support** and **transactional requests**. It handles natural language queries and returns realistic responses for each intent.

Location: `services/bapService.js`

Features Implemented:

1. **Agent Requests & Support**
    - Connects users to a live agent.
2. **Transactional & Account Support Intents**
    - getBalance(sessionId) → Requests balance via BAP API.
    - getMiniStatement(sessionId) → Returns last few transactions.
    - payUtilityBill(sessionId) → Initiates bill payment flow.
    - getLoanDetails(sessionId) → Returns loan balance and EMI info.
    - requestEStatement(sessionId) → Sends e-statement to registered email.

Example Requests & Responses:

- **Agent Request**

POST `/ivr/conversation`

{

"sessionId": "103", "query": "I want to talk to an agent"

}

Response:

{

"sessionId": "103", "message": "Connecting you to a live agent."

}

- **Mini Statement**

POST `/ivr/conversation`

{

"sessionId": "104", "query": "Show my last transactions"

}

Response:

{

"sessionId": "104", "message": "Your last five transactions are: a debit of $50, a credit of $200, a debit of $25, a debit of $10, and a credit of $500."

}

- **Pay Utility Bill**

POST `/ivr/conversation`

{

"sessionId": "105", "query": "Pay my electricity bill"

}

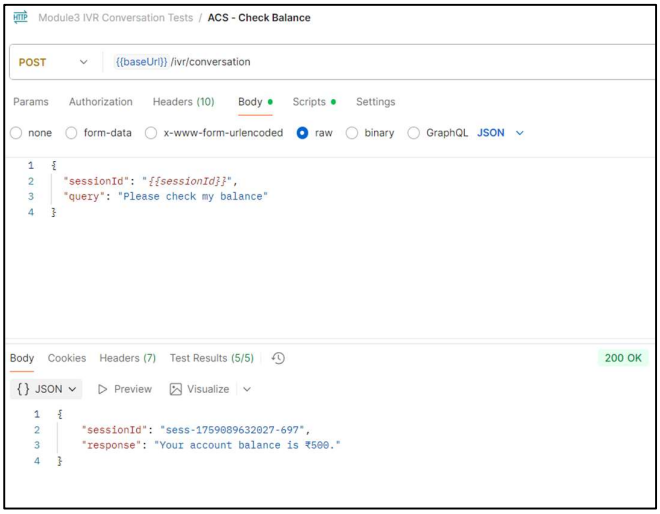
Response:

{

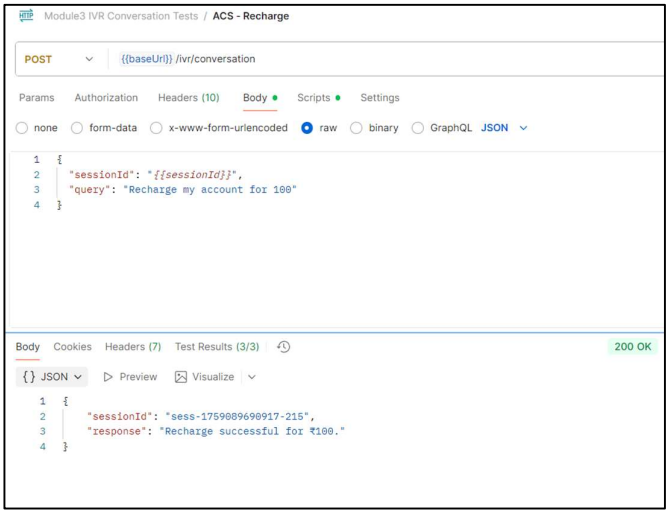
"sessionId": "105", "message": "Please say or enter your 10-digit customer ID for your utility provider."

Step 5 – Testing

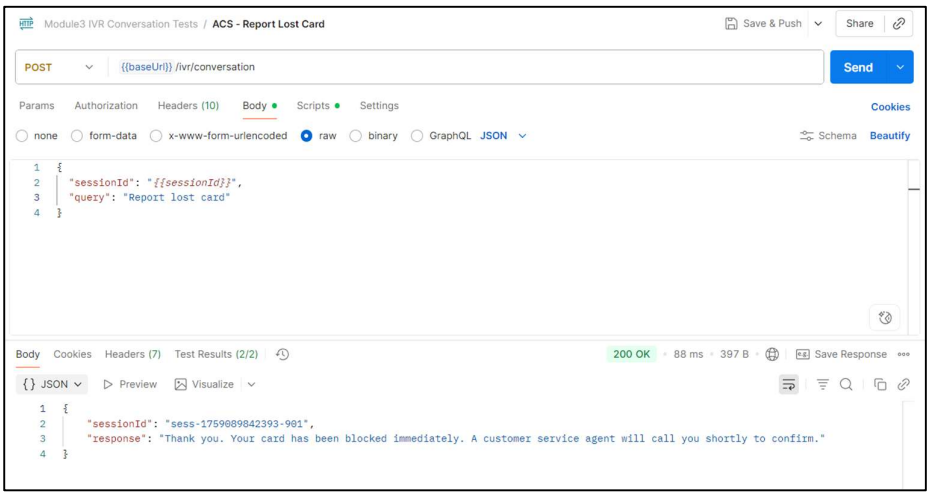
ACS – Check Balance



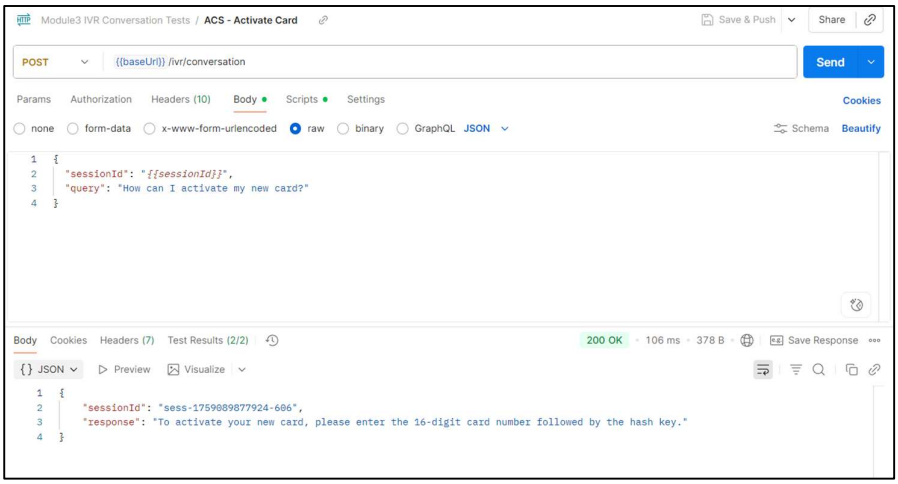
ACS – Recharge



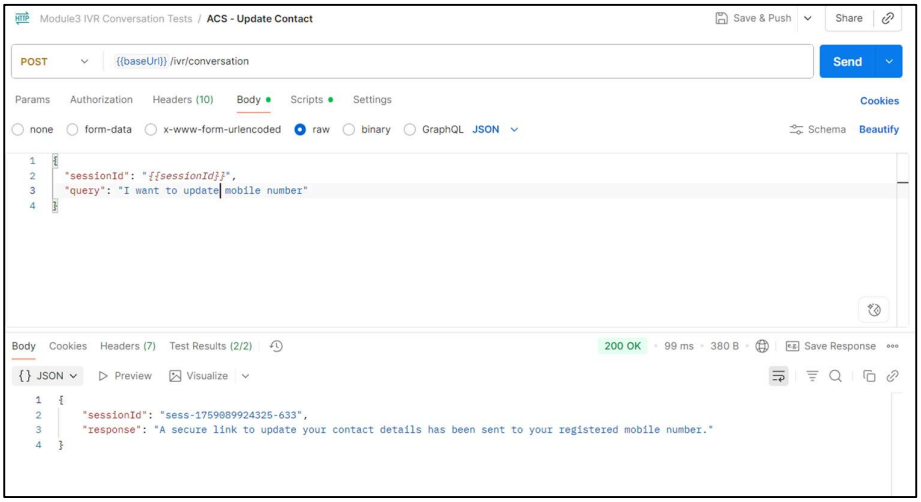
ACS – Report Lost Card



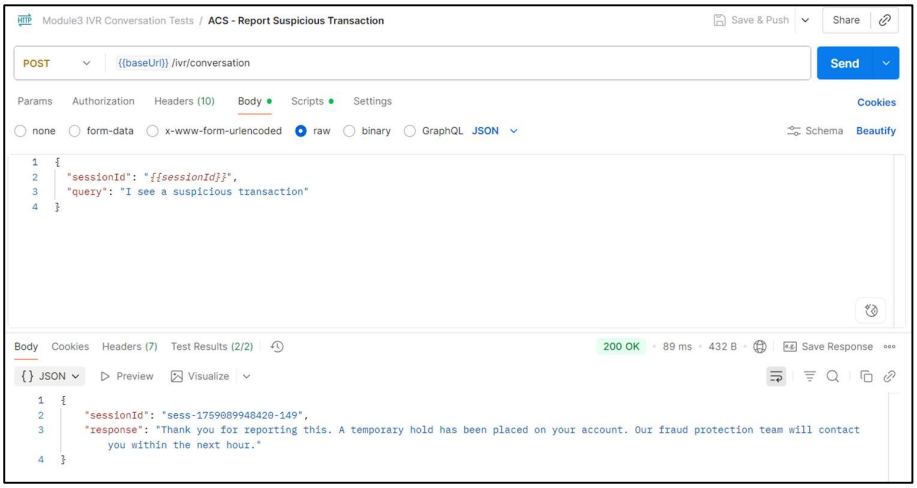
ACS – Activate Card



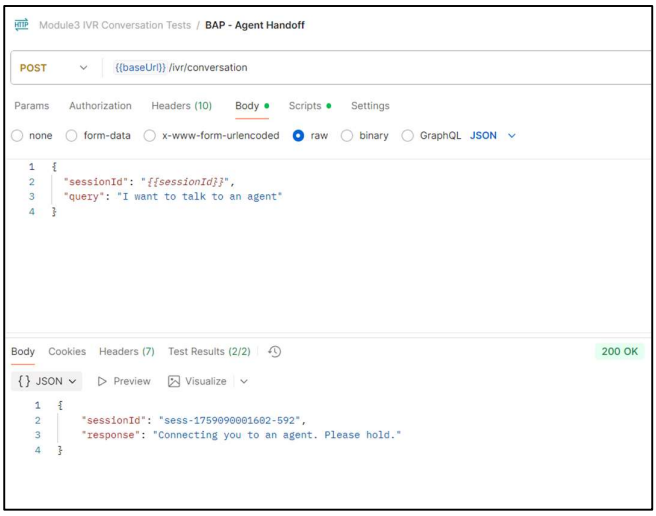
ACS – Update Contact



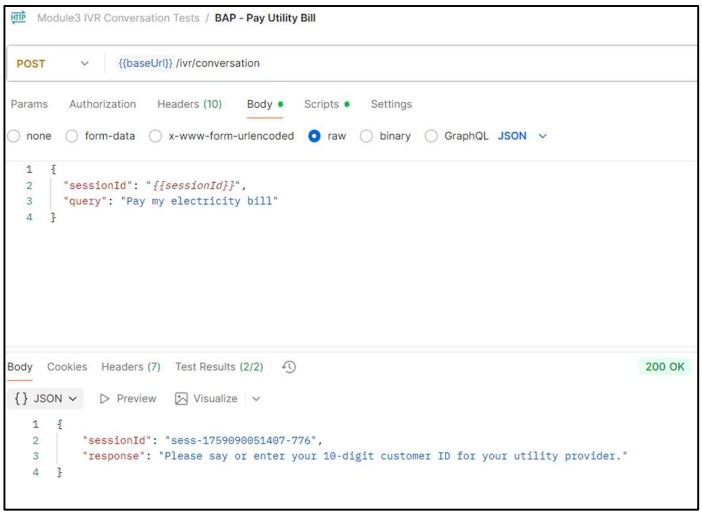
ACS – Report Suspicious Transaction



BAP – Agent Handoff

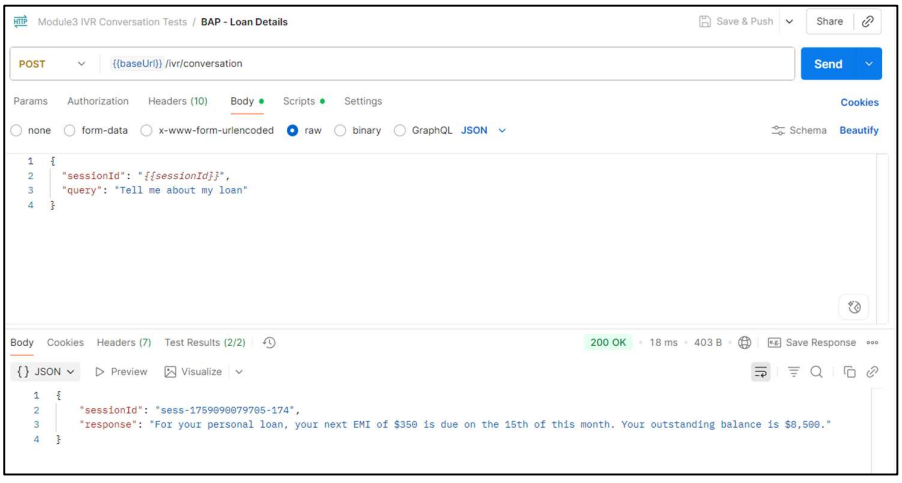


BAP – Pay Utility Bill

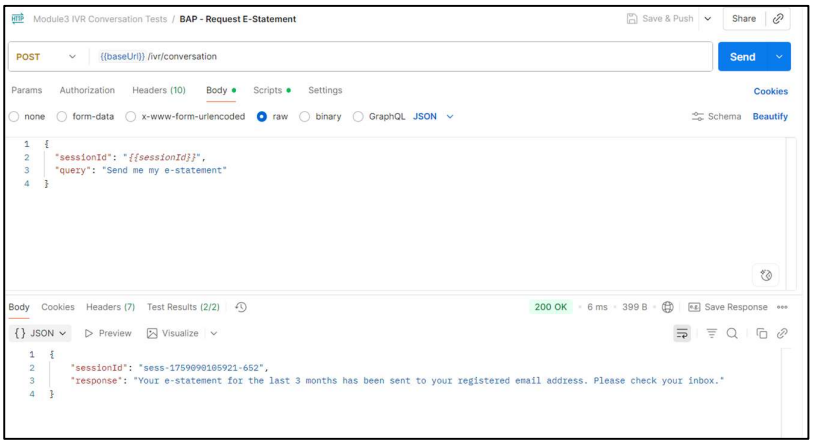


BAP – Mini Statement

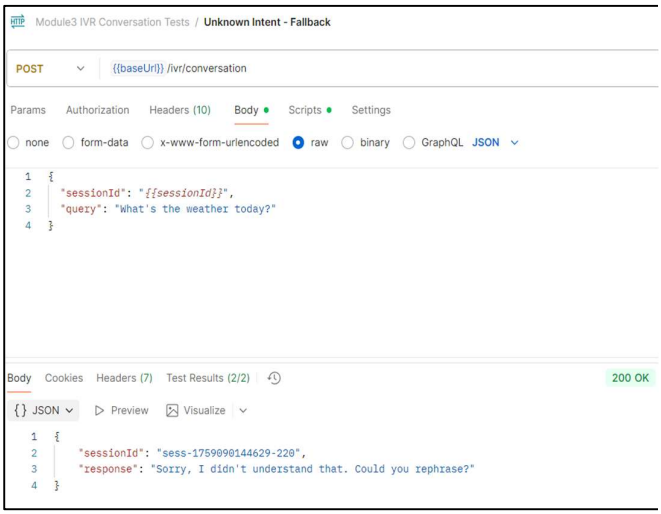
BAP – Loan Details



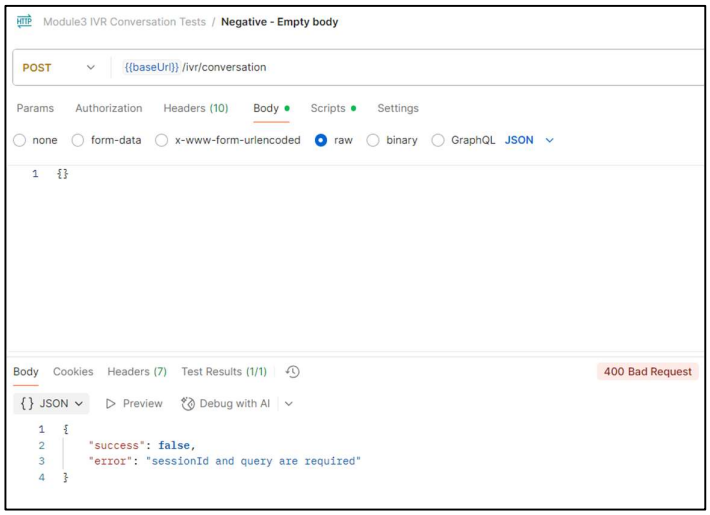
BAP – Request e-statement



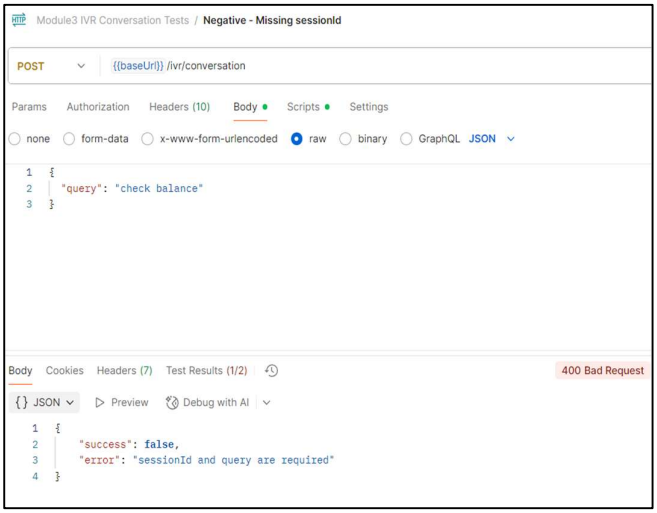
Unknown Intent – Fallback



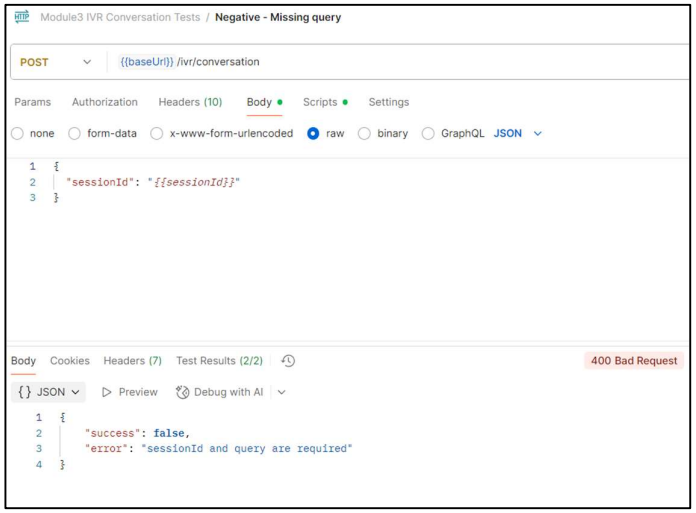
Negative – Empty Body



Negative – Missing sessionId



Negative – Missing Query



Step 6 – Challenges and Learnings

Challenges Faced

1. **Intent Detection Design**
    - Balancing between simple keyword detection and extensibility for future NLP models.
    - Handling overlapping phrases like *"check bill"* (Recharge vs. Utility Bill).
2. **Mock Service Consistency**
    - Ensuring ACS and BAP mock services returned consistent JSON formats across all scenarios.
    - Standardizing responses for QA to avoid breaking Postman tests.
3. **Error Handling**
    - Designing error messages that were clear and reusable.
    - Making sure all negative test cases (missing sessionId, empty body, etc.) behaved uniformly.
4. **Collaboration**
    - Coordinating across different team members (Routes, Controllers, Services, QA) to ensure integration without conflicts.
    - Aligning Postman tests with actual service responses.

Learnings Gained

1. **API-First & Modular Development** – Learned to design clean API contracts and build maintainable middleware using routes, controllers, services, and handlers.
2. **Testing & Validation** – Understood the value of error-driven testing with positive, negative, and fallback cases to ensure reliability.
3. **Collaboration & Real-World Perspective** – Gained hands-on experience in teamwork, documentation, and modernizing legacy IVR systems without disrupting workflows.

Step 7 – Conclusion

This project enabled us to modernize our IVR system by extending it with a conversational interface that supports natural language queries. By introducing the /ivr/conversation endpoint, implementing a keyword-based intent handler, and enhancing ACS and BAP mock services, we created a flexible framework capable of handling both account-related and support-related tasks. Comprehensive Postman testing ensured that positive, negative, and fallback cases were consistently validated. This milestone not only strengthened our technical skills in API-first development, middleware integration, and error handling but also gave us practical exposure to collaborative teamwork in building real-world conversational AI systems.

Next Steps — Milestone 4: Final Deployment & Validation

Objective: Deploy the conversational IVR system in a live environment and validate its performance, stability, and reliability.

Planned Tasks:
• Set up production environment with CI/CD and secure configurations.
• Conduct load testing, negative testing, and end-to-end validation.
• Finalize API documentation and handover materials.

Timeline: Weeks 7–8 — Deployment and performance validation in live environment.

Overall Impact: This milestone delivers a production-ready IVR modernization framework, enhancing customer experience, ensuring scalability, and providing a strong foundation for future AI-driven improvements.