

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

---

## 1. Introduction

Traditional IVR systems are rigid, menu-driven, and difficult to scale. Modern IVR solutions leverage **AI-driven conversational engines** and **cloud telephony platforms** to provide natural, flexible, and intelligent voice interactions.

In this project, a modern IVR integration architecture is designed using **Twilio** for voice call handling and **Rasa** for conversational intelligence. A custom **Integration Layer** bridges VoiceXML-style IVR flows with Rasa APIs, enabling real-time communication.

---

## 2. Problem Statement

- Existing IVR systems rely on static VXML menus
  - No conversational intelligence or context awareness
  - Hard to integrate with modern AI platforms
  - Poor user experience due to long call flows
- 

## 3. Objective

The objectives of Module 2 are:

- To design and implement APIs enabling communication between **VXML-style IVR logic and Rasa**
- To integrate **Twilio voice services** with backend conversational logic

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- To ensure **real-time data handling** and compatibility
  - To validate the integration using sample IVR transactions and flow testing
- 

## 4. Chosen Approach (Single Approach)

- ✓ Twilio for Voice Call Handling
  - ✓ Rasa for Conversational AI
  - ✓ FastAPI-based Integration Layer
- 

## 5. Role of Twilio and Rasa

### 5.1 Purpose of Twilio

- Handles incoming and outgoing voice calls
- Collects DTMF inputs or speech
- Sends user input to Integration Layer APIs
- Plays dynamic voice prompts returned from backend

### 5.2 Purpose of Rasa

- Processes user intent and context
- Manages conversational state
- Generates intelligent responses

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- Replaces static IVR decision trees
- 

## 6. System Architecture

### 6.1 Architecture Overview

Caller

↓

Twilio Voice (IVR / DTMF / Speech)

↓

Integration Layer (FastAPI APIs)

↓

Rasa Conversational Engine

↓

Response back to Twilio

↓

Voice response to Caller

---

## 7. Integration Layer Design

The Integration Layer acts as a **middleware connector** between Twilio and Rasa.

### Key Responsibilities:

- Receive IVR input from Twilio
- Maintain session context
- Forward input to Rasa REST API
- Return conversational response to Twilio

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- Ensure real-time response handling
- 

## 8. API Design and Implementation

### 8.1 Root API (Health Check)

#### Endpoint

GET /

#### Purpose

- Confirms Integration Layer is running
- Displays active IVR sessions

#### Response

```
{  
  "status": "Integration Layer Running",  
  "active_sessions": 1  
}
```

---

### 8.2 IVR Input API

#### Endpoint

POST /ivr/input

#### Purpose

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- Receives IVR input from Twilio
- Routes request to Rasa
- Controls IVR conversational flow

## Request

```
{  
  "session_id": "unique-session-id",  
  "input_value": "1"  
}
```

## Response

```
{  
  "prompt": "Press 1 to book appointment. Press 2 to check status.  
Press 0 to go back.",  
  "state": "APPOINTMENTS",  
  "real_time": "YES",  
  "integration": "Twilio → API → Rasa"  
}
```

---

## 9. IVR Conversational Flow

### States Implemented

- START
- MAIN\_MENU
- APPOINTMENTS
- STATUS

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- END

## Conversational Loop

1. User calls via Twilio
2. Twilio captures input
3. Integration API sends input to Rasa
4. Rasa predicts intent and next action
5. Response returned to Twilio
6. Twilio plays voice prompt
7. Loop continues until call ends

---

## 10. Real-Time Data Handling

- FastAPI ensures low-latency responses
- JSON-based communication
- Stateless APIs with session tracking
- Compatible with cloud and on-prem deployments

---

## 11. Sample Transaction & Flow Testing

### Test Using cURL

```
curl -X POST http://127.0.0.1:8000/ivr/input \
```

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

```
-H "Content-Type: application/json" \  
-d '{"session_id":"123","input_value":"1"}'
```

## Validation Results

Parameter	Result
API Response	Successful
State Transition	Correct
Real-Time Handling	Yes
Twilio Compatibility	Verified
Rasa Integration	Logical

---

## 12. Existing vs Proposed System

Feature	Existing IVR	Proposed IVR
Menu Type	Static	Conversational
Intelligence	None	AI-powered
Flexibility	Low	High
Scalability	Limited	Cloud-based
User Experience	Poor	Improved

---

## 13. Constraints and Compatibility Issues

- Twilio trial limitations (call duration, trial message)

# MODULE 2: Integration Layer Design (Rasa + Twilio Based IVR)

- Rasa requires proper training data
  - Voice quality depends on network
  - In-memory session storage for prototype
- 

## 14. Advantages of Proposed System

- AI-enabled IVR
  - Scalable cloud architecture
  - Reduced call handling time
  - Natural conversation flow
  - Easy replacement of legacy IVRs
- 

## 15. Conclusion

This module successfully implements a **modern IVR integration layer** using **Twilio and Rasa**, replacing legacy ACS/BAP systems. The architecture enables real-time conversational IVR experiences and provides a scalable foundation for intelligent voice assistants in hospital management systems.