BURIDI VENKATA SAI SRINAGA NIKHITA
2022BCSE031
BTECH CSE
NIT SRINAGAR

TELUGU GOVERNMENT SCHEME VOICE ASSISTANT

ARCHITECTURE DOCUMENT

1. Executive Summary

This document describes the architecture of the **Telugu Government Scheme Voice Assistant**, an AI-powered system that enables Telugu-speaking citizens to discover eligible government schemes through natural voice conversation. The system uses a **state-driven agent architecture** with integrated voice processing, eligibility checking, and conversation memory.

2. System Overview

2.1 High-Level Architecture

The system follows a **modular, pipeline-based architecture** where user voice input flows through distinct processing stages to produce personalized scheme recommendations in Telugu.

Component Flow:

1. **Voice Input** → 2. **Speech Processing** → 3. **Agent Logic** → 4. **Tool Execution** → 5. **Response Generation** → 6. **Voice Output**

2.2 Key Architectural Principles

- **Voice-First Design:** Optimized for Telugu speech interaction
- **State-Driven Conversations:** Guided dialogue using finite state machine
- **Modular Independence:** Each component replaceable/upgradable independently
- **Memory-Aware:** Context retention across conversation turns
- **Offline-First:** Minimal external dependencies

3. Agent Lifecycle

3.1 Lifecycle Stages

Stage 1: Initialization

- Load speech recognition models
- Initialize conversation memory
- Load scheme database (50+ schemes)
- Set initial state: START

Stage 2: Listening & Processing

- Capture Telugu audio (10-second timeout)
- Convert speech to text using custom Telugu model
- Handle background noise suppression

Stage 3: Analysis & Planning

- Extract user information (age, occupation, income via regex patterns)
- Determine conversation goal (find_schemes, get_details, apply)
- Check for contradictions with stored memory
- Create execution plan based on goal

Stage 4: Execution & Evaluation

- Execute eligibility checking via Tool1
- Generate recommendations via Tool2
- Evaluate results against success criteria
- Prepare Telugu response using template system

Stage 5: Response & Continuation

- Generate natural Telugu response
- Update conversation history
- Transition to next state or end conversation

3.2 Lifecycle Diagram

text

[Initialization] → [Listening] → [Processing] → [Analysis] → [Planning] →

[Execution] → [Evaluation] → [Response] → [Continue/End]

---

4. Decision Flow Architecture

4.1 State Machine Design

The agent uses a **finite state machine (FSM)** with 7 primary states:

text

START → ASK_OCCUPATION → ASK_AGE → ASK_INCOME →

CHECK_ELIGIBILITY → RECOMMEND_SCHEMES → END

BURIDI VENKATA SAI SRINAGA NIKHITA
2022BCSE031
BTECH CSE
NIT SRINAGAR

4.2 State Transition Logic

Current State	Trigger Condition	Action	Next State
START	Any user input	Greet & ask occupation	ASK_OCCUPATION
ASK_OCCUPATION	Occupation provided	Validate & ask age	ASK_AGE
ASK_AGE	Age provided	Validate & ask income	ASK_INCOME
ASK_INCOME	Income provided	Validate & run eligibility	CHECK_ELIGIBILITY
CHECK_ELIGIBILITY	Schemes found	Prepare recommendations	RECOMMEND_SCHEMES
CHECK_ELIGIBILITY	No schemes found	Show sorry message	END
RECOMMEND_SCHEMES	Recommendations ready	Present schemes	END

4.3 Error & Edge Case Handling

- **Unclear Input:** Re-ask question with clarification
- **Contradiction Detection:** "మునుపు మీరు 'X' అన్నారు, ఇప్పుడు 'Y' అంటున్నారు"
- **Timeout:** "సమాధానం లేదు. మళ్లీ ప్రారంభిద్దాం."
- **Voice Recognition Failure:** "మీ మాటలు అర్థం కాలేదు. దయచేసి మళ్లీ చెప్పండి."

---

5. Memory System Architecture

5.1 Memory Components

ConversationMemory Class:

- history[]: Sequential log of all interactions
- user_profile{}: Current user information
- session_id: Unique identifier per conversation
- contradictions[]: Detected information conflicts

5.2 Data Structure

python

```
memory_structure = {
  "session": {
    "id": "session_20240115_1030",
    "start_time": "2024-01-15 10:30:00",
    "states_visited": ["START", "ASK_OCCUPATION", "ASK_AGE"],
    "current_state": "ASK_INCOME"
  },
  "user": {
    "occupation": "రైతు",
    "age": 45,
    "income": 300000,
    "extracted_fields": ["occupation", "age"],
    "confidence_scores": {"occupation": 0.95, "age": 0.90}
  },
  "conversation": [
    {
      "turn": 1,
      "user": "నేను రైతుని",
      "agent": "మీ వయస్సు ఎంత?",
      "state": "ASK_OCCUPATION",
      "timestamp": "10:30:05"
    }
  ],
  "contradictions": []
}
```

5.3 Memory Operations

1. **Add Interaction:** Store user input + agent response with state context
2. **Update Profile:** Extract and validate new information
3. **Check Contradictions:** Compare new info with existing profile
4. **Get Context:** Retrieve relevant history for current state
5. **Clear Session:** Reset memory for new conversation

5.4 Contradiction Detection Logic

text

```
IF (field exists in profile) AND (new_value ≠ old_value) THEN
  confidence_old = get_confidence(old_value)
  confidence_new = extract_confidence(new_value)

  IF confidence_new > confidence_old + threshold THEN
    update_profile(field, new_value)
  ELSE
    flag_contradiction(field, old_value, new_value)
    ask_clarification_question()
```

---

6. Prompt Engineering System

6.1 Prompt Categories

Category 1: Greeting & Initialization

text

START: "నమస్కారం! నేను ప్రభుత్వ పథకాల సహాయకుడిని. మీరు ఏ పథకం గురించి తెలుసుకోవాలనుకుంటున్నారు?"

WELCOME_BACK: "మళ్లీ స్వాగతం! మీ గత సంభాషణ నుండి మీ సమాచారం నాకు గుర్తు ఉంది."

Category 2: Information Gathering

text

ASK_OCCUPATION: "మీ వృత్తి ఏమిటి? (రైతు, ఉద్యోగి, వ్యాపారి, విద్యార్థి)"

ASK_AGE: "మీ వయస్సు ఎంత? (సంవత్సరాలలో చెప్పండి)"

ASK_INCOME: "మీ వార్షిక ఆదాయం ఎంత? (లక్షలు లేదా వేలులో చెప్పండి)"

CLARIFICATION: "మరోసారి చెప్పండి, అర్థం కాలేదు."

Category 3: Results & Recommendations

text

FOUND_SCHEMES: "మీకు {count} పథకాలు అర్హత ఉన్నాయి: {scheme_list}"

NO_SCHEMES: "క్షమించండి, ప్రస్తుతం మీరు ఎటువంటి పథకాలకు అర్హులు కాదు."

RECOMMENDATION: "మీకు సిఫార్సు చేస్తున్న పథకం: {scheme}. లాభాలు: {benefits}"

Category 4: Error & System Messages

text

CONTRADICTION: "మునుపు మీరు '{old}' అన్నారు, ఇప్పుడు '{new}' అంటున్నారు. ఏది నిజం?"

VOICE_ERROR: "మీ మాటలు అర్థం కాలేదు. దయచేసి మళ్లీ చెప్పండి."

TIMEOUT: "సమాధానం లేదు. మళ్లీ ప్రారంభిద్దాం."

EXIT: "ధన్యవాదాలు! మళ్లీ కలుద్దాం."

6.2 Template System

Dynamic Variable Insertion:

python

```
response = template.format(
    count=len(schemes),
    scheme_list=", ".join([s['name'] for s in schemes[:3]]),
    benefits=", ".join(top_scheme['benefits'][:2])
)
```

Conditional Prompt Selection:

python

```
def get_prompt(state, context):
  if state == "RECOMMEND_SCHEMES":
    if len(context['schemes']) == 0:
      return prompts["NO_SCHEMES"]
    elif len(context['schemes']) == 1:
      return prompts["SINGLE_SCHEME"].format(scheme=context['schemes'][0]['name'])
    else:
      return prompts["MULTIPLE_SCHEMES"].format(count=len(context['schemes']))
```

6.3 Information Extraction Patterns

Age Extraction Regex:

python

```
patterns = [
  r'నా వయస్సు (\d+)',        # "నా వయస్సు 45"
  r'(\d+) సంవత్సరాలు',       # "45 సంవత్సరాలు"
  r'(\d+) ఏళ్ళు',            # "45 ఏళ్ళు"
  r'నేను (\d+) ఏళ్ళ'         # "నేను 45 ఏళ్ళ"
]
```

Income Extraction Regex:

python

```
patterns = [
  r'(\d+)\s*లక్షలు',         # "3 లక్షలు"
  r'(\d+)\s*లక్ష',           # "3 లక్ష"
  r'(\d+)\s*వేలు',           # "50 వేలు"
  r'సాలికి\s*(\d+)'          # "సాలికి 300000"
]
```

BURIDI VENKATA SAI SRINAGA NIKHITA
2022BCSE031
BTECH CSE
NIT SRINAGAR

7. Tool Integration Architecture

7.1 Tool 1: Eligibility Checker

Purpose: Match user profile against scheme criteria

Input:

python

```
{
    "occupation": "రైతు",    # Farmer
    "age": 45,              # Years
    "income": 300000,       # Annual income in ₹
    "state": "Telangana"    # Optional
}
```

Output:

python

```
[
    {
        "name": "PM Kisan Samman Nidhi",
        "category": "Agriculture",
        "eligibility_score": 0.95,
        "benefits": ["₹6,000/year", "Direct transfer"],
        "criteria_met": ["occupation", "age", "income"]
    }
]
```

Matching Logic:

python

```
def check_eligibility(profile, scheme):
    score = 0
    total_criteria = len(scheme['criteria'])
    if profile['occupation'] in scheme['eligible_occupations']:

        score += 1

    if scheme['min_age'] <= profile['age'] <= scheme['max_age']:

        score += 1

    if profile['income'] <= scheme['max_income']:

        score += 1


    return score / total_criteria
```

7.2 Tool 2: Scheme Recommender

Purpose: Prioritize and personalize scheme recommendations

Input: Eligible schemes list + User profile

Output: Sorted recommendations with priority scores

Scoring Algorithm:

text

```
Priority_Score =

  (Eligibility_Score × 0.4) +

  (Benefit_Value × 0.3) +

  (User_Preference_Match × 0.2) +

  (Application_Simplicity × 0.1)
```

---

8. Deployment Architecture

8.1 Development Environment

text

```
User → Microphone → Local Python → All Components → Speaker
```

8.2 Production Architecture

text

```
                  [Load Balancer]

                         ↓
```

```
    [Agent Instance 1] [Agent Instance 2]

              ↓

      [Shared Memory Cache - Redis]

              ↓

      [Scheme Database - PostgreSQL]

              ↓

    [Speech Service Cluster - 3 nodes]
```

8.3 Dependencies

Core Dependencies:

- Python 3.8+
- SpeechRecognition library
- pyttsx3 (Text-to-speech)
- No external API dependencies (offline-first)

Optional Enhancements:

- Google Speech-to-Text API (improved accuracy)
- AWS Polly (better Telugu TTS)
- PostgreSQL (production database)

---

9. Security & Privacy

9.1 Data Protection Measures

- **Voice Data:** Processed in memory, not stored
- **User Profile:** Session-only, cleared on exit
- **Conversation Logs:** Optional opt-in for improvement

9.2 Security Features

1. **Input Sanitization:** All user input validated
2. **Rate Limiting:** 5 requests/minute per session
3. **Memory Isolation:** Separate memory per session
4. **No PII Storage:** Avoid collecting identifiable information

10. Conclusion

This architecture provides a **robust, scalable foundation** for voice-based government scheme assistance. The **state-driven agent design** ensures natural conversations, while **modular components** enable easy maintenance and extension. The system successfully bridges **technical sophistication** with **practical usability** for Telugu-speaking citizens.

Key Architectural Strengths:

1. **Culturally Adapted:** Designed specifically for Telugu speakers
2. **Modular & Maintainable:** Independent components, easy updates
3. **Scalable Design:** Can grow from prototype to production
4. **Privacy-Focused:** Minimal data retention, user privacy first