

**Topic:** AI based Interactive voice Assistant implemented for welcome calling and SMA accounts

1. **Name of the Project** - EchoAssist: AI driven Customer Outreach

**B. Objective:**

* **Enhance Customer Experience:** Develop a seamless, user-friendly, and efficient voice assistant system that provides a personalized welcome experience for new banking customers and those with SMA accounts.
* **Ensure Compliance and Security:** Implement the voice assistant in a secure and compliant manner to ensure that all customer data is handled according to banking regulations, such as confidentiality and data privacy laws.
* **Offer Account-Specific Information:** Provide customers with real-time information about their SMA accounts, such as status, alerts, and recommendations, through the voice assistant.
* **Collect Feedback and Improve Services:** Enable the system to gather customer feedback during calls and provide insights for further improvements in the bank’s services.
* **Reduce Operational Costs:** Decrease costs associated with traditional manual call centers and improve overall operational productivity.

1. **Features Identified:**

The project involves an AI-based Interactive Voice Assistant implemented for welcome calling and SMA (Special Mention Account) account management in a bank.

1. Welcome Calls for New Customers

* Feature: Automatically identify and call customers who joined the bank the previous day.
* Logic: Query the database to fetch customers whose `Joining\_Date` is equal to `SYSDATE - 1` (yesterday).

2. SMA Account Management

* Feature: Call customers with overdue accounts based on their SMA category.
* SMA Categories:
  + SMA0: 1-30 days overdue.
  + SMA1: 31-60 days overdue.
  + SMA2: 61-90 days overdue
* Calling Frequency:
  + SMA0: Call at month-end (28th or later).
  + SMA1: Call every 15 days (15th and 30th/31st).
  + SMA2: Call every week (every Monday).

3. Database Integration

Stores and manages customer and call data.

**Tables:**

* + **CustomerAccounts** (Stores customer and overdue details)
  + **CallLogs** (Tracks call history and status)

**CustomerAccounts Table Schema**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Description** |
| Customer\_Name | VARCHAR2(100) | Name of the customer |
| Account\_No | NUMBER(14) | Unique account number |
| Overdue\_Date | DATE | Overdue date |
| SMA\_category | VARCHAR2(10) | SMA0, SMA1, or SMA2 |
| Type\_of\_Advance | VARCHAR2(10) | Loan, CC, OD |
| Overdue\_Amount | NUMBER | Amount overdue |
| Joining\_Date | DATE | Account opening date |
| Phone\_Number | VARCHAR2(15) | Customer's phone number |

**CallLogs Table Schema:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Description** |
| Call\_ID | NUMBER (Auto) | Unique Call ID |
| Customer\_Name | VARCHAR2(100) | Name of the customer |
| Account\_No | VARCHAR2(20) | Account number |
| Phone\_Number | VARCHAR2(15) | Customer's phone number |
| Call\_Type | VARCHAR2(20) | Welcome/SMA Call |
| SMA\_Category | VARCHAR2(5) | SMA0, SMA1, SMA2 (NULL for Welcome) |
| Call\_Status | VARCHAR2(20) | Initiated, Completed, Failed |
| Call\_Date | DATE | Date of the call |

4. Twilio Integration

* Automates calls using **Twilio API**.
* Customers receive a call with an interactive message.
* If the call is unanswered, an SMS reminder is sent.
* Call statuses (e.g., **Completed, No-Answer, Failed**) are updated in the database.

5. API for Call Handling (Flask)

* **Flask API Endpoint**: /handle\_response
* Handles customer input when they press **1** for assistance.
* Routes the call to customer support.
* Uses **Postman** for testing.

6. Exposing API via ngrok

* **ngrok** is used to provide a public URL for the Flask API.
* This allows Twilio to send webhook requests to the local server.

Rules and Constraints:

* **Unique Account Numbers**: Ensures no duplicate account numbers are generated.
* **SMA Category Rules**: For “Type\_of\_Advance” = "CC" or "OD", the SMA category cannot be "SMA0". It is randomly assigned to "SMA1" or "SMA2".
* **Calling Logic:** Calls are made based on the SMA category and predefined frequencies.

Updated Workflow

1. **Database Query:** Fetch eligible customers.
2. **Call Initiation:** Twilio calls the customer.
3. **Customer Response:**
   * Press **1** → Redirect to customer care.
   * No response → SMS reminder sent.

**Call Log Update:** Call status is recorded in Oracle.

Testing & Execution

* Flask is run locally (python app.py).
* ngrok generates a public URL (ngrok http 5000).
* Twilio webhook is configured with ngrok URL.
* API tested in **Postman**.
* Twilio logs verify call details.

**D. Model Output**

1. Synthetic Data Generation

Output: 500 synthetic customer records are inserted into the `CustomerAccounts` table in the Oracle database.

Confirmation Message

✅ 500 synthetic records inserted successfully. No NULL values exist.

2. Welcome Calls

Output: Simulated welcome calls for customers who joined the bank the previous day.

Confirmation Message:

✅ Welcome calls completed successfully!

Example Call:

📞 Calling Aarav Sharma (Account No: 12345678901234)

Hello Aarav Sharma, welcome to our bank! We are glad to have you onboard. If you need any assistance, feel free to contact our support team.

3. SMA Account Calls

Output: Simulated calls for customers with overdue accounts based on their SMA category and calling frequency.

Confirmation Message:

✅ SMA customer calls completed successfully!

Example Call:

📞 Calling Priya Nair (Account No: 98765432109876)

Dear Priya Nair, this is a reminder from your bank regarding an overdue amount of ₹50000. Please make the payment at the earliest to avoid penalties. Press 1 for assistance.

# Recent Updates

## 1. Twilio Integration

Twilio has been integrated to automate voice calls for welcome messages and SMA payment reminders. A purchased Twilio number is used for making calls. The call flow includes pressing 1 for customer support and handling responses via a Flask webhook.

## 2. Postman Testing

Postman was used to test the '/handle\_response' API by sending simulated user inputs. Successful responses confirmed that the Twilio webhook correctly handles user actions.

## 3. Ngrok for Public API Exposure

Ngrok was configured to expose the local Flask server to the internet, allowing Twilio to reach the webhook endpoint. A secure HTTPS forwarding URL was used for Twilio webhook configuration.

## 4. CallLogs Table in Oracle

A new 'CallLogs' table was added to Oracle to store call details, including customer name, account number, phone number, call type (welcome or SMA), SMA category, call status, and call date.

## 5. Call Status Updates & SMS Reminders

The system now fetches real-time call status from Twilio. If a call is missed, an automated SMS is sent as a reminder to the customer.

6. Automating Calls Using Task Scheduler: Implement Windows Task Scheduler (or CRON jobs) to automate the execution of the script at predefined intervals.

7. Multilingual Support: Enable customers to receive call messages in their native language based on their region, improving accessibility and engagement.

Conclusion

**EchoAssist:** AI-driven Customer Outreach effectively automates welcome calls and overdue reminders for banking customers, enhancing engagement and reducing manual effort. By integrating Twilio with a secure Oracle database and Flask API, the system delivers real-time, personalized voice interactions while ensuring data compliance.

Features like SMS reminders, multilingual support, and automated scheduling make it scalable and user-friendly. Overall, EchoAssist streamlines customer communication and sets the stage for smarter, AI-powered banking services.