Conversational AI Bot for Credit Card Distribution – Bank of Baroda Use Case

1. Project Overview

This project focuses on designing and developing a conversational AI chatbot that automates the credit card distribution process for a banking use case. The chatbot interacts with users through contextual conversations, answers frequently asked questions (FAQs), collects customer information, and updates this data in Google Sheets for further processing by the sales team.

Unlike typical paid solutions, this implementation leverages open-source and free tools such as Google's Flan-T5 language model for generating natural language responses and Google Sheets API for seamless data integration.

2. Technologies Used

- **Python**: Core programming language used to build the chatbot backend.

- **Flask**: Lightweight web framework to serve the chatbot UI and API endpoints.

- **Google Flan-T5 (via Hugging Face Transformers)**: Free, open-source large language model (LLM) used to handle conversational AI tasks such as answering FAQs and generating personalized credit card pitch messages.

- **Google Sheets API**: Integrated using a Google Cloud service account to securely read/write customer data from/to a Google Sheet acting as a lightweight CRM.

- **Service Account & OAuth 2.0**: Used for authentication and authorization with Google APIs.

- **HTML/CSS**: For the frontend UI served by Flask templates.

- **.env & .gitignore**: For managing environment variables and excluding sensitive data like credentials from version control.

- **Virtual Environment (venv)**: To isolate Python dependencies.

- **Git & GitHub**: Version control and remote repository for source code management.

3. System Architecture & Workflow

3.1 Chatbot UI

- The user interacts with the chatbot via a web UI served on Flask.

- Input from the user is sent to the backend for processing.

3.2 Conversational AI Engine

- User queries are sent to the Flan-T5 model hosted locally via Hugging Face Transformers.

- The model is fine-tuned or prompted with a predefined knowledge base to answer FAQs and handle context.

- Personalized credit card pitch messages are generated dynamically based on user responses.

3.3 Google Sheets Integration

- The chatbot uses the Google Sheets API to read existing user data and eligibility.
- New customer data collected during the conversation (name, contact info, preferences) is pushed back into the Google Sheet.
- This ensures real-time synchronization of customer records with the distribution team.

3.4 Authentication and Security

- A Google Cloud service account is created and its credentials (JSON) are securely stored locally and excluded from GitHub.
- OAuth 2.0 is used under the hood by the Google Sheets API client to authorize API calls.
- Sensitive data such as service account credentials and environment variables are stored in `.env` files and ignored in `.gitignore`.

3.5 Summary Email (Planned)

- Though not yet implemented, the design includes generating an end-of-conversation summary email to a sales alias for easy follow-up.

4. Integration Details

- **Google Cloud Setup**:
    - Created a project in Google Cloud Console.
    - Enabled Google Sheets API.
    - Created a Service Account with editor access to the specific Google Sheet.
    - Downloaded service account credentials JSON file.
    - Shared Google Sheet with the service account email.
- **Google Sheets API Usage**:
    - Used the `google-auth` and `google-api-python-client` libraries in Python.
    - Authenticated with the service account credentials.
    - Accessed the specific spreadsheet and worksheet by ID and name.
    - Implemented functions to append new rows or update existing data.
- **Conversational Flow**:
    - The Flask backend accepts user input from the frontend.
    - Passes the input to the Flan-T5 model for response generation.

    - Handles conversation context (simple memory implemented using session or state variables).

    - Updates Google Sheets with customer data after critical interactions.

5. Challenges and Solutions

- **Avoiding Paid APIs**:
  Instead of using paid APIs like OpenAI or Gemini, Google's Flan-T5 was used for generating natural language responses, ensuring zero-cost deployment.

- **Maintaining Conversation Context**:
  Limited context window in Flan-T5 was handled by designing concise prompts and managing session state on the Flask server.

- **Google Sheets API Quotas and Access**:
  Ensured proper service account permissions and API enablement. Rate limits are handled gracefully.

- **Security**:
  Credentials and sensitive info are never committed to GitHub and managed via `.gitignore` and `.env` files.

6. How to Run and Test the Bot

Prerequisites

- Python 3.8+ installed

- Virtual environment (recommended)

- Google Cloud service account JSON credentials file

Setup Steps

1.Clone the repository:

```
bash
CopyEdit
git clone
https://github.com/yuriverma/credit_card_chatbot.git

cd credit_card_chatbot
```

2.Create and activate a virtual environment:

```
bash
CopyEdit
python3 -m venv venv

source venv/bin/activate  # on Mac/Linux
```

3.Install dependencies:

```
bash
CopyEdit
pip install -r requirements.txt
```

4.Place the `credentials.json` file (Google service account credentials) in the root directory.

5.Set environment variables in a `.env` file (example):

```ini
CopyEdit
GOOGLE_SHEETS_CREDENTIALS=credentials.json

SHEET_ID=your_google_sheet_id
```

6.Run the Flask app:

```bash
CopyEdit
python app.py
```

7.Open your browser and go to `http://localhost:5000` to interact with the chatbot.

7. Future Improvements

- **Implement email summaries** after each conversation.
- **Deploy the bot on a cloud platform** for remote access.
- **Enhance context management** using advanced memory techniques or stateful LLM agents.
- **Add CRM integration** beyond Google Sheets, e.g., HubSpot or Zoho.
- **Incorporate voice platform integration** for multimodal interaction.

8. Conclusion

This project successfully demonstrates building a fully functional conversational AI chatbot using free, open-source technologies. It automates customer engagement and data capture for credit card distribution in a scalable and cost-effective manner, providing a strong foundation for further enhancements and deployment.