Project Report: Conversational AI with Groq API

1. Project Overview

This project demonstrates the development of a **conversational AI system** using the **Groq OpenAI-compatible API**. The system allows for:

- Sending and receiving AI chat responses.
- Maintaining a **conversation history**.
- Automatically summarizing long conversations.
- Extracting structured user information from unstructured text using a **JSON schema**.

The project emphasizes simplicity, modularity, and real-world usability, making it suitable for chatbots, virtual assistants, and AI-based customer support systems.

---

2. Technologies Used

- **Python 3.13**
- Libraries: openai, requests, json, getpass, os
- Groq API (OpenAI-compatible endpoint)
- JSON schema for structured data extraction

---

3. Key Features

3.1 Chat Interaction

- Users can send messages to the AI assistant, and receive responses in real-time.
- Example interaction:
- User: Hello, can you help me with Python?
- Assistant: Sure! What do you need help with?

3.2 Conversation History

- The system keeps a **running log** of all messages.
- Supports **truncation**:
  - By number of turns: retains only the last n messages.
  - By character length: ensures the conversation does not exceed a maximum character limit.

3.3 Conversation Summarization

- Long conversations are automatically summarized **every k messages** to reduce memory usage.
- Example summary:

- "Summary: The user is learning Python basics, covering variables, data types, and lists, and has now inquired about dictionaries."

3.4 JSON-Based User Information Extraction

- Extracts structured user info from unstructured text using a **Groq API function call**.
- Fields extracted:
    - name
    - email
    - phone
    - location
    - age
- Example:
- {
-   "name": "Shubham Khedekar",
-   "email": "shubham@example.com",
-   "phone": "9876543210",
-   "location": "Pune",
-   "age": 24
- }

---

4. Implementation Highlights

1. **Reusable API Function**
2. def send_groq_request(messages, model="llama-3.3-70b-versatile", max_tokens=50):
3.   ...
    - Handles all requests to Groq API.
    - Simplifies sending messages, summarization, and extraction tasks.
4. **Conversation Management**
    - Maintains history with add_message.
    - Supports truncation by turns or character count.
5. **Automatic Summarization**
    - add_message_with_summary triggers summarization every k messages.
    - Keeps conversations concise and manageable.
6. **Structured Data Extraction**

  - Uses JSON schema to extract name, email, phone, location, and age.
  - Converts unstructured chat into structured data for analytics or CRM integration.

---

5. Sample Output

Conversation

system: Summary: The user is learning Python basics, covering variables, data types, and lists, and has now inquired about dictionaries.

assistant: Dictionaries are key-value pairs in Python.

User Info Extraction

```
{
  "name": "Shubham Khedekar",
  "email": "shubham@example.com",
  "phone": "9876543210",
  "location": "Pune",
  "age": 24
}
```

---

6. Key Learnings

- How to integrate **Groq API** for AI-based chat.
- Efficient **conversation management** with history, truncation, and summarization.
- Practical use of **JSON schemas** for structured data extraction from unstructured text.
- Writing modular Python code that is beginner-friendly and maintainable.