GeminiClient LLM API Integration

This Java project demonstrates how to integrate and interact with the Gemini LLM (Large Language Model) API from OpenRouter.ai to get AI chat completions. The project provides a simple command-line interface where users can chat with the LLM, track their question history, and limit the number of questions per user.

# Features

* Connects to Gemini API (`google/gemini-flash-1.5` model) using REST HTTP POST requests.
* User authentication via API key stored in environment variable `GEMINI\_API\_KEY`.
* Supports multiple users by username.
* Tracks the number of questions asked by each user (limit set to 5).
* Stores and displays chat history per user.
* Validates user questions for a minimum length, presence of letters, and question mark ending.
* Simple CLI menu with options to start chatting, view history, or exit.
* Clean separation of concerns by defining an interface for the LLM service.

# Prerequisites

Java 8 or above installed.  
Internet connection for API access.  
A valid API key from OpenRouter.ai or your Gemini API provider.  
Set the environment variable `GEMINI\_API\_KEY` with your API key before running the program.  
  
Example (Linux/macOS):  
export GEMINI\_API\_KEY="your\_api\_key\_here"  
  
Example (Windows CMD):  
set GEMINI\_API\_KEY=your\_api\_key\_here

# Project Structure

## LLMService.java

Defines a simple interface for Large Language Model services with the method:  
  
String getChatResponse(String userMessage) throws Exception;  
  
This abstraction allows easy swapping or mocking of different LLM providers.

## GeminiClient.java

Implements LLMService interface to interact with the Gemini API. Handles HTTP requests, authentication, JSON parsing, and returns the LLM's chat response.

## LLMChatApp.java

Provides a command-line interface for users to:  
- Enter username and chat with the LLM.  
- View their chat history.  
- Exit the program.

# How to Run

1. Clone or download this repository.  
2. Ensure `GEMINI\_API\_KEY` environment variable is set.  
3. Compile the Java classes:  
javac -cp gson-<version>.jar org/example/\*.java  
(Ensure you have Gson library in your classpath.)  
4. Run the main application:  
java -cp .:gson-<version>.jar org.example.LLMChatApp  
(On Windows, replace ':' with ';' in the classpath.)

# Usage

- Press 1: Enter your username and start chatting with the LLM. You can ask up to 5 questions per user.  
- Press 2: View previous chat history by entering your username.  
- Press 3: Exit the application.

## Chatting Guidelines

* Questions must be at least 5 characters long.
* Must contain at least one letter.
* Must end with a question mark `?`.
* Type `exit` anytime during chat to end the session.

# Example Interaction

📢 Press 1 to start interacting with the LLM  
📜 Press 2 to view previous chat history  
🚪 Press 3 to exit  
Enter your choice: 1  
👤 Enter your username: alice  
👋 Hello, alice! You’ve already asked 0 question(s).  
💬 Type 'exit' to end the chat.  
📝 You: What is the capital of France?  
🤖 LLM: The capital of France is Paris.  
📝 You: exit  
👋 Chat ended.

# Error Handling

- If the API key is missing or invalid, the program will prompt with:  
 ❌ API key is missing. Please set the GEMINI\_API\_KEY environment variable.  
- If the API returns an error, it will be displayed in the console.  
- Input validation messages guide users to enter valid questions.

# Dependencies

- Gson - For JSON parsing.  
Add the Gson library jar to your classpath when compiling and running the program.

# Notes

- This example uses a synchronous HTTP connection via HttpURLConnection.  
- The API endpoint and model are hardcoded but can be extended for flexibility.  
- The chat history and question count are stored in-memory and reset on program restart.  
- The interface LLMService enables easy extension or replacement of the underlying LLM provider without changing the rest of the app.

# License

This project is open-source and free to use.

Happy chatting with Gemini! 🚀