# Linux AI Help Chat using Groq API

## Problem Statement:

New Linux users often get confused by commands like `chmod`, `grep`, or `tar`. They need simple, plain-English explanations. This chatbot provides exactly that — instantly.

## 1. Introduction

This paper documents the design, development, and troubleshooting of a simple CLI-based chatbot that explains Linux commands in plain English using Groq's free API. The goal is to make Linux more accessible for beginners and contribute to cybersecurity education.

## Step-by-Step Process

### Step 1: Get a Groq API Key

- Go to https://console.groq.com/keys  
- Create an account or sign in with GitHub  
- Generate and copy your free API key

### Step 2: Install Dependencies

Install the requests library:  
  
pip install requests

### Step 3: Write the Python CLI Bot (≤ 15 lines)

import requests  
  
GROQ\_API\_URL = "https://api.groq.com/openai/v1/chat/completions"  
headers = {"Authorization": "Bearer YOUR\_API\_KEY"}  
system = "Explain any Linux command in plain English for beginners."  
  
while True:  
 cmd = input("🔍 Enter Linux command: ")  
 if cmd.lower() in ["exit", "quit"]:  
 break  
 data = {  
 "model": "mixtral-8x7b-32768",  
 "messages": [  
 {"role": "system", "content": system},  
 {"role": "user", "content": f"What does '{cmd}' do in Linux?"}  
 ]  
 }  
 res = requests.post(GROQ\_API\_URL, headers=headers, json=data)  
 print("🤖", res.json()["choices"][0]["message"]["content"].strip())

## 2. Initial Implementation and Issues

The initial chatbot used the Mixtral model (`mixtral-8x7b-32768`). The API request was made using the `requests` library. However, the model was found to be decommissioned, resulting in an API error:  
  
Error: 'choices'  
Full Response: {"error":{"message":"The model `mixtral-8x7b-32768` has been decommissioned and is no longer supported..."}}

## 3. Solution: Switching to a Supported Model

Groq recommends newer models like `llama-3.1-8b-instant` or `gemma2-9b-it`. The script was updated to use `llama-3.1-8b-instant` for compatibility and performance.  
  
Updated model section in the script:

data = {  
 "model": "llama-3.1-8b-instant",  
 "messages": [  
 {"role": "system", "content": system},  
 {"role": "user", "content": f"What does '{cmd}' do in Linux?"}  
 ]  
}

## 4. Final Working Script with Error Handling

Below is the corrected script with error handling and support for Groq's active models:

import requests  
  
GROQ\_API\_URL = "https://api.groq.com/openai/v1/chat/completions"  
headers = {"Authorization": "Bearer YOUR\_VALID\_GROQ\_KEY"}  
system = "Explain any Linux command in plain English for beginners."  
  
while True:  
 cmd = input("🔍 Enter Linux command: ")  
 if cmd.lower() in ["exit", "quit"]:  
 break  
 data = {  
 "model": "llama-3.1-8b-instant",  
 "messages": [  
 {"role": "system", "content": system},  
 {"role": "user", "content": f"What does '{cmd}' do in Linux?"}  
 ]  
 }  
 res = requests.post(GROQ\_API\_URL, headers=headers, json=data)  
 try:  
 print("🤖", res.json()["choices"][0]["message"]["content"].strip())  
 except Exception as e:  
 print("❌ Error:", e)  
 print("📦 Full Response:", res.text)

## 5.Line-by-Line Code Explanation

| Line | Code | Explanation |
| --- | --- | --- |
| 1 | import requests | Imports the HTTP library to send API requests |
| 3 | GROQ\_API\_URL = ... | Sets the API endpoint for Groq's chat system |
| 4 | headers = {...} | Adds your API key for authorization ( required) |
| 5 | system = ... | Defines AI behavior: “Explain Linux commands simply” |
| 7 | while True: | Keeps chatbot running in a loop |
| 8 | cmd = input(...) | Accepts user input (Linux command) from terminal |
| 9 | if cmd.lower()... | Allows user to quit by typing `exit` or `quit` |
| 10–14 | data = {...} | Builds the request body with model and message history |
| 15 | res = requests.post(...) | Sends the API call to Groq’s Mixtral model |
| 16 | print(...) | Displays the AI’s explanation in the terminal |

## Sample Demo Output



## 6. Conclusion

By using Groq's API and a small Python script, it’s possible to create an educational chatbot that supports self-paced learning. Updating deprecated models and adding error handling ensures the system remains reliable and easy to maintain.