LOGISTIC\_CHATBOT

> \_pycache\_

> .streamlit

secret.toml:

GROQ\_API\_KEY = "gsk\_TiJoANcmwEHm7PHaZtN0WGdyb3FYtRXqdkrnSxpt3ks22wjG3voP"

---

### 🎨 Giao diện đẹp hơn - Theme tùy chỉnh (Thêm vào `.streamlit/config.toml`)

```toml

[theme]

base="light"

primaryColor="#4CAF50"

backgroundColor="#FFFFFF"

secondaryBackgroundColor="#F0F2F6"

textColor="#262730"

font="sans serif"

> .venv

> assets

> history

> memory \_files

> uploads

app.py

import streamlit as st

import pandas as pd

import os

from datetime import datetime

from utils import (

ask\_gpt,

ask\_gpt\_with\_data,

save\_chat\_history,

clear\_chat\_history,

validate\_dataframe

)

*# ---- Constants ----*

MAX\_FILE\_SIZE\_MB = 10 *# Increased for larger files*

ALLOWED\_FILE\_TYPES = ["xlsx", "xls", "csv"]

MAX\_DISPLAY\_ROWS = 100 *# For preview purposes*

*# ---- Setup page ----*

st.set\_page\_config(

page\_title="Logistics Analyst",

page\_icon="📊",

layout="centered",

menu\_items={

'About': "### Logistics Data Analyst v3.0\nAI-powered data insights"

}

)

st.title("📊 Logistics Data Analyst")

*# ---- Initialize Session State ----*

if "messages" not in st.session\_state:

st.session\_state.messages = []

if "uploaded\_df" not in st.session\_state:

st.session\_state.uploaded\_df = None

if "file\_name" not in st.session\_state:

st.session\_state.file\_name = None

if "data\_summary" not in st.session\_state:

st.session\_state.data\_summary = None

*# ---- Sidebar ----*

with st.sidebar:

st.header("⚙️ Data Settings")

*# File Uploader with enhanced handling*

st.subheader("📁 Upload Logistics Data")

uploaded\_file = st.file\_uploader(

"Upload your logistics data file",

type=ALLOWED\_FILE\_TYPES,

help=**f**"Supports Excel and CSV files up to {MAX\_FILE\_SIZE\_MB}MB"

)

if uploaded\_file:

try:

file\_size = uploaded\_file.size / (1024 \* 1024) *# MB*

if file\_size > MAX\_FILE\_SIZE\_MB:

st.error(**f**"File too large. Max size: {MAX\_FILE\_SIZE\_MB}MB")

else:

with st.spinner("Analyzing data..."):

if uploaded\_file.name.endswith('.csv'):

df = pd.read\_csv(uploaded\_file)

else:

df = pd.read\_excel(uploaded\_file)

validate\_dataframe(df)

st.session\_state.uploaded\_df = df

st.session\_state.file\_name = uploaded\_file.name

st.success(**f**"✅ {uploaded\_file.name} loaded successfully!")

with st.expander("🔍 Data Preview"):

st.dataframe(df.head(min(len(df), MAX\_DISPLAY\_ROWS)))

st.caption(**f**"Shape: {df.shape[0]} rows, {df.shape[1]} columns")

except Exception as e:

st.error(**f**"❌ Error loading file: {str(e)}")

st.session\_state.uploaded\_df = None

*# Analysis Controls*

st.subheader("🔍 Analysis Options")

analysis\_depth = st.selectbox(

"Analysis Depth",

["Quick Overview", "Detailed Analysis", "Deep Examination"],

help="Choose how thoroughly to analyze the data"

)

*# Chat Management*

st.subheader("💬 Conversation")

if st.button("🧹 Clear Conversation", use\_container\_width=True):

st.session\_state.messages = []

clear\_chat\_history()

st.rerun()

*# ---- Display Chat History ----*

for message in st.session\_state.messages:

avatar = "🛒" if message["role"] == "user" else "📊"

with st.chat\_message(message["role"], avatar=avatar):

st.markdown(message["content"])

if "data\_insights" in message:

with st.expander("View Data Insights"):

st.write(message["data\_insights"])

*# ---- Chat Input ----*

if prompt := st.chat\_input("Ask about the logistics data..."):

*# Display user message*

with st.chat\_message("user", avatar="🛒"):

st.markdown(prompt)

*# Generate and display assistant response*

with st.chat\_message("assistant", avatar="📊"):

with st.spinner("🔍 Analyzing data..."):

try:

if st.session\_state.uploaded\_df is not None:

*# Data-aware analysis*

response = ask\_gpt\_with\_data(

prompt=prompt,

df=st.session\_state.uploaded\_df,

system\_content=**f**"""

You are a Logistics Data Analyst. Provide {analysis\_depth} of this data.

Include specific numbers and actionable insights when possible.

"""

)

else:

*# General logistics knowledge*

response = ask\_gpt(

prompt=prompt,

system\_content="You are a Logistics Expert. Provide helpful information."

)

st.markdown(response)

*# Save to message history*

message = {

"role": "assistant",

"content": response,

"timestamp": datetime.now().isoformat()

}

if st.session\_state.uploaded\_df is not None:

message["data\_insights"] = {

"file": st.session\_state.file\_name,

"shape": st.session\_state.uploaded\_df.shape

}

st.session\_state.messages.append({"role": "user", "content": prompt})

st.session\_state.messages.append(message)

save\_chat\_history(st.session\_state.messages)

except Exception as e:

st.error(**f**"⚠️ Analysis failed: {str(e)}")

*# ---- Data Summary Section ----*

if st.session\_state.uploaded\_df is not None:

with st.expander("📈 Data Summary", expanded=False):

tab1, tab2 = st.tabs(["Statistics", "Sample Data"])

with tab1:

st.subheader("Statistical Summary")

st.write(st.session\_state.uploaded\_df.describe())

with tab2:

st.subheader("Data Sample")

st.dataframe(st.session\_state.uploaded\_df.sample(min(10, len(st.session\_state.uploaded\_df))))

requirements.txt

streamlit==1.32.0

pandas==2.1.4

openpyxl==3.1.2

python-dotenv==1.0.0

groq==0.5.0

protobuf==3.20.0

utils.py

import os

import json

import pandas as pd

from groq import Groq

from dotenv import load\_dotenv

from typing import List, Dict, Optional

from datetime import datetime

*# Load environment variables*

load\_dotenv()

**class** ChatbotError(Exception):

"""Custom exception for chatbot errors"""

pass

**def** get\_groq\_client() -> Groq:

"""Initialize and return Groq client with API key"""

api\_key = os.getenv("GROQ\_API\_KEY")

if not api\_key:

raise ChatbotError("GROQ\_API\_KEY not found in environment variables")

return Groq(api\_key=api\_key)

**def** summarize\_data(df: pd.DataFrame, sample\_size: int = 3) -> str:

"""

Generate a comprehensive summary of the dataframe

Args:

df: DataFrame to summarize

sample\_size: Number of sample values to show for text columns

Returns:

str: Markdown-formatted summary

"""

try:

summary = [

**f**"## Data Summary ({len(df)} rows × {len(df.columns)} columns)",

**f**"\*\*Columns:\*\* {', '.join(**f**'`{col}`' for col in df.columns)}",

**f**"\*\*Missing values:\*\* {df.isna().sum().sum()} total",

]

*# Numeric columns analysis*

numeric\_cols = df.select\_dtypes(include='number').columns

if not numeric\_cols.empty:

summary.append("\n### Numeric Columns")

stats = df[numeric\_cols].describe().transpose()

stats['range'] = stats['max'] - stats['min']

summary.append(stats[['mean', 'min', 'max', 'range', 'std']].to\_markdown())

*# Text columns analysis*

text\_cols = df.select\_dtypes(include=['object', 'string']).columns

if not text\_cols.empty:

summary.append("\n### Text Columns")

for col in text\_cols:

unique\_count = df[col].nunique()

sample\_values = df[col].dropna().sample(min(sample\_size, len(df))).tolist() *# Fixed here*

summary.append(

**f**"- `{col}`: {unique\_count} unique values\n"

**f**" Sample: {', '.join(str(v) for v in sample\_values)}"

)

*# Date columns analysis*

date\_cols = df.select\_dtypes(include=['datetime', 'datetimetz']).columns

if not date\_cols.empty:

summary.append("\n### Date Columns")

for col in date\_cols:

date\_range = **f**"{df[col].min()} to {df[col].max()}"

summary.append(**f**"- `{col}`: {date\_range}")

return "\n".join(summary)

except Exception as e:

raise ChatbotError(**f**"Data summarization error: {str(e)}")

**def** ask\_gpt(prompt: str, system\_content: str = None) -> str:

"""Get response from LLM with proper error handling"""

if not system\_content:

system\_content = """You are an expert Logistics and Supply Chain AI Assistant.

Provide accurate, concise answers with practical recommendations."""

client = get\_groq\_client()

try:

response = client.chat.completions.create(

model="llama3-70b-8192",

messages=[

{"role": "system", "content": system\_content},

{"role": "user", "content": prompt},

],

temperature=0.3,

max\_tokens=4000, *# Increased for data analysis*

top\_p=0.9

)

return response.choices[0].message.content

except Exception as e:

raise ChatbotError(**f**"API Error: {str(e)}")

**def** ask\_gpt\_with\_data(prompt: str, df: pd.DataFrame, system\_content: str = None) -> str:

"""

Enhanced version that includes full data context

Args:

prompt: User question

df: DataFrame to analyze

system\_content: Optional custom system message

Returns:

str: Generated response

"""

if not system\_content:

system\_content = """You are an expert Logistics and Supply Chain AI Assistant

skilled at data analysis and visualization."""

data\_context = summarize\_data(df)

full\_system = **f**"""

{system\_content}

You are analyzing logistics data with these characteristics:

{data\_context}

Provide specific insights from the data when possible.

"""

enhanced\_prompt = **f**"""

When answering this logistics question: {prompt}

Consider this detailed data context:

{data\_context}

Provide specific numbers and insights from the data where relevant.

"""

return ask\_gpt(enhanced\_prompt, full\_system)

**def** save\_chat\_history(history: List[Dict], filename: str = "chat\_history.json") -> None:

"""Save chat history to JSON file"""

try:

with open(filename, "w", encoding="utf-8") as f:

json.dump(history, f, ensure\_ascii=False, indent=2)

except IOError as e:

raise ChatbotError(**f**"Failed to save chat history: {str(e)}")

**def** clear\_chat\_history(filename: str = "chat\_history.json") -> None:

"""Clear chat history from file"""

try:

if os.path.exists(filename):

os.remove(filename)

except IOError as e:

raise ChatbotError(**f**"Failed to clear chat history: {str(e)}")

**def** validate\_dataframe(df: pd.DataFrame) -> bool:

"""Check if dataframe meets analysis requirements"""

if len(df) < 1:

raise ChatbotError("Dataframe has no rows")

if len(df.columns) < 1:

raise ChatbotError("Dataframe has no columns")

return True

README.md:

**# 📚 Chatbot trợ lý phân tích file dữ liệu**

**## Mô tả**

Ứng dụng cho phép:

- Chat trực tiếp với mô hình AI (Groq API - LLaMA3).

- Upload file Excel/CSV và đặt câu hỏi.

- Hỗ trợ nhập câu hỏi bằng giọng nói.

- Tóm tắt dữ liệu tự động.

**## Cài đặt**

```bash

pip install -r requirements.txt