

import os

```
import streamlit as st
from llama_index import GPTSimpleVectorIndex, SimpleDirectoryReader

# NOTE: for local testing only, do NOT deploy with your key hardcoded
# to use this for yourself, create a file called .streamlit/secrets.toml with your api
key
# Learn more about Streamlit on the docs: https://docs.streamlit.io/
os.environ["OPENAI_API_KEY"] = st.secrets["openai_api_key"]

index_name = "./index.json"
documents_folder = "./documents"

@st.cache_resource
def initialize_index(index_name, documents_folder):
    if os.path.exists(index_name):
        index = GPTSimpleVectorIndex.load_from_disk(index_name)
    else:
        documents = SimpleDirectoryReader(documents_folder).load_data()
        index = GPTSimpleVectorIndex(documents)
        index.save_to_disk(index_name)

    return index

@st.cache_data(max_entries=200, persist=True)
def query_index(_index, query_text):
    response = _index.query(query_text)
    return str(response)

# This should be cached and only fully runs once
index = initialize_index(index_name, documents_folder)

st.title("🦙 My Mind Bangun 🦙")
st.header("Welcome to My Mind Bangun")
st.text("Please enter a query about My Mind Bangun")

text = st.text_input("Query text:")

if st.button("Run Query") and text is not None:
    response = query_index(index, text)
    st.markdown(response)

tambahkan fitur mnyimpan history di local dan dapat di tampilkan di csv ketika di
klik
```



Sure, here's an updated version of the code that includes a feature to save the query history to a CSV file:

```
python 📄 Copy code

import os
import csv
import streamlit as st
from llama_index import GPTSimpleVectorIndex, SimpleDirectoryReader

# NOTE: for local testing only, do NOT deploy with your key hardcoded
# to use this for yourself, create a file called .streamlit/secrets.toml with your api
key
# Learn more about Streamlit on the docs: https://docs.streamlit.io/
os.environ["OPENAI_API_KEY"] = st.secrets["openai_api_key"]

index_name = "./index.json"
documents_folder = "./documents"
history_file = "./query_history.csv"

@st.cache_resource
def initialize_index(index_name, documents_folder):
    if os.path.exists(index_name):
        index = GPTSimpleVectorIndex.load_from_disk(index_name)
    else:
        documents = SimpleDirectoryReader(documents_folder).load_data()
        index = GPTSimpleVectorIndex(documents)
        index.save_to_disk(index_name)

    return index

@st.cache_data(max_entries=200, persist=True)
def query_index(_index, query_text):
    response = _index.query(query_text)
    return str(response)

# This should be cached and only fully runs once
index = initialize_index(index_name, documents_folder)

# Initialize the history file with a header row if it doesn't exist
if not os.path.exists(history_file):
    with open(history_file, mode="w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["Query", "Response"])

st.title("🦙 My Mind Bangun 🦙")
st.header("Welcome to My Mind Bangun")
st.text("Please enter a query about My Mind Bangun")

text = st.text_input("Query text:")

if st.button("Run Query") and text is not None:
    response = query_index(index, text)
    st.markdown(response)

    # Save the query and response to the history file
    with open(history_file, mode="a", newline="") as f:
        writer = csv.writer(f)
        writer.writerow([text, response])

# Show the query history if the user clicks the "Show History" button
if st.button("Show History"):
    with open(history_file, mode="r") as f:
        reader = csv.reader(f)
        rows = list(reader)
        st.table(rows)
```

This code creates a new variable called `history_file`, which is the path to the CSV file where the query history will be stored. When the user clicks the "Run Query" button and a query is successfully executed, the code saves the query and its response to the history file using the `csv` module.

Finally, the code adds a new button called "Show History". When the user clicks this button, the code reads the query history from the CSV file and displays it in a `streamlit` table.