```python
from langchain_community.document_loaders import PyPDFLoader
from langchain_core.messages import HumanMessage,SystemMessage
from langchain_core.prompts import ChatPromptTemplate,HumanMessagePromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from langchain_community.vectorstores import Chroma
import streamlit as st

# Title and header for the Streamlit app
st.title("Chatbot Using RAG System")
st.header("Ask me from 'Leave No Context Behind' Paper")

# Gemini-api-key
f = open(r"key.txt")
GEMINI_API_KEY = f.read()

# Create the chat model
chat_model = ChatGoogleGenerativeAI(google_api_key=GEMINI_API_KEY, model="gemini-1.5-pro-latest")

# # Create the embedding model
embedding_model = GoogleGenerativeAIEmbeddings(google_api_key=GEMINI_API_KEY, model="models/embedding-001")

# Set up a connection with the Chroma for retrieval
connection = Chroma(persist_directory=r"chroma", embedding_function=embedding_model)

# Converting CHROMA db_connection to Retriever Object
retriever = connection.as_retriever(search_kwargs={"k": 5})

# User query
user_query = st.text_input("Search your query")

# Chatbot prompt templates
chat_template = ChatPromptTemplate.from_messages([
    SystemMessage(content="You are a Helpful AI Bot. You take the context and question from user. Your answer should be based on the specific context."),
    HumanMessagePromptTemplate.from_template("Answer the question based on the given context.\nContext: {Context}\nQuestion: {question}\nAnswer:")
])

# Output parser for chatbot response
output_parser = StrOutputParser()

# Function to format retrieved documents
def format_docs(docs):
    formatted_content = "\n\n".join(doc.page_content.strip() for doc in docs if doc.page_content.strip())
    return formatted_content if formatted_content else "Sorry relevant context not found."

# RAG chain for chatbot interaction
rag_chain = (
    {"Context": retriever | format_docs, "question": RunnablePassthrough()}
    | chat_template
    | chat_model
    | output_parser
)

if st.button("click here to know"):
    if user_query:
        response = rag_chain.invoke(user_query)
        st.write("Answer:")
        st.write(response)
```