- We try to create a simple chatbot app with a user interface using streamlit and langchain.

1-Packages Used:

1-1-Streamlit:

```
import streamlit as st
```

is a open-source framework to build user inteface for AI applications with an easy way.

1-2-langchain and OpenAI:

```
from langchain_openai import OpenAI
```

This is a LangChain x OpenAI integration package and we going to import the OpenAI LLM.

1-3-Langchain_core:

```
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
```

This package is imported to use the core functionalities of langchain and

we are importing StrOutputParser

it used to convert the output of the LLM from a message into a string,

the second functionalitie is ChatPromptTemplate and is used to create the prompt object.

1-4-Dot_env

```
from dotenv import load_dotenv
```

Used to load environment variables from the .env file.

2-Code explanation:

-First of all we have to create a .env file to store the environment variables

then we add the openAI api key inside this file (this is custom api key)

```
OPENAI_API_KEY = sk-bF4A69ERsIvDFzxNGjj5T3BlbkFJSjuHTSHnCPZlCammiA
```

finally we load it using load_dotenv()

```
load_dotenv()
```

-Second we create the user interface with streamlit

```
st.set_page_config(page_title="Streaming bot", page_icon="🤖")
st.title("BuddyBot 🤖")

# store the user input
user_query = st.chat("Ask your question")
```

st.set_page_config():to set a title and icon to the web app.

st.title(): to display the title on the app.

in the last line we creating a chat input to make the user able to ask questions and we are storing this question inside the user_query variable.

.Now we creating the OpenAI language model object, then we define a template for creating prompt after that we create a chat prompt template using the defined

template.

```
llm = OpenAI()
💡
template = """
You are an AI bot, answer the user questions:

User question: {user_question}
"""

# create the prompt
prompt = ChatPromptTemplate.from_template(template)
```

.Create a output string parser.

```
output_parser = StrOutputParser()
```

after that we creating a chain combining the llm, prompt and the parser so our llm can understand the given user_query verry well and give us a string output .

```
chain = prompt | llm | output_parser

# test the user input if it's empty
if user_query:
    # invoke the chain and pass the us
    response = chain.invoke({
        "user_question": user_query
    })

    # show the response to the user
    st.write(response)
```

after testing if there is any input

we invoke the chain and give it the user input to processe it and returns

a response.

st.write is a streamlit sunction that allow us to display the response on the app and the user can see it.