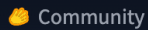


Spaces: DexterSptizu/langchain-airline-enquiry   like 0 ● Running 

Files



Community



Settings



main ▾

langchain-airline-enquiry / app.py 

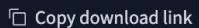
DexterSptizu Create app.py

dc83d6b

VERIFIED

4 minutes ago

</> raw



Copy download link



history



blame



edit



delete



Safe



5.78 kB

```
1 import gradio as gr
2 from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
3 from langchain_core.messages import SystemMessage, HumanMessage, AIMessage
4 from langchain_openai import ChatOpenAI
5 import os
6
7 class AirlineChatbot:
8     def __init__(self):
9         self.chat_history = []
10        self.api_key = None
11        self.chat = None
12
13    def set_api_key(self, api_key):
14        self.api_key = api_key
15        os.environ["OPENAI_API_KEY"] = api_key
16        self.chat = ChatOpenAI(
17            temperature=0.7,
18            model="gpt-4o-mini"
19        )
20        return "API Key set successfully! You can now start chatting."
21
22    def format_chat_history(self, chat_history):
23        formatted_messages = []
24        for message in chat_history:
25            if isinstance(message, HumanMessage):
26                formatted_messages.append(("human", message.content))
27            elif isinstance(message, AIMessage):
28                formatted_messages.append(("assistant", message.content))
29        return formatted_messages
30
31    def respond(self, message, history):
32        if not self.api_key:
33            return "Please set your OpenAI API key first."
34
35        # System prompt template
36        system_prompt = """You are an airline customer service representative.
37        Your role is to assist customers with:
38        - Flight bookings and inquiries
39        - Schedule information
40        - Baggage policies
41        - Check-in procedures
42        - General airline policies
43
44        Be professional, courteous, and provide accurate information.
45        Always maintain the role of an airline representative throughout the conversation."""
46
47        # Convert Gradio history to LangChain format
48        self.chat_history = []
49        for human, ai in history:
50            self.chat_history.append(HumanMessage(content=human))
51            self.chat_history.append(AIMessage(content=ai))
52
```

```

53     # Create appropriate template based on message content
54     if "book" in message.lower():
55         template = ChatPromptTemplate.from_messages([
56             ("system", system_prompt),
57             ("human", "I want to book a flight from {origin} to {destination} on {date}"),
58             ("assistant", "I'll help you book a flight. Let me check available options."),
59             MessagesPlaceholder(variable_name="chat_history"),
60             ("human", "{input}")
61         ])
62     elif "baggage" in message.lower():
63         template = ChatPromptTemplate.from_messages([
64             ("system", system_prompt),
65             ("human", "What's the baggage allowance for {flight_type} flights?"),
66             MessagesPlaceholder(variable_name="chat_history"),
67             ("human", "{input}")
68         ])
69     elif "schedule" in message.lower():
70         template = ChatPromptTemplate.from_messages([
71             ("system", system_prompt),
72             ("human", "I need flight schedules between {origin} and {destination}"),
73             MessagesPlaceholder(variable_name="chat_history"),
74             ("human", "{input}")
75         ])
76     else:
77         template = ChatPromptTemplate.from_messages([
78             ("system", system_prompt),
79             MessagesPlaceholder(variable_name="chat_history"),
80             ("human", "{input}")
81         ])
82
83     # Format the prompt with the current message
84     prompt = template.format_messages(
85         chat_history=self.chat_history,
86         input=message,
87         origin="",
88         destination="",
89         date="",
90         flight_type=""
91     )
92
93     try:
94         # Get response from the model
95         response = self.chat(prompt)
96         return response.content
97     except Exception as e:
98         return f"Error: {str(e)}"
99
100 def create_demo():
101     chatbot = AirlineChatbot()
102
103     with gr.Blocks(theme=gr.themes.Soft()) as demo:
104         with gr.Row():
105             gr.Markdown("## ✈️ Airline Customer Service Assistant")
106
107         with gr.Row():
108             with gr.Column():
109                 api_key_input = gr.Textbox(
110                     label="Enter your OpenAI API Key",
111                     type="password",
112                     placeholder="sk-..."
113                 )
114                 set_key_button = gr.Button("Set API Key")
115                 api_key_status = gr.Textbox(label="Status", interactive=False)
116
117             chatbot_interface = gr.ChatInterface(
118                 fn=chatbot.respond,
119                 description="Welcome to our Airline Customer Service!"

```

```

120         Ask questions about flights, bookings, baggage, and more.""",
121         examples=[
122             "What's the baggage allowance for international flights?",
123             "How early should I arrive for check-in?",
124             "I want to book a flight from New York to London",
125             "What documents do I need for international travel?",
126             "Can I change my flight date?",
127             "Do you provide meals on long-haul flights?"
128         ]
129     )
130
131     set_key_button.click(
132         fn=chatbot.set_api_key,
133         inputs=[api_key_input],
134         outputs=[api_key_status]
135     )
136
137     return demo
138
139     # Custom CSS for better styling
140     custom_css = """
141     .gradio-container {
142         font-family: 'Arial', sans-serif;
143     }
144     .chat-message {
145         padding: 15px;
146         margin: 5px;
147         border-radius: 10px;
148     }
149     """
150
151     if __name__ == "__main__":
152         # Create and launch the demo
153         demo = create_demo()
154         demo.launch(
155             share=True, # Set to False if you don't want to generate a public URL
156             server_name="0.0.0.0",
157             server_port=7860,
158             debug=True
159         )
160

```