Hotel Information Chatbot

SUPERIOR UNIVERSITY

<u>PAI LAB</u>

Task # 11

Name:

M. Hasan Shahid

Roll No:

059

Submitted To:

Sir Rasik

Date:

5 April 2025

Subject:

PAI

Hotel Information Chatbot

INTRODUCTION

This report outlines the design and implementation of a Hotel Information Chatbot. The system uses a Flask back-end with LangChain's prompt-chaining API and Groq's ChatGroq LLM to handle user queries about rooms, pricing, amenities, and location. A simple HTML/JavaScript front-end provides a chat interface.

TOOLS & TECHNOLOGY

- **Python 3.8+**
- **Flask**: web framework for routing and JSON handling
- **python-dotenv**: load environment variables
- **LangChain Core**: RunnablePassthrough, ChatPromptTemplate, StrOutputParser
- **langchain-groq**: ChatGroq LLM integration
- **HTML/CSS/JavaScript**: front-end chat UI

ARCHITECTURE OVERVIEW

1. **app.py**
    - Defines Flask routes (/ and /ask)
    - Constructs a LangChain Runnable pipeline:
        1. Pass-through of user question
        2. Prompt templating
        3. ChatGroq LLM call
        4. Output parsing
    - Returns JSON responses to the front-end.
2. **templates/chat.html**
    - Renders chat interface
    - Sends user messages via fetch to /ask
    - Appends bot replies to the chat log

1. app.py

```
import os
from flask import Flask, render_template, request, jsonify
from dotenv import load_dotenv
from langchain_core.runnables import RunnablePassthrough
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_groq import ChatGroq
```

Load API key from .env

```
load_dotenv()
API_KEY = os.getenv("GROQ_API_KEY")
app = Flask(__name__)
```

Initialize ChatGroq LLM

```
llm = ChatGroq(
    groq_api_key=API_KEY,
    model_name='llama3-8b-8192',
    temperature=0.7
)
```

Define prompt template

```
template = """
You are a helpful hotel concierge. Answer user questions clearly and concisely about hotel details
(rooms, pricing, amenities, location).
If the question is outside those topics, respond politely that you can only help with hotel info.
User: {question}
"""
prompt = ChatPromptTemplate.from_template(template)
parser = StrOutputParser()
```

Build LangChain pipeline

```
chain = (
    {"question": RunnablePassthrough()}
    | prompt
    | llm
    | parser
)

@app.route('/')
def home():
    return render_template('chat.html')

@app.route('/ask', methods=['POST'])
def ask():
    user_msg = request.json.get('message', '')
    # Invoke the prompt → LLM → parser pipeline
    answer = chain.invoke(user_msg)
    return jsonify({'response': answer})

if __name__ == '__main__':
    app.run(debug=True)
```

Description of Key Sections:

Hotel Information Chatbot

1. **Environment Setup**
   - .env stores GROQ_API_KEY for authentication.
2. **LangChain + Groq Initialization**
   - ChatGroq configured with llama3-8b-8192 model and moderate temperature.
3. **Prompt Template**
   - Instructs the LLM to act as a hotel concierge, restricting its domain.
4. **Runnable Pipeline**
   - Chains input passthrough → templating → LLM call → string parsing.
5. **Flask Routes**
   - **/**: Serves the chat UI.
   - **/ask**: Accepts user messages and returns the bot's reply as JSON.

---

2. templates/chat.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Hotel Info Chatbot</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    #chatbox { width: 100%; height: 400px; border: 1px solid #ccc;
           overflow-y: auto; padding: 10px; }
    .user { color: blue; margin: 5px 0; }
    .bot  { color: green; margin: 5px 0; }
    #message { width: 80%; padding: 5px; }
    #send    { width: 18%; padding: 5px; }
  </style>
</head>
<body>
  <h1>Hotel Information Chatbot</h1>
  <div id="chatbox"></div>
  <input type="text" id="message" placeholder="Type your message..."/>
  <button id="send">Send</button>

  <script>
    document.getElementById('send').addEventListener('click', () => {
      const msg = document.getElementById('message').value.trim();
      if (!msg) return;
      append('user', msg);
      document.getElementById('message').value = '';
      fetch('/ask', {
        method: 'POST',
        headers: {'Content-Type':'application/json'},
        body: JSON.stringify({message: msg})
      })
      .then(res => res.json())
```

```
        .then(data => append('bot', data.response))
        .catch(() => append('bot', 'Server error'));
    });

    function append(sender, text) {
        const div = document.createElement('div');
        div.className = sender;
        div.textContent = (sender === 'user' ? 'You: ' : 'Bot: ') + text;
        const cb = document.getElementById('chatbox');
        cb.appendChild(div);
        cb.scrollTop = cb.scrollHeight;
    }
  </script>
</body>
</html>
```

Description of Key Sections:

1. **HTML Structure**
   - A div#chatbox displays conversation history.
   - An input#message and button#send allow user input.
2. **Styling**
   - Simple CSS for layout, scrollable chat area, and distinct colors for user & bot messages.
3. **JavaScript Logic**
   - On “Send” click, the user’s message is appended, then posted to /ask.
   - The bot’s response is appended on receipt; errors display “Server error.”
   - The chatbox auto-scrolls to show the latest message.

CONCLUSION

This Hotel Information Chatbot demonstrates a clean separation between back-end prompt chaining (Flask + LangChain + Groq) and front-end interactivity (HTML/JavaScript). The modular design allows:

- Easy swapping of LLM providers or models.
- Prompt/template adjustments for expanded domains.
- Front-end enhancements (e.g., user authentication, styling upgrades).