**The Superior University, Lahore**

|  |  |  |  |
| --- | --- | --- | --- |
| Course Title: | PAI-LAB | | |
| Programme Name: | BSDS | | |
| Semester: | 4rd | Section: | BSDSM-4A |
| Student’s Name: | Muhammad Faizan Majid | | |
| Instructor Name: | Prof. Rasikh | | |

Project Report:

Booking Chatbot Project Report

Introduction

The Booking Chatbot is a web-based application designed to assist users with hotel booking queries. Built using Flask for the backend and HTML/JavaScript for the frontend, it integrates the Gemini API to process natural language inputs and provide relevant responses. This report outlines the project’s structure, key files, and functionality.

File Descriptions

.env

The .env file stores environment variables, specifically the API key for the Gemini API, which is used for natural language processing.

app.py

The app.py file is the core of the Flask backend. It configures the Gemini API, defines routes for the web interface, and handles user queries by processing them through the Gemini model.

index.html

The index.html file defines the frontend interface, including a chat window, input field, and styling. It uses JavaScript to handle user interactions and communicate with the backend via AJAX.

Functionality

The Booking Chatbot provides the following key features:

- Interactive Chat Interface: Users can type queries in a styled chat window, with responses displayed in real-time.

- Gemini API Integration: Queries are processed using the Gemini 1.5 Flash model, configured to assist with booking-related questions.

- Responsive Design: The frontend uses CSS with a modern gradient background and animations for a user-friendly experience.

- Error Handling: The backend includes basic error handling for API issues, ensuring robust operation.

Appendices

Appendix: .env

GEMINI\_API\_KEY = "your\_api\_key"

Appendix: app.py

from flask import Flask, render\_template, request, jsonify

import datetime

import re

import os

import google.generativeai as genai

# Set your Gemini API key (replace or store in environment variable)

genai.configure(api\_key=os.getenv("AIzaSyDbmTcnEmNRCnXV6pX51WrrusyIAvhkJKc") or "AIzaSyDbmTcnEmNRCnXV6pX51WrrusyIAvhkJKc")

app = Flask(\_\_name\_\_)

# models = genai.list\_models()

# for m in models:

# print(m.name)

def process\_booking\_query(message):

try:

model = genai.GenerativeModel("gemini-1.5-flash")

chat = model.start\_chat()

response = chat.send\_message(

f"""You are a helpful booking assistant.

if user asks questions other than booking tell them to ask about booking.

Keep the tone friendly and clear but well formatted.

Avoid unnecessary details. User asked: '{message}'"""

)

return response.text

except Exception as e:

return f"Error with Gemini API: {str(e)}"

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/chat', methods=['POST'])

def chat():

user\_message = request.json.get('message')

response = process\_booking\_query(user\_message)

return jsonify({'response': response})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Appendix: index.html

<script type="text/javascript">

var gk\_isXlsx = false;

var gk\_xlsxFileLookup = {};

var gk\_fileData = {};

function filledCell(cell) {

return cell !== '' && cell != null;

}

function loadFileData(filename) {

if (gk\_isXlsx && gk\_xlsxFileLookup[filename]) {

try {

var workbook = XLSX.read(gk\_fileData[filename], { type: 'base64' });

var firstSheetName = workbook.SheetNames[0];

var worksheet = workbook.Sheets[firstSheetName];

// Convert sheet to JSON to filter blank rows

var jsonData = XLSX.utils.sheet\_to\_json(worksheet, { header: 1, blankrows: false, defval: '' });

// Filter out blank rows (rows where all cells are empty, null, or undefined)

var filteredData = jsonData.filter(row => row.some(filledCell));

// Heuristic to find the header row by ignoring rows with fewer filled cells than the next row

var headerRowIndex = filteredData.findIndex((row, index) =>

row.filter(filledCell).length >= filteredData[index + 1]?.filter(filledCell).length

);

// Fallback

if (headerRowIndex === -1 || headerRowIndex > 25) {

headerRowIndex = 0;

}

// Convert filtered JSON back to CSV

var csv = XLSX.utils.aoa\_to\_sheet(filteredData.slice(headerRowIndex)); // Create a new sheet from filtered array of arrays

csv = XLSX.utils.sheet\_to\_csv(csv, { header: 1 });

return csv;

} catch (e) {

console.error(e);

return "";

}

}

return gk\_fileData[filename] || "";

}

</script><!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Booking Chatbot</title>

<style>

body {

font-family: 'Poppins', sans-serif;

max-width: 900px;

margin: 0 auto;

padding: 20px;

background: linear-gradient(135deg, #e0eafc 0%, #cfdef3 100%);

min-height: 100vh;

display: flex;

flex-direction: column;

justify-content: space-between;

}

h1 {

color: #1a2a6c;

text-align: center;

margin-bottom: 40px;

font-size: 2.8em;

text-transform: uppercase;

letter-spacing: 3px;

text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);

}

.chat-box {

border-radius: 20px;

box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);

padding: 25px;

background-color: #ffffff;

height: 550px;

overflow-y: auto;

margin-bottom: 30px;

border: 2px solid #d3e0ea;

background: linear-gradient(to bottom, #ffffff, #f9fbfd);

}

.chat-box div {

margin: 15px 0;

padding: 12px 18px;

border-radius: 15px;

max-width: 65%;

animation: fadeIn 0.5s ease-in-out;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

.chat-box .bot {

text-align: left;

background-color: #e8f0f7;

color: #1a2a6c;

border-left: 4px solid #3498db;

}

.chat-box .user {

text-align: right;

background-color: #2ecc71;

color: white;

margin-left: 35%;

border-right: 4px solid #27ae60;

}

.input-container {

display: flex;

justify-content: center;

align-items: center;

gap: 15px;

padding: 20px 0;

background-color: #ffffff;

border-radius: 10px;

box-shadow: 0 -2px 10px rgba(0, 0, 0, 0.05);

}

#user\_input {

width: 70%;

padding: 15px;

border-radius: 30px;

border: 2px solid #d3e0ea;

font-size: 1.1em;

outline: none;

transition: border-color 0.3s, box-shadow 0.3s;

box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.05);

}

#user\_input:focus {

border-color: #3498db;

box-shadow: 0 0 8px rgba(52, 152, 219, 0.3);

}

#send\_btn {

padding: 15px 30px;

background-color: #3498db;

color: white;

border: none;

border-radius: 30px;

cursor: pointer;

font-size: 1.1em;

transition: background-color 0.3s, transform 0.2s;

box-shadow: 0 4px 6px rgba(52, 152, 219, 0.2);

}

#send\_btn:hover {

background-color: #2980b9;

transform: translateY(-2px);

}

#send\_btn:active {

transform: translateY(0);

}

</style>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">

</head>

<body>

<h1>Booking Chatbot</h1>

<div class="chat-box" id="chat-box">

<!-- Chat history will appear here -->

</div>

<div class="input-container">

<input type="text" id="user\_input" placeholder="Type your message..." />

<button id="send\_btn">Send</button>

</div>

<script>

window.onload = function() {

var defaultBotMessage = "Chat Bot: Hi! I'm here to help you. Please tell me how I can assist you with your hotel booking.";

var botDiv = document.createElement("div");

botDiv.className = "bot";

botDiv.textContent = defaultBotMessage;

document.getElementById("chat-box").appendChild(botDiv);

document.getElementById("chat-box").scrollTop = document.getElementById("chat-box").scrollHeight;

};

document.getElementById("send\_btn").addEventListener("click", function() {

var userMessage = document.getElementById("user\_input").value;

if (userMessage) {

var userDiv = document.createElement("div");

userDiv.className = "user";

userDiv.textContent = "You: " + userMessage;

document.getElementById("chat-box").appendChild(userDiv);

fetch("/chat", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ "message": userMessage })

})

.then(response => response.json())

.then(data => {

var formattedResponse = formatChatResponse(data.response);

var botDiv = document.createElement("div");

botDiv.className = "bot";

botDiv.innerHTML = "Chat Bot: " + formattedResponse;

document.getElementById("chat-box").appendChild(botDiv);

document.getElementById("chat-box").scrollTop = document.getElementById("chat-box").scrollHeight;

document.getElementById("user\_input").value = "";

});

}

});

document.getElementById("user\_input").addEventListener("keypress", function(e) {

if (e.key === "Enter") {

document.getElementById("send\_btn").click();

}

});

function formatChatResponse(text) {

text = text.replace(/\\*\\*(.+?)\\*\\*/g, '<strong>$1</strong>');

const lines = text.split('\n');

let formatted = '';

let inList = false;

lines.forEach(line => {

if (line.trim().startsWith('\*')) {

if (!inList) {

formatted += '<ul>';

inList = true;

}

formatted += `<li>${line.replace(/^\\*\s\*/, '')}</li>`;

} else {

if (inList) {

formatted += '</ul>';

inList = false;

}

formatted += `<p>${line}</p>`;

}

});

if (inList) formatted += '</ul>';

return formatted;

}

</script>

<script>(function(){function c(){var b=a.contentDocument||a.contentWindow.document;if(b){var d=b.createElement('script');d.innerHTML="window.\_\_CF$cv$params={r:'93ba9e1c0833bd3c',t:'MTc0NjU1Njk0Ny4wMDAwMDA='};var a=document.createElement('script');a.nonce='';a.src='/cdn-cgi/challenge-platform/scripts/jsd/main.js';document.getElementsByTagName('head')[0].appendChild(a);";b.getElementsByTagName('head')[0].appendChild(d)}}if(document.body){var a=document.createElement('iframe');a.height=1;a.width=1;a.style.position='absolute';a.style.top=0;a.style.left=0;a.style.border='none';a.style.visibility='hidden';document.body.appendChild(a);if('loading'!==document.readyState)c();else if(window.addEventListener)document.addEventListener('DOMContentLoaded',c);else{var e=document.onreadystatechange||function(){};document.onreadystatechange=function(b){e(b);'loading'!==document.readyState&&(document.onreadystatechange=e,c())}}}})();</script></body>

</html>