**Building a Terminal-to-Web Chat Interface with Flask and a Given Dataset**

**Introduction**

In this document, we will guide you through the creation of a basic chat application that serves as both a terminal and a web interface. We will use Flask, a Python web framework, to build the web interface, and a provided dataset for chat messages. This project aims to demonstrate how to connect a command-line application with a web application for chat interactions.

**Prerequisites**

Before you begin, ensure you have the following:

* Python (3.x) installed on your system.
* Flask library installed.
* A provided dataset of chat messages.
* Basic knowledge of Python programming.
* Terminal or Command Prompt.

**Implementation**

**Create Project Directory:**

Start by creating a project directory for your application.

**Create Python Script:**

Create a Python script for the chat application. You can use your favorite code editor to create a file, e.g., chat\_app.py.

**Import Libraries:**

Import the necessary libraries at the beginning of your Python script.

from flask import Flask, render\_template, request

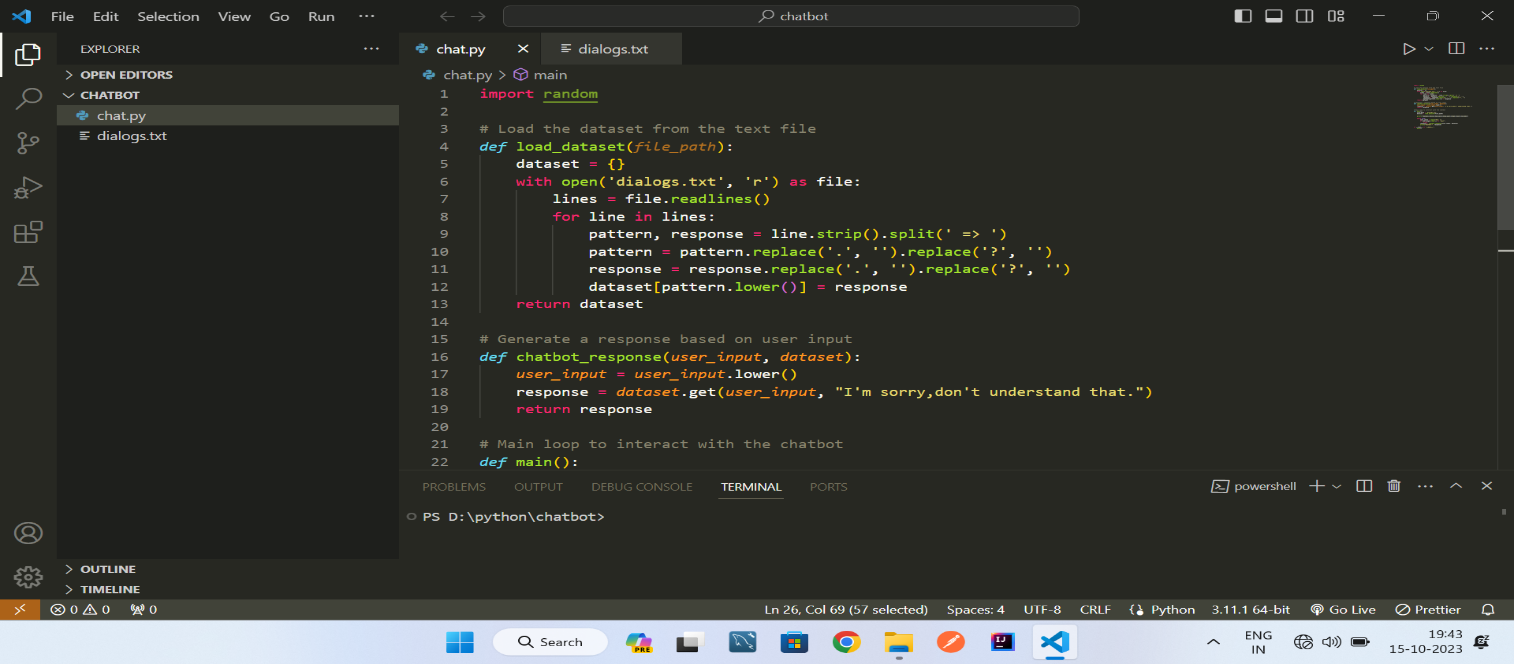
**Set Up Flask App:**

Initialize a Flask web application.

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

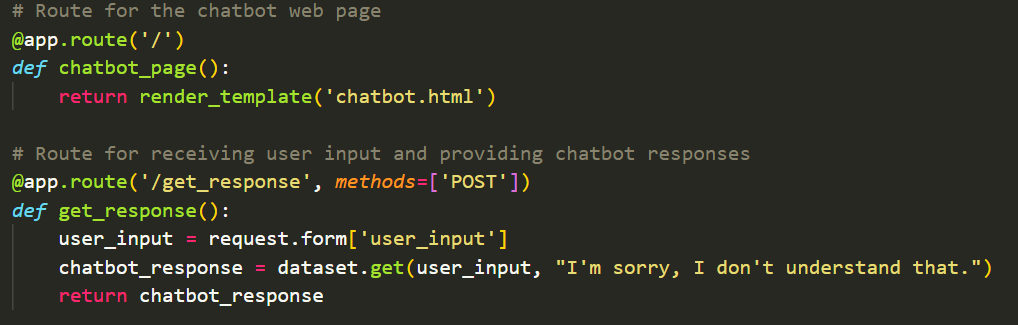
**Load the Dataset:**

Load the provided dataset into your Python script, similar to the previous example.



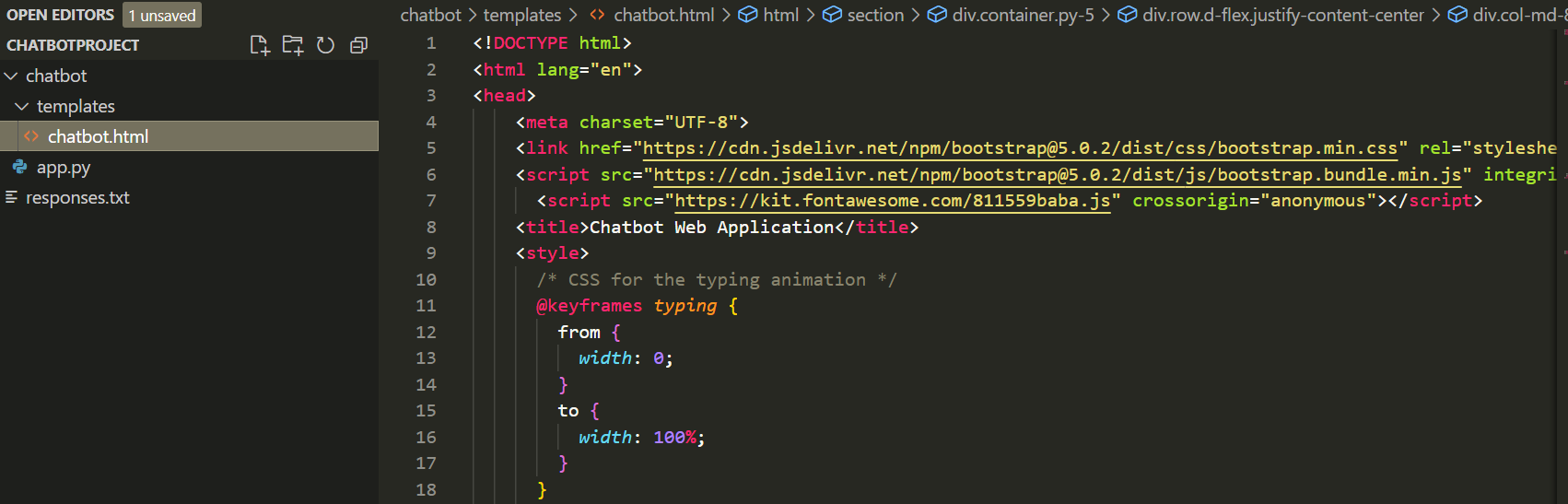
**Create a Route for Chat Interface:**

Define a route in your Flask app to render a web page for the chat interface



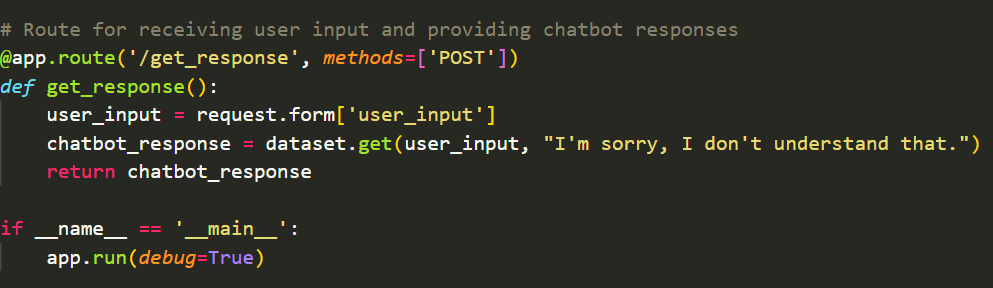
**Create a Chat HTML Template:**

Create an HTML template for the chat interface. You can use the Jinja2 template engine to render chat messages on the web page.



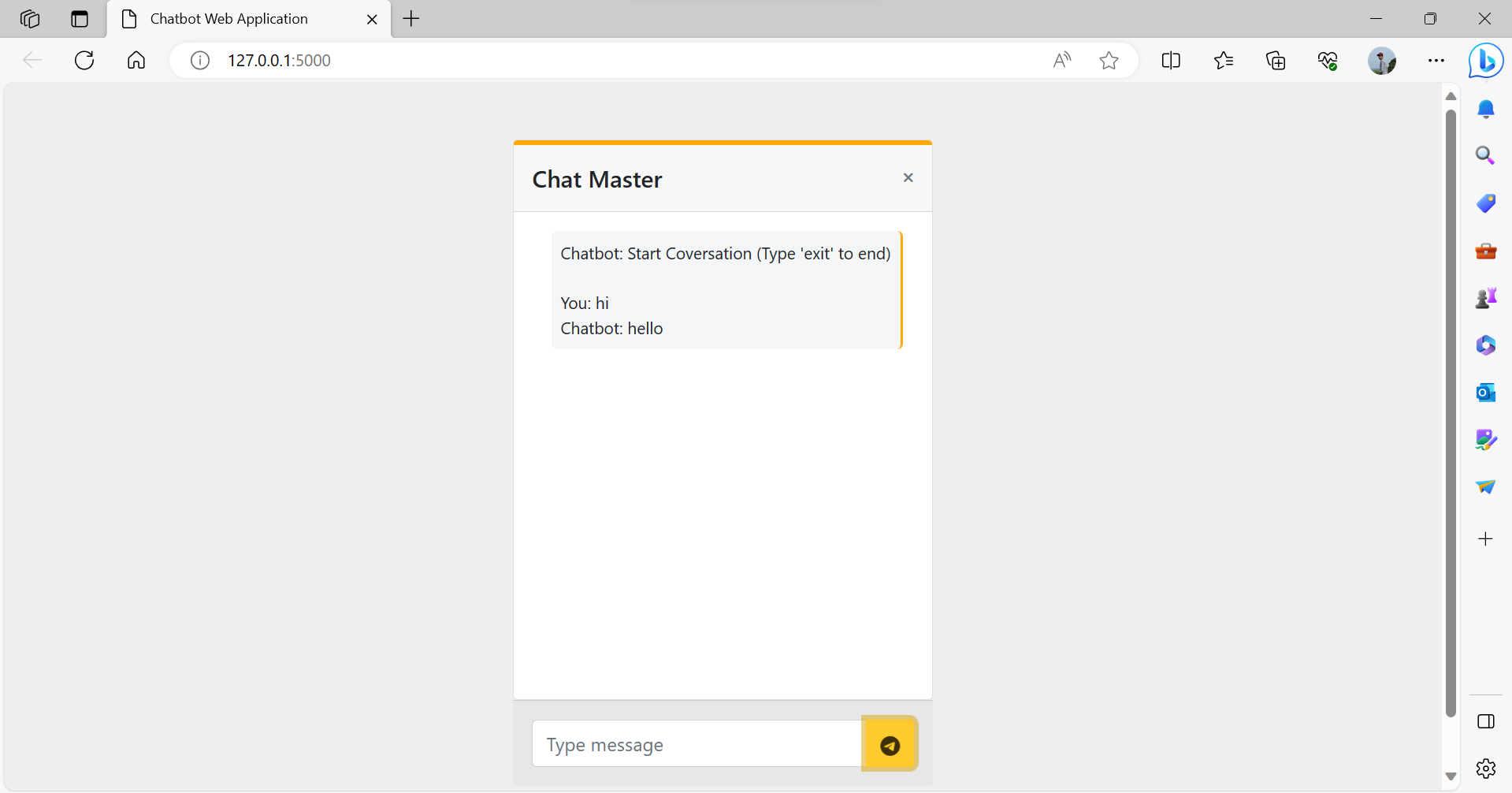
**Run the Flask App:**

Run your Flask application.

****

**Testing:**

Open a web browser and navigate to **http://127.0.0.1:5000/**  to access the chat interface. You should see the chat messages from the provided dataset displayed on the web page.



**Improvements:**

You can extend this application by allowing user input, interactive chat features, and real-time updates.

**Sample code:**

**App.py**

from flask import Flask, render\_template, request

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

# Load responses from the text file

*def* load\_responses():

    dataset = {}

    with open('responses.txt', 'r') as file:

        lines = file.readlines()

        for line in lines:

            pattern, response = line.strip().split(' => ')

            pattern = pattern.replace('.', '').replace('?', '')

            response = response.replace('.', '').replace('?', '')

            dataset[pattern.lower()] = response

    return dataset

dataset = load\_responses()

# Route for the chatbot web page

@app.route('/')

*def* chatbot\_page():

    return render\_template('chatbot.html')

# Route for receiving user input and providing chatbot responses

@app.route('/get\_response', *methods*=['POST'])

*def* get\_response():

    user\_input = request.form['user\_input']

    chatbot\_response = dataset.get(user\_input, "I'm sorry, I don't understand that.")

    return chatbot\_response

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

    app.run(*debug*=True)

**Chatbot Html File**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

      <script src="https://kit.fontawesome.com/811559baba.js" crossorigin="anonymous"></script>

    <title>Chatbot Web Application</title>

    <style>

      /\* CSS for the typing animation \*/

      @keyframes *typing* {

        from {

*width*: 0;

        }

        to {

*width*: 100%;

        }

      }

      .typing-animation {

*display*: inline-block;

*overflow*: hidden;

*white-space*: nowrap;

*border-right*: 2px solid #ffa900; /\* Blinking cursor \*/

*padding-right*: 3px; /\* Spacing for cursor \*/

*animation*: typing 3s steps(30, end);

      }

    </style>

</head>

<section style="background-color: #eee; height: 600px;">

    <div class="container py-5">

      <div class="row d-flex justify-content-center">

        <div class="col-md-8 col-lg-6 col-xl-4">

          <div class="card">

            <div class="card-header d-flex justify-content-between align-items-center p-3"

              style="border-top: 4px solid #ffa900;">

              <h5 class="mb-0">Chat Master</h5>

              <div class="d-flex flex-row align-items-center">

                <i class="fas fa-times text-muted fa-xs"></i>

              </div>

            </div>

            <div class="card-body" data-mdb-perfect-scrollbar="true" style="position: relative; height: auto">

              <div class="d-flex justify-content-between">

                <p class="typing-animation small p-2 ms-3 mb-3 rounded-3 " style="background-color: #f5f6f7;" id="chat-output">

                  Chatbot: Start Coversation (Type 'exit' to end)

                  <br>

                  <br>

                </p>

              </div>

              <br>

              <br>  <br>

              <br>  <br>

              <br>  <br>

              <br>  <br>

              <br>  <br>

                </div>

              </div>

            <div class="card-footer text-muted d-flex justify-content-start align-items-center p-3">

              <div class="input-group mb-0">

                <input type="text" class="form-control" id="user-input" placeholder="Type message"

                  aria-label="Recipient's username" aria-describedby="button-addon2" />

                <button class="btn btn-warning" type="submit"  id="send-button" style="padding-top: .55rem;">

                    <i class="fa-brands fa-telegram fa-beat-fade" value="PLAY" onclick="play()"></i>

                    <audio id="audio" src="https://s27.aconvert.com/convert/p3r68-cdx67/c4lpg-az7kc.mp3"></audio>

                  </button>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

  </section>

  <body>

<script>

*const* chatOutput = document.getElementById('chat-output');

*const* userInput = document.getElementById('user-input');

*const* sendButton = document.getElementById('send-button');

        sendButton.addEventListener('click', *function* () {

*function* play() {

*var* audio = document.getElementById("audio");

                        audio.play();

                      }

*const* message = userInput.value;

            if(message =='exit')

            {

              window.location.reload("Refresh")

              alert('Your Coversation ends')

            }

*var* audio = new *Audio*('sound.mp3');

            audio.play();

            if (message.trim() !== '') {

                appendMessage('You: ' + message);

                userInput.value = '';

                // Send user input to the server and get chatbot response

                fetch('/get\_response', {

                    method: 'POST',

                    body: new *URLSearchParams*({ 'user\_input': message }),

                })

                .then(*response* *=>* *response*.text())

                .then(*data* *=>* {

                    appendMessage('Chatbot: ' + *data*);

                });

            }

        });

*function* appendMessage(*message*) {

*const* messageElement = document.createElement('div');

            messageElement.textContent = *message*;

            chatOutput.appendChild(messageElement);

        }

    </script>

</body>

</html>

**Conclusion**

This document provides a basic foundation for creating a chat application that serves as both a terminal and a web interface using Flask and a provided dataset. You can enhance this application by adding more interactive features and extending the web interface to support real-time chat interactions.