

**Superior University Gold Campus**

**PAI Lab Task # 10**

Name: Syed Ejiz Ul Hassan Kazmi

Roll. No: SU92-BSDSM-F23-025

Section: BSDS-4A

Instructor: Sir Rasikh Ali

# **University Admission Chatbot**

**Files:**

* app.py (Flask backend)
* index.html (Frontend template)
* style.css (Styling)

## **1. app.py (Flask Application)**

This file controls the backend logic of the chatbot. It manages user messages, responses, sessions, and routing.

### Key Points:

* **Imports**:
  + Flask, render\_template, request, session, redirect, url\_for from flask.
  + Session from flask\_session to manage chat history.
* **App Configuration**:
  + Sets the secret\_key for secure sessions.
  + Configures session type to 'filesystem'.
  + Initializes the Session object.
* **Responses Dictionary**:  
  Predefined static responses based on user input categories such as "greeting", "apply", "requirements", etc.
* **Route /**:
  + Handles both GET and POST requests.
  + **GET**: Displays chat history.
  + **POST**: Takes user input, matches it with predefined keywords, and generates a relevant response.
  + Stores user messages and bot replies into the session for maintaining chat history.
  + Special handling for "thanks" to clear chat history after responding.
* **Route /clear**:
  + Clears the entire chat history manually when the user clicks "Clear Chat".
* **Running the App**:
  + The server is started with debug=True and use\_reloader=False.

## **2. index.html (Frontend Template)**

This is the user interface where students interact with the chatbot.

### Key Points:

* **HTML Structure**:
  + Basic structure with title "University Admission Chatbot".
  + A background image is applied using background.jpg.
* **Chat Display**:
  + Loops over chat\_history from the session and displays messages as "You" or "Bot" based on who sent it.
  + Messages are styled differently for user and bot.
* **Suggestion Box**:
  + Provides users hints on what questions they can ask (like "What are the admission requirements?", "How to apply?", etc.).
* **Forms**:
  + A text input form to send new messages.
  + A second form with a "Clear Chat" button to reset the conversation.

## **3. style.css (Styling)**

This file styles the chatbot to look clean, modern, and user-friendly.

### Key Points:

* **General Body**:
  + Centers the chatbot on the screen.
  + Applies a full height (100vh) layout.
* **Chat Container**:
  + White background card.
  + Rounded corners and shadow for a modern effect.
  + Fixed width and scrollable if chat becomes long.
* **Chat Box**:
  + Bordered container to display conversation history.
  + Different background colors for user and bot messages.
* **Input Forms**:
  + Styled text input and send button.
  + Rounded edges and proper padding.
* **Suggestions Box**:
  + Light background with suggestions to guide the user.
* **Clear Button**:
  + Red-colored clear chat button.
  + Hover effect for better UX.

# **Overall Flow:**

1. User opens the website.
2. Background is loaded, and the chatbot UI is shown.
3. User types a message and submits.
4. Flask processes the message, finds the matching response, updates the session.
5. Page reloads with updated chat history.
6. User can also manually clear the chat by clicking "Clear Chat".

# **Features:**

* Simple and clean UI.
* Remembers previous messages using Flask sessions.
* Dynamic responses based on keywords.
* Option to clear chat history after completion.
* Suggestion prompts to help users interact easily.

### Interface:

### 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_