




Topic	CHATBOT INTEGRATION ON WEB	
Class Description	The student will learn to integrate the chatbot on a web page using AJAX and Flask.	
Class	PRO C122	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Learn to integrate the chatbot on the Digi Diary webpage. Creating AJAX call to send and receive data. Creating Flask API. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Smartphone Student Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen 	
Class structure	Warm-Up Teacher-led Activity 1 Student-led Activity 1 Wrap-Up	10 mins 10 mins 20 mins 05 mins
WARM-UP SESSION - 10 mins		
<div>  </div> <p>Teacher Starts Slideshow Slide 1 to 4</p> <p>Refer to speaker notes and follow the instructions on each slide.</p>		

Teacher Action	Student Action
<p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes. 	<p>ESR: Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p>Hey <student's name>! It's great to see you! How are you?</p> <p>Can you tell me what we learned in the previous class?</p> <p><i>Note: Encourage the student to give answers and be more involved in the discussion.</i></p> <p>Amazing!</p>	<p>ESR: Hi, I am good!</p> <p>ESR: In the previous class we created a project called invisibility cloak.</p>
<p>WARM-UP QUIZ Click on In-Class Quiz</p>	
<p>Continue WARM-UP Session Slide 5 to 11</p> 	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. 	
Do you remember we had created Digi-Diary ?	ESR: Yes

<p>What have we done in that project so far?</p> <p>Great!! We had created a chatbot as well. How will it be if we can integrate the chatbot in the same web-page?</p> <p>Amazing! So let's continue with it.</p>	<p>ESR: We created a webpage and in the webpage we integrated the ML model for sentiment analysis.</p> <p>ESR: It'll be fun</p>
<p style="text-align: center;">  Teacher Ends Slideshow </p>	
<p style="text-align: center;">TEACHER-LED ACTIVITY - 10 mins</p>	
<p style="text-align: center;">• Teacher Initiates Screen Share</p>	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Create a function to display the bot. • Create a function to append the messages in the chatbot 	
<p>Note: Open Teacher Activity2 for Boilerplate code.</p> <p>In the boilerplate code we are given with the HTML page. It's the Digi-Diary along with the UI for chatbot. Let's have a look at it and understand how the names are given to different elements which will be used to create an AJAX call.</p> <p>Can you tell me the layout of our previous webpage?</p>	<p>ESR: We had a title, date, text area, button. For sentiment prediction, we had a paragraph and image to display emoticons.</p>

Yes!! Now we'll add some new elements according to our chatbot.

Also columns to save entries.

The HTML page has the following layout:

1. The name of the webpage '**Digi- Diary**' is displayed.
2. A paragraph for displaying the **date**.
3. A text area for entering **text**.
4. A '**Predict Emotion**' button for sending the request
5. A '**Save This Entry**' button for saving the entry.
6. A paragraph for displaying **the predicted emotion** as a result. The display is kept as '**none**' to hide it.
7. An **image (emoticon)** will be shown according to the emotion. Initially, it's also hidden.
8. There are three columns created in a row for displaying the saved entries along with the emoticons and emotions.
9. A button is added (**id=open_button**). When this button is clicked a chatbox will open. We'll see the layout of the chatbox while going through its UI.

Title

Date

Enter Text

Predict Emotion

Save entry

Display Sentiment

Show Entry

Date:
Text:
Emoticon
Sentiment

Show Entry

Date:
Text:
Emoticon
Sentiment

Show Entry

Date:
Text:
Emoticon
Sentiment

Click to chat

When the button for a chatbot is clicked, it shows initial messages.


Digi Diary

Date: 2/22/2022

How was your day?

Predict Emotion

Save This Entry




Your Entries :

Date: 2/22/2022

It was bad.


worry



Date: 2/1/2022

It was great

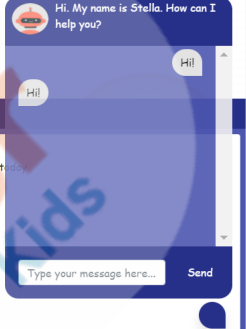
happiness



Date: 2/1/2022

It was awesome we went to zoo today

neutral



Let's check index.html first. It has the required code for the web page. We have already seen the code for the rest of the page elements except the chatbot. So we'll check the UI for the chatbot and try to understand how it is created. The master div '**chatbox**' has the class '**chatbox_button**'.

Note: We'll be using class name while creating AJAX call.

```

90      <!--Chatbot UI-->
91      <div class="chatbox">
92
93      <!--Chatbot Button-->
94      <div class="chatbox_button">
95          <button id="open_button"></button>
96      </div>
  
```

Now, the UI is divided into three parts i.e **header**, **messages** and **footer**.

1. In the header part, an image and a message are displayed.

```
<!--Chatbot Chat-->
<div class="chatbox__chat">

  <!--Chatbot Header-->
  <div class="row chatbox_header">

    <div class="col-2 chatbox_image_header">
      
    </div>

    <div class="col-10 chatbox_content_header">
      <p class="chatbox_description">Hi. My name is Stella. How can I help you?</p>
    </div>
  </div>
</div>
```

2. In the messages section, initial messages are displayed. (This part is very important if it is removed then appended messages will not be displayed. Thus we have to keep the initial message).
3. Class **user_messages** are used to append **user messages**, and **bot_messages** are used to append the **chatbot response**.

```
<!--Chatbot Messages-->
<div class="row chatbox_messages_cotainer">
  <div class="col" id="chat_messages">
    <div class="user_messages">
      Hi!
      <!--User Message gets appended here-->
    </div>
    <div class="bot_messages">
      Hi!
      <!--Bot Message gets appended here-->
    </div>
  </div>
</div>
```

4. In the footer part, **text input** is given with a send button for **sending messages to the bot**.

```
<!--Chatbot Footer-->
<div class="row chatbox__footer">
  <div class="col-9 text-left chatbox__input">
    <input type="text" class="form-control" id="bot_input_text"
      placeholder="Type your message here...">
  </div>
  <div class="col-3 text-left chatbox__send_button">
    <button class="btn" id="send_button">Send</button>
  </div>
</div>
```

In the index.js file, you are given the AJAX call for displaying dates, predicting emotion, and saving the diary entry.

Next, we'll have to write the AJAX call for sending messages and receiving chatbot responses.

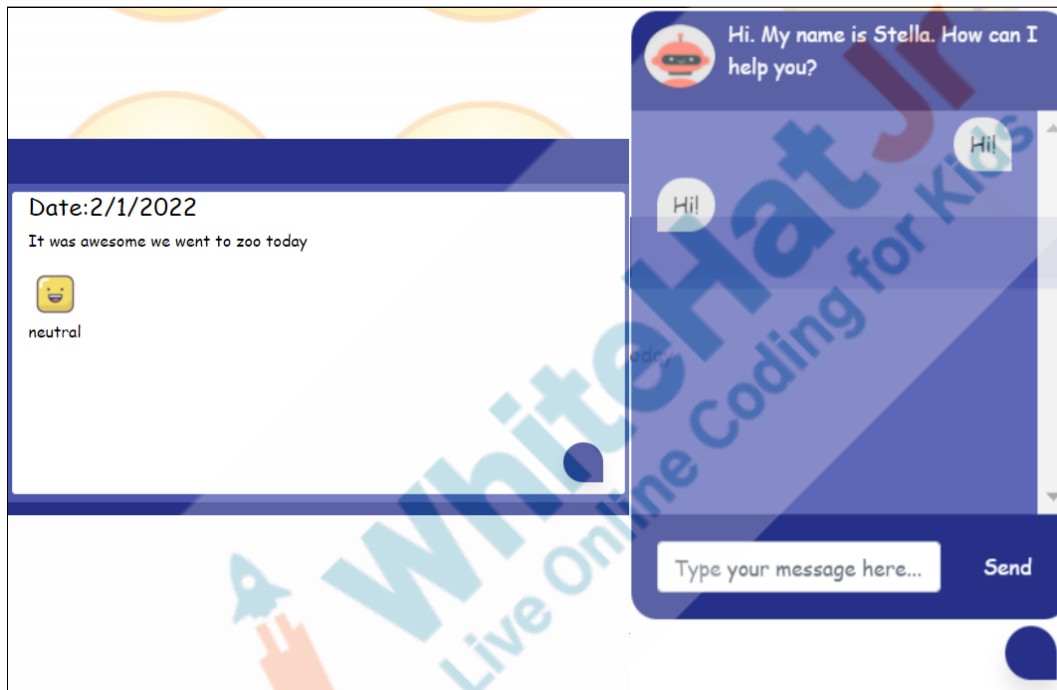
There will be two methods we'll be creating one for displaying chatbot and another is to ask questions to the chatbot.

Note: Refer [Teacher Activity3](#) for jQuery.

In this function, you can observe here we are using the class name so '.' is used as a **selector**. When the **chatbox_button** is clicked, the **chatbox_chat** div will pop up using the **toggle()** function.

After this, The **askBot()** function is called to chat with the bot.


```
59 function displayBot() {  
60     $('.chatbox__button').click(function () {  
61         $('.chatbox__chat').toggle()  
62     });  
63     //Start Conversation with Bot  
64     askBot()  
65 }
```



Let's write the **askBot()** function now.

1. When the send button is clicked the user request or question is given to chatbot and a response should be given according to the trained chatbot.
2. The user message or question is stored in the variable **user_bot_input_text**. The '#' is used as a selector for **bot_input_text**.
3. If the user is sending a message, then we have to display the user input and chatbot response in the **messages div**.

4. Thus, we are appending user messages in the **chat_messages** div.
5. After appending user messages the text input is cleared for sending new messages.
6. Again initialize the variable **user_bot_input_text** for getting new messages from the user.
7. Meanwhile, to get the response from the chatbot we have to write an AJAX call and API.

```

67 function askBot() {
68     $("#send_button").click(function () {
69         var user_bot_input_text = $("#bot_input_text").val()
70         if (user_bot_input_text != "") {
71             $("#chat_messages").append('<div class="user_messages">' + user_bot_input_text + ' </div>')
72             //Clear the text input box after sending message
73             $("#bot_input_text").val('');
74             let chat_input_data = {
75                 "user_bot_input_text": user_bot_input_text
76             }
77         }
78     })
79 }
80
81

```

So, since you know how to write AJAX calls and API. Are you excited to create your own chatbot on your webpage?

ESR: Yes

Ok. So let's start now.

Teacher Stops Screen Share

So now it's your turn.

Please share your screen with me.



Teacher Starts Slideshow
Slide 12 to 15

Refer to speaker notes and follow the instructions on each slide.

STUDENT-LED ACTIVITY - 20 mins

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Fullscreen.

Student Initiates Screen Share

ACTIVITY

- Create an AJAX call and Flask API for sending and receiving messages.

Teacher Action	Student Action
<p>Note: Guide the student to open Student Activity1. Download all the files. If student runs the app.py file they'll be able to see the Digi-Diary which was created previously.</p> <p>You can check the directory structure. It is the same as the previous classes. A trained model for sentiment analysis is added to predict emotions.</p> <p>Also, a trained model for the chatbot is added that can be used for sending responses.</p> <p>Let's continue with writing the AJAX call first. Can you tell me the structure of an AJAX call?</p> <p>Yes, in the same manner, you have to create an AJAX call. Keep in mind all the naming conventions used for the different elements of the HTML page.</p>	<p>ESR: Request type, url, data, data type and success function etc.</p>
<p>Write the AJAX call now use the following steps:</p> <ol style="list-style-type: none"> 1. Write the type of request we are sending to the API. It'll be the 'POST' method. 2. Define a URL for receiving the bot response. 3. Write stringify method to send the data into JSON format. 4. Write the type of data we are sending(JSON). 5. Define content type as 'application/json'. 6. On Success, the response from the chatbot should be displayed for each user message. 	

7. For each user message, append the bot response which is the result we'll be getting from API. let's call it **bot_response**.
8. So, in the messages div, append **bot_response**.
9. Also to make the chatbot scrollable to read previous messages, use the function **scrollTop()**. In this function select the element which should be scrollable and the height of the scrollable area.
10. If any error is present write the method for error which will display a dialogue box with the error message.

```
$.ajax({
  type: 'POST',
  url: "/bot-response",
  data: JSON.stringify(chat_input_data),
  dataType: "json",
  contentType: 'application/json',
  success: function (result) {
    $("#chat_messages").append('<div class="bot__messages">' + result.bot_response + ' </div>')
    $('.chatbox__messages__cotainer').animate({
      scrollTop: $('.chatbox__messages__cotainer')[0].scrollHeight, 1000);
    },
    error: function (result) {
      alert(result.responseJSON.message)
    }
  });
```

11. If the **Enter** key is pressed, send button should be activated.
12. So, select the **bot_input_text** for the event of keypress, the send button should be clicked.
13. Key code for the Enter key is **13** so write **e.which** parameter for comparing the key value.

```
104     $('#bot_input_text').keypress(function(e){  
105         //If Enter key(key code is 13) pressed  
106         if(e.which == 13){  
107             $('#send_button').click(); //Trigger Send Button Click Event  
108         }  
109     });
```

The last step would be to write the API to get the response from the user.

How do we write an API?

Great!

Now to create the API:

1. Use **@app.route** method for defining the endpoint for your chatbot response.
2. Write a function **bot()** for taking the request from AJAX. The request is stored in **user_bot_input_text**.
3. Call the method to get the chatbot response. This is the same method we have created previously to get the chatbot responses.
4. Then, send this response to the AJAX call.

ESR: The URL, get the data, process it, and send the response.

```

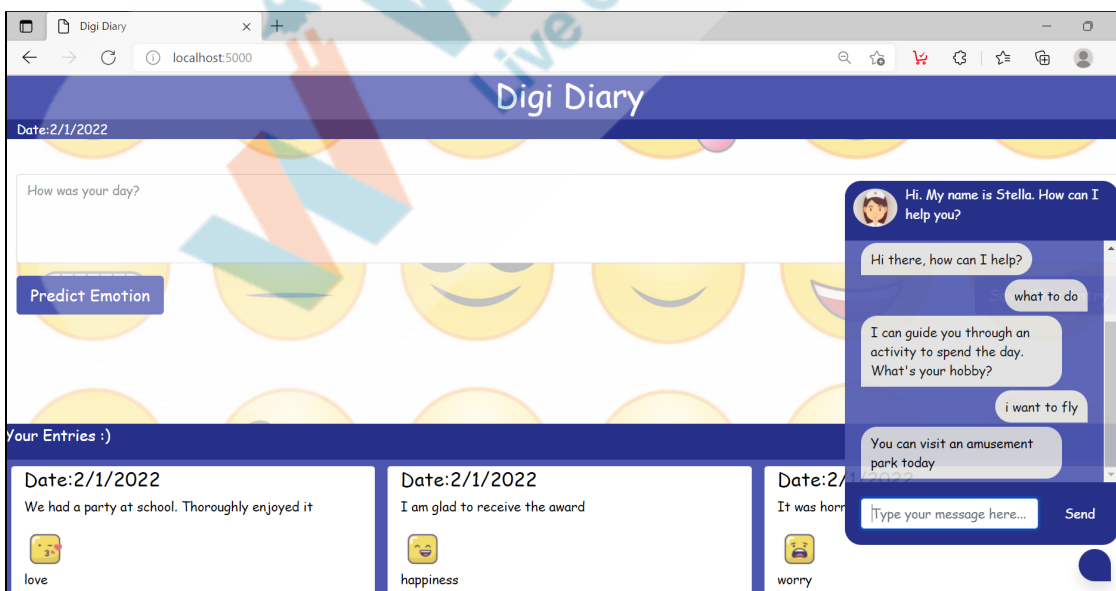
59 @app.route("/bot-response", methods=["POST"])
60 def bot():
61     # Get User Input
62     input_text = request.json.get("user_bot_input_text")
63
64     # Call the method to get bot response
65     bot_res = bot_response(input_text)
66
67     response = {
68         "bot_response": bot_res
69     }
70
71     return jsonify(response)
72
73 if __name__ == '__main__':
74     app.run(debug=True)



```

Let's run the file and check the output.

Go to the command prompt and create a virtual environment and run the file.

Note: Guide the student to create the environment and run the app.py file.



<p>Great!!! So you can chat with the chatbot and have fun. Also, your webpage can predict sentiment.</p> <p>We successfully integrated two Machine Learning models in our Digi Diary.</p>	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 05 mins	
<p>Teacher Starts Slideshow </p> <p>Slide 16 to 21</p>	
<p>Activity details</p> <p>Following are the WRAP-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Revise the current class activities. • Discuss the quizzes. 	
<p>WRAP-UP QUIZ Click on In-Class Quiz</p>	
<p>Continue WRAP-UP Session </p> <p>Slide 22 to 27</p>	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional) 	
<u>FEEDBACK</u>	

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
<p>You get “hats-off” for your excellent work!</p>	<p><i>Make sure you have given at least 2 hats-off during the class for:</i></p> <div> <div>Creatively Solved Activities +10</div> <div>Great Question +10</div> <div>Strong Concentration +10</div> </div>
<p>PROJECT OVERVIEW DISCUSSION</p> <p>Refer the document below in Activity Links Sections</p>	
<p>Teacher Clicks</p>	<p>✕ End Class</p>

ACTIVITY LINKS		
Activity Name	Description	Links
Teacher Activity 1	Previous Class Code	https://github.com/procodingclass/PRO-C120-Reference-Code
Teacher Activity 2	Boilerplate Code	https://github.com/procodingclass/PRO-C122-Teacher-Boilerplate-Code
Teacher Activity 3	SIMPLIFYING JavaScript - BASICS OF jQuery	https://docs.google.com/document/d/1LZmLOdYR6dZvTK9NAbkfNpdrACL68W_kXVIW79OctcM/edit?usp=sharing
Teacher Activity 4	Reference Code	https://github.com/procodingclass/PRO-C122-Reference-Code
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/a49b559f-ea8c-43a7-b389-68dfebf676e8.pdf

Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C122-Project-Solution.git
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/9937007e-1287-4a1a-8a98-50dfa64892cc.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/e19f2092-1498-4aec-93ce-86a1e2645559.pdf
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C122-Student-Boilerplate-Code