Back

Congrats Daniel! This project has been marked as completed.

Project Rating

★★★★☆

Teacher's Comment

"Good Work "

-

Was this helpful?

PRO-C122: WEB CHATBOT Completed

In Class 122, We Learned To Integrate A Chatbot On A Website Using Ajax, Flask And Html. In This Project, You Will Use The Same Principles To Create A Customer Support Chatbot.

Goal of the Project:

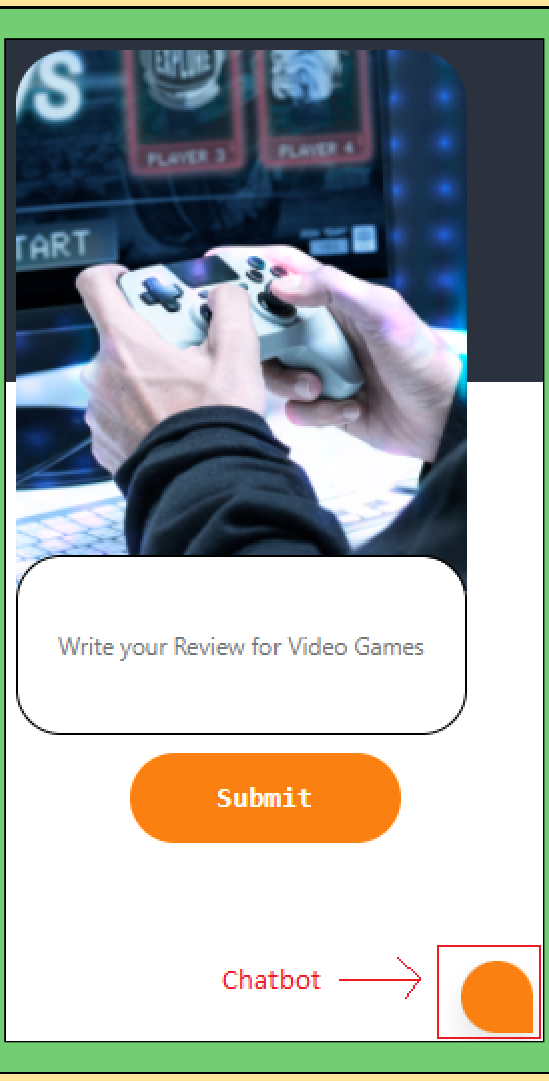
In class 122, we learned to integrate a chatbot on a website using AJAX, Flask and HTML. In this project, you will use the same principles to create a customer support chatbot.

Story:

Alex is very happy with the fact that he was able to create a fully functioning chatbot which will listen to customer queries and suggest them best possible solutions. Now, as the last step, he needs to integrate this chatbot into his website so that he can fully automate his business. Now, he already has a trained chatbot model, but he needs help to integrate that model into his website.

Can you help him to do so?

Project Template Output



Expected Output

Community Link

Publish to Community

Edit Your Project

Last Submitted

Previous Submissions

27th Feb 2024	Open Link

Start Project

Submit Your Project

Learn how to submit your project

Paste your project URL

Submit Project

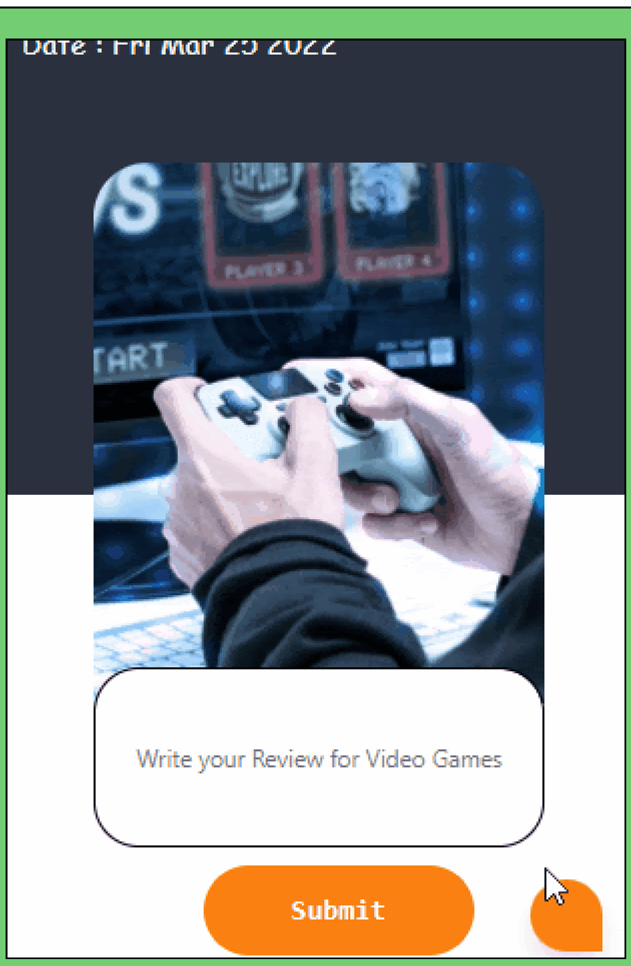
Class Summary

This project is based on your last class PRO-C122

View Class Summary

Ask a doubt to your tea

HELP

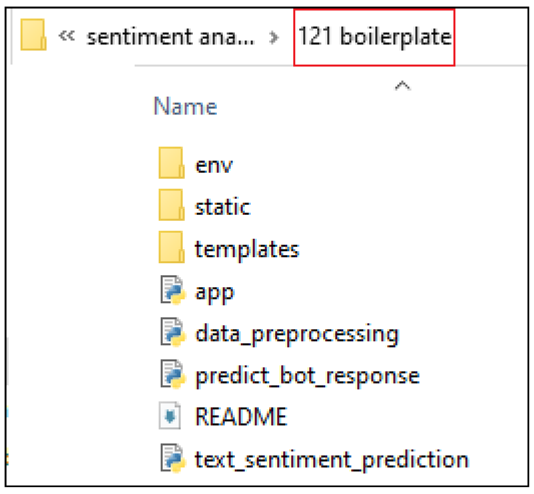


Refer link : https://s3-whjr-curriculum-uploads.whjr.online/f1c02bf1-4654-47b6-a37f-df03bbf1c011.gif

*This is just for your reference. We expect you to apply your own creativity to the project.

Getting Started:

1. Open the Boilerplate link and download all the files within a **new folder** on your system.
2. Open the **command prompt**, traverse to that folder and create a python virtual environment inside it in such a way, so that the **virtual environment**, **intents.json and data_preprocessing.py** files are within the same folder.

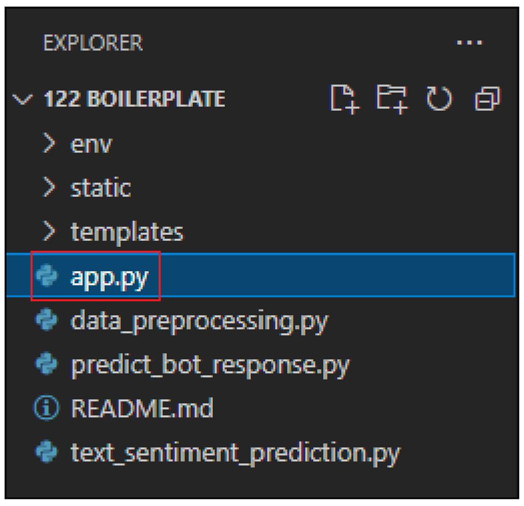


« sentiment ana... › 121 boilerplate

Name

- env
- static
- templates
- app
- data_preprocessing
- predict_bot_response
- README
- text_sentiment_prediction

3. Activate the virtual environment and install the **tensorflow , pandas , flask and nltk** libraries in it, using **pip install tensorflow==2.5.0 , pip install pandas , pip install flask and pip install nltk**.
4. Open the **folder** in Visual Studio code, and click on the **app.py** file.

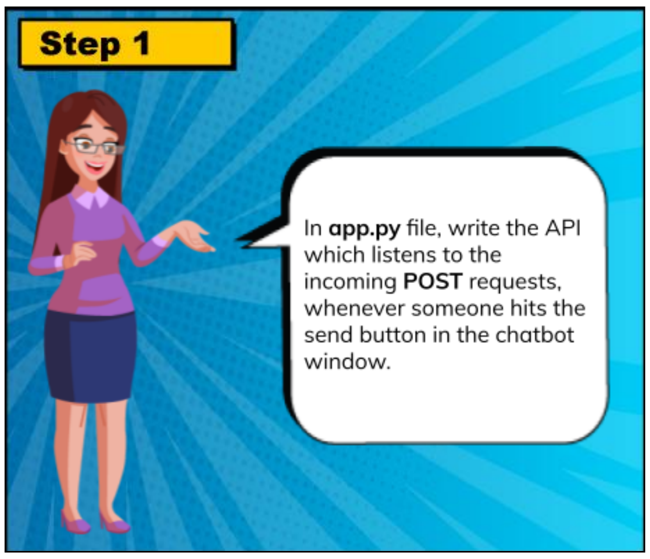


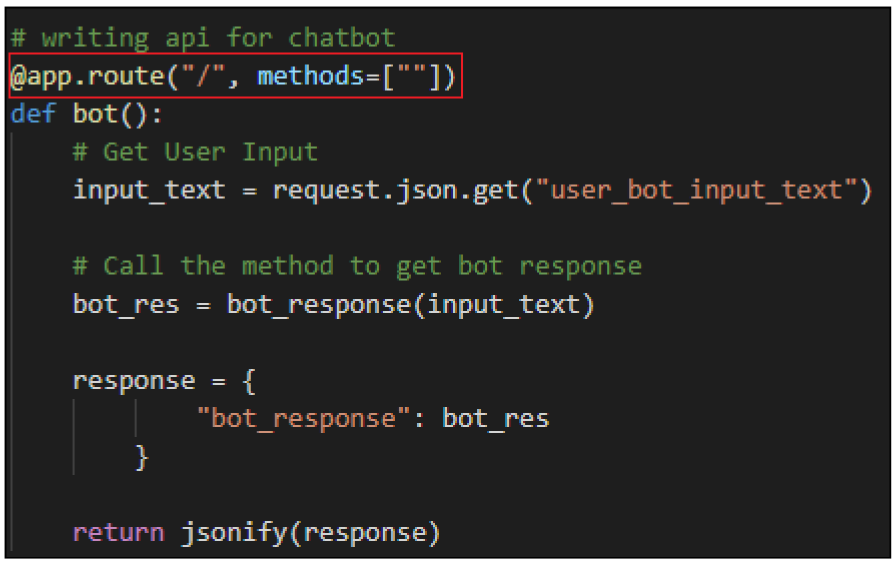
EXPLORER

122 BOILERPLATE

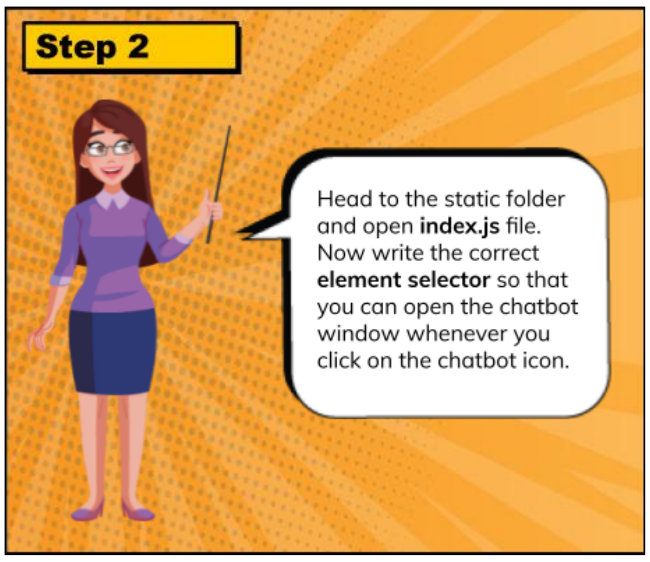
- env
- static
- templates
- app.py
- data_preprocessing.py
- predict_bot_response.py
- README.md
- text_sentiment_prediction.py

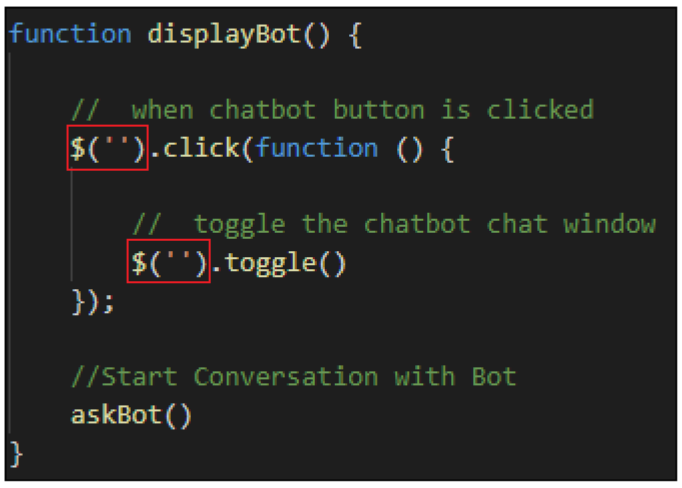
Specific Tasks to complete the Project:


Step 1
In **app.py** file, write the API which listens to the incoming **POST** requests, whenever someone hits the send button in the chatbot window.

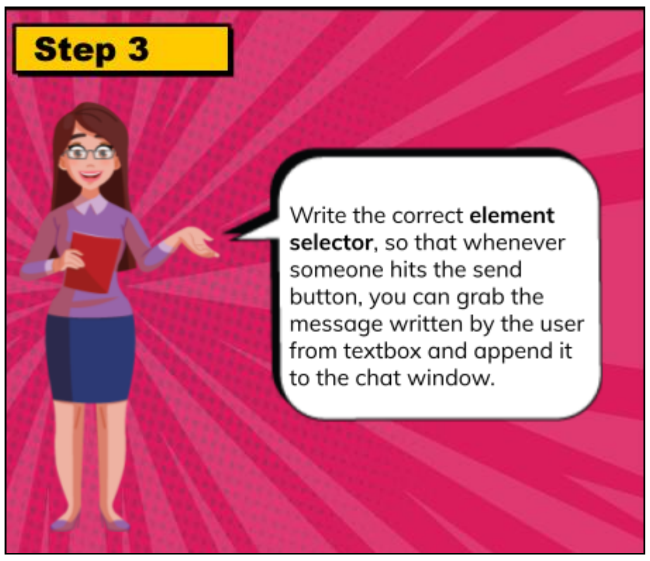


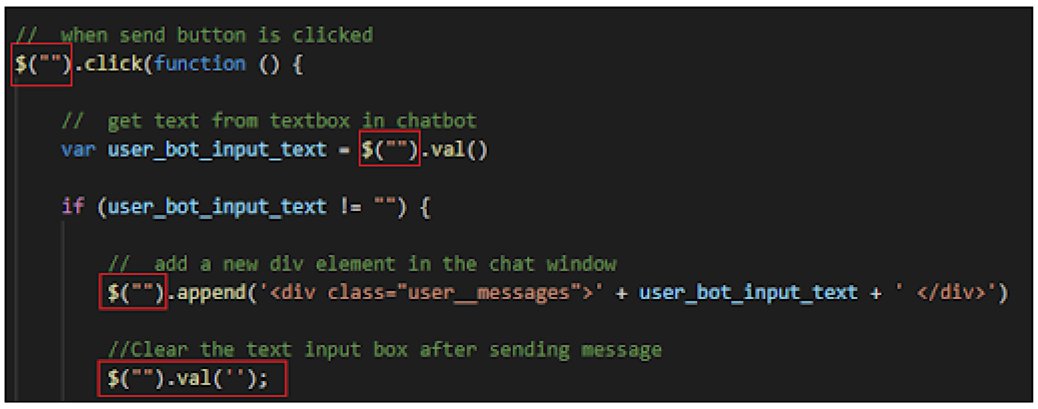
```
# writing api for chatbot
@app.route("/", methods=[""])
def bot():
    # Get User Input
    input_text = request.json.get("user_bot_input_text")

    # Call the method to get bot response
    bot_res = bot_response(input_text)

    response = {
            "bot_response": bot_res
        }

    return jsonify(response)
```


Step 2
Head to the static folder and open **index.js** file. Now write the correct **element selector** so that you can open the chatbot window whenever you click on the chatbot icon.

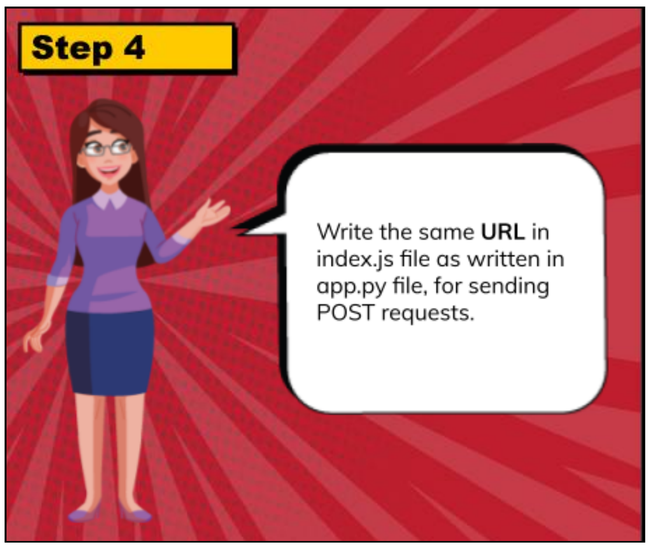


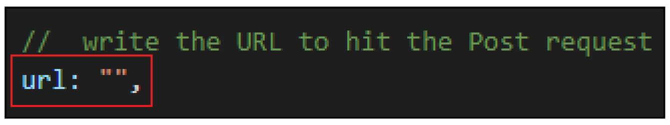
```
function displayBot() {

    //  when chatbot button is clicked
    $('').click(function () {

        //  toggle the chatbot chat window
        $('').toggle()
    });

    //Start Conversation with Bot
    askBot()
}
```


Step 3
Write the correct **element selector**, so that whenever someone hits the send button, you can grab the message written by the user from textbox and append it to the chat window.

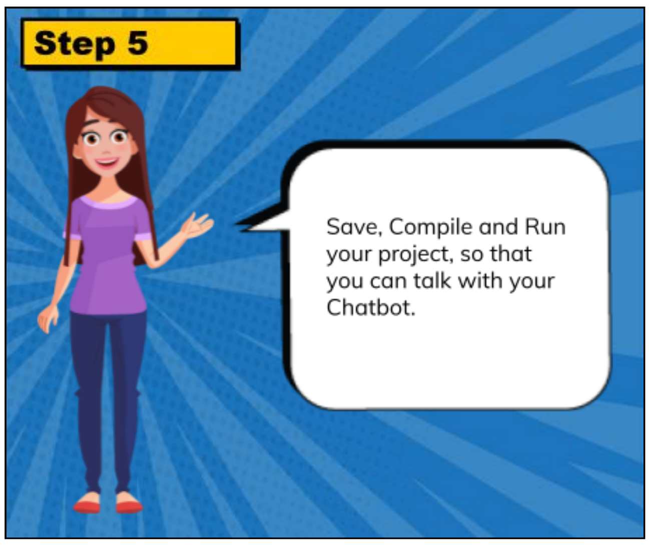


```
//  when send button is clicked
$("").click(function () {

    //  get text from textbox in chatbot
    var user_bot_input_text = $("").val()

    if (user_bot_input_text != "") {

        //  add a new div element in the chat window
        $("").append('<div class="user__messages">' + user_bot_input_text + ' </div>')

        //Clear the text input box after sending message
        $("").val('');
```


Step 4
Write the same **URL** in index.js file as written in app.py file, for sending POST requests.



```
//  write the URL to hit the Post request
url: "",
```


Step 5
Save, Compile and Run your project, so that you can talk with your Chatbot.

Submitting the Project:

1. **SAVE** all the changes made to the project.
2. Click on **"Run"** once to check if it is working.
3. Open GitHub and create a repository named **Project122**.
4. Upload files and click **Commit Changes**.
5. Copy the link and submit it in the Student Dashboard Projects panel against the correct class number.

Hints:

1. For step1, the URL at which we will sending our POST request, should be the same in both **app.py** and **index.js** files.

2. To change the formatting of the chatbot, change the code in **style.css** file.
3. Use the **dot** operator, if you want to select an element, using its class.

HELP