Artificial intelligence

PHASE 4:Building the chatbot by integrating it into a web app using flask

Integrating the chatbot into a web app using Flask involves a series of steps. Here's an overview of the process:

1. **Set Up Flask**: Make sure you have Flask installed. You can use pip to install it: `pip install Flask`.

2. **Create a Flask App**: Create a new Flask app by creating a Python file, for example, `app.py`. Import the necessary modules and create your Flask app instance.

3. **Define Routes**: Define the routes for your web app. For example, you might have a route for the home page and another for handling chatbot interactions.

```python
@app.route('/')
def home():
    # Your home page logic here

@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.form['user_input']
    # Process user input and get chatbot response
    bot_response = generate_response(user_input)
    return jsonify({'response': bot_response})
```

4. **HTML Templates**: Create HTML templates for your web pages. In this case, you might have an

HTML form for user input on the home page and a div to display chatbot responses.

5. **User Interface**: Use JavaScript to handle user input and asynchronously send it to the `/chat` route for processing. Update the chatbot responses on the web page without requiring a full page reload.

6. **Chatbot Logic**: Integrate your chatbot logic into the `/chat` route. You'll need to call your chatbot model to generate responses based on user input.

7. **Run the Flask App**: Start your Flask app by adding the following code at the end of your `app.py` file:

```python
if __name__ == '__main__':
    app.run(debug=True)
```

8. **Testing**: Test your web app locally to ensure it's working as expected.

9. **Deployment**: When your web app is ready, you can deploy it on a web server or a platform of your choice.

Remember that this is a high-level overview, and there are many details and customization options depending on your specific project requirements. You'll need to implement the chatbot logic, handle sessions, and consider user authentication and data storage if necessary.

PYTHON CODE:

Flask Framework Basic:

Getting Started

First, let make a very basic chatbot using basic Python skills like input/output and basic condition statements, which will take basic information from the user and print it accordingly.

Step 1: First, we will variables store user information.

```
User_Name  = None

User_Age  = None

User_Job = None
```

Step 2: Then, we will take input from the user.

```
print("Hello, I'm a Chatbot \n")

User_Name =  input("What is your name? ")

print("How are you {0}. \n".format(User_Name))

User_Age = input("What is your age? ")

print("Oh, so your age is {0}. \n".format(User_Age))

User_Job = input("What is your job profile? ")

print("So you're a  {0}. \n".format(User_Job))
```

When we run the above program, we will get the following output:

Hello, I'm a Chatbot

What is your name? pythonscholar

How are you pythonscholar.

What is your age? 26

Oh, so your age is 26.

What is your job profile? python developer

So you're a python developer.

So now, let's start creating a real chatbot and deploy it on Flask.

We will use the ChatterBot Python library, which is mainly developed for building chatbots.

What is ChatterBot, and how does it work?

ChatterBot is a machine learning library that helps to generate an automatic response based on the user's input. It uses a Natural Language Processing-based algorithm to generate repossessed based on the user's contexts.

Step 2: Then, we will name our chatbot. It can be anytime as per our need.

```
chatbot=ChatBot('Pythonscholar')
```

Step 3: We will start training our chatbot using its pre-defined dataset.

```
# Create a new trainer for the chatbot
trainer = ChatterBotCorpusTrainer(chatbot)

# Now, let us train our bot with multiple corpus
trainer.train("chatterbot.corpus.english.greetings",
              "chatterbot.corpus.english.conversations" )
```

chatterbot.corpus.english.greetings and chatterbot.corpus.english.conversations are the pre-defined dataset used to train small talks and everyday conversational to our chatbot.

Step 4: Then, we will check how our chatbot is responding to our question using the below code.

```
response = chatbot.get_response("How are you?")
print(response)
```

Now let's run the whole code and see what our chatbot responds to.

Whole Source Code:

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

chatbot=ChatBot('Pythonscholar')

# Create a new trainer for the chatbot
trainer = ChatterBotCorpusTrainer(chatbot)

# Now let us train our bot with multiple corpus
trainer.train("chatterbot.corpus.english.greetings",
        "chatterbot.corpus.english.conversations" )
response = chatbot.get_response("How are you?")
print(response)
```

The Output will be as follow:

```
I am doing well.
```

How to deploy a chatbot on Flask

First, we will install the Flask library in our system using the below command:

pip install flask

And for Google Colab use the below command, mostly Flask comes pre-install on Google Colab.

!pip install flask

But first, let's understand what the Flask framework in Python is.

What is Python Flask?

The Flask is a Python micro-framework used to create small web applications and websites using Python. Flask works on a popular templating engine called Jinja2, a web templating system combined with data sources to the dynamic web pages.

Now start developing the Flask framework based on the above ChatterBot in the above steps.

We have already installed the Flask in the system, so we will import the Python methods we require to run the Flask microserver.

Step 1: Import necessary methods of Flask and ChatterBot.

from flask import Flask, render_template, request

from chatterbot import ChatBot

```
from chatterbot.trainers import ChatterBotCorpusTrainer
```

Step 2: Then, we will initialize the Flask app by adding the below code.

```
#Flask initialisation
app = Flask(__name__)
```

Flask(name) is used to create the Flask class object so that Python code can initialize the Flask server.

Step 3: Now, we will give the name to our chatbot.

```
chatbot=ChatBot('Pythonscholar')
```

Step 4: Add training code for a chatbot.

```
# Create a new trainer for the chatbot
trainer = ChatterBotCorpusTrainer(chatbot)

# Now, let us train our bot with multiple corpus
trainer.train("chatterbot.corpus.english.greetings",
              "chatterbot.corpus.english.conversations" )
```

Step 5: We will create the Flask decorator and a route in this step.

```
@app.route("/")
def index():
        render_template("index.html")
```

The route() is a function of a Flask class used to define the URL mapping associated with the function.

Then we make an index function to render the HTML code associated with the index.html file using the render_template function.

In the next step, we will make a response function that will take the input from the user, and also, it will return the result or response from our trained chatbot.

Step 6: Create a function to take input from the user and respond accordingly.

```
@app.route("/get", methods=["POST"])
def chatbot_response():
        msg = request.form["msg"]
        response = chatbot.get_response(msg)
```

```
        return response
```

Step 7: Then, we will add the final code that will start the Flask server after interpreting the whole code.

```
if __name__ == "__main__":
   app.run()
```

The complete code will look like this:

```
from flask import Flask, render_template, request
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

#Flask initialization
app = Flask(__name__)

chatbot=ChatBot('Pythonscholar')

# Create a new trainer for the chatbot
trainer = ChatterBotCorpusTrainer(chatbot)

# Now let us train our bot with multiple corpus
trainer.train("chatterbot.corpus.english.greetings",
        "chatterbot.corpus.english.conversations" )
```

```
@app.route("/")
def index():

        return render_template("index.html")



@app.route("/get", methods=["GET","POST"])
def chatbot_response():
   msg = request.form["msg"]
   response = chatbot.get_response(msg)
   return str(response)


if __name__ == "__main__":
   app.run()
```

As per Jinja2 implementation, we have to create two folders for storing HTML and CSS files; while working with the Jinja2 engine, it is necessary to make a folder with a name template to add HTML files, and for other files like CSS, javascript, or image we have to make a folder with the name static but it optional but creating template folder is compulsory.

First, we will make an HTML file called index.html inside the template folder.

```
<!DOCTYPE html>
```

```
<html>


<head>
  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css')}}" />
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>


<body>
  <div class="row">
    <div class="col-md-10 mr-auto ml-auto">
  <h1>Pythonscholar ChatBot</h1>
  <form>
    <div id="chatbox">
      <div class="col-md-8 ml-auto mr-auto">
        <p class="botText"><span>Hi! I'm Pythonscholar.</span></p>
      </div>
    </div>
    <div id="userInput" class="row">
      <div class="col-md-10">
        <input id="text" type="text" name="msg" placeholder="Message" class="form-control">
        <button type="submit" id="send" class="btn btn-warning">Send</button>
      </div>
    </div>
  </form>
</div>
```

```
</div>

<script>
  $(document).ready(function() {
    $("form").on("submit", function(event) {
      var rawText = $("#text").val();
      var userHtml = '<p class="userText"><span>' + rawText + "</span></p>";
      $("#text").val("");
      $("#chatbox").append(userHtml);
      document.getElementById("userInput").scrollIntoView({
        block: "start",
        behavior: "smooth",
      });
      $.ajax({
        data: {
          msg: rawText,
        },
        type: "POST",
        url: "/get",
      }).done(function(data) {
        var botHtml = '<p class="botText"><span>' + data + "</span></p>";
        $("#chatbox").append($.parseHTML(botHtml));
        document.getElementById("userInput").scrollIntoView({
          block: "start",
          behavior: "smooth",
```

```
</div>


<script>
    $(document).ready(function() {
        $("form").on("submit", function(event) {
            var rawText = $("#text").val();
            var userHtml = '<p class="userText"><span>' + rawText + "</span></p>";
            $("#text").val("");
            $("#chatbox").append(userHtml);
            document.getElementById("userInput").scrollIntoView({
                block: "start",
                behavior: "smooth",
            });
            $.ajax({
                data: {
                    msg: rawText,
                },
                type: "POST",
                url: "/get",
            }).done(function(data) {
                var botHtml = '<p class="botText"><span>' + data + "</span></p>";
                $("#chatbox").append($.parseHTML(botHtml));
                document.getElementById("userInput").scrollIntoView({
                    block: "start",
                    behavior: "smooth",
```

```
            });
        });
        event.preventDefault();
    });
  });
</script>
</body>

</html>
```

We will not understand HTML and jquery code as jquery is a vast topic.

We will create a style.css file, save it in the static folder, and use the below code.

```
body {
  font-family: Garamond;
}

h1 {
  color: black;
  margin-bottom: 0;
  margin-top: 0;
  text-align: center;
```

```
    font-size: 40px;
}

h3 {
    color: black;
    font-size: 20px;
    margin-top: 3px;
    text-align: center;
}
.row {
    display: flex;
    flex-wrap: wrap;
    margin-right: -15px;
    margin-left: -15px;
}

.ml-auto{
    margin-left:auto !important;
}
.mr-auto{
    margin-right:auto !important;
}

.col-md-10,.col-md-8,.col-md-4{
    position: relative;
```

```
    width: 100%;
    min-height: 1px;
    padding-right: 15px;
    padding-left: 15px;
}
.col-md-8{flex:0 0 66.666667%;max-width:66.666667%}
.col-md-4{flex:0 0 33.333333%;max-width:33.333333%}
.col-md-10{flex:0 0 83.333333%;max-width:83.333333%}

.form-control {
    background: no-repeat bottom,50% calc(100% - 1px);
    background-image: none, none;
    background-size: auto, auto;
    background-size: 0 100%,100% 100%;
    border: 0;
    height: 36px;
    transition: background 0s ease-out;
    padding-left: 0;
    padding-right: 0;
    border-radius: 0;
    font-size: 14px;
}
.form-control {
    display: block;
    width: 100%;
```

```
    padding: .4375rem 0;
    padding-right: 0px;
    padding-left: 0px;
    font-size: 1rem;
    line-height: 1.5;
    color: #495057;
    border:none;
    background-color: transparent;
    background-clip: padding-box;
    border-bottom: 1px solid #d2d2d2;
    box-shadow: none;
    transition: border-color .15s ease-in-out,box-shadow .15s ease-in-out;
}
.btn {
    float: left;
    text-align: center;
    white-space: nowrap;
    vertical-align: middle;
    user-select: none;
    border: 1px solid transparent;
    padding: .46875rem 1rem;
    font-size: 1rem;
    line-height: 1.5;
    border-radius: .25rem;
    transition: color .15s ease-in-out,background-color .15s ease-in-out,border-color .15s ease-in-out,box-shadow .15s ease-in-out;
```

```
}
.btn-warning {
  color: #fff;
  background-color: #f08f00;
  border-color: #c27400;
}
.btn.btn-warning:active, .btn.btn-warning:focus, .btn.btn-warning:hover {
  box-shadow: 0 14px 26px -12px rgba(255,152,0,.42),0 4px 23px 0 rgba(0,0,0,.12),0 8px 10px -5px rgba(255,152,0,.2);
}

button, input, optgroup, select, textarea {
  margin: 0;
  font-family: inherit;
  font-size: inherit;
  line-height: inherit;
  overflow:visible;
}
#chatbox {
  background-color: cyan;
  margin-left: auto;
  margin-right: auto;
  width: 80%;
  min-height: 70px;
  margin-top: 60px;
}
```

```
#userInput {
    margin-left: auto;
    margin-right: auto;
    width: 40%;
    margin-top: 60px;
}

#textInput {
    width: 87%;
    border: none;
    border-bottom: 3px solid #009688;
    font-family: monospace;
    font-size: 17px;
}

#buttonInput {
    padding: 3px;
    font-family: monospace;
    font-size: 17px;
}

.userText {
    color: white;
    font-family: monospace;
```

```
    font-size: 17px;
    text-align: right !important;
    line-height: 30px;
    margin: 5px;
}

.userText span {
    background-color: #009688;
    padding: 10px;
    border-radius: 2px;
}

.botText {
    color: white;
    font-family: monospace;
    font-size: 17px;
    text-align: left;
    line-height: 30px;
    margin: 5px;
}

.botText span {
    background-color: #ef5350;
    padding: 10px;
    border-radius: 2px;
```

```
}

#tidbit {
    position: absolute;
    bottom: 0;
    right: 0;
    width: 300px;
}
```

We are all set to run our Flask application.

So let's start our chatbot by running a Python file