CREATE A CHATBOT IN PYTHON

Chatbot:

Chatbots are conversational tools that perform routine tasks efficiently. It Communicates with users using interactive text or speech capabilities. People likethem because they help them get through those tasks quickly so they can focus
their attention on high-level, strategic, and engaging activities

Documentation

Problem statement:

Chatbots offer a convenient way for customers to interact with businesses and get the information they need 24/7. However, traditional chatbots can be limited in their ability to understand and respond to complex questions. Additionally, many chatbots are not integrated with other systems, making it difficult for them to providecomprehensive answers.

Design thinking process:

The design thinking process is a human-centered approach to innovation that focuses on understanding the needsof users and developing solutions that meet those needs. To develop a chatbot that addresses the limitations of traditionalchatbots, we used the following design thinking steps:

- **Empathize:** We interviewed customer service representatives to understand the most common questions that customers ask and the challenges they face inanswering those questions. We also analyzed customer feedback to identify areas where the chatbot could improve.
- **Define:** Based on our findings, we defined the problem as follows:
    - Traditional chatbots are limited in their ability to understand and respondto complex questions.
- **Ideate:** We generated a variety of ideas for how to address the problem. Someof our ideas included:
    - Developing a chatbot that uses natural language processing (NLP) tobetter understand user queries.
    - Integrating the chatbot with other systems, such as the customer relationship management (CRM) system and the knowledge base.
- **Prototype:** We developed a prototype of the chatbot using a chatbotdevelopment platform. The prototype included the following features:
    - NLP capabilities to better understand user queries
    - Integration with the CRM system to retrieve customer information
    - The ability to escalate to a human agent
- **Test:** We tested the prototype with a group of users to get feedback on its usability and functionality. Based on the feedback, we made necessary changesto the prototype.

Phases of development:

The chatbot development process can be divided into the following phases:

1. **Requirements gathering and analysis:** This phase involves identifying the needs of the business and the users, and developing a set of requirements for the chatbot.
2. **Design:** This phase involves designing the chatbot's conversation flow, user interface, and integration with other systems.
3. **Development:** This phase involves developing the chatbot's code and implementing the features that were designed in the previous phase.
4. **Testing:** This phase involves testing the chatbot to ensure that it meets the requirements and that it is easy to use.
5. **Deployment:** This phase involves deploying the chatbot to the production environment.

2. Libraries Used and Integration of NLP Techniques

- **TensorFlow:** A machine learning library that we used to train the chatbot's NLP model.
- **Flask:** A web framework that we used to develop the chatbot's web application.

We integrated NLP techniques into the chatbot as follows:

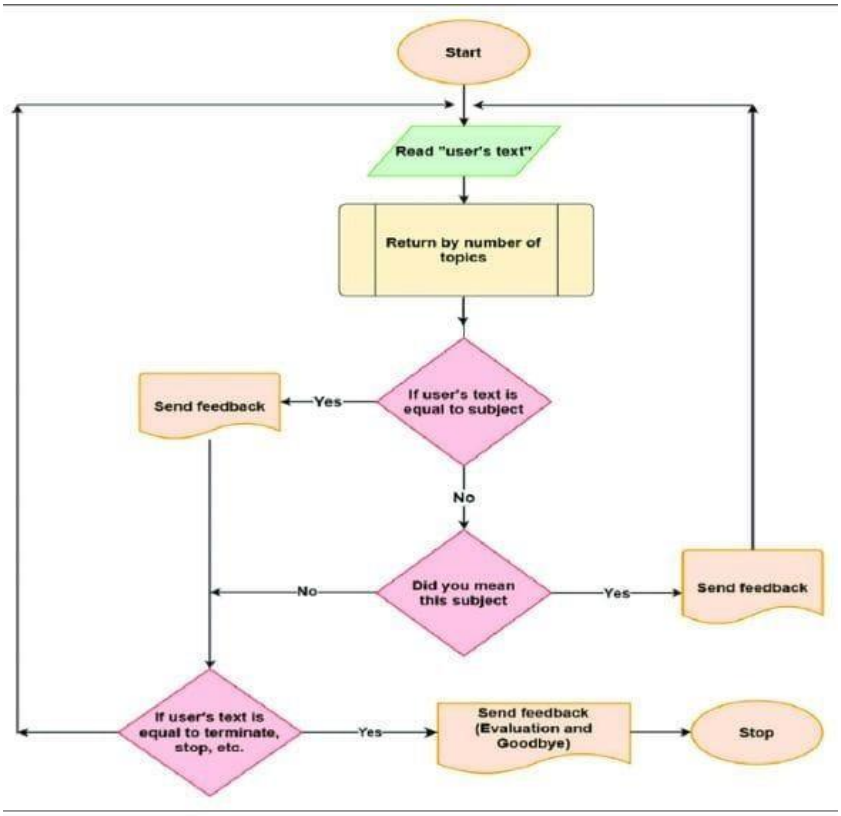
- Natural language understanding (NLU): We used a pre-trained NLU model to extract relevant information from user queries. For example, if a user asks "What is the status of my order?", the NLU model would extract the order number from the query.
- Natural language generation (NLG): We used an NLG model to generate responses to user queries. For example, if a user asks "What is the status of my order?", the NLG model would generate a response like "Your order is currently being processed and is expected to ship within 2 business days."

3. How the Chatbot Interacts with Users andthe Web Application

- The chatbot interacts with users through a web-based interface. Users can access the chatbot by visiting the company's website and clicking on the "Chat with us" button.
- When a user starts a chat session, the chatbot will greet the user and ask them how it can help. The user can then type their question into the chat window. The chatbot will use its NLP capabilities to understand the user's query and generate a response.
- If the chatbot is unable to answer the user's question, it will provide the user with the ability to escalate to a human agent.

The chatbot is integrated with the web application in the following ways:

- The chatbot can access customer information from the CRM system.
- The chatbot can access product information from the e-commerce platform.
- The chatbot can access knowledge base articles from the knowledge management system.



Submission

Installation:

Flask:

```
[ ] pip install flask
```

Transformers:

```
[ ] pip install transformers
```

Torch:

```
[ ] pip install torch
```

Import Libraries:

```
[ ] from flask import Flask, render_template, request, jsonify
    from transformers import AutoModelForCausalLM, AutoTokenizer
    import torch
```

program:

```
[ ] app = Flask(__name__)

    responses = {}
    with open('dialogs.txt', 'r') as file:
        for line in file:
            user_input, bot_dialogs = line.strip().split(':')
            responses[user_input.lower()] = bot_dialogs

    @app.route('/')
    def index():
        return render_template('chat.html')

    @app.route("/get", methods=["GET", "POST"])
    def chat():
      user_message = request.form['user_message'].lower()
      bot_dialogs = responses.get(user_message, "I'm sorry, I don't understand that.")
      return bot_dialogs

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

Chat.html

colab.research.google.com/drive/1Ga1Wcl_es44zOJS6POpyn1kO7cYTP3kY#scrollTo=B-HAeQuXAmFL

+ Code + Text Connect

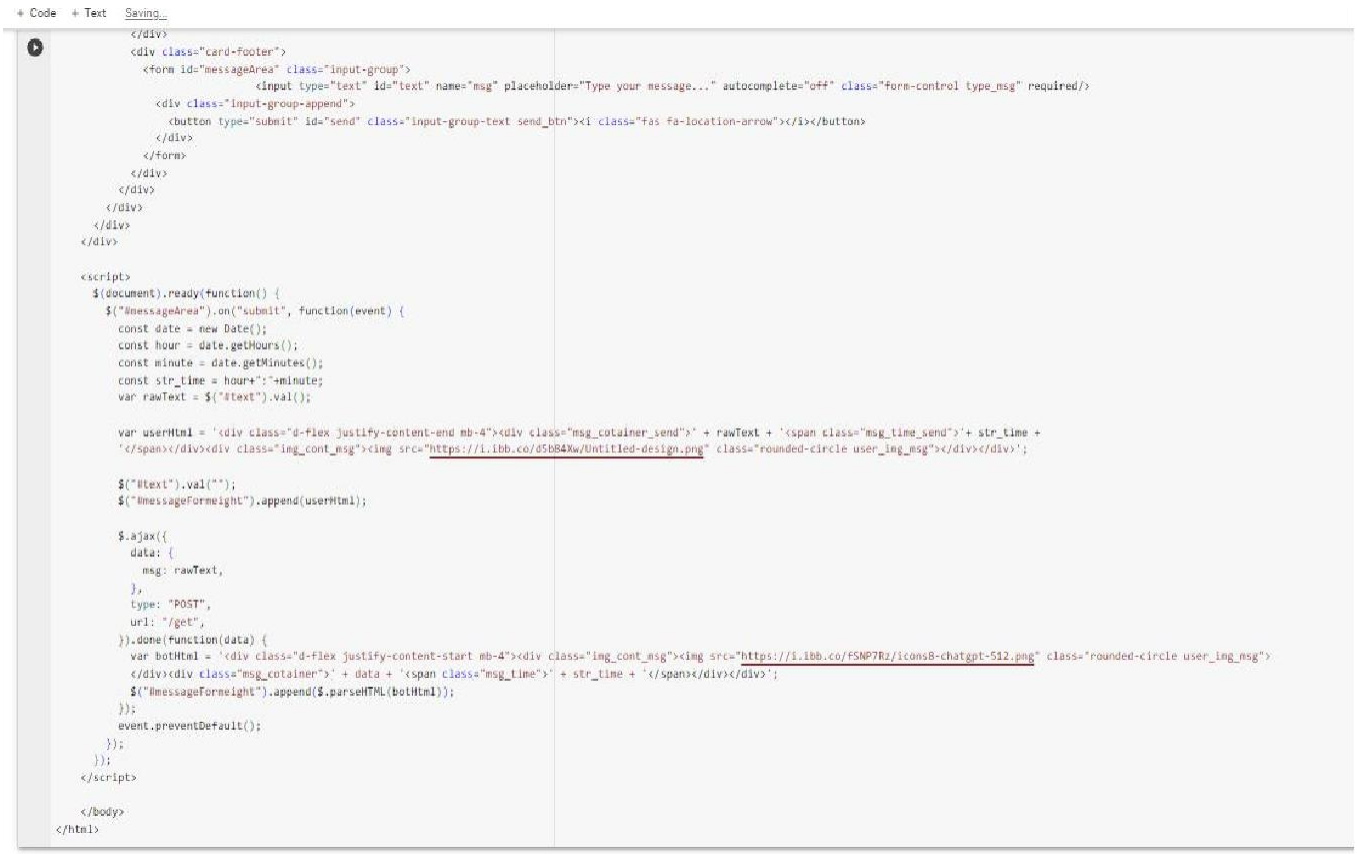
```
<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">
<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<!DOCTYPE html>
<html>
  <head>
    <title>Chatbot</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU" crossorigin="anonymous">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css')}}"/>
  </head>


  <body>
    <div class="container-fluid h-100">
      <div class="row justify-content-center h-100">
        <div class="col-md-8 col-xl-6 chat">
          <div class="card">
            <div class="card-header msg_head">
              <div class="d-flex bd-highlight">
                <div class="img_cont">
                  <img src="https://i.ibb.co/fSNP7Rz/icons8-chatgpt-512.png" class="rounded-circle user_img">
                  <span class="online_icon"></span>
                </div>
                <div class="user_info">
                  <span>ChatBot</span>
                  <p>Ask me anything!</p>
                </div>
              </div>
            </div>
            <div id="messageFormeight" class="card-body msg_card_body">


            </div>
            <div class="card-footer">
              <form id="messageArea" class="input-group">
                              <input type="text" id="text" name="msg" placeholder="Type your message..." autocomplete="off" class="form-control type_msg" required/>
                <div class="input-group-append">
                  <button type="submit" id="send" class="input-group-text send_btn"><i class="fas fa-location-arrow"></i></button>
```

Type here to search ENG 20:43 01-11-2023



```
+ Code + Text  Saving...
            </div>
            <div class="card-footer">
              <form id="messageArea" class="input-group">
                                <input type="text" id="text" name="msg" placeholder="Type your message..." autocomplete="off" class="form-control type_msg" required/>
                <div class="input-group-append">
                  <button type="submit" id="send" class="input-group-text send_btn"><i class="fas fa-location-arrow"></i></button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>

    <script>
      $(document).ready(function() {
        $("#messageArea").on("submit", function(event) {
          const date = new Date();
          const hour = date.getHours();
          const minute = date.getMinutes();
          const str_time = hour+":"+minute;
          var rawText = $("#text").val();

          var userHtml = '<div class="d-flex justify-content-end mb-4"><div class="msg_cotainer_send">' + rawText + '<span class="msg_time_send">'+ str_time +
          '</span></div><div class="img_cont_msg"><img src="https://i.ibb.co/d5bB4kw/Untitled-design.png" class="rounded-circle user_img_msg"></div></div>';

          $("#text").val("");
          $("#messageFormeight").append(userHtml);

          $.ajax({
            data: {
              msg: rawText,
            },
            type: "POST",
            url: "/get",
          }).done(function(data) {
            var botHtml = '<div class="d-flex justify-content-start mb-4"><div class="img_cont_msg"><img src="https://i.ibb.co/fSNP7Rz/icons8-chatgpt-512.png" class="rounded-circle user_img_msg">
            </div><div class="msg_cotainer">' + data + '<span class="msg_time">' + str_time + '</span></div></div>';
            $("#messageFormeight").append($.parseHTML(botHtml));
          });
          event.preventDefault();
        });
      });
    </script>

    </body>
</html>
```

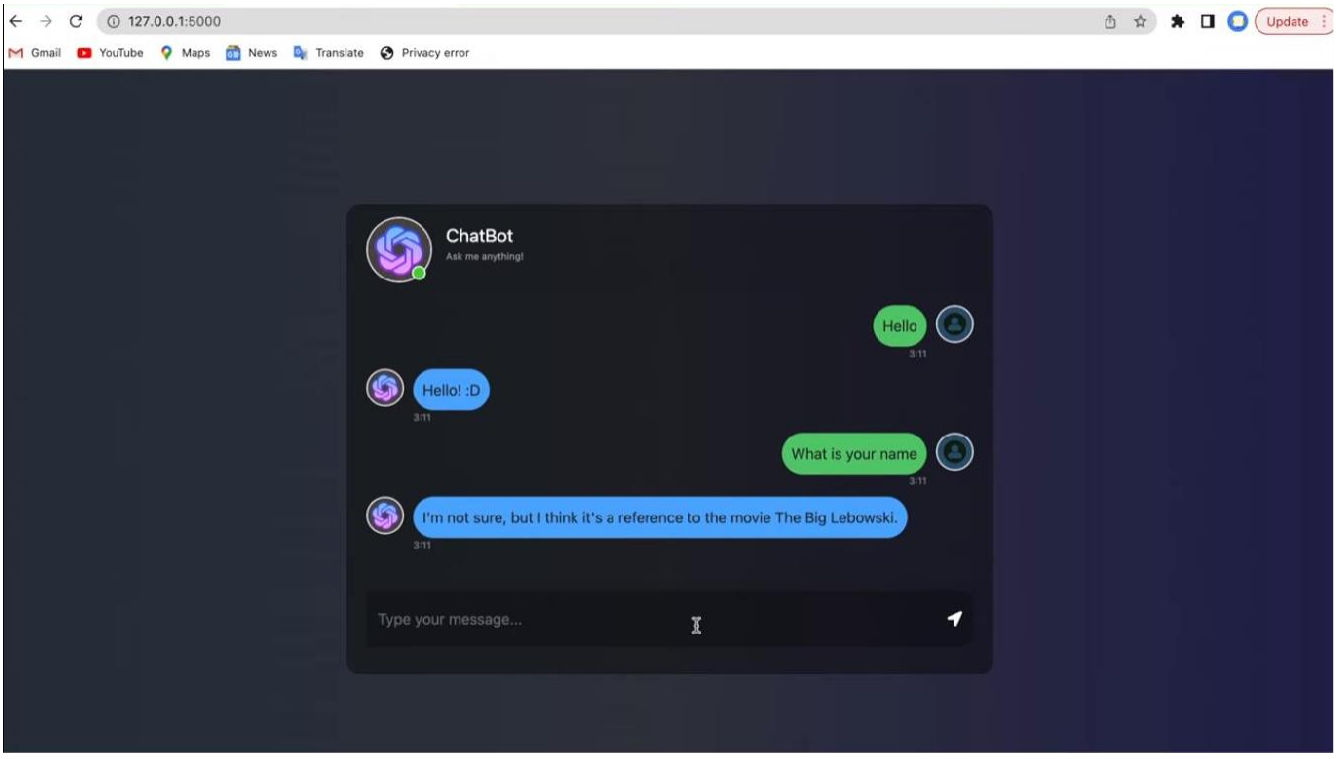
Run the Flask application:

- Save the above code in a Python file (app.py).
- Run the application using python app.py.

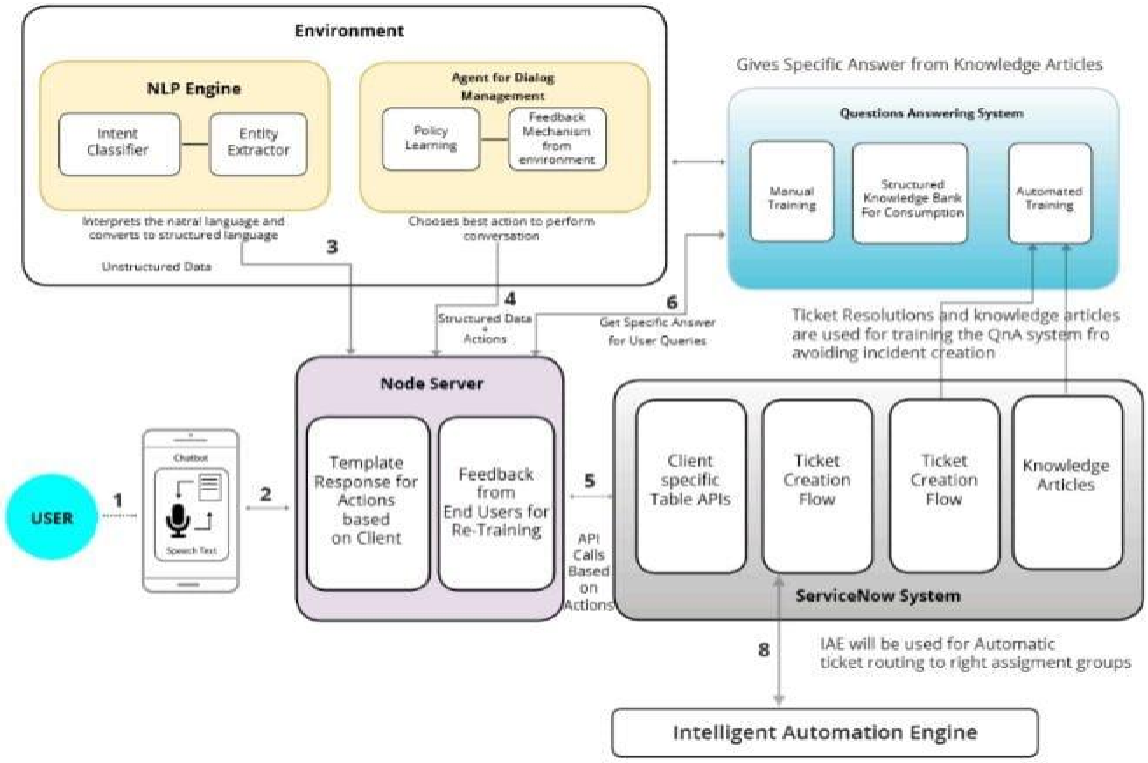
Interact with your chatbot:

- Visit http://127.0.0.1:5000 in youí web bíowseí to check if the app is íunning.
- I'o chat with the bot, you can send a POSI' íequest tohttp://127.0.0.1:5000 with a useí_message paíameteí containing youí input, and the bot will íespond with the appíopíiate íesponse fíom the text file.

Output:



Architectural Diagram in Chatbot:



Conclusion:

Building a chatbot by integrating it into a web app using Flask is a relatively straight forward process. By following the steps outlined above, you can create a chatbot that can interact with users in a natural and engaging way.