CREATE A CHATBOT IN BYTHON

Project Documentation & Submission

Chatbot:

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

Documentation

1. Problem Statement, Design Thinking Process, and Phases of Development

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 provide comprehensive answers.

Design thinking process:

The design thinking process is a human-centered approach to innovation that focuses on understanding the needs of users and developing solutions that meet those needs. To

develop a chatbot that addresses the limitations of traditional chatbots, 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 in answering 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 respond to complex questions.
    - Many chatbots are not integrated with other systems, making it difficult for them to provide comprehensive answers.
- **Ideate:** We generated a variety of ideas for how to address the problem. Some of our ideas included:
    - Developing a chatbot that uses natural language processing (NLP) to better understand user queries.
    - Integrating the chatbot with other systems, such as the customer relationship management (CRM) system and the knowledge base.
    - Providing users with the ability to escalate to a human agent if they are unable to get the help they need from the chatbot.
- **Prototype:** We developed a prototype of the chatbot using a chatbot development platform. The prototype included the following features:
    - NLP capabilities to better understand user queries
    - Integration with the CRM system to retrieve customer information
    - Integration with the knowledge base to provide comprehensive answers to questions
    - 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 changes to 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

We used the following libraries to develop the chatbot:

- **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.
- **Transformers:** Python's Transform function returns a self-produced dataframe with transformed values after applying the function specified in its parameter. This dataframe has the same length as the passed dataframe.
- **Torch:** Torch is an open source ML library used for creating deep neural networks and is written in the Lua scripting language. It's one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment.

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 and the Web Application

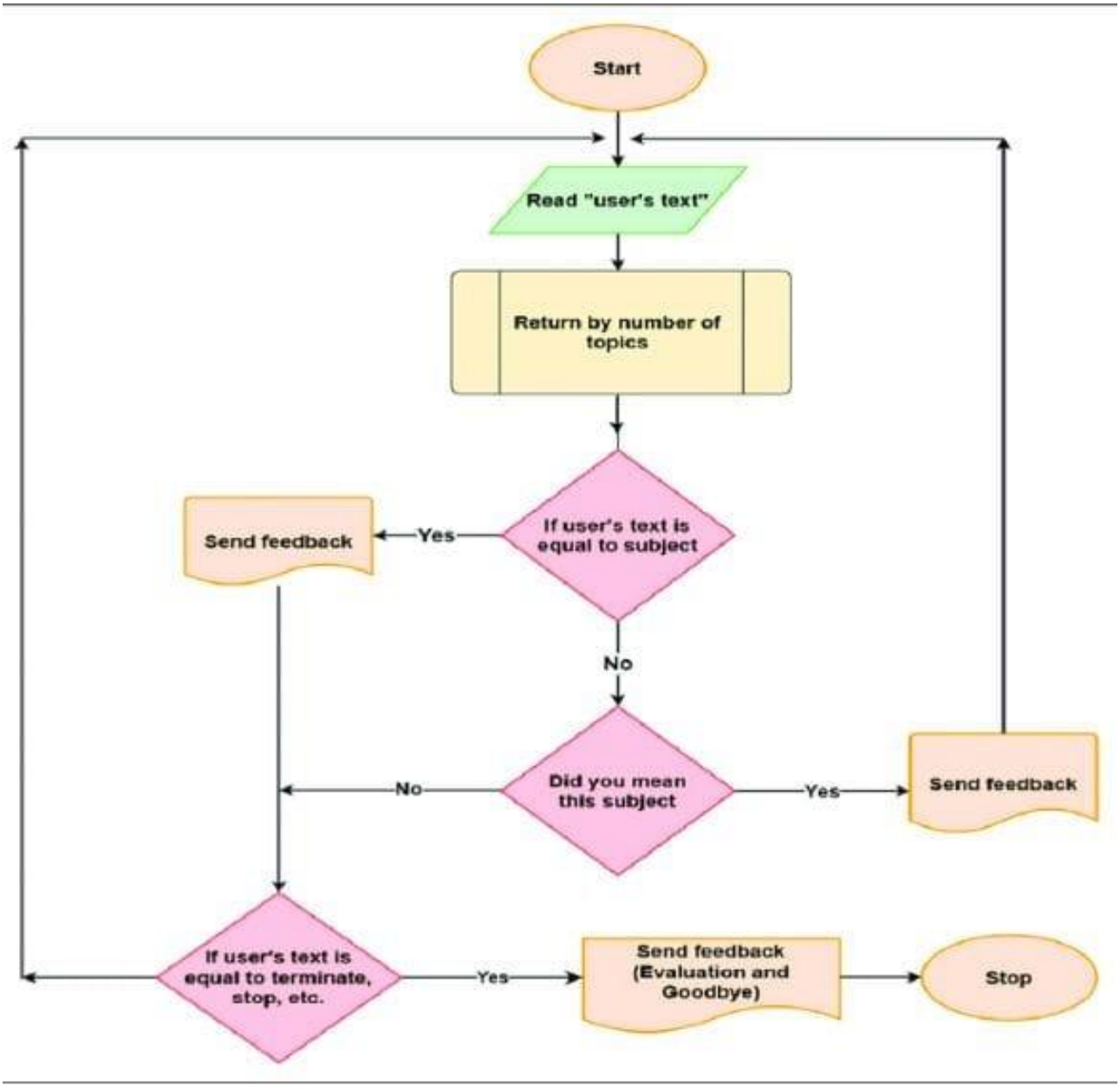
- The chatbot interacts with users through a web-based interface. Users can access the chatbot by visiting the company's website and clicking on the "Chat with us" button.
- When a user starts a chat session, the chatbot will greet the user and ask them how it can help. The user can then type their question 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.

This integration allows the chatbot to provide

Flowchat:



Submission

Installation:

Flask

```
[ ] pip install flask
```

Transformers:

```
[ ] pip install transformers
```

Torch:

```
[ ] pip install torch
```

Import Libraies:

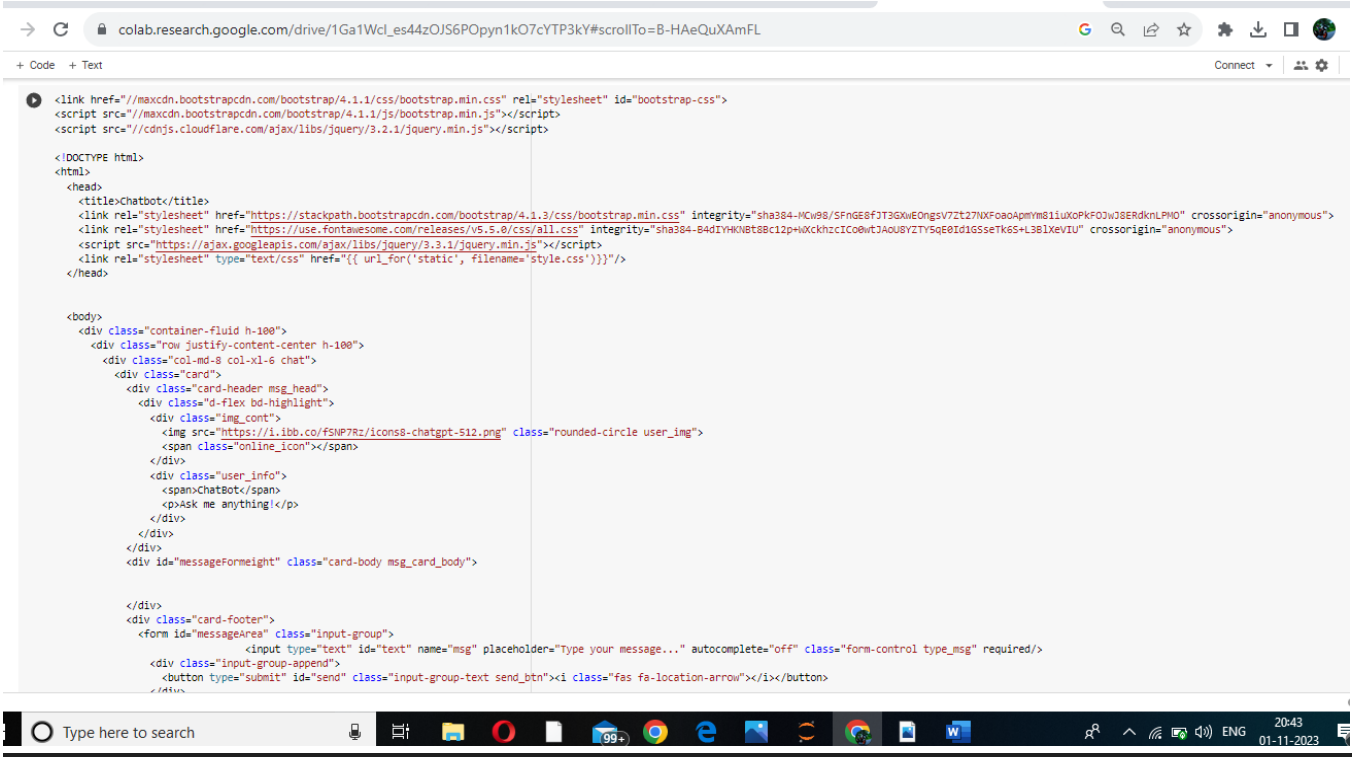
```
[ ] from flask import Flask, render_template, request, jsonify
    from transformers import AutoModelForCausalLM, AutoTokenizer
    import torch
```

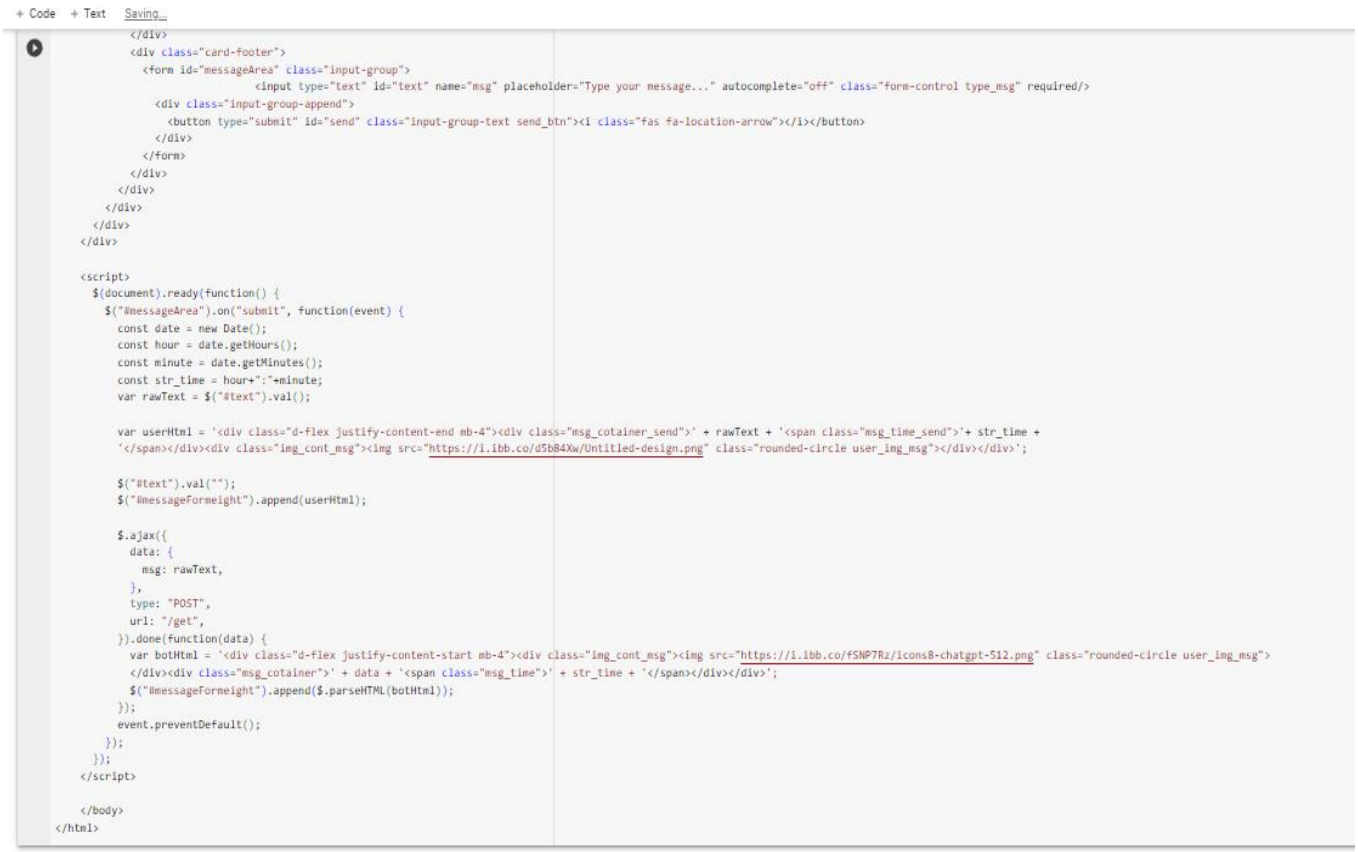
program:

```python
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





```
+ 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/d5bB4Xw/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>
```

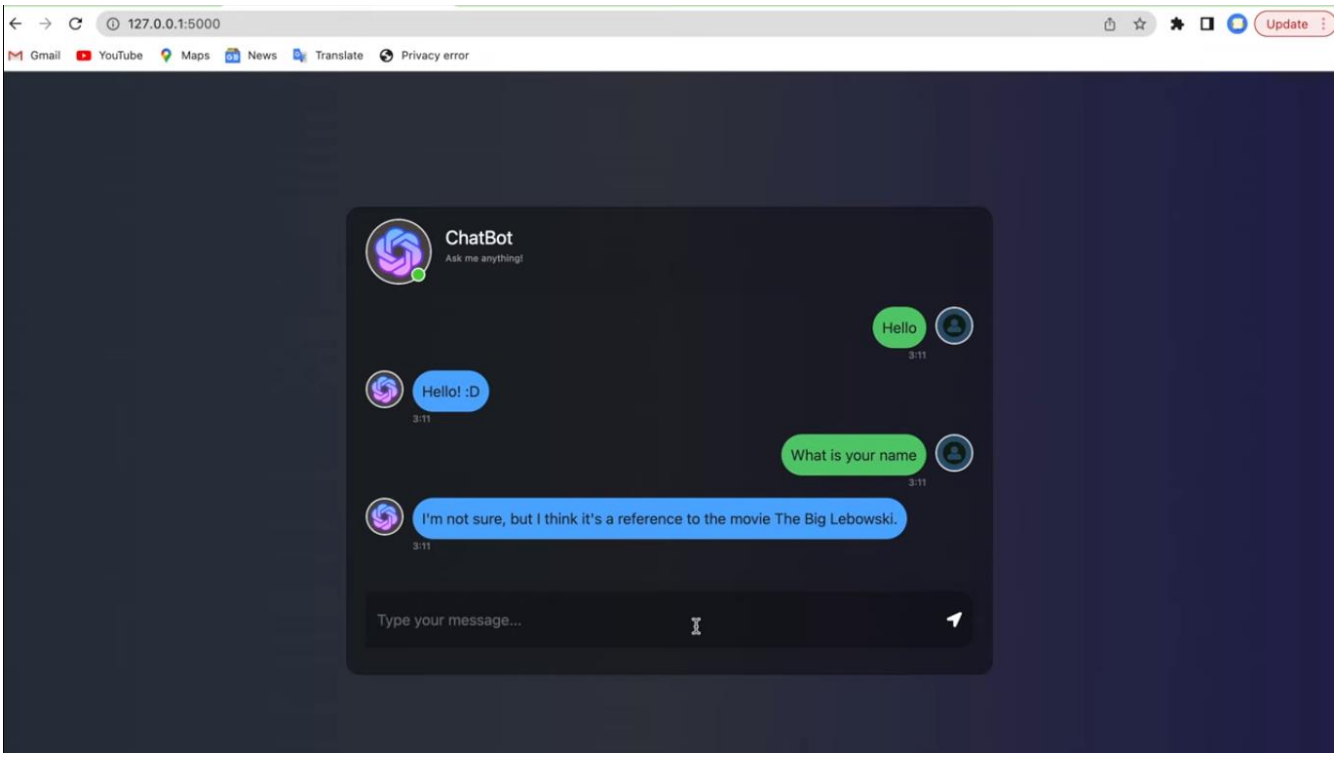
Run the Flask application:

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

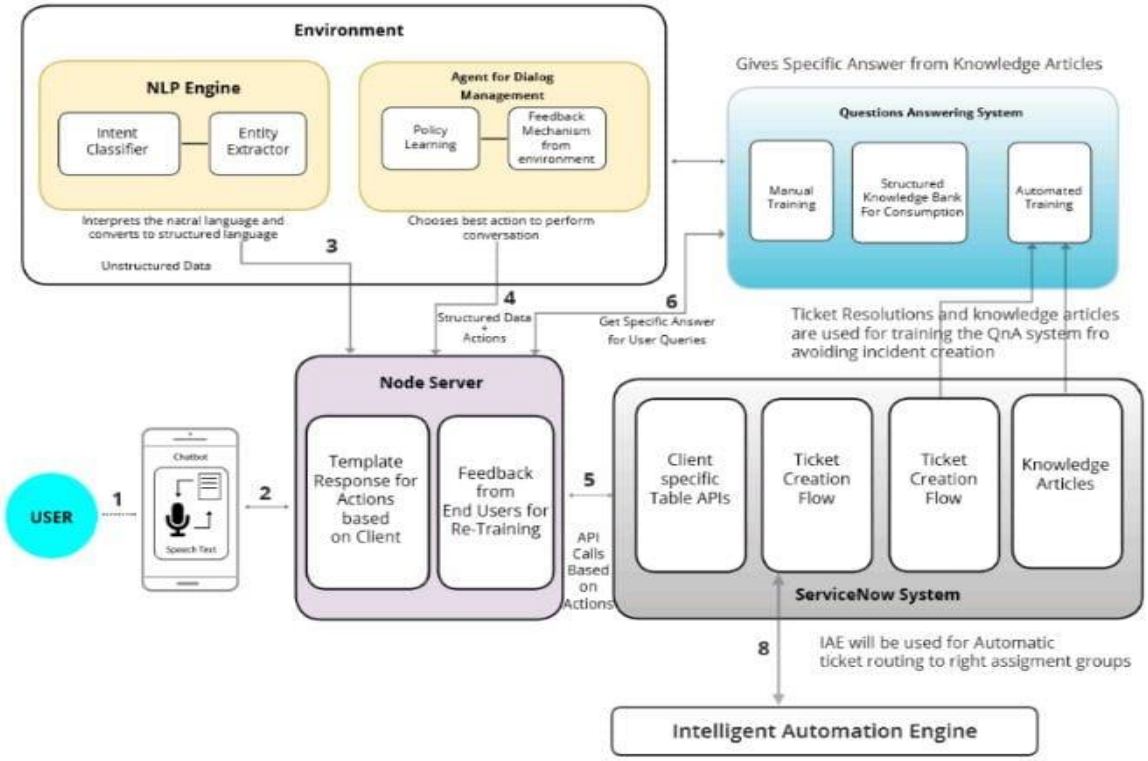
Interact with your chatbot:

- Visit http://127.0.0.1:5000 in your web browser to check if the app is running.
- To chat with the bot, you can send a POST request tohttp://127.0.0.1:5000 with a user_message parameter containing your input, and the bot will respond with the appropriate response from the text file.

Output:



Architectural Diagram in Chatbot:



Conclusion

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

THANK YOU..!