| **Date** | **30-09-2023** |
| --- | --- |
| **Team ID** |  |
| **Project Name** | **Create a chatbot in Python** |

## Create a chatbot in Python

**Problem Definition and Design Thinking**

**Introduction**

On creating a chatbot in Python. In today's fast-paced world, customers expect quick and efficient service from businesses. This is where chatbots come in. A chatbot is an AI-powered tool that can interact with customers in a conversational manner, providing them with instant support and assistance.

**Definition**

Chatbots are becoming increasingly popular among businesses due to their ability to improve customer satisfaction, reduce costs, and increase efficiency. By providing customers with 24/7 support, businesses can ensure that their customers are always taken care of, even outside of regular business hours. Additionally, chatbots can handle a large volume of queries simultaneously, reducing the need for human resources.

**Key Concerns**

1.**Define the Use Case and Objectives**: Start by specifying the chatbot's purpose and objectives. Understanding the use case will dictate the architecture and features your chatbot needs.

3. **Data Collection and Preparation:** If you opt for a machine learning approach, assembling a relevant dataset is critical. For rule-based bots, compile a knowledge base or rule set. Preprocessing the data involves tokenization, stop-word removal**,** and handling special characters.

**4. Building the Chatbot:**

* **Rule-Based:** Craft a set of rules and associated responses. Regular expressions or predefined patterns can aid in recognizing user input**.**
* **Machine Learning-Based:** Train a model with your dataset. This often entails designing a neural network architecture and fine-tuning it for your specific task.

**5. Natural Language Understanding (NLU**): Implement NLU components to extract intents and entities from user inputs. Tools like Rasa NLU or Dialogflow can simplify this process.

**6. Response Generation:** Based on the intents and entities identified in user messages, generate appropriate responses. This could involve a lookup from a predefined list or utilizing the output from your trained model.

**7. User Experience:** Prioritize a smooth and user-friendly experience. Ensure your chatbot offers clear responses, gracefully handles errors, and maintains a coherent conversation flow.

**Design Thinking**

Design thinking is a dynamic and iterative process that encourages empathy for users, problem definition, creative idea generation, prototypingtesting, and, the implementation of user-centered solutions. Each of these subtopics plays a crucial role in guiding designers through the process of creating meaningful and effective designs

**1. Empathize:**

* **Understand User Needs:** Gather insights into the users' pain points, preferences, and behaviors.
* **Identify User Goals**: Define the specific goals and objectives that users are trying to achieve.

**2. Define:**

* **Problem Statement**: Clearly articulate the problem or challenge that your design project aims to address.
* **User Needs and Goals**: Document the specific needs and goals of the users as a reference point for your design process.

**3. Ideate:**

* **Brainstorming:** Generate a wide range of creative ideas and potential solutions to the defined problem.
* **Storyboarding:** Create visual representations or narratives that illustrate how your ideas might unfold in practice.

**4. Prototype:**

* **Low-Fidelity Prototypes**: Develop rough, initial versions of your design concepts that can be quickly and inexpensively tested.
* Iterative Testing: Continuously refine and improve your prototypes through repeated testing and feedback.

**5. Test:**

* **User Testing**: Collect feedback from actual users to evaluate the usability and effectiveness of your prototypes.
* **Iterate:** Make iterative design improvements based on the feedback received during testing.

**6. Implement:**

* **Development**: Transform your refined design into a functional product, service, or solution.
* **Integration:** ensure that implemented Design fits into context

**Conclusion**

Creating a chatbot in Python is a multifaceted endeavor that involves various technical and user-centric considerations. Whether you approach it from a programmer's or engineer's perspective or aim for a more casual and creative tone, the key steps and concerns remain consistent

**Customer Prediction:**

Chatbots are becoming increasingly popular among businesses due to their ability to improve customer satisfaction, reduce costs, and increase efficiency. By providing customers with 24/7 support, businesses can ensure that their customers are always taken care of, even outside of regular business hours. Additionally, chatbots can handle a large volume of queries simultaneously, reducing the need for human resources.

**Define the Use Case and Objectives**:

Start by specifying the chatbot's purpose and objectives. Understanding the use case will dictate the architecture and features your chatbot needs.

**Data Collection and Preparation:**

If you opt for a machine learning approach, assembling a relevant dataset is critical. For rule-based bots, compile a knowledge base or rule set. Preprocessing the data involves tokenization, stop-word removal**,** and handling special characters.

**Building the Chatbot:**

Craft a set of rules and associated responses. Regular expressions or predefined patterns can aid in recognizing user input**.**

**Machine Learning-Based:**

Train a model with your dataset. This often entails designing a neural network architecture and fine-tuning it for your specific task.

**Natural Language Understanding (NLU**):

Implement NLU components to extract intents and entities from user inputs. Tools like Rasa NLU or Dialogflow can simplify this process.

**Response Generation:**

Based on the intents and entities identified in user messages, generate appropriate responses. This could involve a lookup from a predefined list or utilizing the output from your trained model.

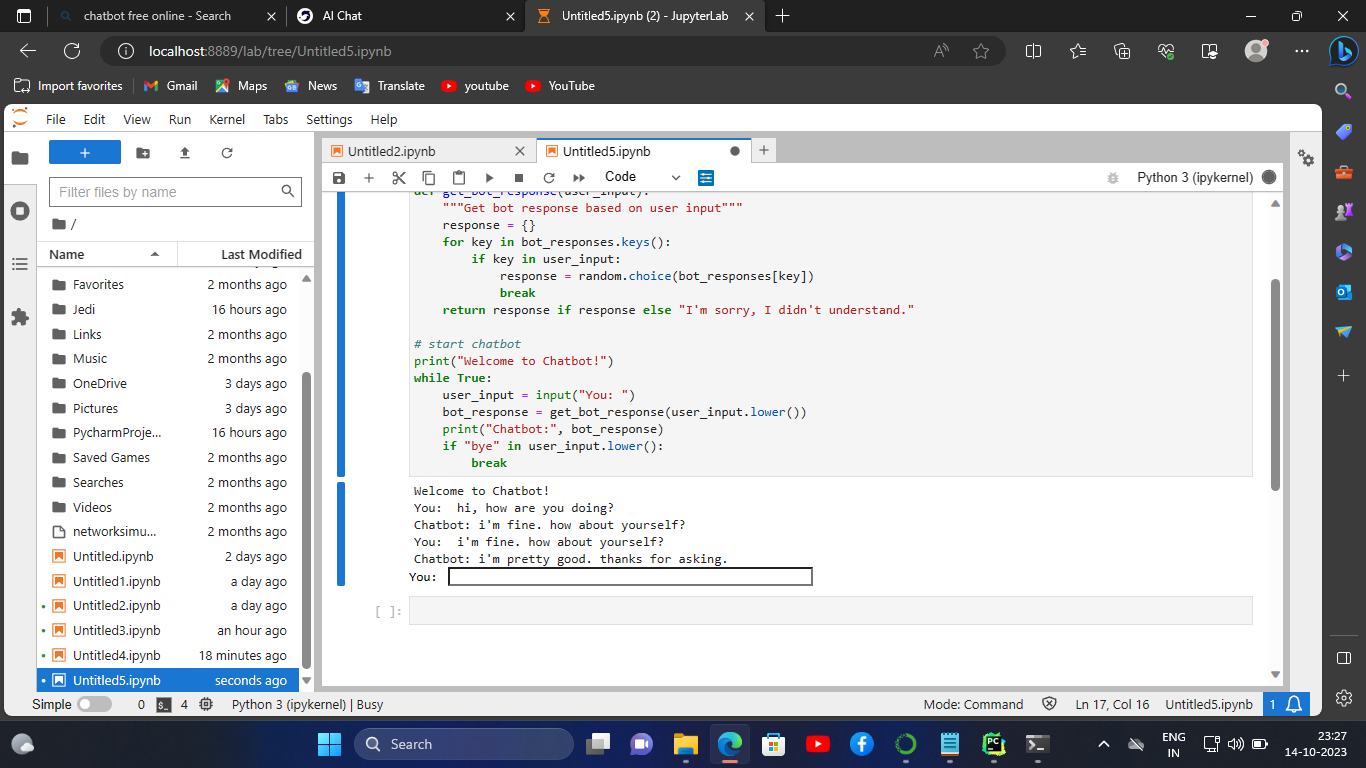
**Conclusion**

Creating a chatbot in Python is a multifaceted endeavor that involves various technical and user-centric considerations. Whether you approach it from a programmer's or engineer's perspective or aim for a more casual and creative tone, the key steps and concerns remain consistent.

**Development Part 1**

In this part you will begin building your project by loading and pre-processing the dataset.

Start building the Chatbot by preparing the environment and implementing basic user interactions. Install required libraries, like transformers for GPT-3 integration and flask for web app development.



In this program, we imported the dataset.Then, Get data from the dataset.So, the dataset is clear and check the punctuation. Then, The data is preprocessed.

**Development Part 2:**

* In this part you will continue building your project.
* Continue building the chatbot by integrating it into a web app using Flask.

* In this part you will continue building your project.
* Continue building the chatbot by integrating it into a web app using Flask.
* We training our Chatbot with help of the below dataset link.

[**https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**](https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot)

**STEP:1**

Creating the app.py file. Then,Run it.

Create the Flask app to serving in main application

**APP.PY CODE:**

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return 'Chatbot is running!'

@app.route('/chat', methods=['POST'])

def chat():

message = request.form['message']

# Process the user's message and generate a response from the chatbot

response = process\_message(message)

return response

def process\_message(message):

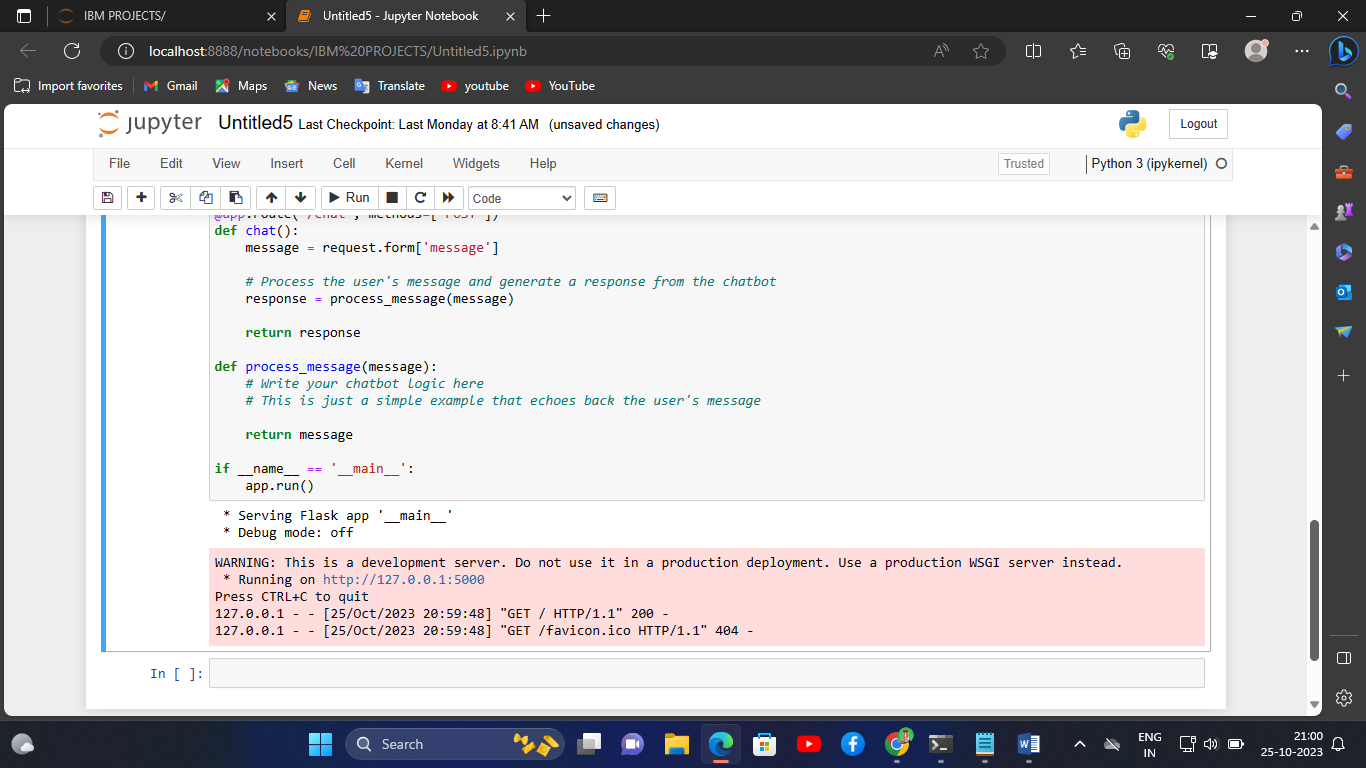
# Write your chatbot logic here

# This is just a simple example that echoes back the user's message

return message

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

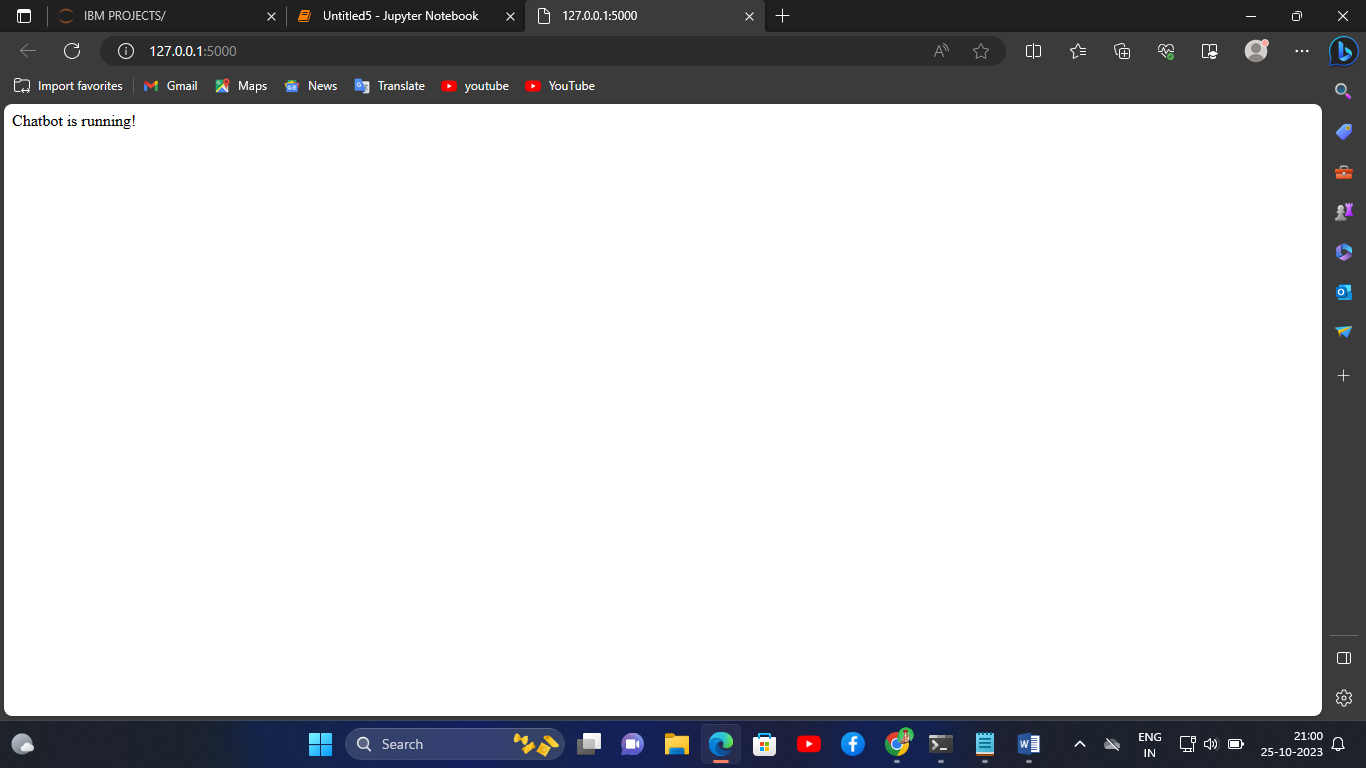


**STEP:2**

Run the app.py code in the jupyter notebook.

Then, the server is running on the website.

In that website, Chatbot is running! is shown on the web application.

****

**PROGRAM CODE:**

import random

# define bot responses

bot\_responses = {"hi, how are you doing?":["i'm fine. how about yourself?"],

"i'm fine. how about yourself?":["i'm pretty good. thanks for asking."],

"i'm pretty good. thanks for asking.":["no problem. so how have you been?"],

"no problem. so how have you been?":["i've been great. what about you?"],

"i've been great. what about you?":["i've been good. i'm in school right now."],

"i've been good. i'm in school right now.":["what school do you go to?"],

"what school do you go to?":["i go to pcc."]

}

# define function for getting bot response

def get\_bot\_response(user\_input):

"""Get bot response based on user input"""

response = {}

for key in bot\_responses.keys():

if key in user\_input:

response = random.choice(bot\_responses[key])

break

return response if response else "I'm sorry, I didn't understand."

# start chatbot

print("Welcome to Chatbot!")

while True:

user\_input = input("You: ")

bot\_response = get\_bot\_response(user\_input.lower())

print("Chatbot:", bot\_response)

if "bye" in user\_input.lower():

break

**OUTPUT:**

