

# CREATE A CHATBOT USING PYTHON:

---

College Code:5113

## **TEAM MEMBERS:**

D.Aakash (511321106001)

S.Chandrsekran (511321106005)

A.Dinesh (511321106010)

J.Yukendiran (511321106306)

## **Phase 5:Project Submission**

**Project Title:** Create A Chatbot Using Python

**Phase 5:** Project Documentation and Submission



# Introduction:

A chatbot is a computer program that can simulate conversation with human users, typically through written or spoken text. Chatbots are used in a wide variety of applications, including customer service, education, entertainment, and research.

Chatbots work by using a combination of artificial intelligence (AI) and natural language processing (NLP) techniques. AI allows chatbots to learn from data and improve their performance over time. NLP allows chatbots to understand and respond to human language.

There are two main types of chatbots: rule-based chatbots and AI-powered chatbots. Rule-based chatbots follow a set of pre-defined rules to respond to user input. AI-powered chatbots use machine learning to learn from data and generate more human-like responses.

Chatbots have many benefits. They can provide 24/7 customer service, answer questions quickly and accurately, and be used to collect feedback from users. Chatbots can also be used to automate tasks and provide personalized support.

Here are some examples of how chatbots are used today:

- Customer service: Chatbots can be used to answer customer questions, resolve issues, and provide support. For example, a chatbot can be used to help customers book an appointment, find information about a product, or troubleshoot a problem.
- Education: Chatbots can be used to provide personalized instruction, answer student questions, and give feedback.
- For example, a chatbot can be used to help students learn a new language, practice math problems, or prepare for a test.
- Entertainment: Chatbots can be used to create games, tell stories, and provide companionship. For example, a chatbot can be used to play a game

of trivia, tell a bedtime story, or simply chat with someone who is feeling lonely.

- Research: Chatbots can be used to collect data from users, conduct surveys, and interview participants. For example, a chatbot can be used to collect data about user preferences, conduct a survey about a new product, or interview participants about a particular topic.

As chatbot technology continues to develop, chatbots are likely to become even more widely used in a variety of applications.

Here are some of the potential benefits of using chatbots:

- Improved customer service: Chatbots can provide 24/7 customer service, which can help businesses to improve their customer satisfaction ratings.
- Reduced costs: Chatbots can automate tasks that are currently performed by human employees. This can help businesses to reduce their costs.
- Increased efficiency: Chatbots can help businesses to operate more efficiently by automating tasks and providing personalized support to customers.
- Improved decision-making: Chatbots can collect data from users and provide insights that can help businesses to make better decisions.

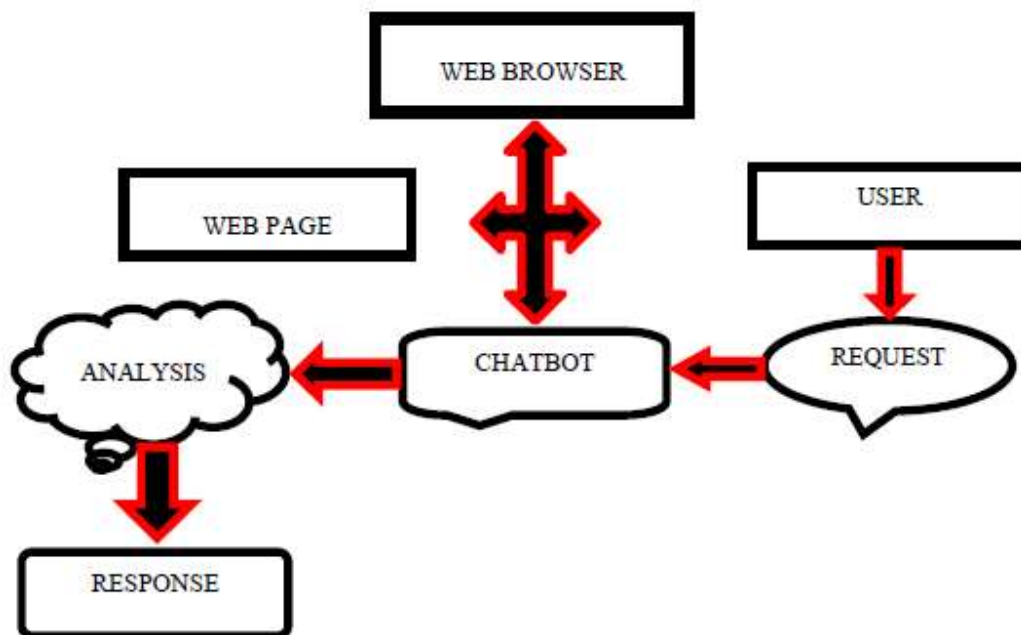
Overall, chatbots are a versatile tool that can be used to improve customer service, reduce costs, increase efficiency, and improve decision-making.

### **What is a Chatbot?**

A chatbot, short for "chat robot," is an artificial intelligence (AI) program designed to simulate human conversation. Chatbots can understand user input, process it, and generate relevant responses, making them valuable tools for automating customer interactions and providing instant assistance.

### **SYSTEM ARCHITECTURE:**

The main purpose of this chatbot is to respond to user queries without man power. User can work with the chatbot in any web browser. The chatbot receives the request sent by the user, analyses it and responds to the user in return. This analysis is done, using the machine learning algorithm. The queries are predefined with a particular tag for each set. This tag is nothing but a keyword that helps the chatbot to analyze the user request. After the analysis, the chatbot responds to the user with required reply. If the user request is not clear to the chatbot, the response will be a default message defined by the developer. Almost all the queries from the user will be clearly responded, only few are exceptional cases.



### Creating a Chatbot Using Python:

Creating a chatbot using Python involves several key steps:

1. **Design and Planning:** Define the chatbot's purpose, target audience, and desired functionalities. Determine whether it will be rule-based, AI-driven, or a combination of both.
2. **Data Collection:** Gather or create the necessary datasets and training data to teach the chatbot how to understand and respond to user input.
3. **Natural Language Processing (NLP):** Choose an NLP library or framework (e.g., NLTK, spaCy, Rasa) to process and understand natural language.

4. **Model Selection:** Decide on the chatbot's architecture. You can opt for rule-based chatbots, machine learning models, or pre-trained language models like GPT-3 or BERT.
5. **Training:** Train the chatbot on the collected data or a pre-existing dataset, teaching it how to recognize intents and generate relevant responses.
6. **Response Generation:** Implement logic to generate responses based on user input and the chatbot's learned knowledge.
7. **User Interface:** Develop a user interface for interacting with the chatbot, whether it's a web application, messaging platform integration, or custom app.
8. **Testing and Evaluation:** Test the chatbot for accuracy, performance, and user satisfaction. Collect user feedback and iteratively improve its responses.
9. **Deployment:** Deploy the chatbot in the desired environment, making it accessible to users.
10. **Maintenance and Updates:** Regularly update and maintain the chatbot to adapt to changing user needs and maintain relevance.

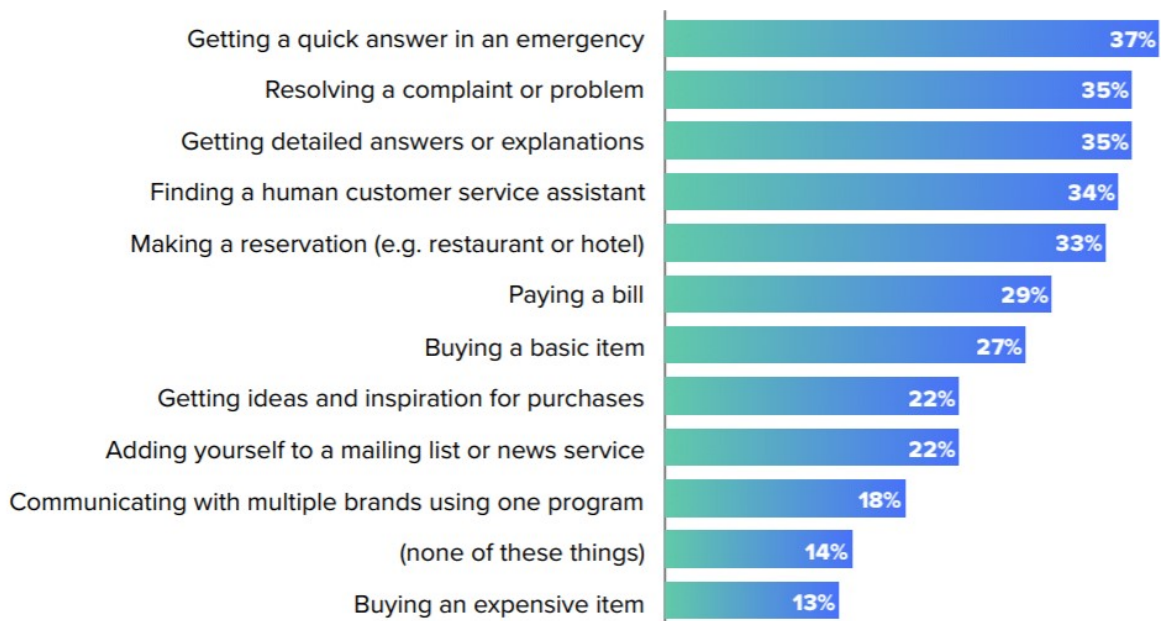
Python, with its rich ecosystem of libraries and frameworks, is a popular choice for chatbot development. It offers a wide range of tools for NLP, machine learning, and integration with various platforms, making it a versatile language for building effective and engaging chatbots.

Creating a chatbot is an exciting journey that combines linguistics, AI, and software development. Whether you're developing a customer support chatbot, a virtual assistant, or a fun conversational bot, Python provides the tools and resources needed to bring your chatbot project to life.

### **Purpose of Using Chatbot:**

Chatbots allow businesses to connect with customers in a personal way without the expense of human representatives. For example, many of the questions or issues customers have are common and easily answered. That's why companies create FAQs and troubleshooting guides.

*What do you predict you would use a chatbot for?*



### Understanding Chatbots:

Chatbots are designed to understand user input, process it, and generate appropriate responses, all in a conversational manner. They can be rule-based (using predefined rules and patterns) or powered by machine learning and AI techniques, such as natural language processing and deep learning.

### Key Steps in Creating a Chatbot with Python:

Creating a chatbot using Python involves a series of steps:

1. **Defining Purpose:** Clearly define the chatbot's purpose, including its intended use, target audience, and goals.

2. **Data Collection:** Gather or create datasets to train the chatbot. This may include conversation data and domain-specific knowledge.
3. **Natural Language Processing (NLP):** Select NLP libraries or frameworks (e.g., NLTK, spaCy, Rasa) to process and understand human language.
4. **Model Selection:** Decide on the chatbot's architecture, whether it's rule-based, machine learning-based, or a combination of both.
5. **Training:** Train the chatbot on the collected data, teaching it to recognize user intents and generate appropriate responses.
6. **Response Generation:** Implement logic for generating responses based on user input and the chatbot's learned knowledge.
7. **User Interface:** Create a user interface or integration method (e.g., web interface, messaging platform) for users to interact with the chatbot.
8. **Testing and Evaluation:** Thoroughly test the chatbot's accuracy, performance, and user satisfaction. Collect user feedback and iterate on improvements.
9. **Deployment:** Deploy the chatbot in the desired environment, making it accessible to users.
10. **Maintenance and Updates:** Regularly update and maintain the chatbot to adapt to changing user needs and maintain relevance.

Here's a list of tools and software commonly used in the process:

1. **Programming Language:**

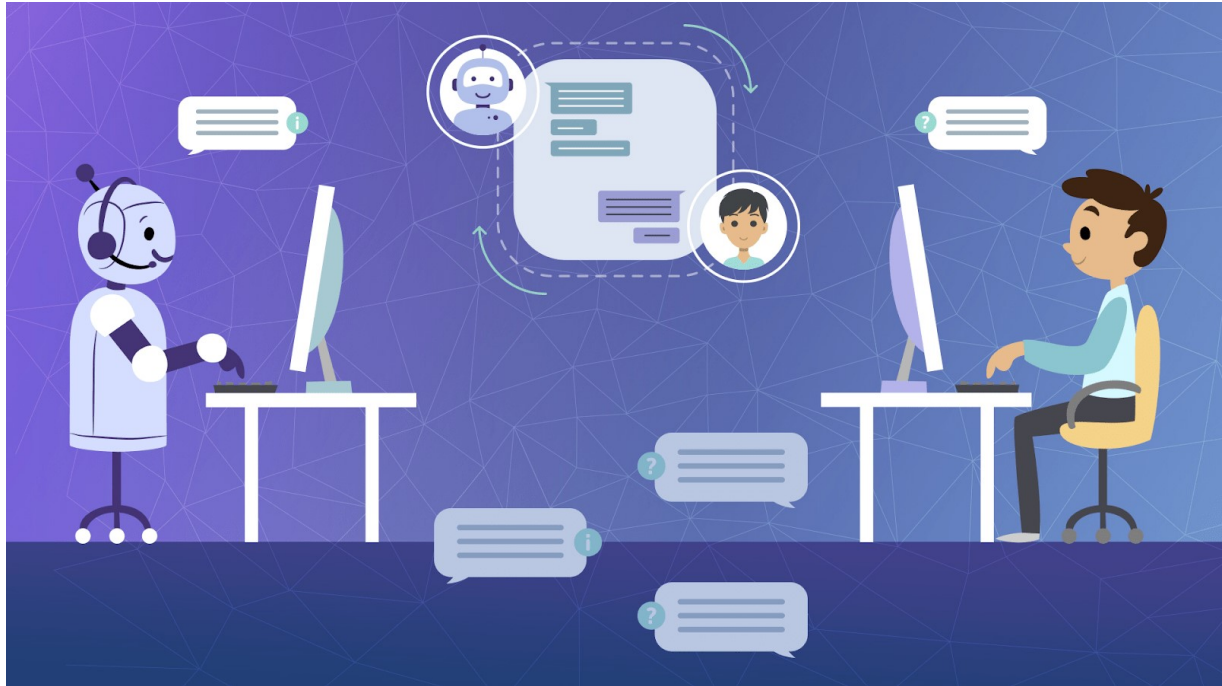
Python is the primary programming language used for developing chatbots. It's known for its simplicity, readability, and a vast ecosystem of libraries and frameworks.

2. **Natural Language Processing (NLP) Libraries:**

- **NLTK (Natural Language Toolkit):** NLTK is a powerful library for working with human language data. It offers tools for tokenization, stemming, parsing, and more.
  - **spaCy:** spaCy is a fast and efficient NLP library with pre-trained models for various languages. It's widely used for text processing and named entity recognition.
  - **TextBlob:** TextBlob is a simple NLP library that offers easy-to-use APIs for tasks like part-of-speech tagging, sentiment analysis, and translation.
3. **Rasa:** Rasa is an open-source framework for building conversational AI. It provides tools for natural language understanding, dialogue management, and chatbot training. Rasa allows you to create rule-based or machine learning-based chatbots.
  4. **Hugging Face Transformers:** Hugging Face offers pre-trained transformer models that can be fine-tuned for chatbot applications. Models like GPT-3 and BERT can be used for more advanced conversational agents.
  5. **Botpress:** Botpress is an open-source bot-building platform that provides a visual interface for building chatbots. It's based on Node.js but can be integrated with Python.
  6. **Dialogflow:** Dialogflow, a Google Cloud product, is a cloud-based chatbot development platform that offers natural language understanding and conversation management.
  7. **Microsoft Bot Framework:** Microsoft's Bot Framework allows developers to build chatbots for various platforms, including Microsoft Teams, Skype, and more. It supports Python through the Bot Framework SDK.
  8. **Facebook Wit.ai:** Wit.ai is a platform for building natural language processing models. It's owned by Facebook and can be used to create chatbots for Facebook Messenger.



9. **IBM Watson Assistant:** IBM Watson offers a chatbot development platform that allows you to create AI-powered chatbots. It supports Python for integration and customization.
10. **Jupyter Notebook:** Jupyter Notebook is commonly used for developing and testing chatbot components. It provides an interactive and visual environment for code development and experimentation.
11. **Version Control Systems:** Tools like Git and platforms like GitHub or GitLab are essential for collaborative chatbot development and code version control.
12. **IDEs (Integrated Development Environments):** IDEs such as PyCharm, Visual Studio Code, and JupyterLab provide code editing and debugging capabilities.
13. **Web Development Tools:** If you are creating chatbots with web interfaces, web development tools like HTML, CSS, and JavaScript, along with web frameworks like Flask or Django, might be required.
14. **Databases:** Databases like PostgreSQL, MySQL, or NoSQL databases may be used to store chatbot-related data, user profiles, and conversation histories.
15. **Cloud Services:** Cloud platforms like AWS, Google Cloud, and Azure provide cloud-based infrastructure for deploying and hosting chatbots.
16. **Testing and Evaluation Tools:** Various testing frameworks and tools for evaluating chatbot performance and user experience.



**Keywords:** Chabot, Chabot architecture, Artificial Intelligence, NLU (Natural language understanding), NLP (Natural language processing) Artificial Intelligence Markup Language (AIML), Latent Semantic Analysis (LSA)

## 1.Design Thinking For chatbot:

### 1. Empathize:

- Define your target audience: Understand who your chatbot users are and what their needs, preferences, and pain points are.
- Gather user feedback: Conduct surveys, interviews, or user testing to gain insights into how potential users expect the chatbot to assist them.

### 2. Define:

- Identify the problem or opportunity: Clearly define the purpose of your chatbot. What specific problem is it intended to solve, or what opportunity does it address?

- Create user personas: Develop user profiles that represent the characteristics and needs of your target users.
- Set clear objectives: Establish measurable goals for the chatbot's performance and user satisfaction.

### **3. Ideate:**

- Brainstorm chatbot features: Collaborate with your team to generate creative ideas for chatbot functionality and interaction design.
- Create user scenarios: Develop scenarios that illustrate how users will interact with the chatbot to achieve their goals.
- Explore potential use cases: Consider different ways your chatbot can provide value to users and the organization.

### **4. Prototype:**

- Design conversation flows: Create visual representations of how the chatbot's conversations will flow. Use tools like diagrams or wireframes.
- Develop a minimum viable product (MVP): Build a basic version of the chatbot to test its core functionalities and gather feedback.

### **5. Test:**

- Conduct usability testing: Involve potential users to interact with the chatbot prototype. Observe their interactions and collect feedback.
- Iterate on the design: Make improvements based on user feedback, ensuring the chatbot aligns better with user expectations.

### **6. Implement:**

- Choose the technology stack: Select the libraries, frameworks, and tools needed to develop the chatbot using Python.

- Develop the chatbot: Write the code for the chatbot's NLP capabilities, dialogue management, and response generation.

#### **7. Test Again:**

- Conduct rigorous testing: Perform thorough testing, including functional, performance, and security testing.
- Address issues and bugs: Fix any identified problems and ensure the chatbot's reliability and responsiveness.

#### **8. Launch:**

- Deploy the chatbot: Make the chatbot accessible to users through the chosen platform, such as a website, messaging app, or voice interface.
- Communicate the launch: Inform users about the availability of the chatbot and provide clear instructions on how to use it.

#### **9. Gather Feedback:**

- Continue to gather user feedback: Encourage users to provide feedback on their experiences with the chatbot.
- Monitor performance: Use analytics and user data to assess the chatbot's effectiveness and identify areas for improvement.

#### **10. Iterate and Improve:**

- Continuously refine the chatbot: Based on user feedback and performance data, make iterative improvements to the chatbot's capabilities and user experience.
- Expand functionality: Consider adding new features and integrations as the chatbot matures.

## **2.Data Innovation for chatbot:**

### **1.Chatbot's Purpose and Goals:**

- Determine the specific problem or task your chatbot will address.
- Identify the goals and objectives you want to achieve with the chatbot.

### **2. Collect and Prepare Data:**

- Gather relevant data for training and fine-tuning your chatbot.
- Preprocess and clean the data to make it suitable for training the model.

### **3. Choose a Pre-trained Language Model:**

- Select a pre-trained language model like GPT-3 or other alternatives (e.g., GPT-4, BERT) based on your project requirements and budget.

### **4. Set Up the Development Environment:**

- Install the necessary Python libraries (e.g., **openai**, **numpy**, **tensorflow**) and dependencies for working with the chosen language model.

### **5. API Integration:**

- Obtain API access credentials for the selected language model.
- Configure your Python code to interact with the model's API using the provided credentials.

### **6. Design Conversational Flows:**

- Plan the conversation flows and user interactions.
- Define how the chatbot should respond to various user inputs and scenarios.

### **7. Implement Natural Language Processing (NLP) Techniques:**

- Use NLP libraries like spaCy or NLTK for tasks such as tokenization, entity recognition, and sentiment analysis.

- Apply techniques like dialogue management to maintain context in multi-turn conversations.

## **8. Train and Fine-Tune the Model:**

- If required, fine-tune the pre-trained model using your specific data and objectives.
- Experiment with hyperparameters and training configurations to optimize performance.

## **9. Test and Evaluate:**

- Test the chatbot extensively to ensure it provides accurate and contextually relevant responses.
- Implement automated tests and collect user feedback for improvement.

**10. Implement Advanced Features:** - Consider adding advanced features like: - Multilingual support for global users. - Voice input/output for speech-based interactions. - Integration with external APIs and databases for dynamic data retrieval. - Personalization for customizing responses based on user preferences.

**11. Implement User Authentication and Security:** - If the chatbot handles sensitive information, implement user authentication and ensure data security.

**12. Deploy the Chatbot:** - Choose a hosting platform or cloud service to deploy your chatbot. - Set up scaling mechanisms to handle increased traffic.

**13. Monitor and Maintain:** - Continuously monitor the chatbot's performance and user interactions. - Update the model and responses as needed to adapt to changing requirements.

**14. User Feedback Loop:** - Implement a feedback mechanism for users to report issues and provide suggestions. - Use user feedback to iterate and improve the chatbot's performance.

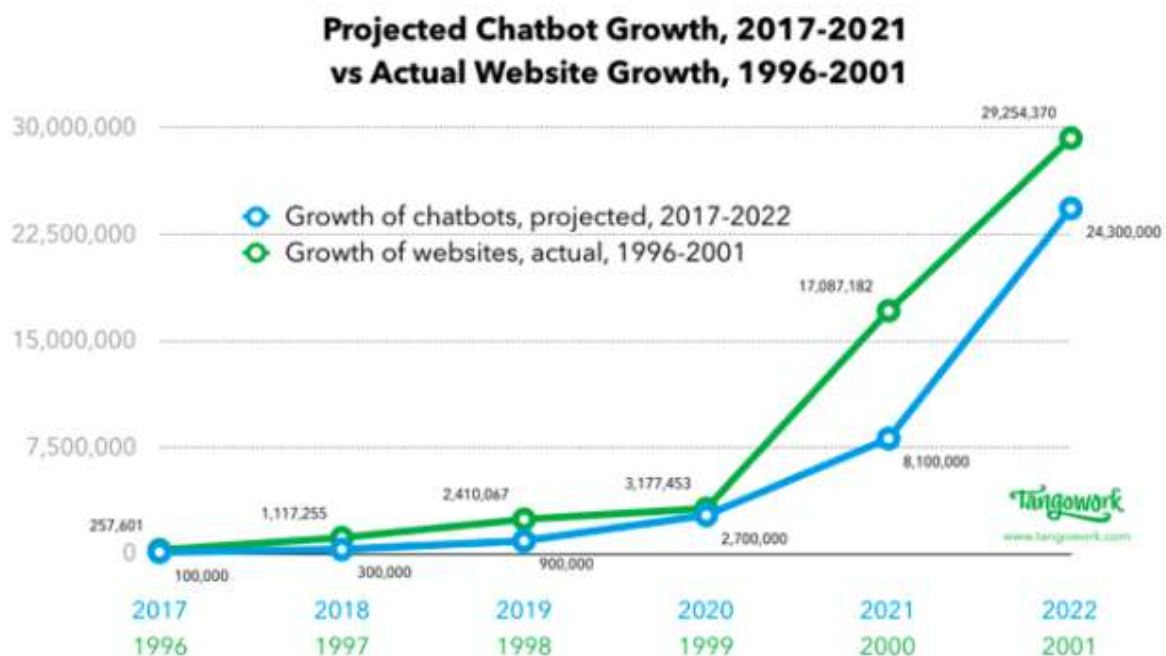
**15. Compliance and Ethical Considerations:** - Ensure that your chatbot complies with legal and ethical standards, including data privacy regulations.

**16. Marketing and Promotion:** - Develop a strategy for promoting your chatbot to potential users. - Consider social media marketing, partnerships, or app store listings.

**17. Continuous Innovation:** - Stay updated with the latest advancements in NLP and AI. - Continuously innovate and enhance your chatbot's capabilities to remain competitive and relevant.

Building an innovative chatbot using advanced techniques like GPT-3 requires a combination of technical expertise, creativity, and a user-centric approach. Regularly gather user feedback and iterate on your chatbot to make it increasingly valuable to your target audience.

## GROWTH OF CHATBOT:



## REQUIRMENTS:

1. Python 3.x
2. OpenAI GPT-3 API access
3. **openai** library (install with **pip install openai**)
4. **numpy** library (install with **pip install numpy**)

**Step 1:** Sign up for the OpenAI GPT-3 API and obtain your API key.

**Step 2:** Here's a more advanced example of a chatbot script:

### **PROGRAM:**

```
import openai

import numpy as np

# Replace with your GPT-3 API key

api_key = "YOUR_API_KEY"

# Initialize the OpenAI API client

openai.api_key = api_key

def generate_response(prompt, max_tokens=50, temperature=0.7):

    response = openai.Completion.create(

        engine="davinci", # You can use other engines like "curie" or "babbage" as well.

        prompt=prompt,

        max_tokens=max_tokens,

        temperature=temperature

    )

    return response.choices[0].text

# Initialize conversation

conversation_history = []

print("Chatbot: Hi, how can I assist you today? (Type 'exit' to end)")
```



```
while True:

    user_input = input("You: ")

    if user_input.lower() == "exit":

        print("Chatbot: Goodbye!")

        break

    # Add the user's message to the conversation history

    conversation_history.append(f"You: {user_input}")

    # Generate a response from the chatbot

    chatbot_message = generate_response("\n".join(conversation_history))

    # Display the chatbot's response

    print(f"Chatbot: {chatbot_message}")

    # Add the chatbot's message to the conversation history

    conversation_history.append(f"Chatbot: {chatbot_message}")
```

In this script:

- We initialize the OpenAI API client with your API key.
- The **generate\_response** function sends a prompt to GPT-3 and retrieves a response.
- We maintain a conversation history to keep track of the chat's context.
- The chatbot continually interacts with the user, extending the conversation history as needed.

### Step 3: Run the Chatbot

Save the script to a Python file (e.g., **chatbot.py**) and run it. You can have a conversation with your chatbot by entering text messages.

This is a basic example, and you can further enhance it by adding features like context management, sentiment analysis, or custom responses for specific user inputs. You can also consider using other NLP libraries like spaCy or NLTK to process and analyze user input for more sophisticated interactions.

### 3.BUILD LOADING AND PREPROCESSING THE DATASET:

#### **Preprocessing the Dataset:**

The quality of a chatbot heavily relies on the dataset it's trained on. Preprocessing the dataset is a critical step to ensure the chatbot can understand and respond appropriately to user inputs. Here's an overview of dataset preprocessing in chatbot development:

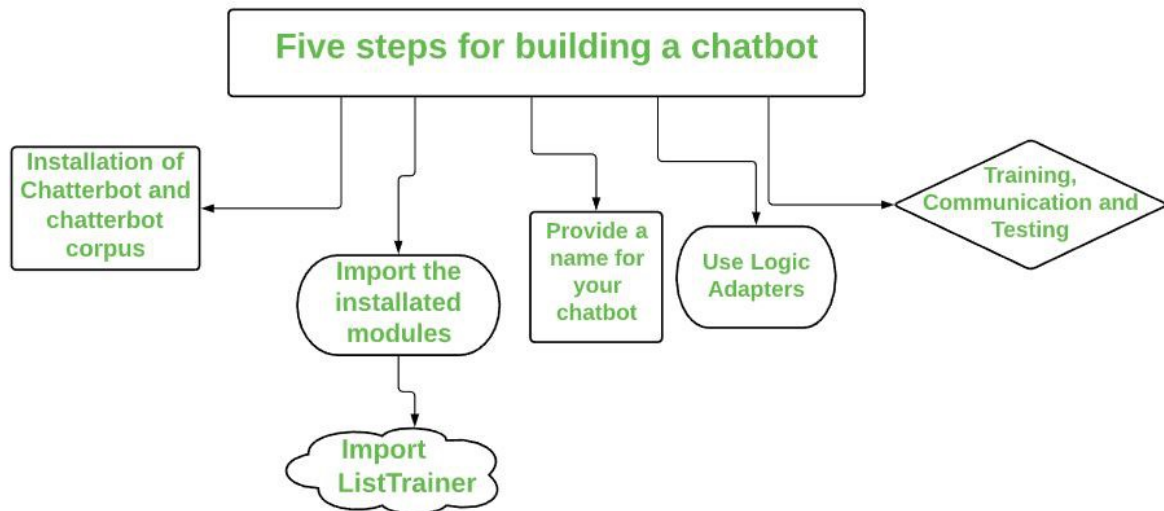
1. **Data Collection:** Begin by collecting a dataset that consists of questions or user inputs and corresponding responses. This dataset can be gathered from various sources, such as FAQs, chat logs, or custom-created dialogues.
2. **Data Cleaning:** Raw text data often contains noise, special characters, or unnecessary formatting. Clean the data by removing unwanted elements, such as HTML tags, non-alphanumeric characters, and irrelevant metadata.
3. **Text Normalization:** Normalize the text by converting it to lowercase. This ensures that the chatbot's responses are consistent and independent of the user's input casing.
4. **Tokenization:** Tokenization is the process of splitting text into individual words or tokens. This step allows the chatbot to analyze and understand the text on a word-by-word basis.
5. **Stopword Removal:** Stopwords are common words like "and," "the," "is," etc., that often don't carry significant meaning. Removing stopwords can help improve the efficiency and accuracy of the chatbot's responses.

6. **Lemmatization or Stemming:** Reducing words to their root form helps the chatbot recognize different word forms as equivalent. You can use lemmatization (reducing words to their base form) or stemming (removing prefixes and suffixes) for this purpose.
7. **Handling Synonyms and Variations:** Addressing synonyms and variations in the dataset is crucial. You can create mappings or equivalence tables to link synonyms or different ways of phrasing the same question to a common response.
8. **Data Structure:** Organize the preprocessed data into a suitable format, such as a JSON file, CSV, or a database, making it accessible for training and future use.

### **Requirment Packages:**

- Numpy
- Pandas
- Keras
- Json
- Pickels
- Warnings
- Tensorflow
- Keras
- Nltk
- Matplotlib.pyplot

## 5 Steps For Building A Chatbot:



Necessary step to follow:

**Step 1: Define Your Objective and Collect Data:** Before you start coding, define the purpose and objective of your chatbot. Then, gather a dataset containing questions and their corresponding responses. For simplicity, let's use a JSON file for the dataset.

```
import json
```

```
# Load your dataset from a JSON file (questions and responses).
```

```
with open("dataset.json", "r") as file:
```

```
    data = json.load(file)
```

**Step 2: Preprocess the Data**

Preprocessing the data is crucial to make it suitable for training and usage by the chatbot. You can tokenize and clean the text, remove punctuation, and convert it to lowercase.

```
import string
```

```
import nltk

nltk.download('punkt')

# Tokenize and clean the text

def preprocess_text(text):

    text = text.lower()

    text = text.translate(str.maketrans("", "", string.punctuation))

    return text

# Preprocess the dataset

for pair in data:

    pair['question'] = preprocess_text(pair['question'])

    pair['response'] = preprocess_text(pair['response'])
```

### **Step 3: Build a Simple Rule-Based Chatbot**

For simplicity, let's create a basic rule-based chatbot that looks for exact matches in the dataset and responds accordingly.

```
def simple_chatbot(input_text):

    input_text = preprocess_text(input_text)

    for pair in data:

        if pair['question'] == input_text:

            return pair['response']

    return "I'm sorry, I don't understand your question."

# Testing the chatbot

while True:
```

```

user_input = input("You: ")

if user_input.lower() == 'exit':

    break

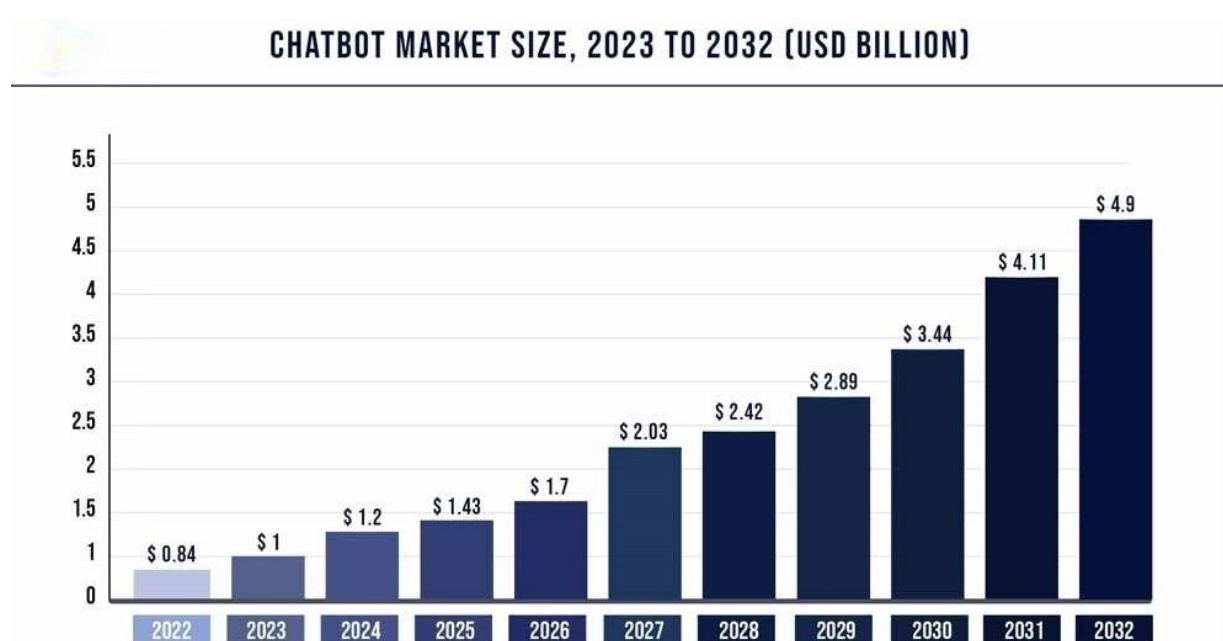
response = simple_chatbot(user_input)

print("Chatbot:", response

```

## Devlopment of Chatbot:

The chatbot market is estimated to grow from USD 5.27 billion in 2023 and is likely to grow at a CAGR of 23.28% during 2023-2028 to reach USD 14.95 billion by 2028.



## Program:

In [1]:

```

import nltk

nltk.download('punkt')#Sentence tokenizer

```

[nltk\_data] Downloading package punkt to /usr/share/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

Out[1]:

True

In [2]:

```
import nltk
```

```
from nltk.stem import WordNetLemmatizer
```

```
lemmatizer = WordNetLemmatizer()
```

```
import json
```

```
import pickle
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

Out[2]:True

In [3]:

```
import numpy as np
```

```
import tensorflow as tf
```

```
from keras.models import Sequential
```

```
from keras.layers import Dense, Activation, Dropout
```

```
from tensorflow.keras.optimizers import SGD
```

```
import random
```

Out[3]:True

**Preprocessing:**

In [4]:

```
words=[]
```

```
classes = []
```

```
documents = []
```

```
ignore_words = ['?', '!']
```

```
data_file = open('/kaggle/input/chatbot-dataset/intents.json').read() # read json file
```

```
intents = json.loads(data_file) # load json file
```

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Based on the requirements we need to apply various operations to preprocess the data.

- Tokenizing is the most basic and first thing you can do on text data.
- Tokenizing is the process of breaking the whole text into small parts like words.
- Here we iterate through the patterns and tokenize the sentence using `nltk.word_tokenize()` function and append each word in the words list. We also create a list of classes for our tags.

In [5]:

```
for intent in intents['intents']:
```

```
    for pattern in intent['patterns']:
```

```
        #tokenize each word
```

```
        w = nltk.word_tokenize(pattern)
```

```
        words.extend(w)# add each elements into list
```

```
        #combination between patterns and intents
```



```
documents.append((w, intent['tag']))#add single element into end of list
```

```
# add to tag in our classes list
```

```
if intent['tag'] not in classes:
```

```
    classes.append(intent['tag'])
```

In [6]:

```
nlTK.download('wordnet') #lexical database for the English language
```

```
[nlTK_data] Downloading package wordnet to /usr/share/nltk_data...
```

```
[nlTK_data] Package wordnet is already up-to-date!
```

Out[6]:

True

In [7]:

```
nlTK.download('omw-1.4')
```

```
[nlTK_data] Downloading package omw-1.4 to /usr/share/nltk_data...
```

Out[7]:

True

Now we will lemmatize each word and remove duplicate words from the list.

- Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

In [8]:

```
# lemmatize, lower each word and remove duplicates
```

```
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in  
ignore_words]
```

```

words = sorted(list(set(words)))

# sort classes

classes = sorted(list(set(classes)))

# documents = combination between patterns and intents

print (len(documents), "documents\n", documents, "\n")

# classes = intents[tag]

print (len(classes), "classes\n", classes, "\n")

# words = all words, vocabulary

print (len(words), "unique lemmatized words\n", words, "\n")

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(classes,open('classes.pkl','wb'))

```

405 documents

```

([['Hi'], 'greeting'], [['How', 'are', 'you', '?'], 'greeting'], [['Is', 'anyone', 'there', '?'],
'greeting'], [['Hello'], 'greeting'], [['Good', 'day'], 'greeting'], [['What', '"s", 'up'],
'greeting'], [['how', 'are', 'ya'], 'greeting'], [['hey'], 'greeting'], [['whatsup'],
'greeting'], [['?', '?', '?', '?', '?', '?', '?', '?'], 'greeting'], [['cya'], 'goodbye'], [['see',
'you'], 'goodbye'], [['bye', 'bye'], 'goodbye'], [['See', 'you', 'later'], 'goodbye'],
[['Goodbye'], 'goodbye'], [['I', 'am', 'Leaving'], 'goodbye'], [['Bye'], 'goodbye'],
[['Have', 'a', 'Good', 'day'], 'goodbye'], [['talk', 'to', 'you', 'later'], 'goodbye'],
[['ttyl'], 'goodbye'], [['i', 'got', 'to', 'go'], 'goodbye'], [['gtg'], 'goodbye'], [['what',
'is', 'the', 'name', 'of', 'your', 'developers'], 'creator'], [['what', 'is', 'the', 'name',
'of', 'your', 'creators'], 'creator'], [['what', 'is', 'the', 'name', 'of', 'the', 'developers'],
'creator'], [['what', 'is', 'the', 'name', 'of', 'the', 'creators'], 'creator'], [['who',
'created', 'you'], 'creator'], [['your', 'developers'], 'creator'], [['your', 'creators'],
'creator'], [['who', 'are', 'your', 'developers'], 'creator'], [['developers'], 'creator'],
[['you', 'are', 'made', 'by'], 'creator'], [['you', 'are', 'made', 'by', 'whom'], 'creator'],

```

(['who', 'created', 'you'], 'creator'), (['who', 'create', 'you'], 'creator'), (['creators'], 'creator'), (['who', 'made', 'you'], 'creator'), (['who', 'designed', 'you'], 'creator'), (['name'], 'name'), (['your', 'name'], 'name'), (['do', 'you', 'have', 'a', 'name'], 'name'), (['what', 'are', 'you', 'called'], 'name'), (['what', 'is', 'your', 'name'], 'name'), (['what', 'should', 'I', 'call', 'you'], 'name'), (['whats', 'your', 'name', '?'], 'name'), (['what', 'are', 'you'], 'name'), (['who', 'are', 'you'], 'name'), (['who', 'is', 'this'], 'name'), (['what', 'am', 'i', 'chatting', 'to'], 'name'), (['who', 'am', 'i', 'taking', 'to'], 'name'), (['what', 'are', 'you'], 'name'), (['timing', 'of', 'college'], 'hours'), (['what', 'is', 'college', 'timing'], 'hours'), (['working', 'days'], 'hours'), (['when', 'are', 'you', 'guys', 'open'], 'hours'), (['what', 'are', 'your', 'hours'], 'hours'), (['hours', 'of', 'operation'], 'hours'), (['when', 'is', 'the', 'college', 'open'], 'hours'), (['college', 'timing'], 'hours'), (['what', 'about', 'college', 'timing'], 'hours'), (['is', 'college', 'open', 'on', 'saturday'], 'hours'), (['tell', 'something', 'about', 'college', 'timing'], 'hours'), (['what', 'is', 'the', 'college', 'hours'], 'hours'), (['when', 'should', 'i', 'come', 'to', 'college'], 'hours'), (['when', 'should', 'i', 'attend', 'college'], 'hours'), (['what', 'is', 'my', 'college', 'time'], 'hours'), (['college', 'timing'], 'hours'), (['timing', 'college'], 'hours'), (['more', 'info'], 'number'), (['contact', 'info'], 'number'), (['how', 'to', 'contact', 'college'], 'number'), (['college', 'telephone', 'number'], 'number'), (['college', 'number'], 'number'), (['What', 'is', 'your', 'contact', 'no'], 'number'), (['Contact', 'number', '?'], 'number'), (['how', 'to', 'call', 'you'], 'number'), (['College', 'phone', 'no', '?'], 'number'), (['how', 'can', 'i', 'contact', 'you'], 'number'), (['Can', 'i', 'get', 'your', 'phone', 'number'], 'number'), (['how', 'can', 'i', 'call', 'you'], 'number'), (['phone', 'number'], 'number'), (['phone', 'no'], 'number'), (['call'], 'number'), (['list', 'of', 'courses'], 'course'), (['list', 'of', 'courses', 'offered'], 'course'), (['list', 'of', 'courses', 'offered', 'in'], 'course'), (['what', 'are', 'the', 'courses', 'offered', 'in', 'your', 'college', '?'], 'course'), (['courses', '?'], 'course'), (['courses', 'offered'], 'course'), (['courses', 'offered', 'in', '(', 'your', 'univrsity', '(', 'UNI', ')', 'name', ')'], 'course'), (['courses', 'you', 'offer'], 'course'), (['branches', '?'], 'course'), (['courses', 'available', 'at', 'UNI', '?'], 'course'), (['branches', 'available', 'at', 'your', 'college', '?'], 'course'), (['what', 'are', 'the', 'courses', 'in', 'UNI', '?'], 'course'), (['what', 'are', 'branches', 'in', 'UNI', '?'], 'course'), (['what', 'are', 'courses', 'in', 'UNI', '?'], 'course'), (['branches', 'available', 'in', 'UNI', '?'], 'course'), (['can', 'you', 'tell', 'me', 'the', 'courses', 'available', 'in',

'UNI', '?'], 'course'), ([ 'can', 'you', 'tell', 'me', 'the', 'branches', 'available', 'in', 'UNI', '?'], 'course'), ([ 'computer', 'engineering', '?'], 'course'), ([ 'computer'], 'course'), ([ 'Computer', 'engineering', '?'], 'course'), ([ 'it'], 'course'), ([ 'IT'], 'course'), ([ 'Information', 'Technology'], 'course'), ([ 'AI/ML'], 'course'), ([ 'Mechanical', 'engineering'], 'course'), ([ 'Chemical', 'engineering'], 'course'), ([ 'Civil', 'engineering'], 'course'), ([ 'information', 'about', 'fee'], 'fees'), ([ 'information', 'on', 'fee'], 'fees'), ([ 'tell', 'me', 'the', 'fee'], 'fees'), ([ 'college', 'fee'], 'fees'), ([ 'fee', 'per', 'semester'], 'fees'), ([ 'what', 'is', 'the', 'fee', 'of', 'each', 'semester'], 'fees'), ([ 'what', 'is', 'the', 'fees', 'of', 'each', 'year'], 'fees'), ([ 'what', 'is', 'fee'], 'fees'), ([ 'what', 'is', 'the', 'fees'], 'fees'), ([ 'how', 'much', 'is', 'the', 'fees'], 'fees'), ([ 'fees', 'for', 'first', 'year'], 'fees'), ([ 'fees'], 'fees'), ([ 'about', 'the', 'fees'], 'fees'), ([ 'tell', 'me', 'something', 'about', 'the', 'fees'], 'fees'), ([ 'What', 'is', 'the', 'fees', 'of', 'hostel'], 'fees'), ([ 'how', 'much', 'is', 'the', 'fees'], 'fees'), ([ 'hostel', 'fees'], 'fees'), ([ 'fees', 'for', 'AC', 'room'], 'fees'), ([ 'fees', 'for', 'non-AC', 'room'], 'fees'), ([ 'fees', 'for', 'Ac', 'room', 'for', 'girls'], 'fees'), ([ 'fees', 'for', 'non-Ac', 'room', 'for', 'girls'], 'fees'), ([ 'fees', 'for', 'Ac', 'room', 'for', 'boys'], 'fees'), ([ 'fees', 'for', 'non-Ac', 'room', 'for', 'boys'], 'fees'), ([ 'where', 'is', 'the', 'college', 'located'], 'location'), ([ 'college', 'is', 'located', 'at'], 'location'), ([ 'where', 'is', 'college'], 'location'), ([ 'where', 'is', 'college', 'located'], 'location'), ([ 'address', 'of', 'college'], 'location'), ([ 'how', 'to', 'reach', 'college'], 'location'), ([ 'college', 'location'], 'location'), ([ 'college', 'address'], 'location'), ([ 'wheres', 'the', 'college'], 'location'), ([ 'how', 'can', 'I', 'reach', 'college'], 'location'), ([ 'whats', 'is', 'the', 'college', 'address'], 'location'), ([ 'what', 'is', 'the', 'address', 'of', 'college'], 'location'), ([ 'address'], 'location'), ([ 'location'], 'location'), ([ 'hostel', 'facility'], 'hostel'), ([ 'hostel', 'servive'], 'hostel'), ([ 'hostel', 'location'], 'hostel'), ([ 'hostel', 'address'], 'hostel'), ([ 'hostel', 'facilities'], 'hostel'), ([ 'hostel', 'fees'], 'hostel'), ([ 'Does', 'college', 'provide', 'hostel'], 'hostel'), ([ 'Is', 'there', 'any', 'hostel'], 'hostel'), ([ 'Where', 'is', 'hostel'], 'hostel'), ([ 'do', 'you', 'have', 'hostel'], 'hostel'), ([ 'do', 'you', 'guys', 'have', 'hostel'], 'hostel'), ([ 'hostel'], 'hostel'), ([ 'hostel', 'capacity'], 'hostel'), ([ 'what', 'is', 'the', 'hostel', 'fee'], 'hostel'), ([ 'how', 'to', 'get', 'in', 'hostel'], 'hostel'), ([ 'what', 'is', 'the', 'hostel', 'address'], 'hostel'), ([ 'how', 'far', 'is', 'hostel', 'from', 'college'], 'hostel'), ([ 'hostel', 'college', 'distance'], 'hostel'), ([ 'where', 'is', 'the', 'hostel'], 'hostel'), ([ 'how', 'big', 'is', 'the', 'hostel'], 'hostel'), ([ 'distance', 'between', 'college', 'and', 'hostel'], 'hostel'),

(['distance', 'between', 'hostel', 'and', 'college'], 'hostel'), (['events', 'organised'], 'event'), (['list', 'of', 'events'], 'event'), (['list', 'of', 'events', 'organised', 'in', 'college'], 'event'), (['list', 'of', 'events', 'conducted', 'in', 'college'], 'event'), (['What', 'events', 'are', 'conducted', 'in', 'college'], 'event'), (['Are', 'there', 'any', 'event', 'held', 'at', 'college'], 'event'), (['Events', '?'], 'event'), (['functions'], 'event'), (['what', 'are', 'the', 'events'], 'event'), (['tell', 'me', 'about', 'events'], 'event'), (['what', 'about', 'events'], 'event'), (['document', 'to', 'bring'], 'document'), (['documents', 'needed', 'for', 'admission'], 'document'), (['documents', 'needed', 'at', 'the', 'time', 'of', 'admission'], 'document'), (['documents', 'needed', 'during', 'admission'], 'document'), (['documents', 'required', 'for', 'admission'], 'document'), (['documents', 'required', 'at', 'the', 'time', 'of', 'admission'], 'document'), (['documents', 'required', 'during', 'admission'], 'document'), (['What', 'document', 'are', 'required', 'for', 'admission'], 'document'), (['Which', 'document', 'to', 'bring', 'for', 'admission'], 'document'), (['documents'], 'document'), (['what', 'documents', 'do', 'i', 'need'], 'document'), (['what', 'documents', 'do', 'i', 'need', 'for', 'admission'], 'document'), (['documents', 'needed'], 'document'), (['size', 'of', 'campus'], 'floors'), (['building', 'size'], 'floors'), (['How', 'many', 'floors', 'does', 'college', 'have'], 'floors'), (['floors', 'in', 'college'], 'floors'), (['floors', 'in', 'college'], 'floors'), (['how', 'tall', 'is', 'UNI', 's', 'College', 'of', 'Engineering', 'college', 'building'], 'floors'), (['floors'], 'floors'), (['Syllabus', 'for', 'IT'], 'syllabus'), (['what', 'is', 'the', 'Information', 'Technology', 'syllabus'], 'syllabus'), (['syllabus'], 'syllabus'), (['timetable'], 'syllabus'), (['what', 'is', 'IT', 'syllabus'], 'syllabus'), (['syllabus'], 'syllabus'), (['What', 'is', 'next', 'lecture'], 'syllabus'), (['is', 'there', 'any', 'library'], 'library'), (['library', 'facility'], 'library'), (['library', 'facilities'], 'library'), (['do', 'you', 'have', 'library'], 'library'), (['does', 'the', 'college', 'have', 'library', 'facility'], 'library'), (['college', 'library'], 'library'), (['where', 'can', 'i', 'get', 'books'], 'library'), (['book', 'facility'], 'library'), (['Where', 'is', 'library'], 'library'), (['Library'], 'library'), (['Library', 'information'], 'library'), (['Library', 'books', 'information'], 'library'), (['Tell', 'me', 'about', 'library'], 'library'), (['how', 'many', 'libraries'], 'library'), (['how', 'is', 'college', 'infrastructure'], 'infrastructure'), (['infrastructure'], 'infrastructure'), (['college', 'infrastructure'], 'infrastructure'), (['food', 'facilities', 'canteen'], 'canteen'), (['canteen', 'facilities'], 'canteen'), (['canteen', 'facility'], 'canteen'), (['is', 'there', 'any',

'canteen'], 'canteen'), ([ 'Is', 'there', 'a', 'cafeteria', 'in', 'college'], 'canteen'),  
([ 'Does', 'college', 'have', 'canteen'], 'canteen'), ([ 'Where', 'is', 'canteen'],  
'canteen'), ([ 'where', 'is', 'cafeteria'], 'canteen'), ([ 'canteen'], 'canteen'), ([ 'Food'],  
'canteen'), ([ 'Cafeteria'], 'canteen'), ([ 'food', 'menu', 'menu'], ([ 'food', 'in',  
'canteen'], 'menu'), ([ 'Whats', 'there', 'on', 'menu', 'menu'], ([ 'what', 'is',  
'available', 'in', 'college', 'canteen', 'menu'], ([ 'what', 'foods', 'can', 'we', 'get', 'in',  
'college', 'canteen', 'menu'], ([ 'food', 'variety', 'menu'], ([ 'What', 'is', 'there', 'to',  
'eat', '?'], 'menu'), ([ 'What', 'is', 'college', 'placement'], 'placement'), ([ 'Which',  
'companies', 'visit', 'in', 'college'], 'placement'), ([ 'What', 'is', 'average', 'package'],  
'placement'), ([ 'companies', 'visit', 'placement'], ([ 'package'], 'placement'),  
([ 'About', 'placement', 'placement'], ([ 'placement', 'placement'], ([ 'recruitment'],  
'placement'), ([ 'companies', 'placement'], ([ 'Who', 'is', 'HOD'], 'ithod'), ([ 'Where',  
'is', 'HOD'], 'ithod'), ([ 'it', 'hod', 'ithod'], ([ 'name', 'of', 'it', 'hod', 'ithod'], ([ 'Who',  
'is', 'computer', 'HOD'], 'computerhod'), ([ 'Where', 'is', 'computer', 'HOD'],  
'computerhod'), ([ 'computer', 'hod', 'computerhod'], ([ 'name', 'of', 'computer',  
'hod', 'computerhod'], ([ 'Who', 'is', 'extc', 'HOD'], 'extchod'), ([ 'Where', 'is', 'extc',  
'HOD'], 'extchod'), ([ 'extc', 'hod', 'extchod'], ([ 'name', 'of', 'extc', 'hod', 'extchod'],  
([ 'what', 'is', 'the', 'name', 'of', 'principal'], 'principal'), ([ 'whatv', 'is', 'the',  
'principal', 'name'], 'principal'), ([ 'principal', 'name'], 'principal'), ([ 'Who', 'is',  
'college', 'principal'], 'principal'), ([ 'Where', 'is', 'principal', "'s", 'office'],  
'principal'), ([ 'principal', 'principal'], ([ 'name', 'of', 'principal'], 'principal'), ([ 'exam',  
'dates', 'sem'], ([ 'exam', 'schedule', 'sem'], ([ 'When', 'is', 'semester', 'exam'],  
'sem'), ([ 'Semester', 'exam', 'timetable', 'sem'], ([ 'sem'], 'sem'), ([ 'semester',  
'sem'], ([ 'exam', 'sem'], ([ 'when', 'is', 'exam'], 'sem'), ([ 'exam', 'timetable', 'sem'],  
([ 'exam', 'dates', 'sem'], ([ 'when', 'is', 'semester', 'sem'], ([ 'what', 'is', 'the',  
'process', 'of', 'admission'], 'admission'), ([ 'what', 'is', 'the', 'admission', 'process'],  
'admission'), ([ 'How', 'to', 'take', 'admission', 'in', 'your', 'college'], 'admission'),  
([ 'What', 'is', 'the', 'process', 'for', 'admission'], 'admission'), ([ 'admission'],  
'admission'), ([ 'admission', 'process'], 'admission'), ([ 'scholarship', 'scholarship'],  
([ 'Is', 'scholarship', 'available'], 'scholarship'), ([ 'scholarship', 'engineering'],  
'scholarship'), ([ 'scholarship', 'it'], 'scholarship'), ([ 'scholarship', 'ce'], 'scholarship'),  
([ 'scholarship', 'mechanical'], 'scholarship'), ([ 'scholarship', 'civil'], 'scholarship'),  
([ 'scholarship', 'chemical'], 'scholarship'), ([ 'scholarship', 'for', 'AI/ML'],

'scholarship'), ([ 'available', 'scholarships'], 'scholarship'), ([ 'scholarship', 'for',  
'computer', 'engineering'], 'scholarship'), ([ 'scholarship', 'for', 'IT', 'engineering'],  
'scholarship'), ([ 'scholarship', 'for', 'mechanical', 'engineering'], 'scholarship'),  
([ 'scholarship', 'for', 'civil', 'engineering'], 'scholarship'), ([ 'scholarship', 'for',  
'chemical', 'engineering'], 'scholarship'), ([ 'list', 'of', 'scholarship'], 'scholarship'),  
([ 'comps', 'scholarship'], 'scholarship'), ([ 'IT', 'scholarship'], 'scholarship'),  
([ 'mechanical', 'scholarship'], 'scholarship'), ([ 'civil', 'scholarship'], 'scholarship'),  
([ 'chemical', 'scholarship'], 'scholarship'), ([ 'automobile', 'scholarship'],  
'scholarship'), ([ 'first', 'year', 'scholarship'], 'scholarship'), ([ 'second', 'year',  
'scholarship'], 'scholarship'), ([ 'third', 'year', 'scholarship'], 'scholarship'), ([ 'fourth',  
'year', 'scholarship'], 'scholarship'), ([ 'What', 'facilities', 'college', 'provide'],  
'facilities'), ([ 'College', 'facility'], 'facilities'), ([ 'What', 'are', 'college', 'facilities'],  
'facilities'), ([ 'facilities'], 'facilities'), ([ 'facilities', 'provided'], 'facilities'), ([ 'max',  
'number', 'of', 'students'], 'college intake'), ([ 'number', 'of', 'seats', 'per', 'branch'],  
'college intake'), ([ 'number', 'of', 'seats', 'in', 'each', 'branch'], 'college intake'),  
([ 'maximum', 'number', 'of', 'seats'], 'college intake'), ([ 'maximum', 'students',  
'intake'], 'college intake'), ([ 'What', 'is', 'college', 'intake'], 'college intake'), ([ 'how',  
'many', 'student', 'are', 'taken', 'in', 'each', 'branch'], 'college intake'), ([ 'seat',  
'allotment'], 'college intake'), ([ 'seats'], 'college intake'), ([ 'college', 'dress', 'code'],  
'uniform'), ([ 'college', 'dresscode'], 'uniform'), ([ 'what', 'is', 'the', 'uniform'],  
'uniform'), ([ 'can', 'we', 'wear', 'casuals'], 'uniform'), ([ 'Does', 'college', 'have', 'an',  
'uniform'], 'uniform'), ([ 'Is', 'there', 'any', 'uniform'], 'uniform'), ([ 'uniform'],  
'uniform'), ([ 'what', 'about', 'uniform'], 'uniform'), ([ 'do', 'we', 'have', 'to', 'wear',  
'uniform'], 'uniform'), ([ 'what', 'are', 'the', 'different', 'committe', 'in', 'college'],  
'committee'), ([ 'different', 'committee', 'in', 'college'], 'committee'), ([ 'Are', 'there',  
'any', 'committee', 'in', 'college'], 'committee'), ([ 'Give', 'me', 'committee',  
'details'], 'committee'), ([ 'committee'], 'committee'), ([ 'how', 'many', 'committee',  
'are', 'there', 'in', 'college'], 'committee'), ([ 'I', 'love', 'you'], 'random'), ([ 'Will',  
'you', 'marry', 'me'], 'random'), ([ 'Do', 'you', 'love', 'me'], 'random'), ([ 'fuck'],  
'swear'), ([ 'bitch'], 'swear'), ([ 'shut', 'up'], 'swear'), ([ 'hell'], 'swear'), ([ 'stupid'],  
'swear'), ([ 'idiot'], 'swear'), ([ 'dumb', 'ass'], 'swear'), ([ 'asshole'], 'swear'),  
([ 'fucker'], 'swear'), ([ 'holidays'], 'vacation'), ([ 'when', 'will', 'semester', 'starts'],  
'vacation'), ([ 'when', 'will', 'semester', 'end'], 'vacation'), ([ 'when', 'is', 'the',

'holidays'], 'vacation'), ([ 'list', 'of', 'holidays'], 'vacation'), ([ 'Holiday', 'in', 'these', 'year'], 'vacation'), ([ 'holiday', 'list'], 'vacation'), ([ 'about', 'vacations'], 'vacation'), ([ 'about', 'holidays'], 'vacation'), ([ 'When', 'is', 'vacation'], 'vacation'), ([ 'When', 'is', 'holidays'], 'vacation'), ([ 'how', 'long', 'will', 'be', 'the', 'vacation'], 'vacation'), ([ 'sports', 'and', 'games'], 'sports'), ([ 'give', 'sports', 'details'], 'sports'), ([ 'sports', 'infrastructure'], 'sports'), ([ 'sports', 'facilities'], 'sports'), ([ 'information', 'about', 'sports'], 'sports'), ([ 'Sports', 'activities'], 'sports'), ([ 'please', 'provide', 'sports', 'and', 'games', 'information'], 'sports'), ([ 'okk'], 'salutaion'), ([ 'okie'], 'salutaion'), ([ 'nice', 'work'], 'salutaion'), ([ 'well', 'done'], 'salutaion'), ([ 'good', 'job'], 'salutaion'), ([ 'thanks', 'for', 'the', 'help'], 'salutaion'), ([ 'Thank', 'You'], 'salutaion'), ([ 'its', 'ok'], 'salutaion'), ([ 'Thanks'], 'salutaion'), ([ 'Good', 'work'], 'salutaion'), ([ 'k'], 'salutaion'), ([ 'ok'], 'salutaion'), ([ 'okay'], 'salutaion'), ([ 'what', 'can', 'you', 'do'], 'task'), ([ 'what', 'are', 'the', 'thing', 'you', 'can', 'do'], 'task'), ([ 'things', 'you', 'can', 'do'], 'task'), ([ 'what', 'can', 'u', 'do', 'for', 'me'], 'task'), ([ 'how', 'u', 'can', 'help', 'me'], 'task'), ([ 'why', 'i', 'should', 'use', 'you'], 'task'), ([ 'ragging'], 'ragging'), ([ 'is', 'ragging', 'practice', 'active', 'in', 'college'], 'ragging'), ([ 'does', 'college', 'have', 'any', 'antiragging', 'facility'], 'ragging'), ([ 'is', 'there', 'any', 'ragging', 'cases'], 'ragging'), ([ 'is', 'ragging', 'done', 'here'], 'ragging'), ([ 'ragging', 'against'], 'ragging'), ([ 'antiragging', 'facility'], 'ragging'), ([ 'ragging', 'juniors'], 'ragging'), ([ 'ragging', 'history'], 'ragging'), ([ 'ragging', 'incidents'], 'ragging'), ([ 'hod'], 'hod'), ([ 'hod', 'name'], 'hod'), ([ 'who', 'is', 'the', 'hod'], 'hod')]

38 classes

[ 'admission', 'canteen', 'college intake', 'committee', 'computerhod', 'course', 'creator', 'document', 'event', 'extchod', 'facilities', 'fees', 'floors', 'goodbye', 'greeting', 'hod', 'hostel', 'hours', 'infrastructure', 'ithod', 'library', 'location', 'menu', 'name', 'number', 'placement', 'principal', 'ragging', 'random', 'salutaion', 'scholarship', 'sem', 'sports', 'swear', 'syllabus', 'task', 'uniform', 'vacation']

263 unique lemmatized words

["'s", '(', ')', 'a', 'about', 'ac', 'active', 'activity', 'address', 'admsion', 'admission', 'against', 'ai/ml', 'allotment', 'am', 'an', 'and', 'antiragging', 'any', 'anyone', 'are', 'as', 'asshole', 'at', 'attend', 'automobile', 'available', 'average', 'be', 'between',



'big', 'bitch', 'book', 'boy', 'branch', 'bring', 'building', 'by', 'bye', 'cafeteria', 'call', 'called', 'campus', 'can', 'canteen', 'capacity', 'case', 'casuals', 'ce', 'chatting', 'chemical', 'civil', 'code', 'college', 'come', 'committe', 'committee', 'comp', 'company', 'computer', 'conducted', 'contact', 'course', 'create', 'created', 'creator', 'cya', 'date', 'day', 'designed', 'detail', 'developer', 'different', 'distance', 'do', 'document', 'doe', 'done', 'dress', 'dresscode', 'dumb', 'during', 'each', 'eat', 'end', 'engineering', 'event', 'exam', 'extc', 'facility', 'far', 'fee', 'first', 'floor', 'food', 'for', 'fourth', 'from', 'fuck', 'fucker', 'function', 'game', 'get', 'girl', 'give', 'go', 'good', 'goodbye', 'got', 'gtg', 'guy', 'have', 'held', 'hell', 'hello', 'help', 'here', 'hey', 'hi', 'history', 'hod', 'holiday', 'hostel', 'hour', 'how', 'i', 'idiot', 'in', 'incident', 'info', 'information', 'infrastructure', 'intake', 'is', 'it', 'job', 'junior', 'k', 'later', 'leaving', 'lecture', 'library', 'list', 'located', 'location', 'long', 'love', 'made', 'many', 'marry', 'max', 'maximum', 'me', 'mechanical', 'menu', 'more', 'much', 'my', 'name', 'need', 'needed', 'next', 'nice', 'no', 'non-ac', 'number', 'of', 'offer', 'offered', 'office', 'ok', 'okay', 'okie', 'okk', 'on', 'open', 'operation', 'organised', 'package', 'per', 'phone', 'placement', 'please', 'practice', 'principal', 'process', 'provide', 'provided', 'ragging', 'reach', 'recruitment', 'required', 'room', 'saturday', 'schedule', 'scholarship', 'seat', 'second', 'see', 'sem', 'semester', 'serve', 'should', 'shut', 'size', 'something', 'sport', 'start', 'student', 'stundent', 'stupid', 'syllabus', 'take', 'taken', 'taking', 'talk', 'tall', 'technology', 'telephone', 'tell', 'thank', 'thanks', 'the', 'there', 'these', 'thing', 'third', 'this', 'time', 'timetable', 'timing', 'to', 'ttyl', 'u', 'uni', 'uniform', 'univrsity', 'up', 'use', 'vacation', 'variety', 'visit', 'we', 'wear', 'well', 'what', 'whats', 'whatsup', 'whatv', 'when', 'where', 'wheres', 'which', 'who', 'whom', 'why', 'will', 'work', 'working', 'ya', 'year', 'you', 'your']

#### 4. PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.,

**1. Feature Engineering:** Feature engineering is the process of preparing and transforming raw data into a format suitable for training machine learning models. When developing a chatbot, this often involves the extraction of meaningful features from text and user interactions. Techniques like tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis can help in understanding user input and generating appropriate responses. Moreover, feature engineering can also involve incorporating user context, historical conversations, and user preferences to make the chatbot's responses more contextually relevant.

**2. Model Training:** The heart of a chatbot lies in its underlying machine learning model. In Python, popular libraries like TensorFlow, PyTorch, and scikit-learn offer robust tools for training chatbot models. Most chatbots leverage deep learning techniques, such as recurrent neural networks (RNNs) or transformer models like GPT (Generative Pre-trained Transformer), which can be fine-tuned for specific tasks. Model training involves feeding the chatbot with extensive data, including conversation transcripts, to enable it to learn how to respond to user queries effectively. Reinforcement learning and supervised learning are common approaches used for model training in chatbot development.

## **Necessary Steps:**

### **Step 1: Install Required Libraries**

to install NLTK if you haven't already. You can do this using pip:

```
pip install nltk
```

### **Step 2: Import Required Libraries**

```
import nltk
```

```
from nltk.chat.util import Chat, reflections
```

### Step 3: Define Chat Patterns and Responses

Create patterns and responses for your chatbot. You can customize these based on your chatbot's purpose. For this example, we'll create a simple chatbot that responds to greetings and questions.

```
# Define chat patterns and responses
```

```
patterns = [
```

```
    (r'hi|hello|hey', ['Hello!', 'Hi there!']),
```

```
    (r'how are you', ['I am just a chatbot, but thanks for asking!']),
```

```
    (r'what is your name', ['I am a chatbot. You can call me ChatGPT.']),
```

```
    (r'bye|goodbye', ['Goodbye!', 'See you later!']),
```

```
]
```

```
# Create a chatbot using patterns and reflections
```

```
chatbot = Chat(patterns, reflections)
```

### Step 4: Test the Chatbot

Now, you can test your chatbot by taking user input and generating responses using the patterns and responses you defined earlier.

```
print("ChatGPT: Hello! I'm your chatbot. You can start a conversation, or type 'bye' to exit.")
```

```
while True:
```

```
user_input = input("You: ")  
if user_input.lower() == 'bye':  
    print("ChatGPT: Goodbye!")  
    break  
response = chatbot.respond(user_input)  
print("ChatGPT:", response)
```

## **Step 5: Evaluation**

Evaluating the performance of a chatbot is critical to ensuring it meets the intended goals and provides a satisfying user experience. Several evaluation metrics can be used to assess the chatbot's performance, including accuracy, precision, recall, and F1 score. Moreover, user-centric metrics such as user satisfaction, response time, and conversation coherence play a pivotal role in determining the chatbot's success. Python libraries like NLTK, spaCy, and scikit-learn can assist in calculating these metrics. Continuous evaluation and improvement are essential as chatbots interact with users and adapt to their evolving needs.

In this comprehensive journey of chatbot development using Python, you will explore the art of feature engineering to understand user intent, train models that provide meaningful responses, and rigorously evaluate the chatbot's performance. As technology continues to advance, chatbots are becoming increasingly intelligent and capable of handling complex, context-aware conversations. This guide will equip you with the knowledge and tools needed to create chatbots that

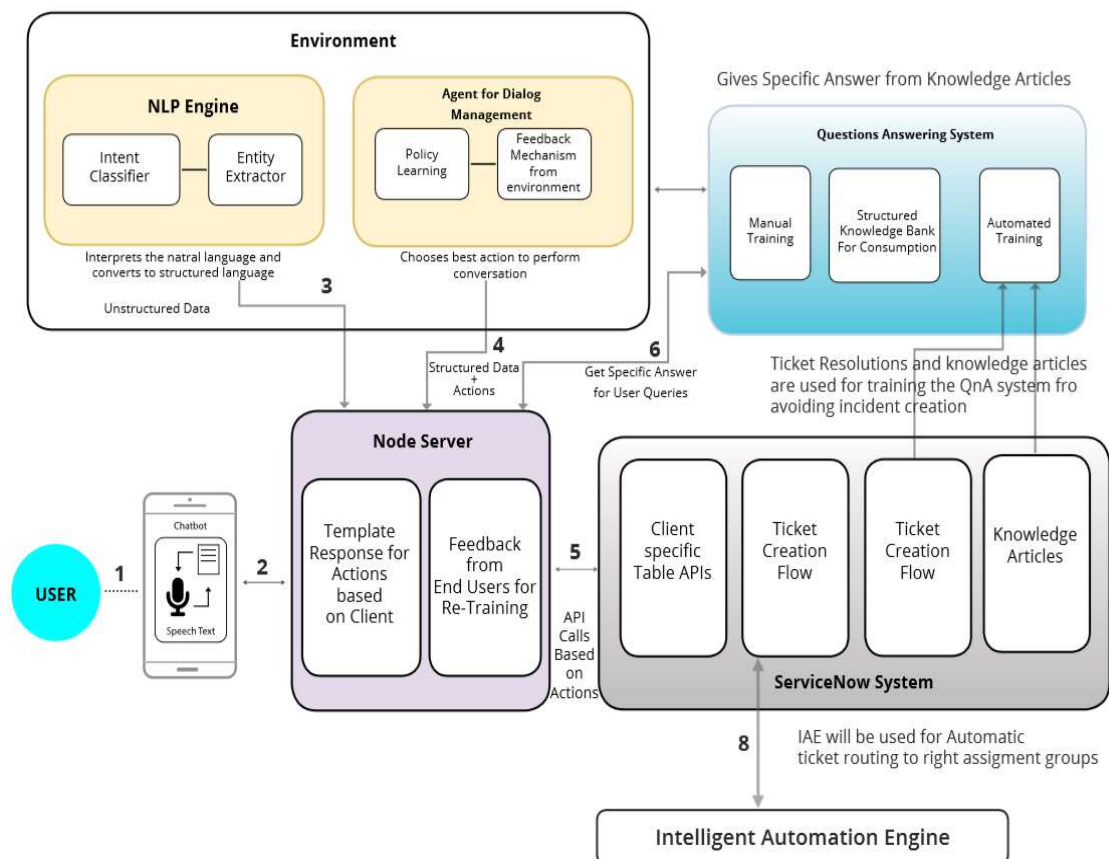
engage, assist, and delight users, contributing to the evolution of human-computer interactions.

This rule-based chatbot doesn't require traditional model training and evaluation as it relies on predefined patterns and responses. Instead, you can evaluate the chatbot's performance through user interactions and improve it by adding more patterns and responses for a better user experience.

However, if you want to create a more sophisticated chatbot that uses machine learning models, you would need to perform the following additional steps:

1. **Data Collection:** Collect and prepare a dataset of conversational data.
2. **Feature Engineering:** Extract features from the dataset, such as text preprocessing, tokenization, and vectorization.
3. **Model Training:** Train a machine learning model (e.g., a neural network using libraries like TensorFlow or PyTorch) on the prepared dataset.
4. **Model Evaluation:** Evaluate the chatbot's performance using metrics like accuracy, F1 score, or user satisfaction.
5. **Deployment:** Deploy the chatbot in a suitable environment (e.g., a web application, chat platform, or a mobile app).
6. **Continuous Improvement:** Continuously gather user feedback and iteratively improve the chatbot by retraining the model and updating its responses.

## Architecture diagram For Chatbot:



From the above diagram was the architecture diagram for chatbot.

## Coding Part:

The coding part is fully worked with python. This includes many library functions like NLTK, TensorFlow, NumPy and few other. These library functions help the chatbot to analyze the user request and decide the response to be given. Python itself has a package for chatbots, which is mainly required for the development of a user-friendly chatbot. Json is a python package that helps the query data set to get parsed by the Python code. This json file has the query with different tags. Each tag has a set of patterns and responses. This tag has default tag with it. The GUI is also developed using a Python package called tkinter. Tkinter is standard interface for GUI creations.

## PROGRAM FOR CHATBOT:

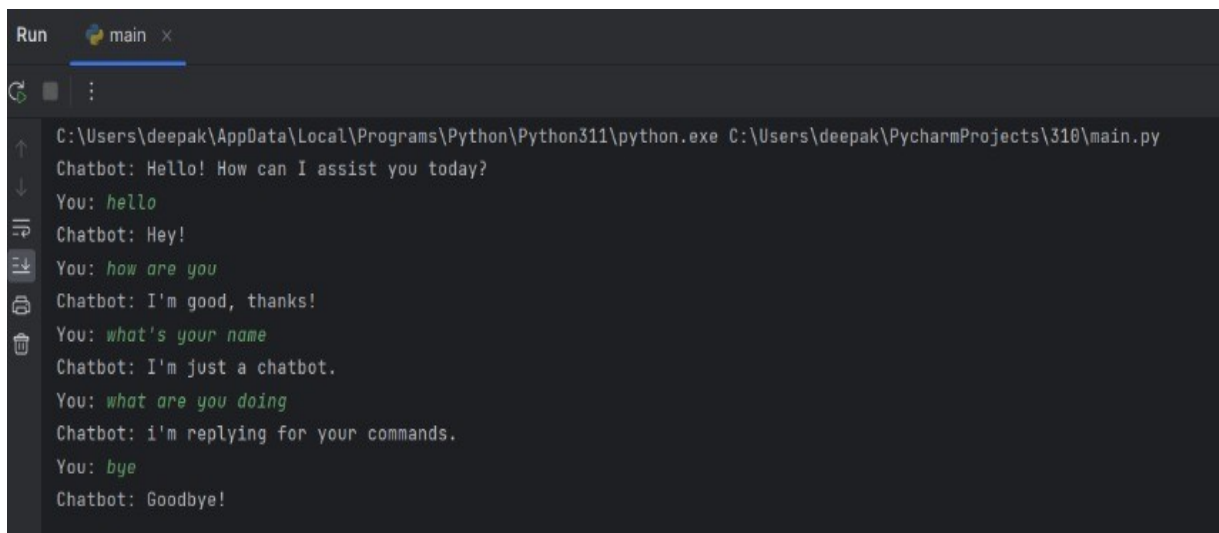
```
import random
# Define a list of responses

responses = {
    "hello": ["Hi there!", "Hello!", "Hey!"],
    "how are you": ["I'm good, thanks!", "I'm just a computer program, so I don't
have feelings, but I'm here to help!"],
    "what's your name": ["I'm just a chatbot.", "My name is Aakash."],
    "what are you doing": ["i'm replying for your commands.", "i'm replying for your
commands"],
    "bye": ["Goodbye!", "See you later!", "Have a great day!"]
}

def get_response(user_input):
    user_input = user_input.lower()
    for key in responses:
        if key in user_input:
            return random.choice(responses[key])
    return "I'm not sure how to respond to that."
```

```
print("Chatbot: Hello! How can I assist you today?")
while True:
    user_input = input("You: ")
    if user_input.lower() == "bye":
        print("Chatbot: Goodbye!")
        break
    response = get_response(user_input)
    print("Chatbot:", response)
```

## OUTPUT FOR CHATBOT:

A screenshot of a terminal window titled 'Run' with a sub-tab 'main'. The terminal shows the execution of a Python script. The output consists of alternating lines of chatbot responses and user inputs. The chatbot responses are in a standard font, while the user inputs are in a green monospace font. The chatbot's responses are: 'Hello! How can I assist you today?', 'Hey!', 'I'm good, thanks!', 'I'm just a chatbot.', 'i'm replying for your commands.', and 'Goodbye!'. The user's inputs are: 'hello', 'how are you', 'what's your name', 'what are you doing', and 'bye'. The terminal window has a dark background and a light-colored border.

```
Run main x
C:\Users\deepak\AppData\Local\Programs\Python\Python311\python.exe C:\Users\deepak\PycharmProjects\310\main.py
Chatbot: Hello! How can I assist you today?
You: hello
Chatbot: Hey!
You: how are you
Chatbot: I'm good, thanks!
You: what's your name
Chatbot: I'm just a chatbot.
You: what are you doing
Chatbot: i'm replying for your commands.
You: bye
Chatbot: Goodbye!
```

## Few Lines Of Coding:

By Using Windows Shell,command prompt,



```

import json
import pickle
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!', '']
data_file = open('intents.json').read()
intents = json.loads(data_file)
for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))
        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))

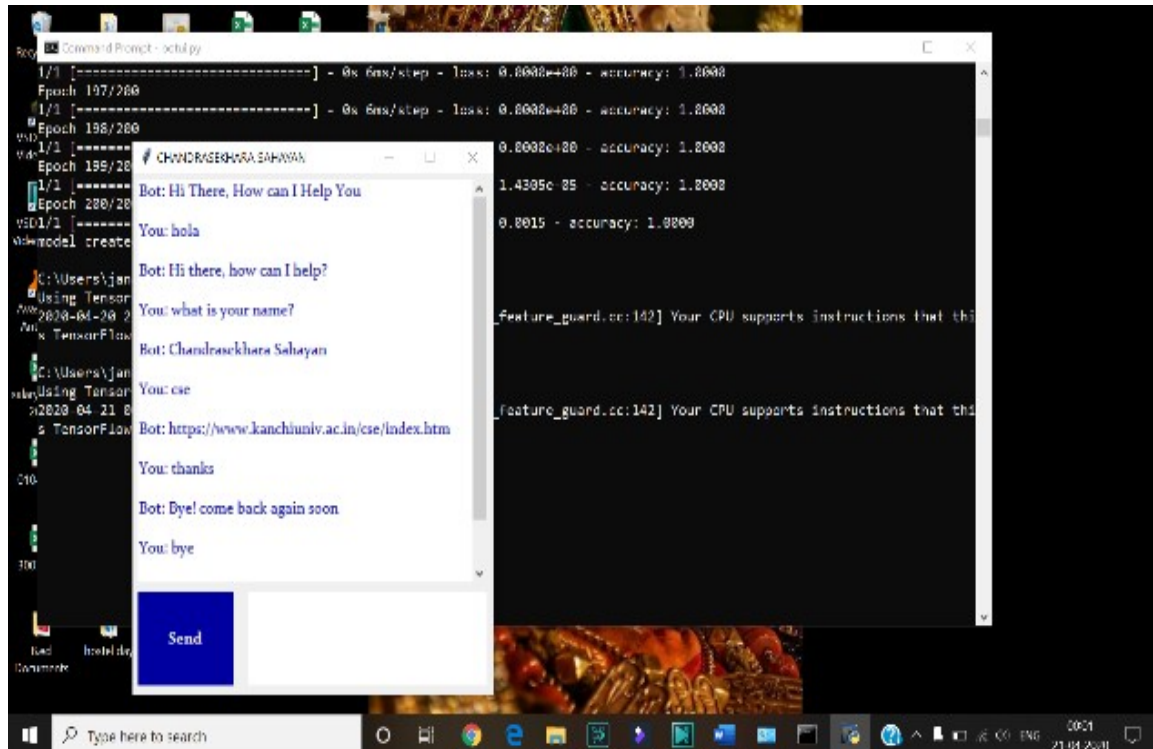
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")

```

```
intents - Notepad
File Edit Format View Help
{"intents": [
  {
    "tag": "greeting",
    "patterns": ["Hi there", "How are you", "Is anyone there?", "Hola", "Good day", "Hello"],
    "responses": ["Hello, thanks for asking", "good to see you again", "Hi there, how can I help?"]
  },
  {
    "tag": "goodbye",
    "patterns": ["bye", "see you later", "goodbye", "Nice chatting with you, bye", "till next time"],
    "responses": ["see you!", "have a nice day", "bye! come back again soon"],
    "context": [""]
  },
  {
    "tag": "thanks",
    "patterns": ["Thasnk", "Thank you", "that's helpful"],
    "responses": ["happy to help!", "any time", "my pleasure"],
    "context": [""]
  },
  {
    "tag": "no answer",
    "patterns": [],
    "responses": ["sorry, cant understand you", "please give me more info"],
    "context": [""]
  },
  {
    "tag": "about scsvm",
    "patterns": ["about the college", "about"],
    "responses": ["https://www.kanchiuniv.ac.in/aboutus.htm"],
    "context": [""]
  },
  {
    "tag": "fees",
    "patterns": ["fees payment website", "fees structure", "how to pay fees online", "online fees payment"],
    "responses": ["https://www.kanchiuniv.ac.in/online-payment.php"],
    "context": [""]
  },
  {
    "tag": "Admission",
    "patterns": ["admission details", "online application", "admissions"],
    "responses": ["https://www.kanchiuniv.ac.in/admission/index.htm", "https://www.kanchiuniv.ac.in/application2020/"],
    "context": [""]
  }
]
```

## Working Of Code:

```
C:\Windows\system32\cmd.exe - python trainbot.py
C:\Users\janan>cd onedrive
C:\Users\janan\OneDrive>cd desktop
C:\Users\janan\OneDrive\Desktop>cd chatbots
C:\Users\janan\OneDrive\Desktop\chatbots>python trainbot.py
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\janan\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]   C:\Users\janan\AppData\Roaming\nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data]   date!
List Trainer: [#####] 100%
You:Hi
Hi
ChatBot: k
You:how are you doing?
how are you doing?
ChatBot: /
You:attendance details of a student
attendance details of a student
ChatBot: - check your attendance in the given link http://studentportal.kanchiuniv.ac.in/scsvmonline/students/loginManager/youlogin.jsp
You:fees payment website
fees payment website
ChatBot: /
You:how to check the faculty details of each department
how to check the faculty details of each department
ChatBot: -https://www.kanchiuniv.ac.in/ in the given link check the academics list to find the faculty details.
You:admission details
admission details
ChatBot: -check your attendance in the given link http://studentportal.kanchiuniv.ac.in/scsvmonline/students/loginManager/youlogin.jsp
You:placement details
placement details
ChatBot: -check your attendance in the given link http://studentportal.kanchiuniv.ac.in/scsvmonline/students/loginManager/youlogin.jsp
You:circular of events
circular of events
ChatBot: - https://www.kanchiuniv.ac.in/circulars.htm
```



## Working Algorithm:

- Step 1) Start
- Step 2) Select a data set, for which we need to develop a chatterbot.
- Step 3) Prepare the set of tags with the patterns and responses.
- Step 4) Install the required packages in Python.
- Step 5) Train the Chatbot with predefined queries.
- Step 6) Develop the GUI
- Step 7) Execute the codes for the results.
- Step 8) Stop.

## Advantages of Creating Chatbots:

The creation of chatbots in Python offers several advantages:

1. **Automation:** Chatbots can handle routine and repetitive tasks, saving time and resources for businesses and organizations.

2. **24/7 Availability:** Chatbots are available around the clock, providing support and information to users even outside regular business hours.
3. **Scalability:** Chatbots can handle multiple conversations simultaneously, making them ideal for handling a high volume of inquiries.
4. **Consistency:** Chatbots deliver consistent responses, reducing the risk of human errors and ensuring uniform customer experiences.
5. **Cost-Effective:** Chatbots can significantly reduce operational costs by replacing or assisting human agents.
6. **Improved Customer Engagement:** Chatbots can engage users in natural conversations, enhancing user experience and satisfaction.

### **Disadvantages For Creating Chatbots:**

1. Limited Understanding:
  - Chatbots often struggle to understand complex or ambiguous user queries, particularly when users express themselves idiomatically, use slang, or have strong accents. This can result in frustration for users.
2. Inconsistent Quality:
  - The quality of responses from chatbots can vary widely depending on their training data and algorithms. Inaccurate or irrelevant responses can lead to user dissatisfaction.
3. Security Concerns:
  - Chatbots may interact with sensitive or private information. Inadequate security measures can lead to data breaches or privacy violations, making security a critical concern.
4. Maintenance and Updates:

- Chatbots require continuous maintenance and updates to remain relevant and accurate. Neglecting these aspects can result in a chatbot that becomes outdated and ineffective.
5. Resistance to Automation:
- Some users may resist automation and prefer to speak with a human. Chatbots should be used judiciously in contexts where human interaction is preferred.
6. User Adoption Challenges:
- Convincing users to adopt and trust chatbots can be a barrier. Users may have reservations about security, privacy, or the chatbot's abilities.
7. Costs and ROI:
- Developing, maintaining, and training chatbots can incur costs. Organizations should carefully assess the return on investment (ROI) to ensure the chatbot is economically viable.

## **Benefits:**

The development of chatbots in Python offers several benefits:

1. **Automation:** Chatbots can automate routine and repetitive tasks, reducing the need for human intervention.
2. **Scalability:** They can handle multiple conversations simultaneously, making them ideal for managing high volumes of user interactions.
3. **Availability:** Chatbots are available 24/7, providing support and information to users at any time.
4. **Consistency:** Chatbots deliver consistent responses, ensuring uniform customer experiences.

5. **Cost-Effective:** Chatbots can reduce operational costs by replacing or assisting human agents.
6. **Improved User Experience:** Well-designed chatbots can engage users in natural conversations, enhancing user satisfaction and retention.

## Conclusion:

In conclusion, creating a chatbot is a dynamic and innovative endeavor that combines technology, user-centered design, and artificial intelligence. Chatbots have evolved to become valuable assets across a wide range of industries, from customer service and healthcare to education and entertainment. Here are some key takeaways:

1. **User-Centered Approach:** A successful chatbot begins with a deep understanding of user needs and expectations. Empathize with your target audience to create a chatbot that truly addresses their pain points and delivers value.
2. **Diverse Use Cases:** Chatbots can be applied in diverse contexts, from automating routine tasks to providing personalized recommendations and assistance. Identifying the right use case is crucial to the chatbot's success.
3. **Technology Stack:** Python, along with a variety of NLP libraries and frameworks, is a popular choice for chatbot development. Selecting the right technology stack is essential to meet your chatbot's requirements.
4. **Design Thinking:** Applying design thinking principles can help you create user-friendly, effective, and innovative chatbots. Empathize, define, ideate, prototype, test, implement, gather feedback, and iterate to build a chatbot that truly resonates with users.
5. **Innovation:** Innovation is key to standing out in the world of chatbots. Explore novel features and functionalities, such as emotional intelligence, contextual memory, multimodal interaction, and more, to create chatbots that captivate users and provide unique value.

6. **User Feedback:** Continuously gather and act on user feedback. Chatbots are not static; they require ongoing refinement and improvements to maintain relevance and user satisfaction.
7. **Integration and Deployment:** Consider where and how your chatbot will be used. Integration with existing systems and platforms is critical, and the choice of deployment method should align with user accessibility.
8. **Privacy and Security:** Ensure that your chatbot handles user data with the utmost care. Implement robust security measures to protect user information and comply with data privacy regulations.
9. **Cost-Benefit Analysis:** Assess the costs and potential return on investment for your chatbot project. A well-planned and executed chatbot can offer significant advantages, but it's essential to manage costs effectively.
10. **Human-AI Collaboration:** Recognize that chatbots are tools to augment human capabilities, not necessarily replace them. Collaboration between humans and chatbots can lead to more efficient and satisfying outcomes.

In today's digital landscape, chatbots have become an integral part of improving user experiences and streamlining processes. When thoughtfully designed and effectively implemented, chatbots have the potential to be game-changers, enhancing customer engagement, automating tasks, and providing innovative solutions.