CREATE AN CHATBOT IN PYTHON

**PHASE 5:PROJECT** **DOCUMENTATION&SUBMISSION**

**PROBLEM STATEMENT:**

The problem is to develop a chatbot that can understand user input, process it, and generate appropriate responses in a conversational manner. This involves natural language understanding, context management, and generating relevant and coherent replies.

**Design Thinking Process:**

**1.Define the Purpose and Scope:**

* Determine the specific tasks the chatbot will perform (e.g., customer support, information retrieval, etc.
* Understand the target audience and their preferences

**2. Choose a Development Approach:**

* **Rule-based approach**: Define predefined rules and patterns for responses.
* **Machine learning approach**: Use models like RNNs, LSTMs, or transformer-based models (like GPT-3, GPT-4).

**3. Data Collection and Preprocessing:**

* Gather and curate a dataset for training (if using ML-based approach).
* Tokenize and clean the text data.

**4. Model Selection (if ML-based):**

* Choose a suitable machine learning or deep learning architecture.
* Fine-tune or train the model on the prepared dataset.

**5. Natural Language Understanding (NLU):**

* Implement techniques for understanding user intent and extracting relevant information from input sentences**.**
* Apply techniques like Named Entity Recognition (NER) if necessary.

**6. Context Management:**

* Maintain context of the conversation to provide coherent and relevant responses.
* Use techniques like memory networks or context windowing.

**7. Response Generation:**

* Based on the understanding of user input, generate appropriate responses.
* For rule-based systems, design response templates.
* For ML-based models, use the trained model to generate responses.

**8. Integration with External Systems (if needed):**

* Connect the chatbot to databases, APIs, or other external systems for information retrieval or processing**.**

**9. Testing and Validation:**

* Test the chatbot with various inputs, including edge cases.
* Evaluate its performance against predefined metrics (e.g., accuracy, response time,
* user satisfaction).

**Phases of Development:**

**1.Planning and Design:**

* Define the purpose, target audience, and scope of the chatbot.
* Decide on the development approach (rule-based or ML-based).
* Create a high-level architectural design.

**2. Data Collection and Preprocessing:**

* Gather a diverse dataset for training (if using ML-based approach).
* Preprocess the data by cleaning, tokenizing, and normalizing text.

**3. Model Development (if ML-based):**

* Select or design a suitable ML model.
* Train and fine-tune the model on the preprocessed data**.**

**4. Natural Language Understanding (NLU):**

* Implement techniques for intent recognition and information extraction.

**5. Context Management:**

* Develop a mechanism to maintain context and history of the conversation**.**

**6. Response Generation:**

* Implement logic for generating responses based on user input and context.

**7. Integration and Testing:**

* Connect the chatbot to external systems if required.
* Conduct extensive testing, including unit tests, integration tests, and user acceptance tests.

**8. Deployment and Monitoring:**

* Deploy the chatbot in the desired environment (e.g., web, mobile, messaging platforms).
* Monitor its performance and gather user feedback for further improvements.

**9. Maintenance and Updates:**

* Regularly update and improve the chatbot based on user feedback and changing requirements.
* Following these steps and phases will help in the systematic development of a chatbot in Python, ensuring it meets its intended purpose effectively.

**Libraries:**

**Import libraries:**

**pip install Flask**

**pip install ChatterBot==1.0.8**

**pip install chatterbot-corpus==1.2.0**

**pip install spacy==2.1.9**

**pip install nltk==3.8.1**

**1. NLTK (Natural Language Toolkit):**

* NLTK provides a suite of libraries and programs for natural language processing.
* It includes modules for tokenization, stemming, tagging, parsing, and semantic reasoning.

**2. SpaCy:**

* SpaCy is a popular NLP library that provides pre-trained models for various tasks like
* named entity recognition (NER), part-of-speech tagging, dependency parsing, and more.

**3. Gensim:**

* Gensim is a library for topic modeling, document indexing, and similarity retrieval with large corpora. It particularly useful for tasks like document similarity analysis and text summarization.

**4. Transformers (Hugging Face):**

* The Transformers library by Hugging Face provides easy-to-use APIs for working with
* state-of-the-art pre-trained models like BERT, GPT, etc. It&#39;s particularly powerful for
* tasks like sentiment analysis, text generation, and more.
* 5. TensorFlow / PyTorch:
* These deep learning frameworks are essential for developing and training custom NLP
* models. They are used to implement architectures like LSTMs, RNNs, and transformers.

**6. Scikit-learn:**

* While not specifically an NLP library, Scikit-learn provides tools for text feature
* extraction (like TF-IDF), which can be useful in NLP tasks.

**NLP Techniques:**

**1.Tokenization:**

* Divide text into smaller units (tokens) like words or subwords. This is a fundamentalstep in NLP.

**2. Stopword Removal:**

* Eliminate common words (e.g., &quot;the&quot;, &quot;is&quot;, &quot;and&quot;) that don&#39;t carry much information.

**3. Stemming and Lemmatization:**

* Reduce words to their root or base form (e.g., &quot;running&quot; to &quot;run&quot;). This helps in reducing the dimensionality of the data.

**4. Named Entity Recognition (NER):**

* Identify entities in text like names of people, places, organizations, etc.

**5. Part-of-Speech (POS) Tagging:**

* Assign grammatical parts of speech to words in a sentence.

**6. Dependency Parsing:**

* Analyze the grammatical structure of a sentence and establish relationships between words.

**7. Sentiment Analysis:**

* Determine the sentiment or emotion expressed in a piece of text (e.g., positive,negative, neutral).

**8. Intent Recognition:**

* Understand the intention behind a user&#39;s input (e.g., asking a question, making a request).

**9. Text Generation:**

* Use models like GPT to generate coherent and contextually relevant text.

**10. Context Management:**

* Keep track of the conversation history to generate responses that are contextually relevant.

**Integration:**

**1. Preprocessing:**

* Use libraries like NLTK, SpaCy, or custom code for tasks like tokenization,stopword removal, and stemming/lemmatization.

**2. NLP Libraries for Tasks:**

* Utilize libraries like SpaCy or NLTK for tasks such as NER, POS tagging, and dependency parsing.

**3. Custom Model Integration:**

* Integrate pre-trained models (e.g., from Hugging Face Transformers) for tasks like sentiment analysis or text generation.

**4. Context Management:**

* Implement logic to maintain context in the conversation, ensuring that the chatbot can generate coherent responses.

**Interaction Flow:**

**1. User Input:**

* The interaction starts when a user provides input through the web application. This can be in the form of text messages, voice input, or other input methods.

**2. Frontend Processing:**

* The frontend of the web application is responsible for capturing and sending the input to the backend for processing. This can be done using HTML forms, JavaScript, or other frontend technologies.

**3. Backend Processing:**

* The backend of the web application receives the input. This is where the chatbot logic resides.

**4. NLP Processing:**

* If the chatbot uses Natural Language Processing (NLP) techniques, the backend may apply tasks like tokenization, entity recognition, sentiment analysis, etc., to understand the input in context.

**5. Context Management:**

* If the chatbot needs to maintain context (e.g., in a conversation), the backend keeps track of the conversation history to generate relevant responses.

**6. Generating a Response:**

* Based on the processed input and context, the chatbot generates a response. This can
* be a text message, a voice response, or even a combination of various media types.

**7. Backend to Frontend Communication:**

* The backend sends the generated response back to the frontend using APIs or other communication protocols. This is typically done using technologies like HTTP requests or WebSockets.

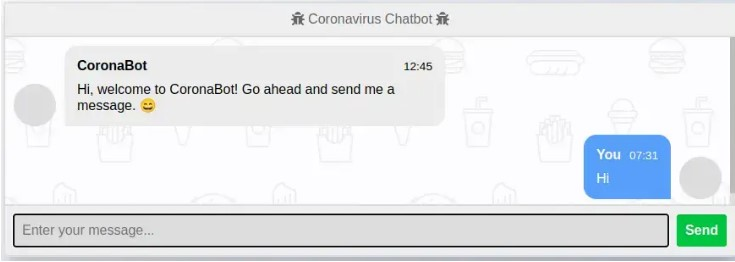
**8. Frontend Presentation:**

* The frontend receives the response from the backend and presents it to the user. This

can be done through the web interface using HTML, CSS, and JavaScript.

**9. User Interaction Loop:**

* The user can continue the interaction by providing further input. The process repeats,allowing for a back-and-forth conversation.



Our chatbot will greet users, engage in interactive conversations, and provide accurate and helpful information about Covid-19. Through this project, we aim to create a web-based chatbot that serves as a reliable resource for users seeking information about the pandemic and its various aspects.

**Initializing Flask App:**

**from chatbot import chatbot**

**from flask import Flask, render\_template, request**

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

**app.static\_folder = 'static'**

**@app.route("/")**

**def home():**

**return render\_template("index.html")**

**@app.route("/get")**

**def get\_bot\_response():**

**userText = request.args.get('msg')**

**return str(chatbot.get\_response(userText))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**app.run()**

**Run the Flask App:**

**python3.7 app.py**

**Technologies Involved:**

**Frontend:**

* HTML/CSS/JavaScript: These are the fundamental building blocks of web
* development for creating the user interface and handling user interactions.

**Backend:**

* Python (or any backend language): This is used to implement the chatbot logic,including NLP processing and context management.

**folder structure your project:**

**chatbot\_project/**

**├── app.py**

**├── chatbot.py**

**├── venv**

**├── database.sqlite3.py**

**├── templates/**

**│ └── index.html**

**├── training\_data/**

**│ ├── ques\_ans.txt**

**│ └──personal\_ques.txt**

**└── static/**

**└── style.css**

**Web Framework:**

* A web framework (e.g., Django, Flask for Python) is used to handle HTTP requests and responses, manage sessions, and facilitate communication between the frontend and

backend.

**NLP Libraries and Tools:**

* Libraries like NLTK, SpaCy, or Hugging Face Transformers are used for NLP tasks such as tokenization, entity recognition, sentiment analysis, etc.

**APIs and Communication Protocols:**

* APIs or communication protocols (e.g., RESTful APIs, WebSockets) are used for communication between the frontend and backend.

**Database (if necessary):**

* If the chatbot needs to store or retrieve information from a database, a database system is integrated into the backend.

## innovative for Export a WhatsApp Chat:

## you’ll have downloaded a TXT file that contains the chat history of a WhatsApp conversation. If you don’t have a WhatsApp account or don’t want to work with your own conversational data

## To export the history of a conversation that you’ve had on WhatsApp, you need to open the conversation on your phone. Once you’re on the conversation screen, you can access the export menu:

* **Click on the three dots (⋮) in the top right corner to open the main menu.**
* **Choose *More* to bring up additional menu options.**
* **Select *Export chat* to create a TXT export of your conversation.**

## 

**Once you’ve clicked on Export chat, you need to decide whether or not to include media, such as photos or audio messages. Because your chatbot is only dealing with text, select WITHOUT MEDIA. Then, you can declare where you’d like to send**