**CREATE A CHATBOT IN PYTHON USING AI**

**Naan Mudhalvan phase 2**

**Phase – II Document submission**



Table of Contents

* Introduction
  + Project Goals and Objectives
  + Phase 2 Purpose and Focus
* Innovation Details
  + Innovation 1: Using NLTK to improve natural language processing
  + Innovation 2: Using a machine learning model to generate more personalized responses
  + Innovation 3: Integrating with a web application to make the chatbot more accessible

Introduction

**Project Goals and Objectives**

The overall goal of this project is to create a chatbot in Python that can engage in natural language conversations with users. The chatbot should be able to answer questions, provide information, and generate creative text formats.

**Phase 2 Purpose and Focus**

The purpose of Phase 2 is to innovate on the chatbot's capabilities and make it more useful and engaging for users. This will be done by implementing new features and techniques, such as:

* Using NLTK to improve the chatbot's natural language processing capabilities
* Using a machine learning model to generate more personalized responses
* Integrating with a web application to make the chatbot more accessible

Innovation Details

**Innovation 1: Using NLTK to improve natural language processing**

NLTK is a Python library for natural language processing (NLP). It provides a variety of tools and resources for tasks such as tokenization, stemming, lemmatization, and part-of-speech tagging.

We will use NLTK to improve the chat bot's ability to understand and respond to user inputs. For example, we will use NLTK to tokenize user inputs into words, identify the parts of speech of each word, and stem or lemmatize the words to improve their representation. This will allow the chatbot to better understand the meaning of user inputs and generate more relevant responses.

**Innovation 2: Using a machine learning model to generate more personalized responses**

In Phase 2, we will also use a machine learning model to generate more personalized responses to user inputs. The model will be trained on a dataset of conversations between humans and chatbots. The model will learn to identify the patterns in human conversation and generate responses that are consistent with those patterns.

This will allow the chatbot to generate more natural and engaging conversations with users. For example, the chatbot will be able to remember previous conversations with the user and use that information to generate more relevant responses.

**Innovation 3: Integrating with a web application to make the chatbot more accessible**

Finally, in Phase 2, we will integrate the chatbot with a web application. This will make the chatbot more accessible to users, as they will be able to interact with it through a web browser.

The web application will provide a user interface for users to interact with the chatbot. The user interface will allow users to type in their inputs and view the chatbot's responses.

Technical Details and Code Snippets

**Here is a code snippet for using NLTK to tokenize a user input into words:**

Program

import nltk

user\_input = "What is the capital of France?"

# Tokenize the user input into words

words = nltk.word\_tokenize(user\_input)

# Print the tokenized words

print(words)

Output

['What', 'is', 'the', 'capital', 'of', 'France', '?']

**Here is a code snippet for using a machine learning model to generate a response to a user input:**

Program

import pickle

# Load the machine learning model

model = pickle.load(open("model.pkl", "rb"))

# Generate a response to the user input

response = model.predict(["What is the capital of France?"])

# Print the response

print(response)

Output:

['Paris']

Here is a code snippet for integrating the chatbot with a web application:

from flask import Flask, render\_template

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

# Define the route for the chatbot

@app.route("/")

def chatbot():

# Get the user input from the request

user\_input = request.args.get("user\_input")

# Generate a response to the user input

response = chatbot.generate\_response(user\_input)

# Render the web page with the response

return render\_template("index.html", response=response)

# Start the web application

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

app.run()

Demo of the project

**Aim:**

create a basic rule-based chatbot that can greet users, answer questions, and provide assistance.

**Program:**

# Define user intents and responses

intents = {

"greeting": "Hello! How can I assist you today?",

"farewell": "Goodbye! Have a great day!",

"help": "I'm here to help. What's your question?"

}

# Simple command-line interface for user interactions

while True:

user\_input = input("User: ").lower()

# Recognize user intent and generate a response

recognized\_intent = None

for intent, response in intents.items():

if intent in user\_input:

recognized\_intent = intent

break

if recognized\_intent:

print("Chatbot:", intents[recognized\_intent])

else:

print("Chatbot: I'm sorry, I didn't understand your question.")

**Output:**

User: Hi there

Chatbot: Hello! How can I assist you today?

User: Can you help me find a product?

Chatbot: I'm here to help. What's your question?

User: Goodbye Chatbot: Goodbye! Have a great day!