CHATBOT FOR AGRICULTURE

A PROJECT REPORT

Submitted by

HELEN SHALINI A (2116210701082)
VARSSHA BALASUNDARAM (2116210701325)
KARISHMA KANNADASAN (2116210701106)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Thesis titled **“CHATBOT FOR AGRICULTURE**” is the bonafide work of “**HELEN SHALINI A (2116210701082), VARSSHA BALASUNDARAM (2116210701325), KARISHMA KANNADASAN (2116210701106)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr . K.Anand M.E.,Ph.D.,

PROJECT COORDINATOR

Professor

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai - 602 105

Submitted to Project Viva-Voce Examination held on_________________

Internal Examiner **External Examine**r

ABSTRACT

With technological innovation, agriculture has the potential to undergo considerable revolution in the current digital era. This project offers a user-friendly web application with a chatbot that is intended to give farmers and individuals useful information about agriculture. The chatbot uses machine learning techniques to provide specific assistance on a variety of issues, from best practices to crop production and pest management. Users can interact with an abundance of information and resources that are suited to their needs through an easy-to-use interface while having natural language discussions. The chatbot seeks to empower members of the farming community by making access to agricultural knowledge and resources, promoting sustainable farming practices, increasing production, and encouraging economic growth. The chatbot, with its user-centered design and continuous improvement, is a possibly helpful device for knowledge cultivation and teamwork.

ACKNOWLEDGMENT

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college Principal, **Dr. S. N. Murugesan M.E., PhD.,** and **Dr. P. KUMAR M.E., PhD, Director computing and information science , and Head Of Department of Computer Science and Engineering** and our project coordinator **Dr. K.Anand M.E.,Ph.D.,** for his encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

HELEN SHALINI A

VARSSHA BALASUNDARAM

KARISHMA KANNADASAN

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This project presents the development of an interactive chatbot designed to serve as a comprehensive resource for agricultural inquiries. Leveraging a pre-processed JSON dataset encompassing a wide range of agricultural topics, the chatbot utilizes machine learning models and natural language processing (NLP) techniques to interpret user queries and deliver relevant, context-aware responses. This functionality empowers the chatbot to address diverse user needs, including crop recommendations, pest management strategies, weather forecasts, and various agricultural practices.

At the core lies a trained neural network model, constructed using Keras and TensorFlow, that effectively predicts user intent based on the provided query. This model is supported by the pre-processed dataset, which meticulously categorizes intents and corresponding responses. These intents encompass a comprehensive spectrum of agricultural knowledge, including crop selection, pest control methods, weather impact analysis, and diverse farming techniques. By parsing the JSON formatted dataset, the chatbot efficiently matches user input to the most pertinent response, ensuring informative and accurate replies.

The user interface prioritizes a user-centric approach, facilitating seamless interaction through a web-based platform. Developed with Flask, HTML, and JavaScript, the interface offers a straightforward chat window for users to pose questions and receive real-time responses. Colour-coded message boxes effectively differentiate user inputs from chatbot replies, enhancing the overall user experience.

1.1 PROBLEM STATEMENT

The agricultural community, including both farmers and enthusiasts, faces a critical challenge in obtaining timely and relevant information on crucial aspects like crop management, pest control, weather conditions, and best practices. This limited access to knowledge hinders informed decision-making, potentially leading to suboptimal agricultural outcomes. This project proposes to address this challenge by developing an intelligent chatbot solution. By interpreting user queries and delivering accurate, context-aware responses, the chatbot aims to empower the agricultural community and enhance overall knowledge accessibility.

1.2 SCOPE OF THE WORK

This project encompasses the development of an intelligent chatbot designed to empower the agricultural community. Natural Language Processing (NLP) techniques will be integrated to facilitate effective comprehension and response to user queries. Additionally, machine learning models will be leveraged to accurately predict user intent, ensuring the delivery of relevant and helpful agricultural information. The chatbot's scope encompasses a comprehensive range of agricultural topics, including crop cultivation, pest management, weather forecasting, best practices, and market trends.

1.3 AIM AND OBJECTIVES OF THE PROJECT

This project aims to build an intelligent chatbot for farmers. It will use natural language processing (NLP) to understand user questions and machine learning to predict user intent. The chatbot will address various agricultural topics with a user-friendly interface for seamless interaction. This comprehensive and scalable tool empowers farmers with timely information for better decision-making.

1.4 RESOURCES

This project demands high-performance computing power (GPUs) for training models and real-time chatbot interaction. Software-wise, Python, Flask, Keras/TensorFlow, NLTK, IDEs, and Git are crucial. A comprehensive, well-annotated agricultural dataset (JSON format) is vital for training and testing. Finally, a team of data scientists, web developers, and agricultural experts is needed, along with documentation, training materials, and testing tools for a robust and reliable chatbot.

1.5 MOTIVATION

The Challenge and its Significance:

The agricultural sector plays a pivotal role in global food security and economic stability. However, farmers face a multitude of challenges that threaten agricultural sustainability and profitability. Unpredictable weather patterns, escalating pest infestations, and volatile market conditions significantly impact crop yields and income. Access to reliable and timely information is crucial for farmers to navigate these complexities and make informed decisions.

The Proposed Solution and its Impact:

This project addresses this critical need by harnessing the power of artificial intelligence. We propose an intelligent chatbot that leverages machine learning and natural language processing to empower farmers. This user-friendly platform will provide access to expert advice and tailored solutions, enabling farmers to make informed decisions regarding crop management and optimize their agricultural practices. The project extends beyond farmer empowerment, aiming to contribute to broader goals of food security and sustainable agricultural practices. Ultimately, by delivering precise and actionable insights, this chatbot can help farmers build resilience and thrive in a challenging environment.

CHAPTER 2

LITERATURE SURVEY

Artificial Intelligence (AI) chatbots [1] provide a novel format for individuals to interact with large language models (LLMs). Recently released tools allow nontechnical users to develop chatbots using natural language. Surgical education is an exciting area in which chatbots developed in this manner may be rapidly deployed, though additional work will be required to ensure their accuracy and safety. In this paper, we outline our initial experience with AI chatbot creation in surgical education and offer considerations for future use of this technology

Artificial intelligence chatbots are making waves in education [2] , particularly for English speaking skills. A recent review examined research on chatbots used in English language learning. While the field is young, the findings suggest chatbots can improve learning speed, confidence, and motivation. This research offers valuable insights for teachers, chatbot designers, and future research.

Land access is a major hurdle for young people in African agriculture. Land titling programs are seen as a solution [3], but their effectiveness is unclear. This study in Tanzania found that titled land leads young people to invest more time in farming. It also highlights education and farm size as important factors. The study recommends targeted land titling programs for youth, along with relevant educational programs, to boost youth participation in agriculture for Tanzania and potentially other African countries.

This [4] research analyzes power's impact across four stages: generation, transmission, distribution, and consumption. Surprisingly, the consumption stage, measured by residential electricity use and service users, has the strongest influence on agricultural growth. This suggests focusing investment and upgrades at the consumption level for better agricultural outcomes. The study also introduces a data-driven model for predicting future trends, allowing for more informed policy decisions. Overall, this research highlights the need to move beyond broad assessments and delve deeper to understand how power infrastructure truly supports rural agriculture.

In [5] research tackles a challenge in machine reading comprehension (MRC), where AI models struggle to understand complex relationships between questions and paragraphs. The current method using RoBERTa, a powerful AI technique, suffers from divided attention.This study proposes a new approach with three separate inputs: question, paragraph, and combined RoBERTa outputs. These are then processed to improve focus and co-attention between question and answer. Experiments show this method significantly outperforms existing models, suggesting a path for better AI reading comprehension.

As smart farms teem with devices [6], secure communication across domains becomes crucial. Existing authentication methods, reliant on a central hub, struggle with this. To address this, researchers propose a blockchain-based scheme. This system enables secure communication between devices on different farms, even with a surge of devices. It achieves this through a decentralized architecture, efficient group verification for faster authentication, and pseudonym updates to safeguard device privacy. This approach offers enhanced security for data and devices in smart agriculture networks.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

In this section, we would like to show how the general outline of how all the components end up working when organized and arranged together. It is further represented in the form of a flow chart below.

3.2 SYSTEM ARCHITECTURE DIAGRAM

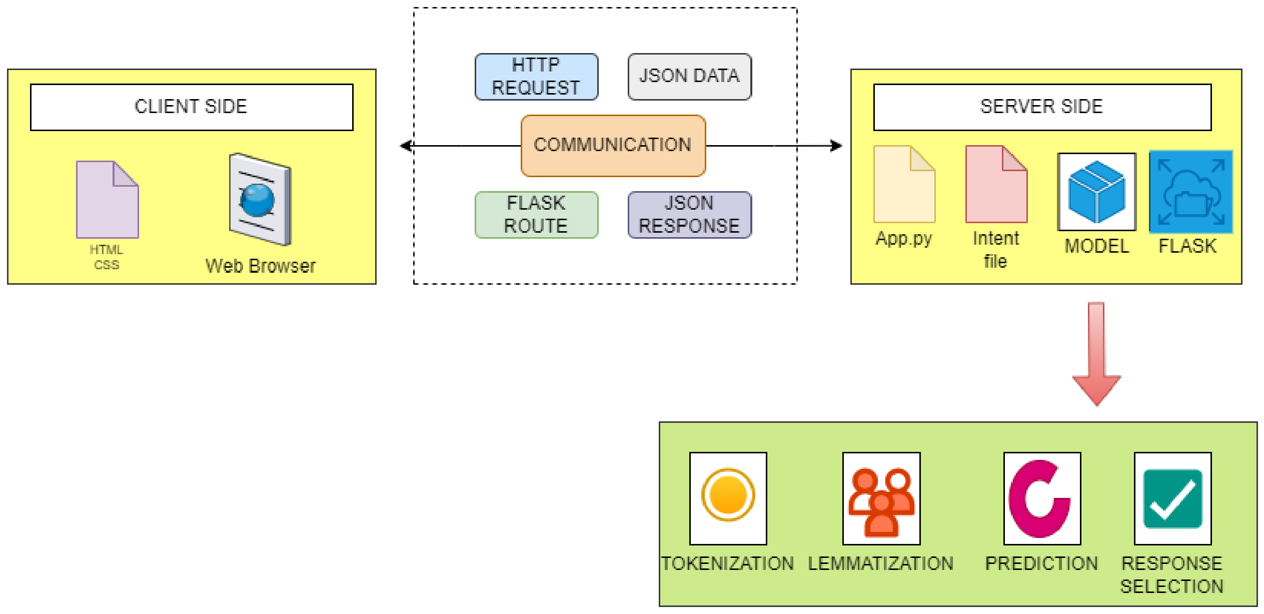


Fig 3.2: System Architecture

3.3 DEVELOPMENTAL ENVIRONMENT

3.3.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the system's implementation. It should therefore be a complete and consistent specification of the entire system. It is generally used by software engineers as the starting point for the system design.

Table 3.1 Hardware Requirements

COMPONENTS	SPECIFICATION
PROCESSOR	Intel Core i5
RAM	8 GB RAM
GPU	NVIDIA GeForce GTX 1650
MONITOR	15" COLOR
HARD DISK	512 GB
PROCESSOR SPEED	MINIMUM 1.1 GHz

3.3.2 SOFTWARE REQUIREMENTS

The software requirements document is the specifications of the system. It should include both a definition and a specification of requirements. It is aset of what the system should rather be doing than focus on how it should be done. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating the cost, planning team activities, performing tasks, tracking the team, and tracking the team's progressthroughout the development activity.

Python IDLE or Visual Studio and **Chrome** would all be required.

CHAPTER 4

PROJECT DESCRIPTION

4.1 METHODOLODGY

The first steps of the project involve problem definition and requirements collecting, with an emphasis on the target audience and the goal of the chatbot. At this stage, it's essential to comprehend the kinds of queries the chatbot should be able to respond to as well as the features it must have. Data collection and preparation are the next steps after the requirements are defined. In order to train the chatbot, this involves gathering data, such as sample conversations or FAQs. The text is then cleaned and standardized using preprocessing on the data. The nltk (Natural Language Toolkit) module is used for tokenization, lemmatization, and stop word removal during this preparation step. Following data pretreatment, the project moves on to model training. The preprocessed data is used to build and train a neural network model. Classifying user inputs into specified intents is the model's intended function. To do this, we build and train the model using the Keras framework. The model is stored for later use after it has been trained. Integrating the learned model into a web application is the next step. Flask is a lightweight web framework that is used to build the chatbot's web interface. The application's backend processes user inputs, applies the trained model to forecast the input's class, and produces relevant responses. The responses are obtained from a pre-established JSON file called intentions, which lists potential user intents along with the appropriate bot responses. The frontend is created with JavaScript, CSS, and HTML. It offers an easy-to-use interface for consumers to communicate with the chatbot. AJAX calls are used to send user messages to the backend, and the web page instantly displays the bot's responses. The user interface (UI) is designed to resemble social network chat apps, with user and bot messages encased in distinct colored boxes and aligned to the right.

4.2 MODULE DESCRIPTION

1. **Requirement Analysis**: Understand the objectives and requirements of the chatbot project. Determine the scope, functionalities, and target audience. Define the types of interactions the chatbot will support and the information it needs to provide.

2. **Data Collection and Annotation**: Gather relevant data for training the chatbot. This may include conversation logs, FAQs, or other textual data sources. Annotate the data with labels or intents to indicate the purpose or meaning of each message.

3. **Data Preprocessing**: Clean and preprocess the collected data to prepare it for training. This involves tasks like tokenization, lemmatization, removing stopwords, and converting text into numerical representations suitable for machine learning models.

4. **Model Selection and Training**: Choose an appropriate machine learning or deep learning model architecture for the chatbot. Train the selected model using the preprocessed data. Optimize hyperparameters and evaluate the model's performance using metrics like accuracy, precision, recall, and F1 score.

5. **Integration with Web Framework**: Develop the backend of the chatbot using a web framework like Flask or Django. Create API endpoints to handle user requests, predict intents using the trained model, and generate appropriate responses.

6. **Frontend Development**: Design and implement the user interface (UI) for the chatbot. Use HTML, CSS, and JavaScript to create an intuitive and user-friendly

interface where users can interact with the chatbot. Implement features like message input, message display, and dynamic updates.

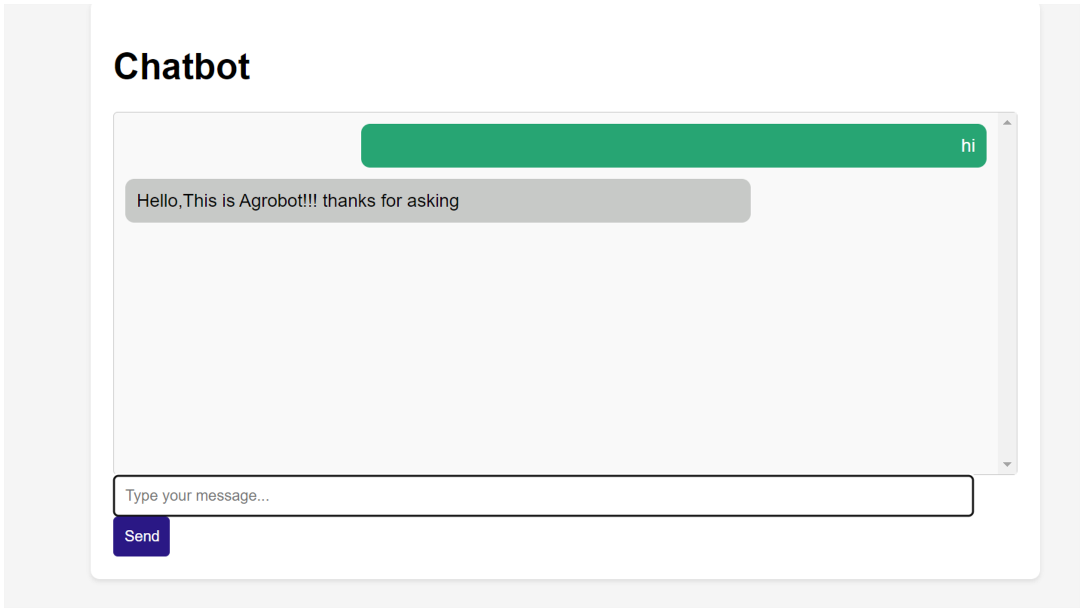
7. **Testing and Evaluation**: Conduct thorough testing of the chatbot system to ensure its functionality, usability, and reliability. Test for various scenarios, including normal interactions, edge cases, and error handling. Gather feedback from users and stakeholders to identify areas for improvement.

8. **Deployment and Maintenance**: Deploy the chatbot system to a production environment where it can be accessed by users. Monitor the system's performance, scalability, and security. Continuously update and improve the chatbot based on user feedback and changing requirements.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 OUTPUT

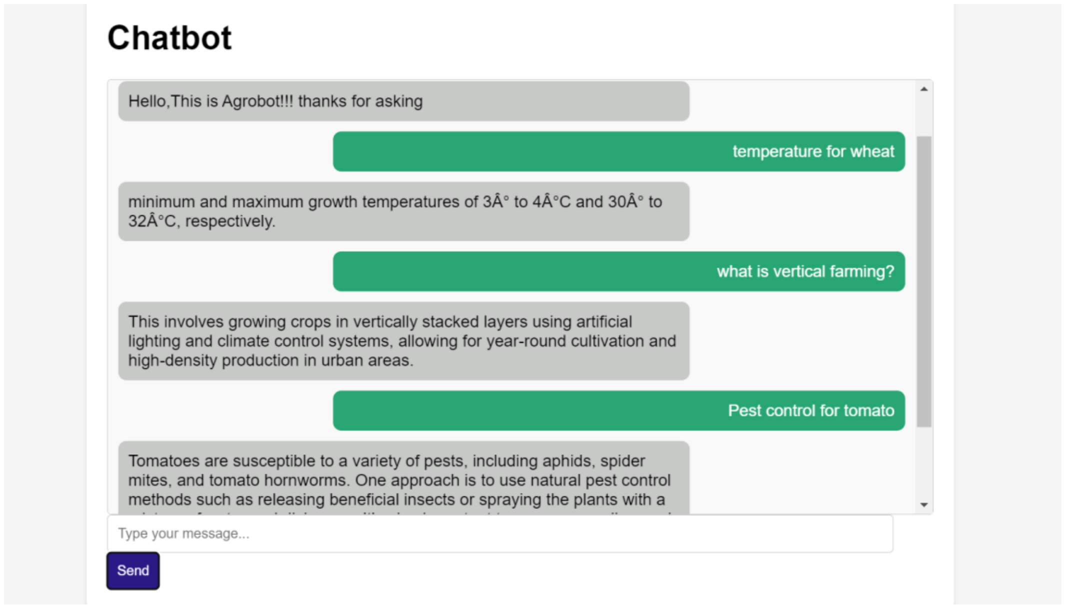


Chatbot

hi

Hello,This is Agrobot!!! thanks for asking

Fig 5.1: Chatbot ui



Chatbot

Hello,This is Agrobot!!! thanks for asking

temperature for wheat

minimum and maximum growth temperatures of 3Â° to 4Â°C and 30Â° to 32Â°C, respectively.

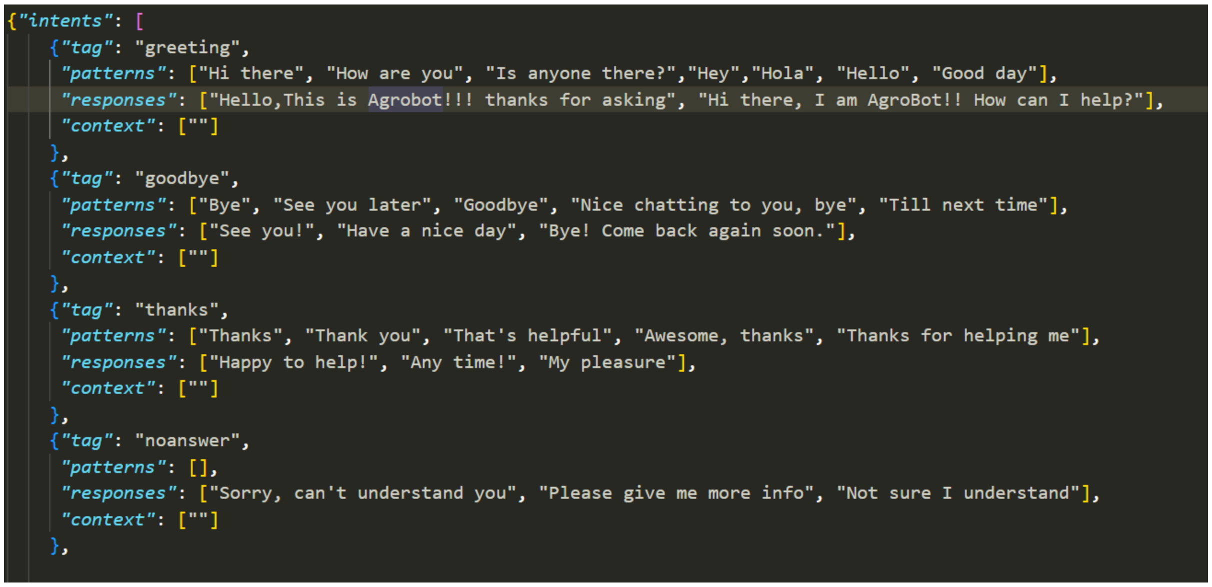
what is vertical farming?

This involves growing crops in vertically stacked layers using artificial lighting and climate control systems, allowing for year-round cultivation and high-density production in urban areas.

Pest control for tomato

Tomatoes are susceptible to a variety of pests, including aphids, spider mites, and tomato hornworms. One approach is to use natural pest control methods such as releasing beneficial insects or spraying the plants with a

Fig 5.2:Chatbot queries and answers



```
{"intents": [
    {"tag": "greeting",
     "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
     "responses": ["Hello,This is Agrobot!!! thanks for asking", "Hi there, I am AgroBot!! How can I help?"],
     "context": [""]
    },
    {"tag": "goodbye",
     "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
     "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
     "context": [""]
    },
    {"tag": "thanks",
     "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
     "responses": ["Happy to help!", "Any time!", "My pleasure"],
     "context": [""]
    },
    {"tag": "noanswer",
     "patterns": [],
     "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
     "context": [""]
    },
```

Fig 5.3:Data in JSON format

5.2 RESULT

After the chatbot web application has been developed, it is clear that the project has successfully met its goals. Users can communicate with the chatbot in a smooth and simple way thanks to the user interface. The design, which is reminiscent of well-known social media chat programs, increases user familiarity and engagement. Messages are easily entered by users and processed by the backend for response generation and classification.

Based on the trained model, the chatbot's functionality is strong, correctly reading user messages and delivering pertinent responses. Users receive relevant and useful information in answer to their inquiries by having their intents classified and suitable responses selected from a prepared intents JSON file.Scalability and maintainability of the project are guaranteed by its modular design. To accommodate upgrades or changes in the future, each module—including the frontend HTML/CSS/JavaScript interface and the backend Flask application—can be separately expanded or modified. Smooth scaling as user traffic grows and new features are added is made possible by this flexibility.

The chatbot performs well, handling user messages and producing responses with quickness. When faced with a range of user inputs, the trained model responds quickly and without noticeable lag. Performance, however, could differ based on outside variables like server load and user query complexity.To sum up, the web application for chatbots effectively offers an interactive interface that allows consumers to communicate with the chatbot. The application's strong capabilities, scalability, performance, and user-friendly design make it an invaluable tool for resolving user inquiries and improving overall user experience.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The development of the chatbot online application is an important step in the use of technology to assist farmers and those with an interest in agriculture. Through its simple design, the program enables access to agricultural counsel and information while bridging the gap between current technical solutions and traditional farming practices. Farmers now have easy access to insightful advice on a variety of topics, including crop cultivation, pest control, and more—especially those who work in distant or overlooked locations. By enabling farmers to make well-informed decisions, enhance farming methods, and raise crop yields, this accessibility eventually contributes to agricultural sustainability and food security.

In addition, the chatbot is a useful educational resource for people who want to learn more about farming and agriculture. Even those with no prior experience in agriculture can learn about all aspects of farming by providing a platform for interactive engagement with the chatbot. The open access of agricultural information creates opportunities for newcomers to the farming sector and sparks the curiosity and excitement of hobbyists and potential farmers alike. The potential for innovation and entrepreneurship in agriculture grows as more individuals use the chatbot to access information and services, supporting the growth and development of rural areas.The chatbot comprehends the intent of users, processes their inquiries quickly, and provides contextually relevant responses by utilizing machine learning models. The chatbot may continuously learn and enhance its performance over time, according to changing user needs and preferences, thanks to the application of sophisticated algorithms and methodologies.

6.2 FUTURE ENHANCEMENT

When considering potential updates and future possibilities for the chatbot online application, lots of routes come to mind. First off, adding more functions like crop disease detection, real-time weather updates, and tailored agricultural advice based on user-specific data might greatly increase the application's usefulness and worth. The chatbot may obtain real-time data streams by utilizing IoT devices and agricultural sensors. This would enable it to provide farmers with timely alerts and insights to enhance their farming practices and reduce hazards.

Improving the chatbot's natural language processing powers to provide multilingual capability will also increase its accessibility and reach, making it suitable for a variety of farming groups and geographical areas across the globe. Including a feedback feature in the software would help with constant enhancement by letting users know what they think about how accurate and relevant the responses are.Additionally, establishing partnerships with academic institutions, agribusinesses, and agricultural specialists could unlock doors to resources and knowledge that will improve the chatbot's functioning and content offers. Working together with stakeholders from each step along the value chain for farming would guarantee that the application stays up-to-date flexible, and open to changing to user demands and market trends.

APPENDIX

SOURCE CODE:

Agro.py

```
import nltk
from nltk.stem import WordNetLemmatizer
import numpy as np
import json
import pickle
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
from keras.preprocessing.sequence import pad_sequences
import random

# Download NLTK resources
nltk.download('punkt')
nltk.download('wordnet')

# Initialize WordNet lemmatizer
lemmatizer = WordNetLemmatizer()

# Load intents from JSON file
data_file = open('data.json').read()
intents = json.loads(data_file)

words = []
classes = []
documents = []
ignore_words = ['?', '!']

# Loop through each intent and its patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # Tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        # Add documents to the corpus
        documents.append((w, intent['tag']))
        # Add intent tag to the classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# Lemmatize words and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
classes = sorted(list(set(classes)))

# Save words and classes to pickle files
pickle.dump(words, open('texts.pkl', 'wb'))
pickle.dump(classes, open('labels.pkl', 'wb'))

# Create training data
training = []
output_empty = [0] * len(classes)

# Convert documents into bag of words format
for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])

# Shuffle and convert training data
random.shuffle(training)
max_length = len(training[0][0])  # Max length is the length of the bag of words

# Convert training list into NumPy arrays
train_x = pad_sequences([row[0] for row in training], maxlen=max_length,
padding='post')  # Pad sequences
train_y = np.array([row[1] for row in training])  # Extract labels

# Build and compile the model
model = Sequential()
model.add(Dense(128, input_shape=(max_length,), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
```

```
# Update optimizer arguments
sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# Train the model
hist = model.fit(train_x, train_y, epochs=200, batch_size=5, verbose=1)

# Save the model using the new format
model.save('model.keras')

print("Model created and saved successfully.")
```

app.py

```
from flask import Flask, render_template, request, jsonify
import numpy as np
import pickle
from keras.models import load_model
import json
import random
import nltk
from nltk.stem import WordNetLemmatizer

app = Flask(__name__)

# Load necessary files
model = load_model('model.keras')
words = pickle.load(open('texts.pkl', 'rb'))
classes = pickle.load(open('labels.pkl', 'rb'))
lemmatizer = WordNetLemmatizer()

# Load intents file
with open('data.json') as json_data:
    intents = json.load(json_data)

# Define the home route to render the chatbot interface
@app.route('/')
def home():
    return render_template('index.html')
```

```
# Define the chat route to handle user messages
@app.route('/chat', methods=['POST'])
def chat():
    msg = request.form['msg']
    if msg:
        # Process the user message and generate a response
        ints = predict_class(msg, model)
        res = get_response(ints, intents)
        return jsonify({'response': res})

# Function to predict the class
def predict_class(sentence, model):
    # Tokenize and lemmatize the sentence
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    # Create a bag of words
    bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                bag[i] = 1
    # Convert to NumPy array and predict
    res = model.predict(np.array([bag]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
    # Sort by probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

# Function to get the response
def get_response(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if i['tag'] == tag:
            return random.choice(i['responses'])

if __name__ == '__main__':
    app.run(debug=True)
```

REFERENCES

[1] A. Miklosik, N. Evans and A. M. A. Qureshi, "The Use of Chatbots in Digital Business Transformation: A Systematic Literature Review," in IEEE Access, vol. 9, pp. 106530-106539, 2021, doi: 10.1109/ACCESS.2021.3100885.

[2]G. A. Santos, G. G. de Andrade, G. R. S. Silva, F. C. M. Duarte, J. P. J. D. Costa and R. T. de Sousa, "A Conversation-Driven Approach for Chatbot Management," in IEEE Access, vol. 10, pp. 8474-8486, 2022, doi: 10.1109/ACCESS.2022.3143323.

[3] R. Ren, M. Zapata, J. W. Castro, O. Dieste and S. T. Acuña, "Experimentation for Chatbot Usability Evaluation: A Secondary Study," in IEEE Access, vol. 10, pp. 12430-12464, 2022, doi: 10.1109/ACCESS.2022.3145323.

[4] G. Attigeri, A. Agrawal and S. V. Kolekar, "Advanced NLP Models for Technical University Information Chatbots: Development and Comparative Analysis," in IEEE Access, vol. 12, pp. 29633-29647, 2024, doi: 10.1109/ACCESS.2024.3368382.

[5]P. K. Maduri, P. Dhiman, M. R. Shukla, S. Anand and S. P. Singh, "Farmers Agriculture Assistance Chatbot," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 1884-1889, doi: 10.1109/ICAC3N53548.2021.9725634.