DOCUMENTATION OF METANONG

Prepared By

Joeren B. Soriano

Eldred M. Palanas

June 2024

TABLE OF CONTENTS

Introduction

Machine learning (ML) is a powerful technique used in artificial intelligence (AI) systems. Unlike traditional programming where explicit instructions are coded, machine learning allows AI to learn from data and make predictions or decisions on its own. This is achieved by training a model on a dataset containing input data and corresponding desired outputs. The model analyzes patterns and relationships within the data, enabling it to make predictions on new, unseen data.

A Python-based chatbot for Mets logistics improves inventory management and operational efficiency for both cold and dry storage. It facilitates seamless communication between warehouse staff and the logistics system, offering real-time help with tracking inventory, monitoring storage conditions, and processing orders. In cold storage, it ensures optimal temperature control by alerting staff to any issues, preventing spoilage. In dry storage, it helps organize and locate items quickly. The chatbot also generates reports on stock levels, expiration dates, and order statuses, aiding managers in making informed decisions. With natural language processing, it provides an intuitive user experience through text or voice commands, enhancing productivity and accuracy in warehouse operations.

The purpose of METanong user manual is to guide users on how to effectively use the chatbot. It provides clear instructions on interacting with the chatbot, including starting conversations, entering commands, and receiving responses. The manual also includes troubleshooting tips, FAQs, and best practices to ensure users can easily manage tasks like inventory and order processing. This helps users quickly learn and efficiently use the chatbot, enhancing their overall experience.

Product Overview

The METanong Chatbot is a Python-based solution designed to streamline inventory management and operational processes for both cold and dry storage facilities. This intelligent chatbot provides real-time assistance to warehouse staff, facilitating efficient communication and task execution. Key features include:

Order Processing: Simplifies the processing of orders, updating status, and managing shipments.

Report Generation: Automatically sends detailed reports on inventory levels, expiration dates, and order fulfillment.

User-Friendly Interface: Accessible via text commands, the chatbot leverages natural language processing to offer an intuitive and seamless user experience.

Voice Capability: The chatbot can interact via voice commands, leveraging natural language processing for an intuitive user experience.

Safety Information

To ensure METanong operates securely and effectively in Mets logistics, follow these key practices: protect sensitive data with strong encryption, control access through role-based permissions, train users on proper usage and issue escalation, regularly update software for security, and monitor interactions to promptly detect and address any anomalies. These steps help maintain the integrity and safety of your chatbot system.

Protect Your Data: Ensure sensitive information is securely handled and transmitted using strong encryption methods.

Control Access: Use role-based access controls to restrict who can use and manage the chatbot to authorized personnel only.

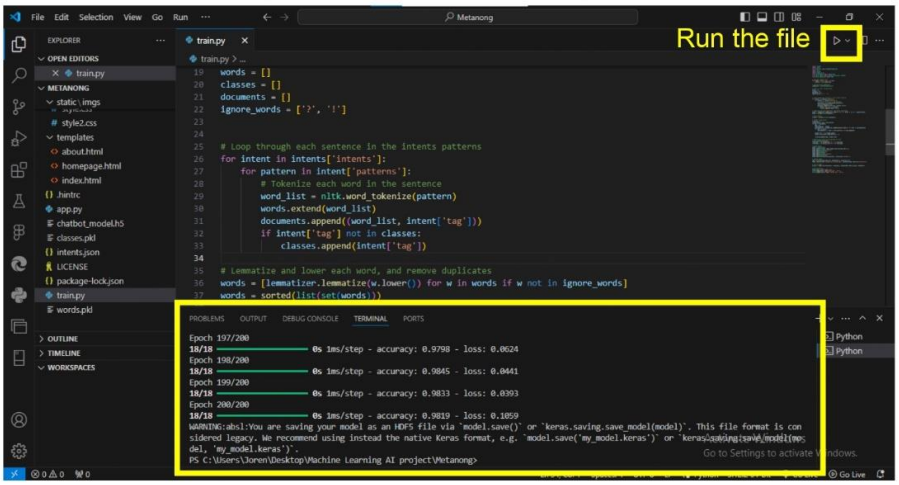
Stay Informed: Train users on how to use the chatbot properly and escalate issues when needed.

Keep Updated: Regularly update the chatbot software and security measures to prevent vulnerabilities.

Monitor Activity: Monitor chatbot interactions and audit regularly to detect and respond to any unusual activities promptly.

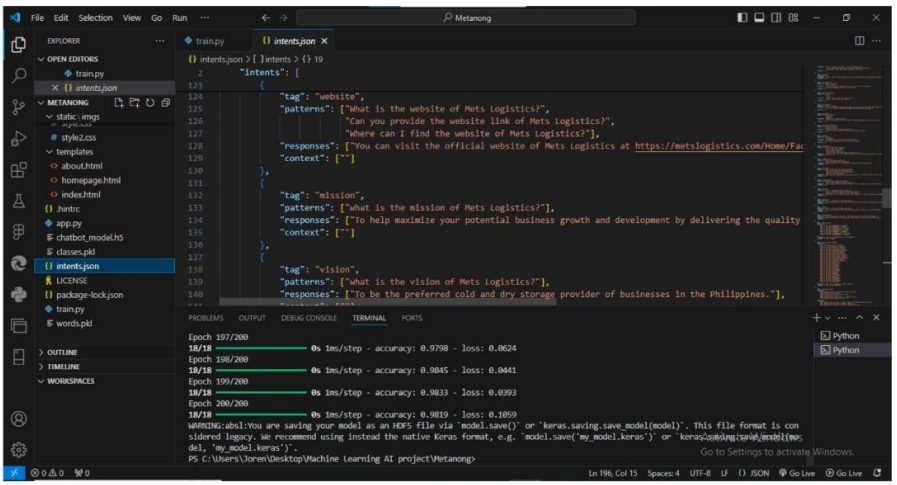
Getting Started

Installation Guide



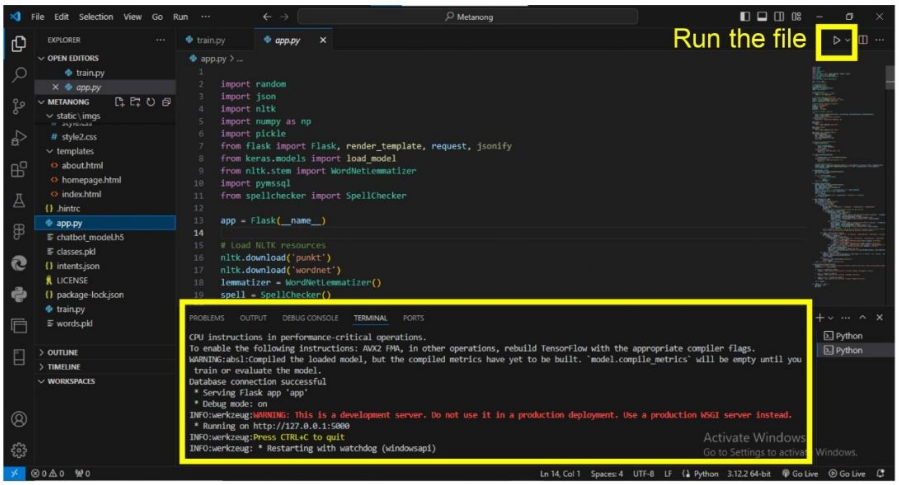
Run train.py: Begin by running train.py to initiate the training process for the chatbot.

Prepare for train.py Installation: Ensure all dependencies required by train.py are installed and up to date.



Train intent.json: train.py will train the chatbot using the intent.json file, which contains predefined intents and responses.

Wait for Training Completion: Allow train.py to complete the training of intent.json. This process may take some time depending on the size of the dataset and complexity of intents.



Run app.py: Once training is finished, execute app.py to launch the chatbot application.

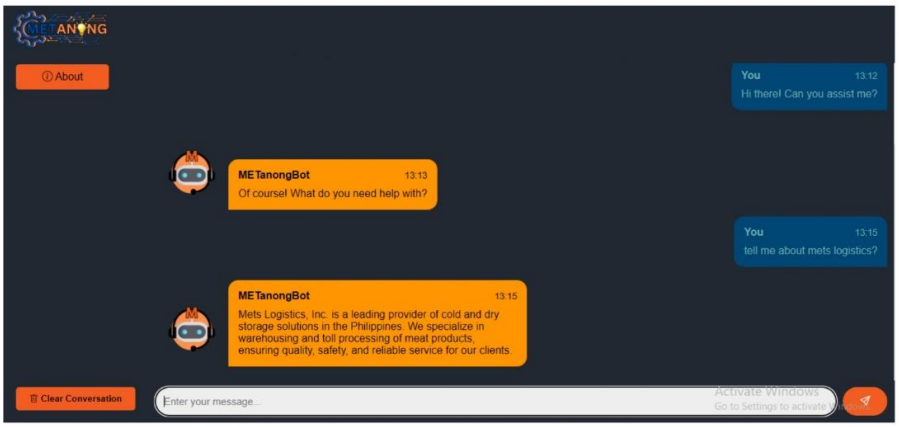
Access the Application: app.py will provide a link or URL that you can use to access the chatbot via a website interface.

Follow these steps sequentially to successfully install and deploy the chatbot application. Ensure you have Python and Visual Studio Code installed along with necessary libraries as specified by train.py and app.py dependencies.

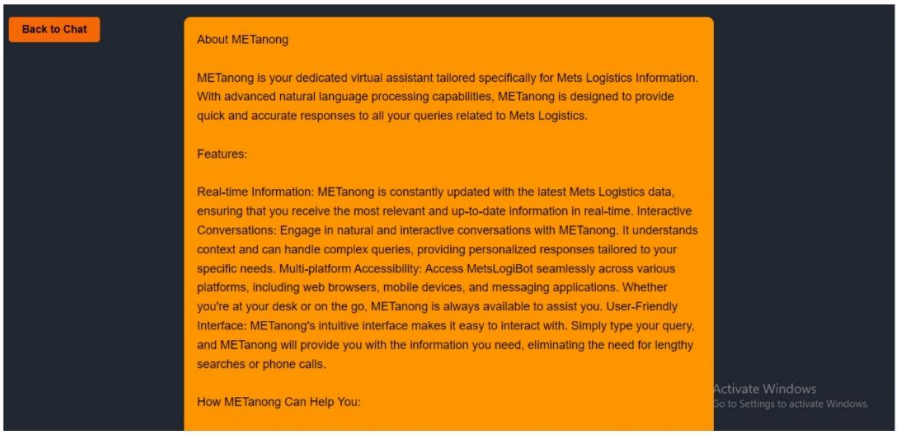
How to Use Metanong

1. Once the website loads, the chat interface will be visible on the homepage.
2. Click on the chatbox at the bottom of the screen to activate it.

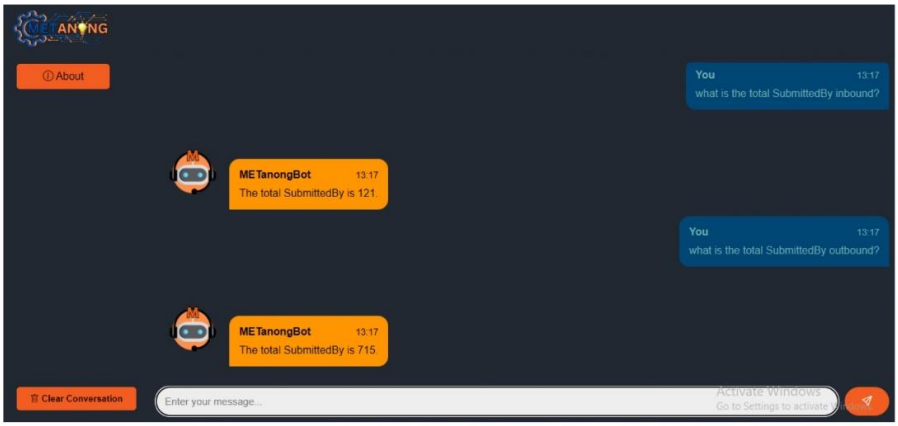
3. Type a message or question in the chatbox.



4. Press Enter or click the "Send" button to submit the query.



5. METanong will respond instantly with relevant information or actions.



List Of Questions That Can Be Asked

Name Inquiry

- What's your name?
- Who are you?

About Robot

- Who are you?

Assistance Preference

- Can you help me?
- Need assistance

Company Info

- Tell me about METS Logistics
- Tell me about this company
- What is mets logistics?
- What does METS Logistics do?
- Can you provide information about METS Logistics?

Founder Info

- Who founded METS Logistics?
- Who is the CEO of METS Logistics?

Offerings

- What services does METS Logistics offer?
- What are the products of METS Logistics?
- Does METS Logistics have its own products?
- Service of METS logistics?

Location Info

- Where is METS Logistics located?
- What are the locations of METS Logistics?

Website

- What is the website of METS Logistics?

- Can you provide the website link of METS Logistics?
- Where can I find the website of METS Logistics?

Mission

- What is the mission of METS Logistics?

Vision

- What is the vision of METS Logistics?

Accreditation

- Is METS Logistics accredited?
- Does METS Logistics have any accreditations?

Total Storage

- What is the total STORAGECOLD in inbound?
- What is the total STORAGEDRY in inbound?
- What is the total STORAGECHILLER in inbound?
- What is the total STORAGEAIRCON in inbound?
- What is the total STORAGECOLD in outbound?
- What is the total STORAGEDRY in outbound?
- What is the total STORAGECHILLER in outbound?
- What is the total STORAGEAIRCON in outbound?

Total Inbound Outbound

- What is the total Arrival inbound?
- What is the total Departure inbound?
- What is the total StartUnload inbound?
- What is the total ICNTotalQty inbound?
- What is the total CompleteUnload inbound?
- What is the total StorageType inbound?
- What is the total Supplier inbound?
- What is the total ArrivalTime outbound?
- What is the total DepartureTime outbound?
- What is the total StartLoading outbound?
- What is the total CompleteLoading outbound?

- What is the total TotalQty outbound?
- What is the total StorageType outbound?
- What is the total Supplier?
- What is the total SubmittedBy?
- What is the total PostedBy?
- What is the total ApprovedBy?
- What is the total RejectBy?
- What is the total CheckedBy?
- What is the total CancelledBy?
- What is the total DocumentBy?
- What is the total PutAwayBy?
- What is the total AddedBy?
- What is the total LastEditedBy?
- What is the total AcceptBy?
- What is the total RFPutAwayBy?
- What is the total CheckedBy?
- What is the total AuthorizedBy?
- What is the total SuppliedBy?

Facility Location

- Where are Mets Logistics facilities located in Cavite?
- Where specifically in Cavite?
- Where are Mets Logistics facilities in Cebu?
- Where specifically in Cebu?
- Where are Mets Logistics facilities in Cagayan de Oro?
- Where specifically in Cagayan de Oro?

Code Documentation

Front-end

Index.html

```
<html lang="en">


<head>

  <meta charset="UTF-8">

  <title>METanong</title>

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='imgs/style.css') }}">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

  <link rel="icon" href="{{ url_for('static', filename='imgs/icon1.ico') }}" type="image/x-
icon">

</head>


<body>


  <aside class="sidebar">

    <a href="{{ url_for('about') }}" class="show-about-btn">

      <i class="bi bi-info-circle"></i> About

    </a>
```

```
  <button class="clear-conversation-btn" onclick="clearConversation()">
    <i class="bi bi-trash"></i> Clear Conversation
  </button>
</aside>

<header class="msger-header">
  <div class="logo-header">
    <a href="img-logo"><img src="/static/imgs/logo1.png"></a>
  </div>
</header>

<section class="msger">
  <main class="msger-chat">
    <div class="msg left-msg">
      <div class="msg-img">
        <img class="img-logo" src="{{ url_for('static', filename='imgs/chatbot.png') }}">
      </div>
      <div class="msg-bubble">
        <div class="msg-info">
          <div class="msg-info-name">METanongBot
          </div>
          <div class="msg-info-time"></div>
        </div>
        <div class="msg-text">
```

Hi, I'm Metanong, your personal assistant for Mets Logistics Data Exploration. Send your message. 😄

```
          </div>
        </div>
      </div>
    </main>

    <form class="msger-inputarea" id="messageForm">
      <input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">
      <button type="submit" class="msger-send-btn"><i class="bi bi-send"></i></button>
    </form>
  </section>

  <script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
  <script>
    const msgerForm = document.getElementById("messageForm");
    const msgerInput = document.getElementById("textInput");
    const msgerChat = document.querySelector(".msger-chat");

    msgerForm.addEventListener("submit", event => {
      event.preventDefault();
      const msgText = msgerInput.value.trim();
      if (!msgText) return;
```

```
    appendMessage("You", "right", msgText);
    msgerInput.value = "";
    botResponse(msgText);
  });
  msgerInput.addEventListener("keydown", event => {
    if (event.key === "Enter") {
      event.preventDefault();
      msgerForm.dispatchEvent(new Event("submit"));
    }
  });
  function appendMessage(name, side, text) {
    const msgHTML = `
      <div class="msg ${side}-msg">
        ${side === 'left' ? '<div class="msg-img"><img class="img-logo" src="{{
url_for('static', filename='imgs/chatbot.png') }}"></div>': ''}
        <div class="msg-bubble">
          <div class="msg-info">
            <div class="msg-info-name">${name}</div>
            <div class="msg-info-time">${formatDate(new Date())}</div>
          </div>
          <div class="msg-text">${text}</div>
        </div>
    `;
    msgerChat.insertAdjacentHTML("beforeend", msgHTML);
    msgerChat.scrollTop += 500;
```

```
    }
    function speakText(text) {
      const speech = new SpeechSynthesisUtterance();
      speech.text = text;
      window.speechSynthesis.speak(speech);
    }
    function botResponse(rawText) {
      $.post("/get", { msg: rawText }).done(function (data) {
        const msgText = data;
        appendMessage("METanongBot", "left", msgText);
        speakText(msgText);
      });
    }
    function formatDate(date) {
      const h = "0" + date.getHours();
      const m = "0" + date.getMinutes();
      return `${h.slice(-2)}:${m.slice(-2)}`;
    }
    function clearConversation() {
      location.reload();
    }
  </script>
</body>
</html>
```

style.css

```
:root {
  --body-bg: linear-gradient(135deg, #222831 0%, #222831 100%);
  --msger-bg: #222831;
  --border: #222831;
  --left-msg-bg: #ff9500;
  --right-msg-bg: #004776;
}

/* Box sizing */
html {
  box-sizing: border-box;
}

*, *:before, *:after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

/* Sidebar styles */
.sidebar {
  margin-top: 69px;
  position: fixed;
  left: 0;
  top: 0;
  bottom: 0;
  width: calc(200px + 20px);
  background-color: #222831;
  color: #fff;
  padding: 20px;
```

```
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  z-index: 2;
}

.sidebar ul {
  list-style: none;
  padding: 0;
}

.sidebar ul li {
  margin-bottom: 10px;
}

.sidebar ul li a {
  color: #fff;
  text-decoration: none;
  font-size: 16px;
}

.sidebar ul li a:hover {
  color: #ccc;
}

/* About button styles */
.show-about-btn {
  background-color: rgb(243, 93, 33);
  color: #000;
  font-weight:400;
```

```
  cursor: pointer;
  border: none;
  border-radius: 5px;
  padding: 10px;
  text-align: center;
  transition: background 0.3s;
  width: calc(100% - 40px);
  margin-bottom: 10px;
  position: relative;
  z-index: 1;
  text-decoration: none;
}

.show-about-btn:hover {
  background-color: #ff7d03;
}

/* Conversation interface styles */
.msger {
  position: relative;
  margin-left: calc(200px + 20px);
  margin-top: 15px;
  display: flex;
  flex-flow: column wrap;
  justify-content: space-between;
  width: 84%;
  height: 90%;
  max-height: calc(100% - 50px);
  border: var(--border);
  border-radius: 5px;
```

```
  background: var(--msger-bg);
  box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}

@media (max-width: 768px) {
  .msger {
    margin-left: 10px;
    width: 96%;
  }
}

.msger-header {
  padding: .5vh;
  border-bottom: var(--border);
  background: #222831;
  color: #76ABAE;
}

.msger-chat {
  flex: 1;
  overflow-y: auto;
  padding: 10px;
}

.msger-chat::-webkit-scrollbar {
  width: 10px;
}

.msger-chat::-webkit-scrollbar-track {
  background: #ddd;
```

```
}

.msger-chat::-webkit-scrollbar-thumb {
  background: #bdbdbd;
}

.msg {
  display: flex;
  align-items: flex-end;
  margin-bottom: 10px;
}

.msg-img img {
  padding-top: 30px;
  display: flex;
  max-width: 110px;
  height: auto;
  border-color: #0066ff;
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  border-radius: 50%;
}

.msg-bubble {
  max-width: 450px;
  padding: 15px;
  border-radius: 15px;
  background: var(--left-msg-bg);
}
```

```
.msg-info {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 10px;
}

.msg-info-name {
  margin-right: 10px;
  font-weight: bold;
}

.msg-info-time {
  font-size: 0.85em;
}

.left-msg .msg-bubble {
  border-bottom-left-radius: 0;
}

.right-msg {
  flex-direction: row-reverse;
}

.right-msg .msg-bubble {
  background: var(--right-msg-bg);
  color: #76ABAE;
  border-bottom-right-radius: 0;
}
```

```
.right-msg .msg-img {
  margin: 0 0 0 10px;
}

/* Message input area styles */
.msger-inputarea {
  display: flex;
  padding: 10px;
  border-top: var(--border);
  background: #222831;
  border-bottom-right-radius: 5px;
  border-bottom-left-radius: 5px;
  margin-right: 10px;
  margin-bottom: 13px;
}

.msger-inputarea * {
  font-size: 15px;
  padding: 13px;
  border: none;
  border-radius: 50px;
}

.msger-input {
  margin-right: 10px;
  flex: 1;
  background: #EEEEEE;
}
```

```
.msger-send-btn {
  background: rgb(243, 93, 33);
  color: #fff;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.23s;
}

.msger-send-btn:hover {
  background: rgb(255, 125, 3);
}

.msger-chat {
  background-color: #222831;
  border-radius: 5px;
}

/* Responsive styles */
@media (max-width: 768px) {
  .sidebar {
    display: none;
  }

  .clear-conversation-btn,
  .show-about-btn {
    display: none;
  }

  .sidebar.open {
    transform: translateX(0);
```

```
  }
}

/* Logo header styles */
.logo-header {
  margin-left: 10px;
  display: flex;
  justify-content: left;
  align-items: center;
  max-width: 150px;
}

.logo-header img {
  max-width: 100%;
  height: 80%;
}

/* Body styles */
body {
  justify-content: center;
  align-items: center;
  height: 95vh;
  background-image: var(--body-bg);
  font-family: Helvetica, sans-serif;
}

.clear-conversation-btn {
  background-color: rgb(243, 93, 33);
  color: #000;
  font-weight: bold;
```

```
  cursor: pointer;
  border: none;
  border-radius: 5px;
  padding: 10px;
  text-align: center;
  transition: background 0.3s;
  width: calc(100% - 40px);
  position: absolute;
  bottom: 20px;
  z-index: 1;
}

.clear-conversation-btn:hover {
  background-color: #ff7d03;
}
```

About.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>About METanong</title>
  <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='imgs/style2.css') }}">
  <link rel="icon" href="{{ url_for('static', filename='imgs/icon.ico') }}" type="image/x-
icon">
  <link rel="icon" href="{{ url_for('static', filename='imgs/icon1.ico') }}" type="image/x-
icon">
</head>

<body>
  <div class="about-container">
    <a href="/" class="back-btn">Back to Chat</a>
    <div class="bubble">
```

About METanong

</br>

METanong is your dedicated virtual assistant tailored specifically for Mets Logistics Information. With advanced natural language processing capabilities, METanong is designed to provide quick and accurate responses to all your queries related to Mets Logistics.

</br>

Features:

</br>

Real-time Information: METanong is constantly updated with the latest Mets Logistics data, ensuring that you receive the most relevant and up-to-date information in real-time.

Interactive Conversations: Engage in natural and interactive conversations with METanong. It understands context and can handle complex queries, providing personalized responses tailored to your specific needs.

Multi-platform Accessibility: Access MetsLogiBot seamlessly across various platforms, including web browsers, mobile devices, and messaging applications. Whether you're at your desk or on the go, METanong is always available to assist you.

User-Friendly Interface: METanong's intuitive interface makes it easy to interact with. Simply type your query, and METanong will provide you with the information you need, eliminating the need for lengthy searches or phone calls.

</br>

How METanong Can Help You:

</br>

Address FAQs: Have questions about Mets Logistics policies, services, or procedures? METanong is here to help! It can address frequently asked questions and provide you with the information you need.

Experience the Future of Logistics with METanong. Start Chatting Today!

</body>

</html>

style2.css

```
/* Box sizing */
html {
  box-sizing: border-box;
}

*, *:before, *:after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

/* Body styles */
body {
  font-family: Helvetica, sans-serif;
  background-color: #222831;
  color: #fff;
  padding: 20px;
}

.about-container {
  max-width: 800px;
  margin: 0 auto;
  text-align: center;
}

.about-heading {
  font-size: 24px;
  font-weight: bold;
  color: #ff9500;
  margin-bottom: 20px;
```

```
}

.about-description {
  font-size: 18px;
  line-height: 1.6;
  color: #000;
  margin-bottom: 20px;
}

/* Bubble message styles */
.bubble {
  background-color: #ff9500;
  color: #000;
  padding: 20px;
  border-radius: 10px;
  display: inline-block;
  text-align: left;
}

.back-btn {
  background-color: #f76809;
  color: #000000;
  font-weight: bold;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s, color 0.3s;
  text-decoration: none;
  position: absolute;
```

```
  top: 10px;
  left: 10px;
  margin-top: 10px;
}

@media screen and (max-width: 768px) {
  .back-btn {
    display: none;
  }
}

.back-btn:hover {
  background-color: #ff7d03;
}
```

Back-end

App.py

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

app = Flask(__name__)

# Load NLTK resources
```

```
nltk.download('punkt')
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
spell = SpellChecker()

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

# Load Keras model and pickle files
words = pickle.load(open("words.pkl", "rb"))
classes = pickle.load(open("classes.pkl", "rb"))
model = load_model("chatbot_model.h5")

# Define database connection parameters
server = '192.168.180.22'
database = 'WMS-OJT'
username = 'OJT'
password = '123!@#qwe'

# Establish database connection
try:
    conn = pymssql.connect(server=server, user=username, password=password,
database=database)
    cursor = conn.cursor()
    print("Database connection successful")
except Exception as e:
    print("Error connecting to database:", e)

@app.route("/")
def home():
```

```
    return render_template("index.html")

@app.route("/about")
def about():
    return render_template("about.html")

@app.route("/get", methods=["POST"])
def chatbot_response():
    msg = request.form["msg"]
    ints = predict_class(msg)
    res = get_response(ints, msg)
    return res

# Function to execute database query
def execute_query(query):
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Exception as e:
        print("Error executing query:", e)
        return None

def clean_up_sentence(sentence):
    try:
        sentence_words = nltk.word_tokenize(sentence)
    except Exception as e:
        print("Error tokenizing sentence:", e)
        return []
```

```
    # Correct spelling errors
    sentence_words = [spell.correction(word.lower()) if word.islower() else word for
word in sentence_words]
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words if word.islower()]
    return sentence_words


def bow(sentence, words):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)


    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                bag[i] = 1
    return np.array(bag)


def predict_class(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = bow(sentence, words)
    res = model.predict(np.array([bag]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
    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


def get_response(ints, msg):
    tag = ints[0]["intent"]
```

```
    list_of_intents = intents["intents"]

    for intent in list_of_intents:

        if intent["tag"] == tag:

            if tag == "total_storage":

                storage_types = ["STORAGECOLD", "STORAGEDRY",
"STORAGECHILLER", "STORAGEAIRCON"]

                response = ""

                # Retrieve counts for the specified storage type and direction

                for storage_type in storage_types:

                    if storage_type.lower() in msg.lower():

                        if "outbound" in msg.lower():

                            query_outbound = f"SELECT COUNT(*) FROM WMS.Outbound
WHERE StorageType = '{storage_type}'"

                            result_outbound = execute_query(query_outbound)

                            count_outbound = result_outbound[0][0] if result_outbound else 0

                            response = f"For {storage_type} storage:\nOutbound count:
{count_outbound}"

                            return response

                        elif "inbound" in msg.lower():

                            query_inbound = f"SELECT COUNT(*) FROM WMS.Inbound
WHERE StorageType = '{storage_type}'"

                            result_inbound = execute_query(query_inbound)

                            count_inbound = result_inbound[0][0] if result_inbound else 0

                            response = f"For {storage_type} storage:\nInbound count:
{count_inbound}"

                            return response

                # If no specific storage type is mentioned or direction is not clear, return a
generic response

                return "Please specify both the storage type and direction
(inbound/outbound)."


            elif tag == "total_inbound_outbound":
```

```
            categories = ["Arrival", "Departure", "StartUnload", "CompleteUnload",
"ICNTotalQty", "ArrivalTime", "DepartureTime", "StartLoading", "CompleteLoading",
"TotalQty", "StorageType", "Supplier", "SubmittedBy", "PostedBy", "ApprovedBy",
"RejectBy", "CheckedBy", "CancelledBy", "DocumentBy", "PutAwayBy", "AddedBy",
"LastEditedBy", "AcceptBy", "RFPutAwayBy", "AuthorizedBy", "SuppliedBy",
"RFCheckBy"]

            for category in categories:

                if category.lower() in msg.lower():

                    if category in ["Arrival", "Departure", "StartUnload",
"CompleteUnload", "ICNTotalQty", "StorageType", "Supplier", "SubmittedBy",
"PostedBy", "ApprovedBy", "RejectBy", "CheckedBy", "CancelledBy", "DocumentBy",
"PutAwayBy", "AddedBy", "LastEditedBy", "AcceptBy", "RFPutAwayBy",
"AuthorizedBy"]:

                        if "outbound" in msg.lower():

                            query = f"SELECT COUNT(*) FROM WMS.Outbound WHERE
{category} IS NOT NULL"

                        elif "inbound" in msg.lower():

                            query = f"SELECT COUNT(*) FROM WMS.Inbound WHERE
{category} IS NOT NULL"

                    elif category in ["ArrivalTime", "DepartureTime", "StartLoading",
"CompleteLoading", "TotalQty", "StorageType", "AddedBy", "LastEditedBy",
"SubmittedBy", "PostedBy", "RejectBy", "RFCheckBy", "CheckedBy", "SuppliedBy",
"CancelledBy"]:

                        query = f"SELECT COUNT(*) FROM WMS.Outbound WHERE
{category} IS NOT NULL"

                    result = execute_query(query)

                    total_count = result[0][0] if result else "N/A"

                    response =
random.choice(intent["responses"]).format(CATEGORY=category,
TOTAL_COUNT=total_count)

                    return response

        elif tag == "facility_location":

            location = [word for word in msg.split() if word.lower() not in ["where", "is",
"located", "specifically", "in"]]

            response = get_facility_location(location)

            return response

        else:
```

```
            response = random.choice(intent["responses"])
            return response
    return "I'm sorry, I didn't understand that."

def get_facility_location(location):
    response = "I'm sorry, I couldn't find any information related to your query. Please provide specific questions related to Mets Logistics."
    location_str = " ".join(location)

    if "cavite" in location_str.lower():
        response = "Mets Logistics has facilities in Carmona, Banahaw, and Maguyam in Cavite."

    if "cebu" in location_str.lower():
        response = "Mets Logistics has facilities in Mandaue, Cebu City."

    if "cagayan de oro" in location_str.lower():
        response = "Mets Logistics has facilities in Tablon, Cagayan de Oro City."

    return response

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

train.py

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

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

# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Lists to store training data
words = []
classes = []
documents = []
ignore_words = ['?', '!']

# Loop through each sentence in the intents patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # Tokenize each word in the sentence
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# Lemmatize and 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)))

# Create training data
training = []
output_empty = [0] * len(classes)
for doc in documents:
    bag = []
    word_patterns = doc[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
    for word in words:
        bag.append(1) if word in word_patterns 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 to array
random.shuffle(training)
training_x = np.array([i[0] for i in training])
training_y = np.array([i[1] for i in training])

# Create model
model = Sequential()
model.add(Dense(128, input_shape=(len(training_x[0]),)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
```

```
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(len(training_y[0]), activation='softmax'))

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

# Fit the model
history = model.fit(training_x, training_y, epochs=200, batch_size=5, verbose=1)

# Save artifacts
model.save('chatbot_model.h5')
pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))
```

intents.json

```
{
   "intents": [
      {
         "tag": "greetings",
         "patterns": ["hi there",
                     "hello",
                     "hey",
                     "holla",
                     "howdy"],
         "responses": ["Hello! How can I assist you today?",
                      "Hi there! How can I help you?",
                      "Hey! What can I do for you?"],
         "context": [""]
```

```
},
{
    "tag": "goodbye",
    "patterns": ["bye",
                 "see you later",
                 "farewell"],
    "responses": ["Goodbye! Have a great day!",
                  "See you later! Take care!",
                  "Farewell! Come back anytime!"],
    "context": [""]
},
{
    "tag": "thanks",
    "patterns": ["thanks",
                 "thank you",
                 "appreciate it"],
    "responses": ["You're welcome! Let me know if you need anything else.",
                  "Glad I could assist! Feel free to ask if you have more questions.",
                  "No problem! If you need further assistance, just ask!"],
    "context": [""]
},
{
    "tag": "noanswer",
    "patterns": [""],
    "responses": ["Sorry, I didn't understand that. Could you please provide more
information related to Mets Logistics?"],
    "context": [""]
},
{
    "tag": "name_inquiry",
    "patterns": ["what's your name?",
```

```
                "who are you?"],
        "responses": ["I'm an AI chatbot designed to assist you. You can call me
METanong.",
                "My name is METanong. How can I assist you?"],
        "context": [""]
    },
    {
        "tag": "name_declaration",
        "patterns": ["my name is",
                "I'm",
                "I am"],
        "responses": ["Nice to meet you! How can I assist you?",
                "Hello, How can I help you today?"],
        "context": [""]
    },
    {
        "tag": "about_robot",
        "patterns": ["what are you?"],
        "responses": ["I'm an artificial intelligence designed to assist with various
tasks and provide information.",
                "I'm an AI chatbot created to help users with their queries and
tasks."],
        "context": [""]
    },
    {
        "tag": "compliment",
        "patterns": ["you're great",
                "awesome",
                "amazing"],
        "responses": ["Thank you! I'm here to help.",
                "That's very kind of you! Let me know if there's anything else I can
do for you."],
```

```
    "context": [""]
},
{
    "tag": "assistance_preference",
    "patterns": ["can you help me?",
                 "need assistance"],
    "responses": ["Of course! What do you need help with?",
                  "Sure! I'm here to assist you. What do you need help with?"],
    "context": [""]
},
{
    "tag": "apology",
    "patterns": ["sorry",
                 "apologies"],
    "responses": ["No problem! Let me know how I can assist you.",
                  "No need to apologize! How can I assist you today?"],
    "context": [""]
},
{
    "tag": "company_info",
    "patterns": ["Tell me about Mets Logistics",
                 " Tell me about this company",
                 "What is mets logistics?",
                 "what does Mets Logistics do?",
                 "Can you provide information about Mets Logistics?"],
    "responses": ["Mets Logistics, Inc. is a leading provider of cold and dry
storage solutions in the Philippines. We specialize in warehousing and toll
processing of meat products, ensuring quality, safety, and reliable service for our
clients."],
    "context": [""]
},
```

```
{
    "tag": "founder_info",
    "patterns": ["who founded Mets Logistics?",
                "who is the CEO of Mets Logistics?"],
    "responses": ["Mets Logistics inc. was founded by Mr. Eduard Tio in the year 2010."],
    "context": [""]
},
{
    "tag": "offerings",
    "patterns": ["what services does Mets Logistics offer?",
                "what are the products of Mets Logistics?",
                "Does Mets Logistics have its own products?",
                "service of mets logistics?"],
    "responses": ["Mets Logistics specializes in providing Cold and Dry storage facilities, Blast Freezing, and Toll Processing services. While we do not have personal products, we offer comprehensive solutions for storage and processing needs."],
    "context": [""]
},
{
    "tag": "location_info",
    "patterns": ["where is Mets Logistics located?",
                "what are the locations of Mets Logistics?"],
    "responses": ["Mets Logistics has headquarters in Cavite, Cebu, and Cagayan De Oro, with multiple facilities in each region."],
    "context": [""]
},
{
    "tag": "website",
    "patterns": ["What is the website of Mets Logistics?",
                "Can you provide the website link of Mets Logistics?",
```

"Where can I find the website of Mets Logistics?"],

"responses": ["You can visit the official website of Mets Logistics at https://metslogistics.com/Home/Facilities for more information."],

"context": [""]

},

{

"tag": "mission",

"patterns": ["what is the mission of Mets Logistics?"],

"responses": ["To help maximize your potential business growth and development by delivering the quality and care that your customers deserve through our cold and dry storage solutions."],

"context": [""]

},

{

"tag": "vision",

"patterns": ["what is the vision of Mets Logistics?"],

"responses": ["To be the preferred cold and dry storage provider of businesses in the Philippines."],

"context": [""]

},

{

"tag": "accreditation",

"patterns": ["Is Mets Logistics accredited?", "Does Mets Logistics have any accreditations?"],

"responses": ["Yes, Mets Logistics is Accredited by: NMIS, BFAR, BPI, FDA, GMP, GOP, HACCP, IDCP, ISO 9001:2915, and NSF."],

"context": [""]

},

{

"tag": "total_storage",

"patterns": [

"What is the total STORAGECOLD in inbound?",

```
        "What is the total STORAGEDRY in inbound?",
        "What is the total STORAGECHILLER in inbound?",
        "What is the total STORAGEAIRCON in inbound?",
        "What is the total STORAGECOLD in outbound?",
        "What is the total STORAGEDRY in outbound?",
        "What is the total STORAGECHILLER in outbound?",
        "What is the total STORAGEAIRCON in outbound?"
    ],
    "responses": ["The total {STORAGE_TYPE} is {TOTAL_COUNT}."],
    "context": ["total_storage"]
},
{
    "tag": "total_inbound_outbound",
    "patterns": [
        "What is the total Arrival inbound?",
        "What is the total Departure inbound?",
        "What is the total StartUnload inbound?",
        "What is the total ICNTotalQty inbound?",
        "What is the total CompleteUnload inbound?",
        "What is the total StorageType inbound?",
        "What is the total Supplier inbound?",
        "What is the total ArrivalTime outbound?",
        "What is the total DepartureTime outbound?",
        "What is the total StartLoading outbound?",
        "What is the total CompleteLoading outbound?",
        "What is the total TotalQty outbound?",
        "What is the total StorageType outbound?",
        "What is the total Supplier?",
        "What is the total SubmittedBy?",
        "What is the total PostedBy?",
```

```
            "What is the total ApprovedBy?",
            "What is the total RejectBy?",
            "What is the total CheckedBy?",
            "What is the total CancelledBy?",
            "What is the total DocumentBy?",
            "What is the total PutAwayBy?",
            "What is the total AddedBy?",
            "What is the total LastEditedBy?",
            "What is the total AcceptBy?",
            "What is the total RFPutAwayBy?",
            "What is the total CheckedBy?",
            "What is the total AuthorizedBy?",
            "What is the total SuppliedBy?"
        ],
        "responses": ["The total {CATEGORY} is {TOTAL_COUNT}."],
        "context": ["total_inbound_outbound"]
    },
    {
        "tag": "facility_location",
        "patterns": [
            "Where are Mets Logistics facilities located in Cavite?",
            "Where specifically in Cavite?",
            "Where are Mets Logistics facilities in Cebu?",
            "Where specifically in Cebu?",
            "Where are Mets Logistics facilities in Cagayan de Oro?",
            "Where specifically in Cagayan de Oro?"
        ],
        "responses": [
            "Mets Logistics has facilities in Bancal, Maguyam, and Banahaw Carmona in Cavite.",
            "Mets Logistics has facilities in Mandaue, Cebu City.",
```

```
            "Mets Logistics has facilities in Tablon, Cagayan de Oro City."
        ],
        "context": ["facility_location"]
    }
    ]
}
```

Dependencies

- Chatbot_model.h5
- Classes.pkl
- Words.pkl
- nltk
- keras
- pickle
- numpy
- pyspellchecker
- flask
- pymssql

This file will get after running the “train.py”