



Sri Lanka Institute of Information Technology

Information Retrieval and Web Analytics - IT3041

Option 01 - Chatbot- Take Home Assignment Report

Team name - AI minds

## Group Members

Student ID	Name
IT21307294	Gamage D G A S
IT21255106	Nelligahawaththa A D T B
IT21206832	Pabasara S D
IT21014772	Chathuranga K K K V

**Git hub Link:** <https://github.com/it21206832/Ecommerce-chatbot-python>

## Contents

Information Retrieval and Web Analytics - IT3041 .....	1
<b>1. Executive Summary .....</b>	<b>4</b>
<b>2. Introduction .....</b>	<b>5</b>
<b>3. Methodology .....</b>	<b>6</b>
3.1 Data Preprocessing .....	10
3.2 Creating Vocabulary and Classes.....	10
3.3 Training Data Creation.....	11
3.4 Model Creation .....	12
3.5 Compile the Model .....	12
3.6 Model Training and Saving .....	12
<b>4. Chatbot Implementation.....</b>	<b>13</b>
4.1 Database Connection .....	13
4.2 Load Chatbot Model and Data Loading .....	13
4.3 handle natural language processing (NLP).....	13
<b>5. Implement the web Application. ....</b>	<b>19</b>
<b>6. MySQL database.....</b>	<b>20</b>
<b>7. Web Application Interfaces .....</b>	<b>22</b>
<b>8. Project Team and Workload .....</b>	<b>27</b>
<b>9. References .....</b>	<b>28</b>

## 1. Executive Summary

The project aimed to develop an eCommerce chatbot integrated with a MySQL database for a laptop shop, enabling it to handle a wide range of customer inquiries, such as product recommendations, order tracking, and general product and service queries. Using a user-friendly web application built with Flask, this chatbot served as a valuable tool for improving customer service and engagement. The project's primary objectives were as follows:

### **Objectives:**

- Develop a chatbot capable of understanding and responding to various customer inquiries related to laptops and related products.
- Integrate the chatbot with a MySQL database to provide real-time product information, inventory status, and order tracking.
- Create a user-friendly web application (built with Flask) that facilitates user interactions with the chatbot.
- Ensure the chatbot provides accurate recommendations, tracks orders efficiently, and enhances overall customer experience.

### **Key Findings:**

- The project successfully developed a chatbot model using TensorFlow and Keras, trained to handle a diverse range of customer queries.
- Integration with a MySQL database allowed the chatbot to access and update product data, making it a valuable resource for both customers and the business.
- The Flask-based web application provided a seamless interface for customers to engage with the chatbot and access information.
- Initial performance metrics demonstrated high accuracy in responding to user queries and efficient order tracking.
- The project faced challenges related to data collection and preprocessing, which required careful handling to ensure the chatbot's effectiveness.

## 2. Introduction

In a rapidly evolving digital landscape, businesses are continually seeking innovative ways to enhance customer experiences and streamline operations. This project represents a pioneering effort to address these objectives, introducing an eCommerce chatbot solution integrated with a MySQL database tailored for a laptop retail shop. Our mission was to develop a sophisticated chatbot capable of managing a diverse array of customer inquiries, including product recommendations, order tracking, and general product and service queries.

### **Importance of the Project:**

In today's eCommerce landscape, customer interactions are increasingly occurring in digital spaces. Providing seamless and efficient customer service has become not only a competitive advantage but a business necessity. The laptop shop, like many modern businesses, faced the challenge of managing an influx of customer inquiries while maintaining a high standard of service. The importance of this project is underscored by the following considerations:

- A chatbot offers immediate responses, reducing customer wait times and increasing satisfaction.
- Integrating the chatbot with a MySQL database allows real-time access to product data, inventory, and order tracking, automating manual tasks and improving operational efficiency.
- The chatbot collects and analyzes customer interactions, yielding insights into preferences, frequently asked questions, and emerging trends, which can guide future business decisions.

### **Problem Statement:**

Our main challenge was meeting the rising need for efficient customer service in laptop retail. Customers seek diverse information, from product suggestions to order updates. To resolve this, we developed a responsive eCommerce chatbot. Integration with MySQL provided real-time product and order data, enhancing the customer experience.

**Technologies used:** TensorFlow and Keras, Python, Flask, MySQL, Visual Studio Code

### 3. Methodology

- The 'intents.json' file includes tags, patterns, and responses, which is needed to create the chatbot model based on patterns and intents. The model is trained to recognize user input and provide appropriate responses.

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi", "Hello", "Hey", "Good morning","heya","what's up?","how is it going?"],
      "responses": ["Hello! Welcome to our online laptop store. How can I assist you today?", "Hey! It's great to see you. How can I assist you today?", "Hey there! Welcome to our online store. How can I help you?","Hi there! Ready to explore our laptop options today?", "Hello, what can I do for you today? Looking for laptop options?"]
    },
    {
      "tag": "services",
      "patterns": ["What can you do?","What services do you offer?","Tell me about your capabilities","How can you help me?","What can you assist with?","In what ways can you support me?","How can you support me" ,"List your services","What are your functions?", "What assistance can I expect from you?", "List your capabilities","How do you provide assistance?","What tasks can you perform?","Can you assist me"],
      "responses": [
        "\n\nI'm your trusty digital assistant with a bag full of tricks!\nHere's what I can do for you:\n\n-Have questions about our products? I'm here to provide details, availability, and reveal the latest promotions.\n\n-I have a treasure trove of laptop details to share with you.\n\n-I can recommend the perfect products based on your preferences.\n\n-Just type your order ID, and I'll give you all the details about your order.\n\n-Wondering about product availability? Name the product, and I'll peek into our stock status for you.\n\n\nAsk me anything!"
      ]
    },
    {
      "tag": "product_recommendation",
      "patterns": ["what laptops do you have","tell me about laptops","what kind of laptops you sell","what do you sell","selling products","Can you recommend me a laptop?", "Suggest me a good laptop", "Find me a laptop", "Find me a good laptop","Which laptop should I buy?",
```

```

        "I need a laptop suggestion","Help me to choose a laptop","Recommend
a laptop for me","Looking for laptop advice","Laptop recommendations, please",
        "Laptop suggestions for my work","Show me laptop options", "Top
laptops for students","What's a good laptop for gaming?","Laptops for
gaming","laptops for students",
        "any laptop recommendations", "Give me details on", "Find information
about","laptop details","laptop information"],
        "responses": ["\n\nHey! Let's explore the best laptops that suit for
your needs.We have a variety of models to choose from\nLenovo\nAsus\nAsus i7\nHP
ProBook\nAcer\nApple Macbook.\n\nwhich laptop are you interested in?(you must
type laptop at the end)\n"]
    },
    {
        "tag": "laptop_brands",
        "patterns": [ "Show me laptop brands","laptop brands",
            "Tell me about popular laptop brands",
            "List of laptop brands",
            "Which laptop brands are the best?",
            "What are the top laptop manufacturers?",
            "Famous laptop brands",
            "Can you recommend a laptop brand?",
            "I'm looking for laptop brands",
            "Give me options from laptop manufacturers",
            "Brands of laptops",
            "Top-rated laptop brands",
            "Well-known laptop manufacturers",
            "Share some laptop brands with me"],
        "responses": ["Curious about laptop brands? I'll share some details
with you.These are the brands we have.\nLenovo\nAsus\nAsus i7\nHP
ProBook\nAcer\nApple Macbook.\n\nwhich laptop brand are you interested in?(you
must type laptop at the end)\n"]
    },
    {
        "tag": "product_info",
        "patterns": ["Tell me about", "laptop", "product"],
        "responses": ["Sure"],
        "context_set": "product_info"
    },
    {
        "tag": "promotion_info",
        "patterns": ["promotions", "What discounts are available?", "Any
special offers?", "Promotions and discounts","Are there any deals?",
            "Tell me about current promotions",
            "Show me the latest discounts",
            "Promotional offers",

```

```

        "Discounts on products",
        "Special deals",
        "What's on sale?",
        "Can you help me save money?",
        "Promos and offers",
        "Discounted products",
        "Deals and discounts",
        "Share current promotions"],
    "responses": ["You're about to discover the hottest discounts on our
products.\nCurrently we give promotions for these laptops\n\nLenova laptops-Save
upto 30%\n'Asus laptops-Save upto 25%"],
    "context_set": "promotion_info"
},
{
    "tag": "order_info",
    "patterns": ["What can you", "Give me details on my order", "Find
information about my order", "Check my order status",
        "Order details",
        "Tell me about my order",
        "I want to know about my order",
        "Can you provide order information?",
        "Share my order details",
        "Get info on my recent order",
        "What's the status of my order?",
        "Order status update",
        "Give me the details of my recent order",
        "Check my recent purchase"],
    "responses": ["Sure inq"],
    "context_set": "order_info"
},
{
    "tag": "general_inquiry",
    "patterns": ["Tell me about your return policy", "What payment
methods do you accept?", "How can I contact customer support?", "Do you have a
loyalty program?"],
    "responses": ["I can provide information on that. Please specify your
question.", "Sure, I can help with that. What do you want to know?"]
},
{
    "tag": "thanks",
    "patterns": ["Thank you", "Thanks a lot", "Appreciate your help",
"You're awesome", "I'm thankful",
        "You've been so helpful",
        "I appreciate it",
        "Big thanks to you"],

```



```

        "responses": ["You're welcome! If you have more questions, feel free
to ask.", "It was my pleasure assisting you. Don't hesitate to ask if you need
anything else."]
    },
    {
        "tag": "goodbye",
        "patterns": ["Goodbye", "See you later", "Bye for now", "Farewell"],
        "responses": ["Goodbye! Have a great day.", "Take care and have a
wonderful day!"]
    },
    {
        "tag": "fallback",
        "patterns": ["Sorry, I don't know", "I'm not sure", "Can you repeat
that?", "I didn't understand"],
        "responses": ["I apologize, but I couldn't understand your request.
Please try again or ask something else.", "I'm sorry, I couldn't catch that. Can
you please rephrase your question?"]
    },
    {
        "tag": "order_reference_no",
        "patterns": ["My order reference no is {order_reference_no}",
"this is my order reference no is "],
        "responses": ["Let me check the status for you."],
        "context_filter": "order_reference_no"
    }
]
}

```

- **Important Libraries**

```

import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
import mysql.connector
from keras.models import load_model
import json
import random

```

### 3.1 Data Preprocessing

The code starts by reading an 'intents.json' file, which appears to contain a set of intents and associated patterns. It tokenizes and lemmatizes the words in these patterns, preparing them for further processing.

```
• # Data Preprocessing
• 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'])
•
```

### 3.2 Creating Vocabulary and Classes

The script builds a vocabulary by collecting unique lemmatized words from the patterns. It also identifies unique intent classes. This is essential for creating training data and understanding user intents.

```
• # 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)))
• # 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)
```

### 3.3 Training Data Creation

The code creates training data for the chatbot, representing input patterns as bags of words with binary encoding (1 for presence, 0 for absence) for each word. It uses one-hot encoding to represent the target intent as a binary vector. Training data is shuffled to introduce randomness during training.

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

# Prepare training data
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 training data
random.shuffle(training)

# Prepare data for model training
train_x = np.array([i[0] for i in training])
train_y = np.array([i[1] for i in training])

print("Training data created")
```

### 3.4 Model Creation

Using the Keras library, the code creates a neural network model for the chatbot. It consists of three layers: an input layer with 128 neurons, a hidden layer with 64 neurons, and an output layer with a number of neurons equal to the number of intents. The model uses the softmax activation function for multiclass classification.

```
import tensorflow as tf # Import TensorFlow

# Create a model with Keras
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), 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'))
```

### 3.5 Compile the Model

The model is compiled using the Stochastic Gradient Descent (SGD) optimizer with specific learning rates and momentum settings.

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

### 3.6 Model Training and Saving

The training loop fits the model to the training data, running for a specified number of epochs and batch size. The model learns to predict intents based on the input patterns.

```
4 # Fit and save the model
5 hist = model.fit(np.array(train_x), np.array(train_y), epochs=200,
    batch_size=5, verbose=1)
6 model.save('chatbot_model.h5', hist)
```

## 4. Chatbot Implementation

### 4.1 Database Connection

It establishes a connection to a MySQL database, indicating the host, user, password, and database name.

```
#database connection
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="1234",
    database="ecombot"
)
```

### 4.2 Load Chatbot Model and Data Loading

The pre-trained chatbot model ('chatbot\_model.h5') is loaded using Keras' load\_model function. After that, The script loads essential data, such as words, classes, and intents, from serialized pickle files.

```
# Load the pre-trained chatbot model
model = load_model('chatbot_model.h5')

# Load words, classes, and intents from pickle files
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl', 'rb'))
classes = pickle.load(open('classes.pkl', 'rb'))
```

### 4.3 handle natural language processing (NLP)

NLP functions predict user intent and generate chatbot responses. These functions enable intent recognition and context-aware chatbot replies. Resulting in effective and interactive user-bot interactions.

### The functions involved are:

- `clean_up_sentence`: Tokenizes, lemmatizes, and converts text to lowercase.
- `bow` (Bag of Words): Creates a binary matrix for word presence.
- `predict_class`: Predicts user intent using the chatbot model.
- `getResponse`: Retrieves contextually relevant responses from predefined intents.

```
# Function to clean up a sentence
def clean_up_sentence(sentence):
    # Tokenize the pattern - split words into an array
    sentence_words = nltk.word_tokenize(sentence)
    # Lemmatize and convert to lowercase
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# Function to create a bag of words
def bow(sentence, words, show_details=True):
    # Tokenize and clean the pattern
    sentence_words = clean_up_sentence(sentence)
    # Create a bag of words - a matrix of N words (vocabulary matrix)
    bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                # Assign 1 if the current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print("found in bag: %s" % w)
    return np.array(bag)

# Function to predict user intent based on input
def predict_class(sentence, model):
    # Filter out predictions below a threshold
    p = bow(sentence, words, show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
    # Sort results 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 retrieve a response based on the predicted intent
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag'] == tag):
            result = random.choice(i['responses'])
            break
    return result
```

Below code section comprises functions to handle product inquiries, retrieve order details, and generate chatbot responses based on user intents, enhancing interactivity.

- **handle\_product\_inquiry:** Queries the database for product information, including name, description, price, availability, and promotions. Provides detailed product responses and handles errors effectively.
- **get\_order\_by\_id:** Retrieves order details from the database using the order ID. Offers comprehensive order information and gracefully handles cases where orders are not found.
- **chatbot\_response:** Central function for the chatbot. Predicts user intent and generates relevant responses. Includes product information retrieval, order details, and general responses.
- **order\_tracking:** The chatbot can fetch order status information by utilizing the order reference number as a query parameter. In cases where the user submits inaccurate or invalid inputs, the chatbot will promptly issue an error message to guide the user toward providing valid information.

```
def handle_product_inquiry(product_name):
    cursor = db.cursor()
    try:
        cursor.execute("SELECT * FROM products WHERE name = %s", (product_name,))
        product = cursor.fetchone()

        if product:
            product_id, name, description, price, availability, promo_info =
product
            response = f"{name}\n - {description}\n - Price: ${price}\n"
```

```

        # Check for availability and promotions
        if availability:
            response += " - Availability: In Stock\n"
        else:
            response += " - Availability: Out of Stock\n"

        if promo_info:
            response += f" - Promotions: {promo_info}\n"
        else:
            response += " - No promotions for this laptop\n"

    else:
        response = "I couldn't find information about that product. Please
try again."
    except Exception as e:
        response = f"An error occurred while retrieving the information:
{str(e)}"
    finally:
        cursor.close()

    return response

# Get order details by order id

def get_order_by_id(order_id):

    cursor = db.cursor()
    query = "SELECT * FROM orders WHERE order_id = %s"

    # cursor.execute(query, (order_id,))
    cursor.execute("SELECT * FROM orders WHERE order_id = %s", (order_id,))
    order_data = cursor.fetchone()

    # Fetch and print the results
    cursor.close()

    return order_data

def chatbot_response(msg):

    res = "" # Initialize the res variable with an empty string
    ints = predict_class(msg, model)

```



```

    if ints[0]['intent'] == 'product_info':
        product_name = msg # Assuming that the product name is the same as the
user's input
        stopwords = ['laptop', 'can', 'you', 'tell', 'me', 'about',
'product', 'information', 'details', 'I', 'wanna', 'know', 'about', 'info', 'give', 'pleas
e']
        querywords = msg.split()
        resultwords = [word for word in querywords if word.lower() not in
stopwords]
        product_name = ' '.join(resultwords)
        res = "Your requested laptop information\n\n" +
handle_product_inquiry(product_name) + "\n\nContact us for more information.Our
email address ai_minds@gmail.com"
        #res = product_name

    elif msg.startswith("get order by") or msg.startswith("my order id is ") or
msg.startswith("my order id is "):
        order_id = msg.split()[-1]
        order_data = get_order_by_id(order_id)
        if order_data:
            res = f"Order ID: {order_data[0]}\n, Product id: {order_data[1]}\n,
Quantity: {order_data[2]}\n, Total prise: {order_data[3]}\n"
        else:
            res = "Order not found."

    else:
        # If it's not a specific intent, use the previous code to get a response
        res = getResponse(ints, intents)

    return res

```

```

131
132 # Tracking order
133
134 def order_tracking(order_refNo):
135
136     cursor = db.cursor()
137     query = " SELECT * FROM order_tracking WHERE order_refNo = %s "
138
139     # cursor.execute(query, (order_id,))
140     cursor.execute(query, (order_refNo,))
141     # Fetch and print the results
142     order_data = cursor.fetchone()
143     cursor.close() # Close the cursor and database connection
144
145     return order_data
146
147

```

```

elif msg.startswith("LU") or msg.startswith("my reference no is LU00"):
    # Extract the order ID using regular expressions
    import re
    match = re.search(r'\bLU\d+\b', msg)
    if match:
        order_refNo = match.group()
        order_data = order_tracking(order_refNo)
        if order_data:
            res = f"{order_data[1]}"
        else:
            res = "Your order is not in our database. Please check and re-enter."
    else:
        res = "Invalid order reference number format. The reference number should start with 'LU' followed by 3 digits (e.g., LU123)."

```

## 5. Implement the web Application.

The below code essentially creates a web interface for the chatbot, where users can input messages, and the chatbot responds to those messages. We use Flask to handle the web application, rendering templates for the user interface and handling user inputs and chatbot responses.

### Steps:

- Imports necessary libraries, including Flask for web development, JSON for data, and the chatbot\_response function.
- Loads predefined intents from JSON.
- Initializes a Flask app with CORS for cross-domain requests.
- Defines two routes: one to render a chat interface and one to process user messages and return responses.
- Runs the app in debug mode when executed.

```
from flask import Flask, render_template, request, jsonify
import json
from chatgui import chatbot_response
from flask_cors import CORS

intents = json.loads(open('intents.json').read())

app = Flask(__name__)
CORS(app)

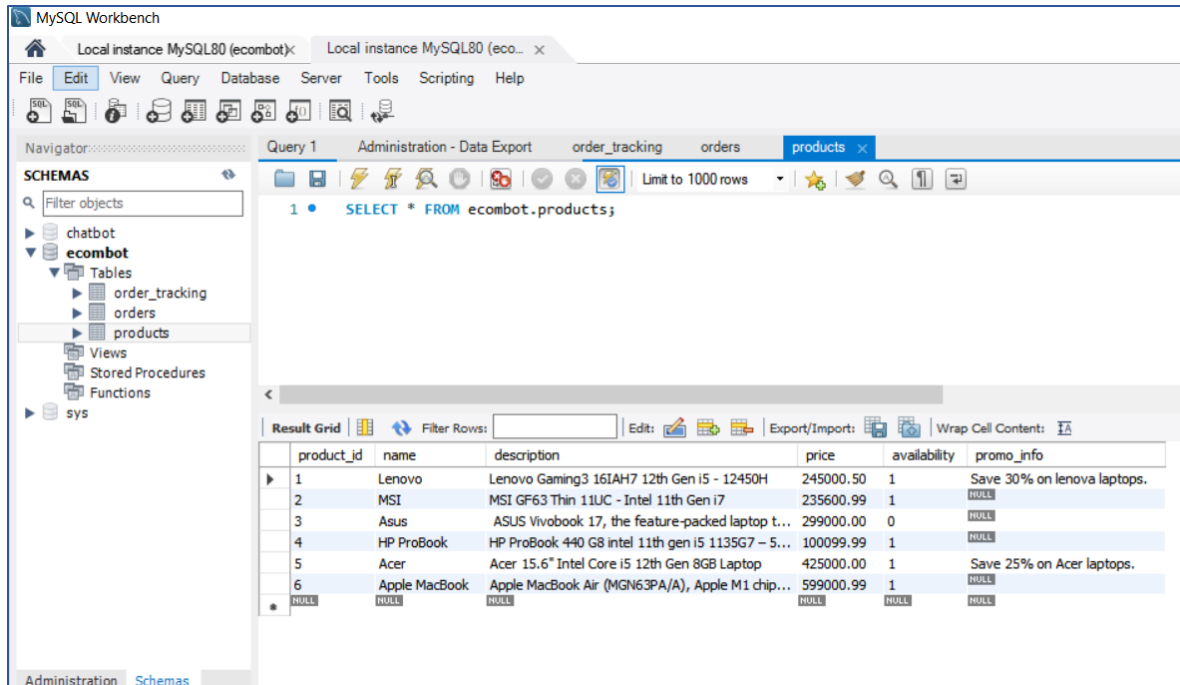
@app.get("/")
def index_get():
    return render_template("base.html")

@app.post("/predict")
def predict():
    text = request.get_json().get("message")
    # TODO: check if text is valid
    res = chatbot_response(text)
    message = {"answer": res}
    return jsonify(message)

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

## 6. MySQL database

We utilized a database named 'ecombot,' consisting of three tables: 'Order\_Tracking,' 'Orders,' and 'Products' to manage and store relevant eCommerce data efficiently.



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'ecombot' database selected, with its tables listed: 'order\_tracking', 'orders', and 'products'. The 'products' table is highlighted. The main workspace shows a query window with the SQL statement: `SELECT * FROM ecombot.products;`. Below the query, the 'Result Grid' displays the data from the 'products' table. The table has columns: 'product\_id', 'name', 'description', 'price', 'availability', and 'promo\_info'. The data is as follows:

product_id	name	description	price	availability	promo_info
1	Lenovo	Lenovo Gaming3 16IAH7 12th Gen i5 - 12450H	245000.50	1	Save 30% on lenova laptops.
2	MSI	MSI GF63 Thin 11UC - Intel 11th Gen i7	235600.99	1	N/A
3	Asus	ASUS Vivobook 17, the feature-packed laptop t...	299000.00	0	N/A
4	HP ProBook	HP ProBook 440 G8 intel 11th gen i5 1135G7 - 5...	100099.99	1	N/A
5	Acer	Acer 15.6" Intel Core i5 12th Gen 8GB Laptop	425000.00	1	Save 25% on Acer laptops.
6	Apple MacBook	Apple MacBook Air (MGN63PA/A), Apple M1 chip...	599000.99	1	N/A
*	N/A	N/A	N/A	N/A	N/A

MySQL Workbench

Local instance MySQL80 (ecombot) Local instance MySQL80 (eco... x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

Filter objects

chatbot

ecombot

Tables

order\_tracking

orders

products

Views

Stored Procedures

Functions

sys

Query 1 Administration - Data Export order\_tracking orders products

1 • SELECT \* FROM ecombot.orders;

Result Grid

order_id	product_id	quantity	total_price	order_refno
1	1	2	490001.00	LU001
2	2	1	235600.99	LU002
3	3	3	897000.00	LU003
4	4	1	100099.99	LU004
5	5	2	850000.00	LU005
6	6	1	599000.99	LU006
7	4	1	100099.99	LU007
8	6	1	599000.99	LU008
9	4	1	100099.99	LU009
10	6	1	599000.99	LU110
*	NULL	NULL	NULL	NULL

Administration Schemas

Information

MySQL Workbench

Local instance MySQL80 (ecombot) Local instance MySQL80 (eco... x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

Filter objects

chatbot

ecombot

Tables

order\_tracking

orders

products

Views

Stored Procedures

Functions

sys

Query 1 Administration - Data Export order\_tracking orders products

1 • SELECT \* FROM ecombot.order\_tracking;

Result Grid

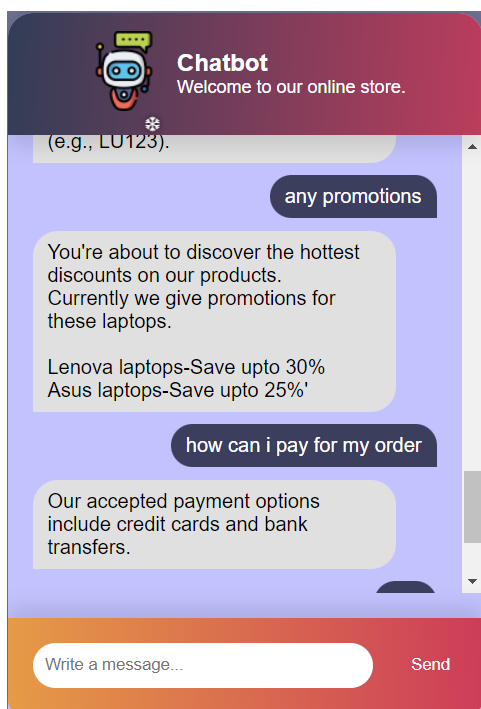
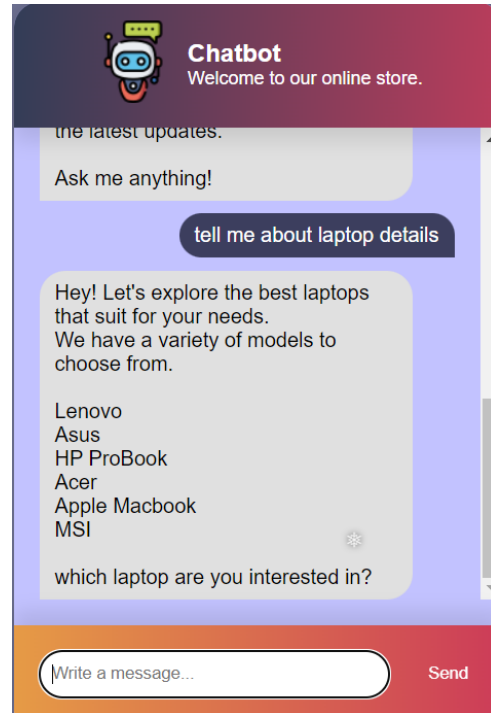
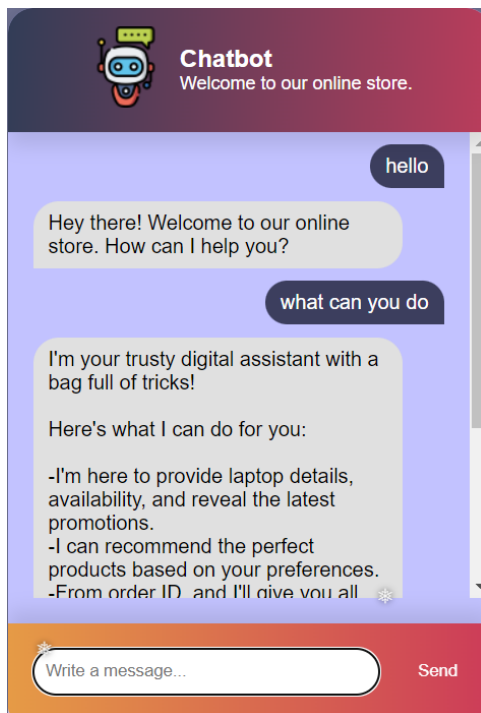
order_id	status	order_refno
1	Your order has been successfully shipped and is...	LU001
2	Your order is currently in the processing stage.	LU002
3	Good news! Your order is being processed and ...	LU003
4	We're in the process of getting your order read...	LU004
5	Your order has been successfully shipped and is...	LU005
6	Your order is currently in the processing stage.	LU006
7	Your order has been successfully shipped and is...	LU007
8	We're in the process of getting your order read...	LU008
9	Your order has been successfully shipped and is...	LU009
10	We're in the process of getting your order read...	LU010
*	NULL	NULL

Administration Schemas

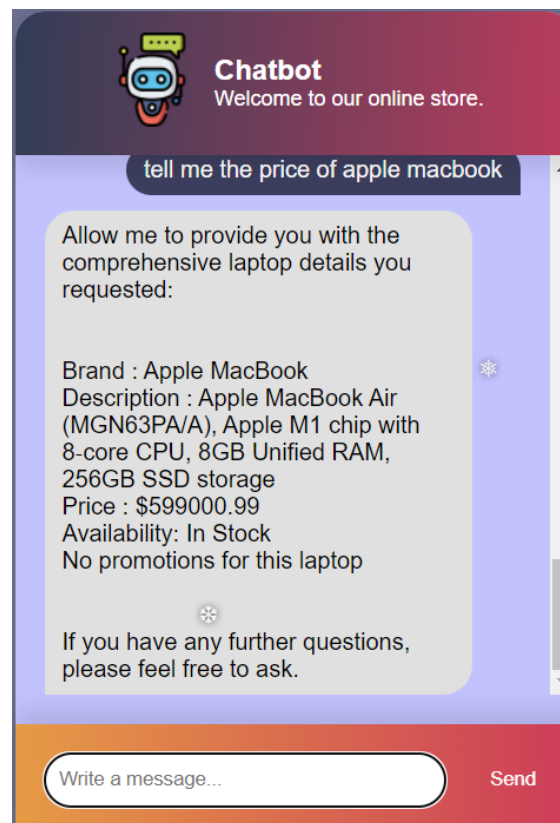
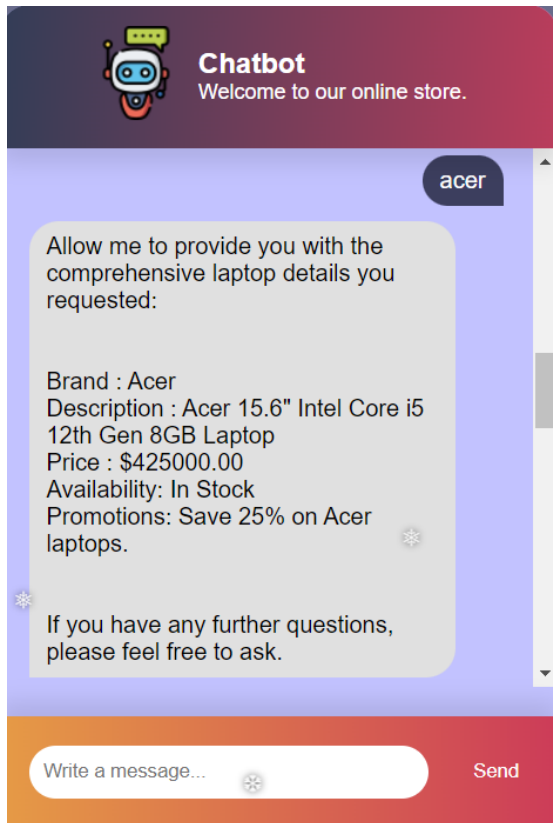
Information

## 7. Web Application Interfaces

The images presented here illustrate the chatbot's response to user input, showcasing its ability to generate appropriate responses based on the provided input.

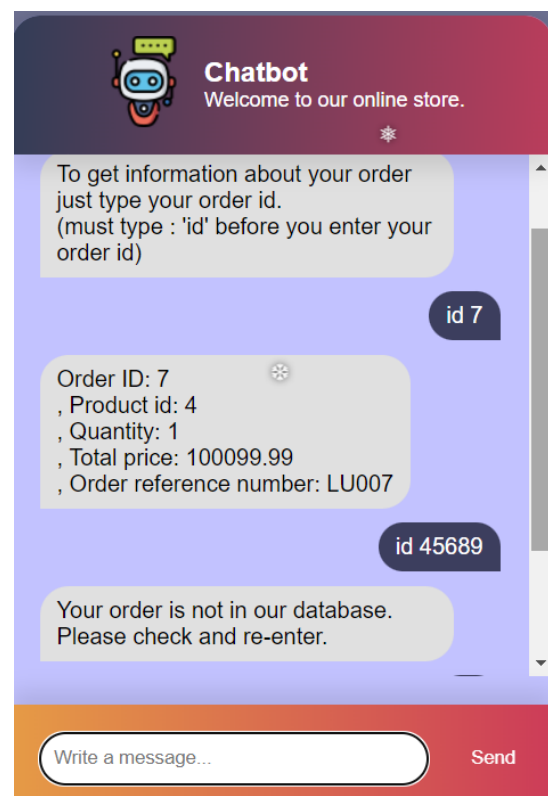
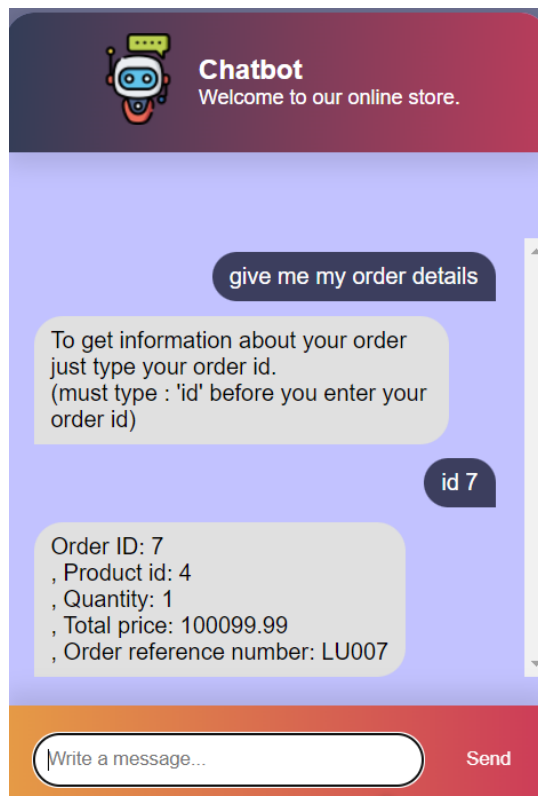


Here chatbot will proceed to retrieve information from the database with respect to the laptop specified by the user, based on the laptop name provided.



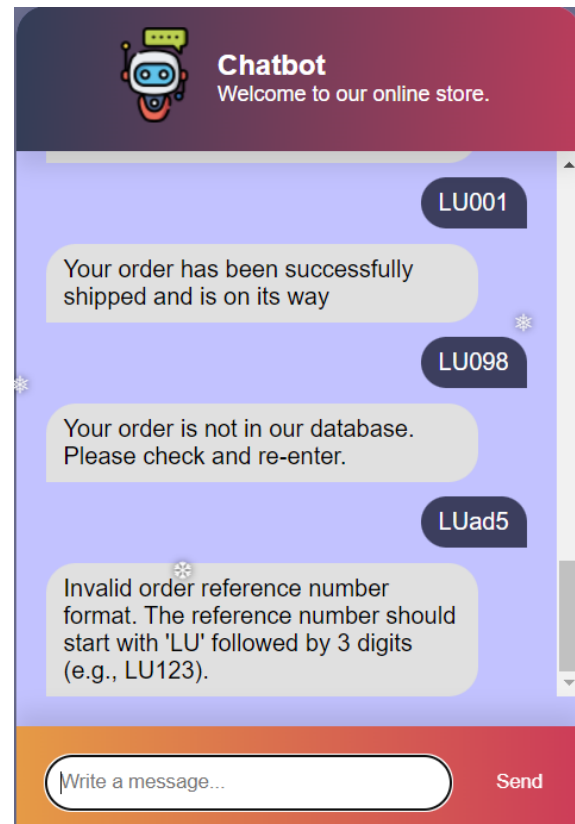
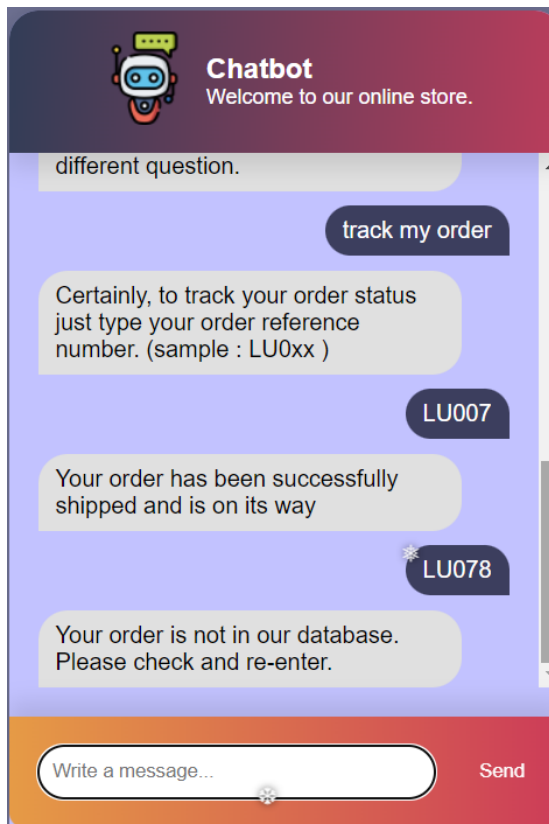
Upon receiving an order ID, the system will initiate the retrieval of pertinent information from the database associated with the specified order ID.

If a user enters an order ID that is not found within the database, the system will generate an error notification to inform the user of the unavailability of the specified order ID in the database.

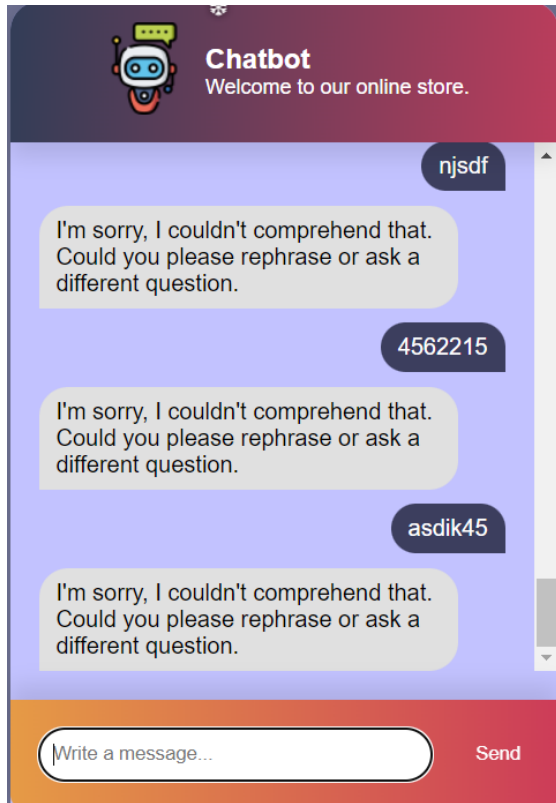




The chatbot can fetch order status information by utilizing the order reference number as a query parameter. In cases where the user submits inaccurate or invalid inputs, the chatbot will promptly issue an error message to guide the user toward providing valid information.



In situations where the user inputs nonsensical or erroneous words or phrases, the chatbot is programmed to respond with an error message to indicate the incomprehensibility of the input.



## 8. Project Team and Workload

Name	Registration Number	Responsibility
<b>IT21307294</b>	<b>Gamage D G A S</b>	NLP Model Training Database Integration Web Application Development Intent and Response Management Deployment and Debugging
IT21206832	Pabasara S D	NLP Model Training Database Integration Web Application Development Intent and Response Management Deployment and Debugging
IT21255106	Nelligahawaththa A D T B	NLP Model Training Database Integration Web Application Development Intent and Response Management Deployment and Debugging
IT21014772	Chathuranga K K K V	NLP Model Training Database Integration Web Application Development Intent and Response Management Deployment and Debugging

## 9. References

- <https://blog.hubspot.com/website/python-ai-chat-bot>
- <https://medium.com/analytics-vidhya/how-to-create-a-chatbot-in-python-7ab924f10125>
- <https://medium.com/databutton/how-to-build-a-chatbot-with-chatgpt-api-and-a-conversational-memory-in-python-8d856cda4542>
- <https://datasciencedojo.com/blog/ai-based-chatbots-in-python/>
- <https://www.digitalocean.com/community/tutorials/how-to-create-an-intelligent-chatbot-in-python-using-the-spacy-nlp-library>
- <https://realpython.com/build-a-chatbot-python-chatterbot/>
- <https://www.simplilearn.com/tutorials/python-tutorial/how-to-make-a-chatbot-in-python>
- <https://www.upgrad.com/blog/how-to-make-chatbot-in-python/>
- <https://www.geeksforgeeks.org/chat-bot-in-python-with-chatterbot-module/>