A

Mini Project

On

PRICE NEGOTIATION CHATBOT ON E-COMMERCE WEBSITE

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

U.HARIHARAN (207R1A05B5)

G.PRANAY (207R1A0579)

M.NIKHILA (207R1A0597)

Under the Guidance of

R. Sai Krishna

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"PRICE NEGOTIATION CHATBOT ON E-COMMERCE WEBSITE"**being submitted by **U.HARIHARAN (207R1A05B5), G.PRANAY(207R1A0597) & M.NIKHILA (107R1A0597)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

R. SAI KRISHNA
(Associate Professor)
INTERNAL GUIDE

Dr. A. RANJITH REDDY
DIRECTOR

Dr. K. SRUJAN RAJU
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **R.Sai Krishna,** Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G.Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Ranjith Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

U.HARIHARAN (207R1A05B5)
G.PRANAY (207R1A0579)
M.NIKHILA (207R1A0597)

ABSTRACT

Negotiation is a key component of real-life transactions. Negotiating is nothing but bargaining. Any big business deal to buying fruits and vegetables from street vendors, it plays a vital role. E- commerce Chatbot project will help us to negotiate the price of the product. Customer satisfaction is the major concern for all the web-based applications and chatbots helps them to get their issues resolved quickly without wasting their time in writing mails and sending them to the responsible authority and waiting for them to answer. Chatbots act as an intermediate source between the company and the user, and having them making it easy to solve the various issues that any customer might face. Negotiation is something that has linguistic as well as reasoning issues eventually which helps to provide a solution. Quite often, customers usually get confused what they are searching and what they actually want but here, the chatbot will help the customer to shop what they exactly desire.

In recent years online shopping has gained a huge boom. With this increase, most of the features of online shopping are developed but some features like negotiating with shopkeepers are not available which is sometimes possible in offline purchasing. We have implemented a chatbot for negotiating on the products. The chatbot interacts with customers and assists them to get a satisfactory price on product(s). With such a system, which impacts on major areas of online shopping there are possibilities in which either the seller of the product or customer's budget gets compromised. To avoid such situations we have developed an algorithm which works along with prediction of old available data to provide a price. Price prediction has less accuracy at times because either irrelevant features/attributes of data are used or some algorithms are not suitable for a particular dataset. Due to this, Ecommerce business does not directly rely on price prediction systems since even a wrong prediction of a single product can result in business losses. Some models also fail when data scales or some feature is unavailable after time on which model prediction was dependent. Then those changes are to be managed to maintain the accuracy and reliability of the model. In our chatbot system we have tried to resolve some of such issues

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT SCOPE

E-commerce websites today apply various AI techniques to determine most liked products or most sold products which eventually are calculated to provide an effortless search for customers shopping on their website. But at times when the best products are sold at high prices, customers have to compromise on their product. There are also some other problems that customers may face on low cost products. These problems can be eliminated by giving them an opportunity to negotiate on the products.

1.2 PROJECT PURPOSE

Negotiation is a key component of real-life transactions. Negotiating is nothing but bargaining. Any big business deal to buying fruits and vegetables from street vendors, it plays a vital role. E- commerce Chatbot project will help us to negotiate the price of the product..

1.3 PROJECT FEATURES

The main features of this project are that the chatbot seamlessly integrates with the eCommerce website's product catalog and pricing system, ensuring accurate and up-to-date information. This integration enables users to explore and negotiate prices for a wide range of products effortlessly. Additionally, the chatbot's 24/7 availability enhances user engagement by providing assistance around the clock, fostering user satisfaction and loyalty. Overall, the project's features combine to create an innovative and interactive eCommerce platform that empowers users, enhances their shopping experience, and sets the website apart in a competitive market.

2.SYSTEM ANALYSIS

System analysis is the process of evaluating, understanding, and defining the requirements and specifications of a system or project. It involves studying the current system (if applicable), identifying deficiencies or opportunities for improvement, gathering user and stakeholder feedback, and formulating a clear plan for the design and development phases. System analysts use various techniques, such as data modeling, workflow analysis, and stakeholder interviews, to ensure that the resulting system meets the needs and objectives of the organization or project. The primary goal of system analysis is to bridge the gap between user expectations and the technical implementation, ultimately leading to the successful development and deployment of a functional and efficient system.

2.1 PROBLEM DEFINITION

Define a chatbot system that enables users to negotiate product prices in real-time on an eCommerce website, ensuring efficient communication, accurate pricing, and a seamless user experience.

2.2 EXISTING SYSTEM

The first chatbots and the brain behind it were Joseph Weizmann. Eliza's key method of operation involves the recognition of cue words or phrases in the input and the output of corresponding pre-prepared or pre-programmed responses that can move the conversation forward in an apparently meaningful way. Thus the key technique here—which characterizes a program as a chatbot rather than as a serious natural language processing system—is the production of responses that are sufficiently vague and non-specific that they can be understood as "intelligent" in a wide range of conversational contexts. More recent notable programs include A.L.I.C.E, Jabberwacky and D.U.D.E. While ELIZA and PARRY were used exclusively to simulate typed conversation, many chatterbots now include functional features such as games and web searching abilities.

Most of the existing virtual agents, also known as chatbots, are mainly for entertainment and research purposes. Successful and Award-winning chatbots like A.L.I.C.E and Clever Bot focus on generic responses to entertain the end user. Some companies like IKEA, Lloyds Banking Group and Royal Bank of Scotland are using automated online assistants as the first points of contact.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

- Lack of Price negotiation
- Limited Personalization
- Limited Understanding of Customer Sentiment
- Limited Interaction method
- Lack of Comprehensive User Management

2.3 PROPOSED SYSTEM

Personal experience has led us to believe that offering discounts are an efficient method of recovering abandoned carts This is primarily due to the way the e-commerce ecosystem has trained consumers to expect discounts all the time. However, for small businesses, providing discounts consistently may not be enough. Additionally, trying to recover abandoned carts through targeted discount emails may not be effective as customers may already have purchased the product from a competitor's website, rather than waiting for a discount email. This also exposes the business's discount strategy, making it more predictable.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- Price negotiating capability
- Personalized pricing
- Sentiment Analysis
- Voice-Based Interaction
- User management
- Enhanced User Experience

2.4 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

2.4.1 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- **System** : Pentium IV 2.4 GHz.
- **Hard Disk** : 40 GB.
- **Monitor** : 15 inch VGA Color.
- **Mouse** : Logitech Mouse.
- **Ram** : 512 MB
- **Keyboard** : Standard Keyboard

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- **Operating System** : Windows XP.
- **Platform** : PYTHON TECHNOLOGY
- **Tool** : Spyder, Python 3.5,Xamp
- **Front End** : Anaconda
- **Back End** : python anaconda script

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification,starting from input to final prediction.

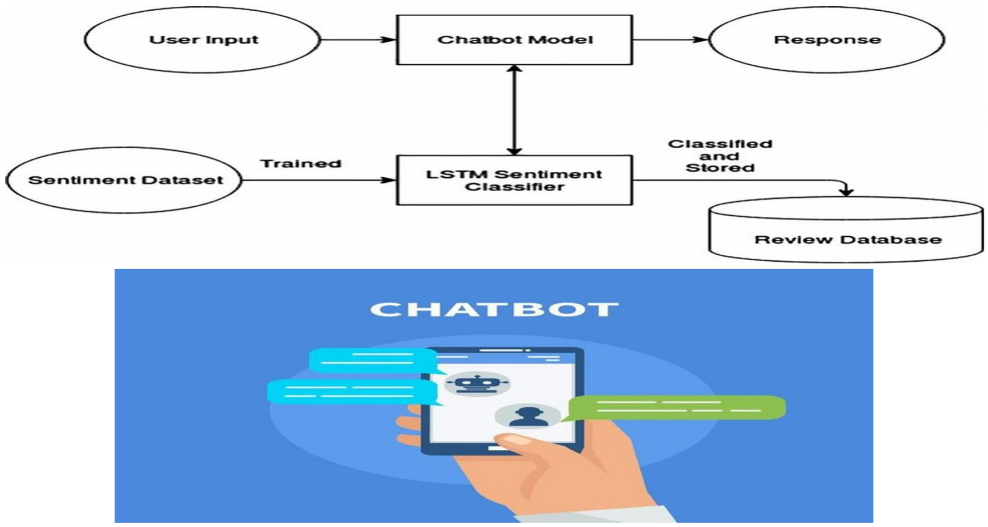


Figure 3.1: Project Architecture of Price Negotiation chatbot on e- commerce website

3.2 DESCRIPTION

The architecture of the Price Negotiation Chatbot on an eCommerce website consists of a front-end interface for user interactions and a back-end system for processing and integration. The front end provides a user-friendly interface for shoppers to engage in natural language conversations with the chatbot, while the back end utilizes technologies like NLP, AI-driven pricing, and database integration to understand user requests, offer real-time price negotiation, and provide personalized product recommendations. Security measures and external system integrations ensure a secure and seamless shopping experience.

3.3 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

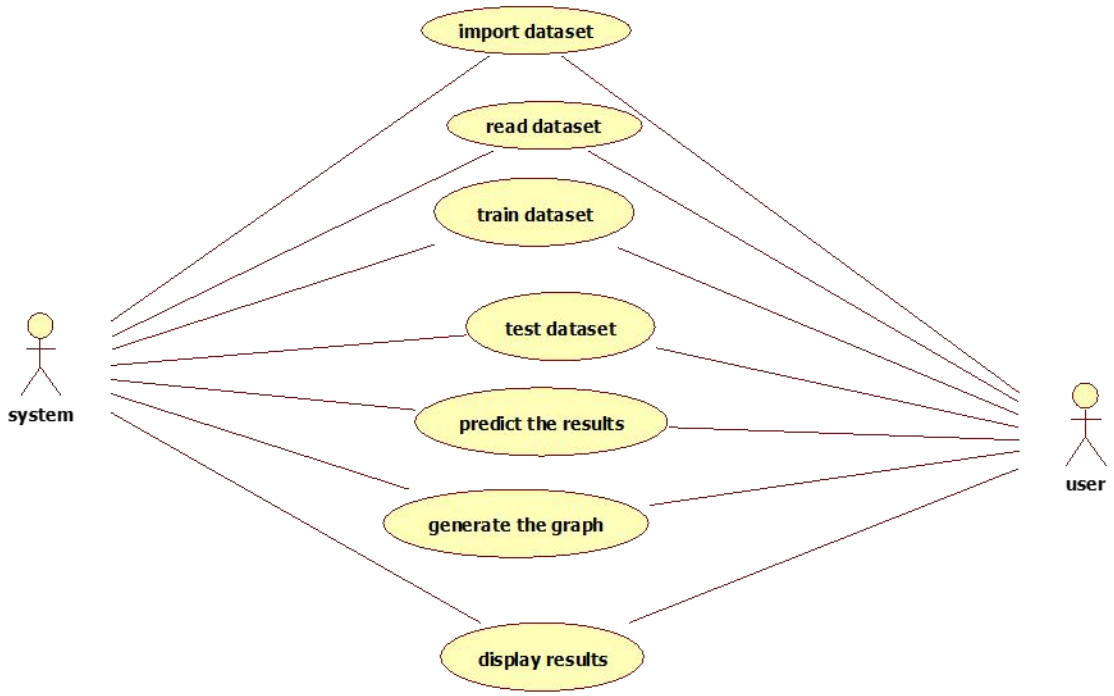


Figure 3.2: Use Case Diagram for price negotiation chatbot on e-commerce website.

3.4 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information

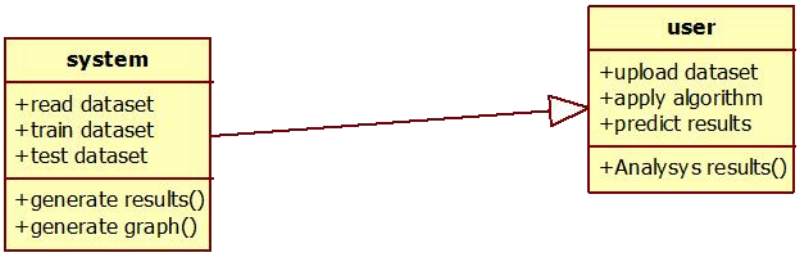


Figure 3.3: Class Diagram for price negotiation chatbot on e-commerce website.

3.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

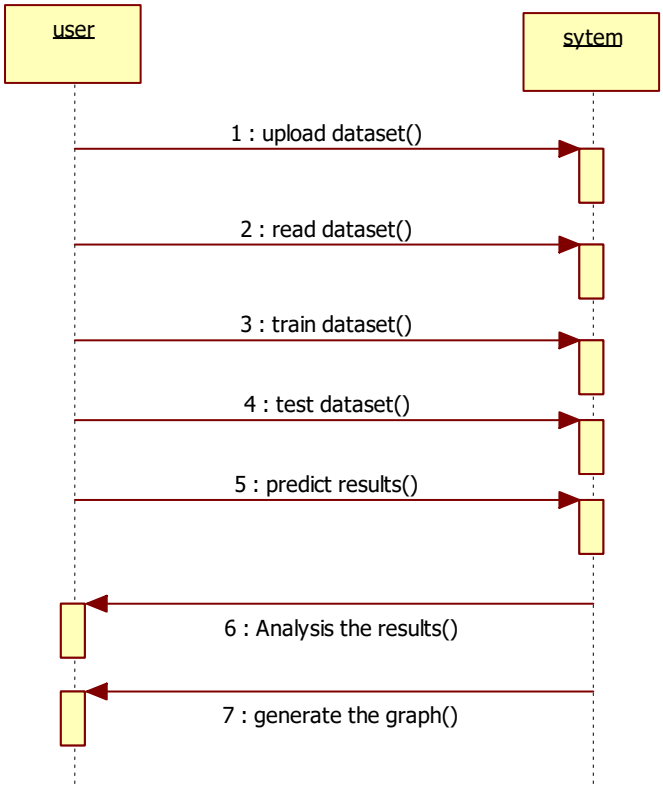


Figure 3.4: Sequence Diagram for price negotiation chatbot on e-commerce website.

3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

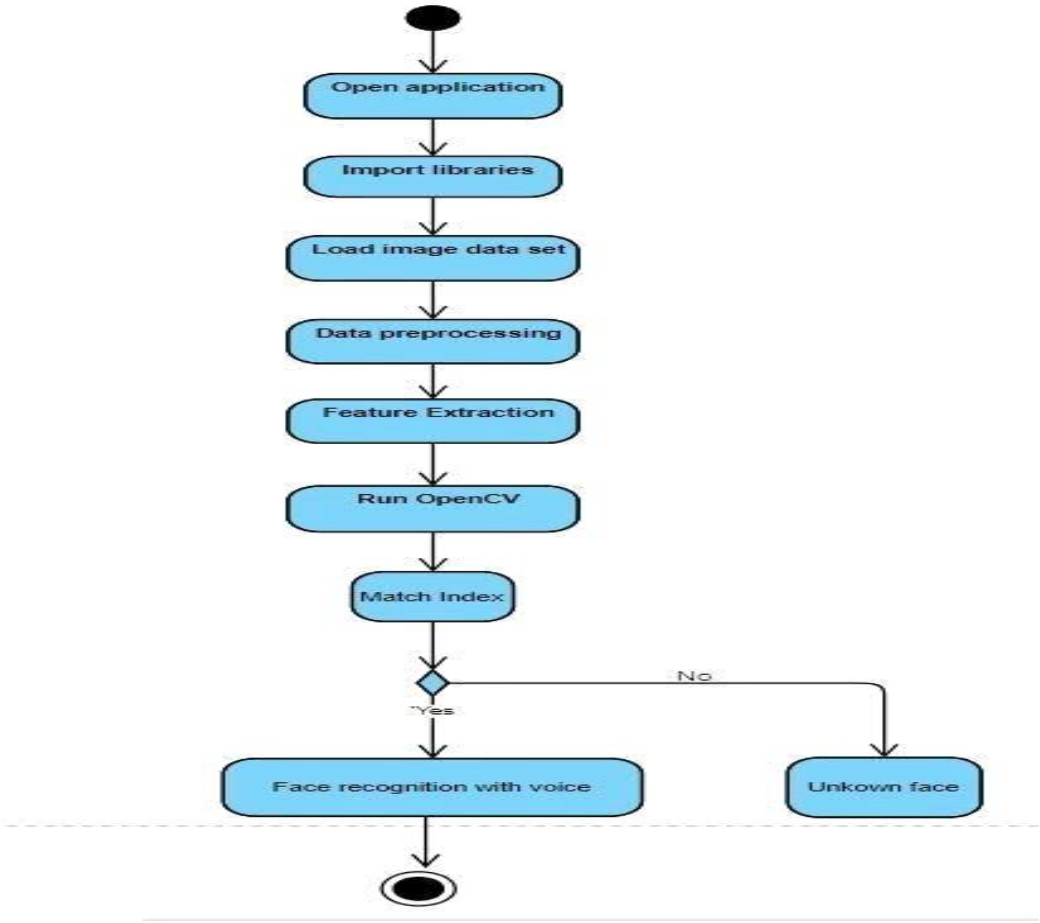


Figure 3.5: Activity Diagram for price negotiation chatbot on e-commerce website.

4.1 SAMPLE CODE

```
from flask import Flask, render_template, request, redirect, url_for, session,
make_response
import pymysql
import datetime
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import os
import subprocess
import speech_recognition as sr

app = Flask(__name__)

app.secret_key = 'welcome'
global uname
global original_price, predicted_price, final_price, product_name, product_id

sid = SentimentIntensityAnalyzer()
recognizer = sr.Recognizer()

@app.route('/ViewReview', methods=['GET', 'POST'])
def ViewReview():
    if request.method == 'GET':
        global uname
        font = '<font size="3" color="black">'
        output = '<table border="1" width="100%">'
        output += '<tr><th><font size="3" color="black">Username</font></th>'
        output += '<th><font size="3" color="black">Review</font></th>'
        output += '<th><font size="3" color="black">Sentiment</font></th></tr>'
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '',
database = 'negotiate',charset='utf8')
        index = 0
        with con:
            cur = con.cursor()
            cur.execute("select * FROM reviews")
            rows = cur.fetchall()
            for row in rows:
```

```
            output += "<tr><td>"+font+str(row[0])+"</td>"
            output += "<td>"+font+str(row[1])+"</td>"
            output += "<td>"+font+str(row[2])+"</td>"
        return render_template('ViewReview.html', msg=output)

@app.route('/ViewOrders', methods=['GET', 'POST'])
def ViewOrders():
    if request.method == 'GET':
        global uname
        font = '<font size="3" color="black">'
        output = '<table border="1" width="100%">'
        output += '<tr><th><font size="3" color="black">Purchaser Name</font></th>'
        output += '<th><font size="3" color="black">Product ID</font></th>'
        output += '<th><font size="3" color="black">Product Name</font></th>'
        output += '<th><font size="3" color="black">Amount</font></th>'
        output += '<th><font size="3" color="black">Purchase Date</font></th></tr>'
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '',
database = 'negotiate',charset='utf8')
        index = 0
        with con:
            cur = con.cursor()
            cur.execute("select * FROM purchaseorder where username='"+uname+"'")
            rows = cur.fetchall()
            for row in rows:
                output += "<tr><td>"+font+str(row[0])+"</td>"
                output += "<td>"+font+str(row[1])+"</td>"
                output += "<td>"+font+str(row[2])+"</td>"
                output += "<td>"+font+str(row[3])+"</td>"
                output += "<td>"+font+str(row[4])+"</td>"
        return render_template('ViewOrders.html', msg=output)

@app.route('/CompleteOrder', methods=['GET', 'POST'])
def CompleteOrder():
    global uname
    global original_price, predicted_price, final_price, product_name, product_id
    if request.method == 'POST':
        if predicted_price != 0:
            now = datetime.datetime.now()
            current_time = now.strftime("%Y-%m-%d %H:%M:%S")
            status = "Error in cinfirming order"
            db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root',
password = '', database = 'negotiate',charset='utf8')
            db_cursor = db_connection.cursor()
            student_sql_query = "INSERT INTO
purchaseorder(username,product_id,product_name,amount,transaction_date)
```

```
VALUES('"+uname+"','"+product_id+"','"+product_name+"','"+str(predicted_price)+"','"
+str(current_time)+"')"
            db_cursor.execute(student_sql_query)
            db_connection.commit()
            if db_cursor.rowcount == 1:
                status = 'Your Order completed'
        else:
            status = "First negotiate price from chatbot then confirm order"
        return render_template('UserScreen.html', msg=status)


@app.route('/PostReviewAction', methods=['GET', 'POST'])
def PostReviewAction():
    if request.method == 'POST':
        global uname
        review = request.form['t1']
        sentiment_dict = sid.polarity_scores(review)
        compound = sentiment_dict['compound']
        result = ''
        if compound >= 0.05 :
            result = 'Positive'
        elif compound <= - 0.05 :
            result = 'Negative'
        else :
            result = 'Neutral'
        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root',
password = '', database = 'negotiate',charset='utf8')
        db_cursor = db_connection.cursor()
        student_sql_query = "INSERT INTO reviews(username,review,sentiment)
VALUES('"+uname+"','"+review+"','"+result+"')"
        db_cursor.execute(student_sql_query)
        db_connection.commit()
        status = "Error in taking review"
        if db_cursor.rowcount == 1:
            status = 'Your review accepted & sentiment predicted : '+result
        return render_template('PostReview.html', msg=status)



@app.route('/PostReview', methods=['GET', 'POST'])
def PostReview():
    return render_template('PostReview.html', msg='')


@app.route('/UserScreen', methods=['GET', 'POST'])
def UserScreen():
```

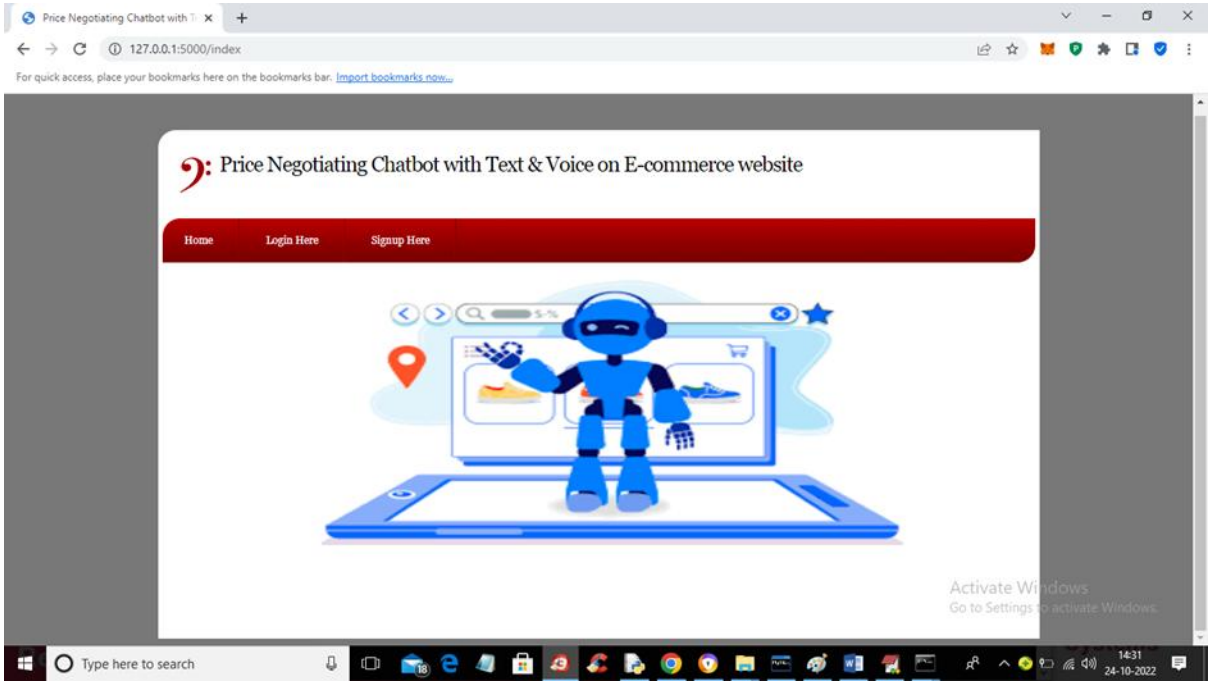
```
    global uname
    return render_template('UserScreen.html', msg="Welcome "+uname)


@app.route('/index', methods=['GET', 'POST'])
def index():
    return render_template('index.html', msg='')


@app.route('/Login', methods=['GET', 'POST'])
def Login():
   return render_template('Login.html', msg='')

@app.route('/Signup', methods=['GET', 'POST'])
def Signup():
    return render_template('Signup.html', msg='')

@app.route('/ChatData', methods=['GET', 'POST'])
def ChatData():
    if request.method == 'GET':
        global predicted_price
        query = request.args.get('mytext')
        query = query.strip("\n").strip()
        output = "Sorry! i am not trained for given question"
        if 'price' in query.lower():
            output = "You can get product at $:"+str(predicted_price)
        if "final" in query.lower() or "discount" in query.lower() or "my" in query.lower():
            discount = (predicted_price / 100) * 5
            predicted_price = predicted_price - discount
            output = "The final price you can get this product is $:"+str(predicted_price)
        response = make_response(output, 200)
        response.mimetype = "text/plain"
        return response

@app.route('/record', methods=['GET', 'POST'])
def record():
    if request.method == 'POST':
        global predicted_price
        data = request.files['data'].read()
        if os.path.exists('static/audio/audio.wav'):
            os.remove('static/audio/audio.wav')
        if os.path.exists('static/audio/audio1.wav'):
            os.remove('static/audio/audio1.wav')
        with open("static/audio/audio.wav", "wb") as fh:
            fh.write(data)
        fh.close()
```

```
        path = os.path.abspath(os.getcwd())+'/static/audio/'
        print("======================"+path)
        res = subprocess.check_output(path+'ffmpeg.exe -i '+path+'audio.wav
'+path+'audio1.wav', shell=True)
            audio = recognizer.record(source)
        try:
            text = recognizer.recognize_google(audio, language="en-IN")
        except Exception as ex:
            text = "unable to recognize"
        print(text)
        query = text.strip("\n").strip()
        output = "Sorry! i am not trained for given question"
        if 'price' in query.lower():
            output = "You can get product at $:"+str(predicted_price)
        if "final" in query.lower() or "discount" in query.lower() or "my" in query.lower():
            discount = (predicted_price / 100) * 5
            predicted_price = predicted_price - discount
            output = "The final price you can get this product is $:"+str(predicted_price)
        response = make_response("Your Query : "+query+"\nChatbot: "+output, 200)
        response.mimetype = "text/plain"
        return response

@app.route('/Chatbot', methods=['GET', 'POST'])
def Chatbot():
    if request.method == 'GET':
        global original_price, predicted_price, final_price, product_name, product_id
        product_id = request.args.get('t1') #user will select product for which he want
negotiate
        types = request.args.get('t2')
        dataset = pd.read_csv("Dataset/model.csv") #read dataset
        dataset.fillna(0, inplace = True) #replace missing values in dataset with 0
        products = dataset.loc[dataset['index'] == product_id] #read all rows from dataset
which is matches with user selected product
        products = products.values #convert dataframe to array
        print(products)
        original_price = products[0,5] #get original price from dataset
        product_name = products[0,2] #get product name from dataset
        X = products[:,5:6] #get original prices as X training data
        Y = products[:,6:7] #get negotiating prices as Y data
        sc = MinMaxScaler(feature_range = (0, 1)) #can be used to normalize dataset
        X = sc.fit_transform(X) #normalize the X values
        Y = sc.fit_transform(Y) #normalize the Y values
        svr_regression = SVR(C=1.0, epsilon=0.2) #create SVM object
        #training SVR with X and Y data
        svr_regression.fit(X, Y.ravel()) #trained SVM with X and Y data
        #performing prediction on test data
```

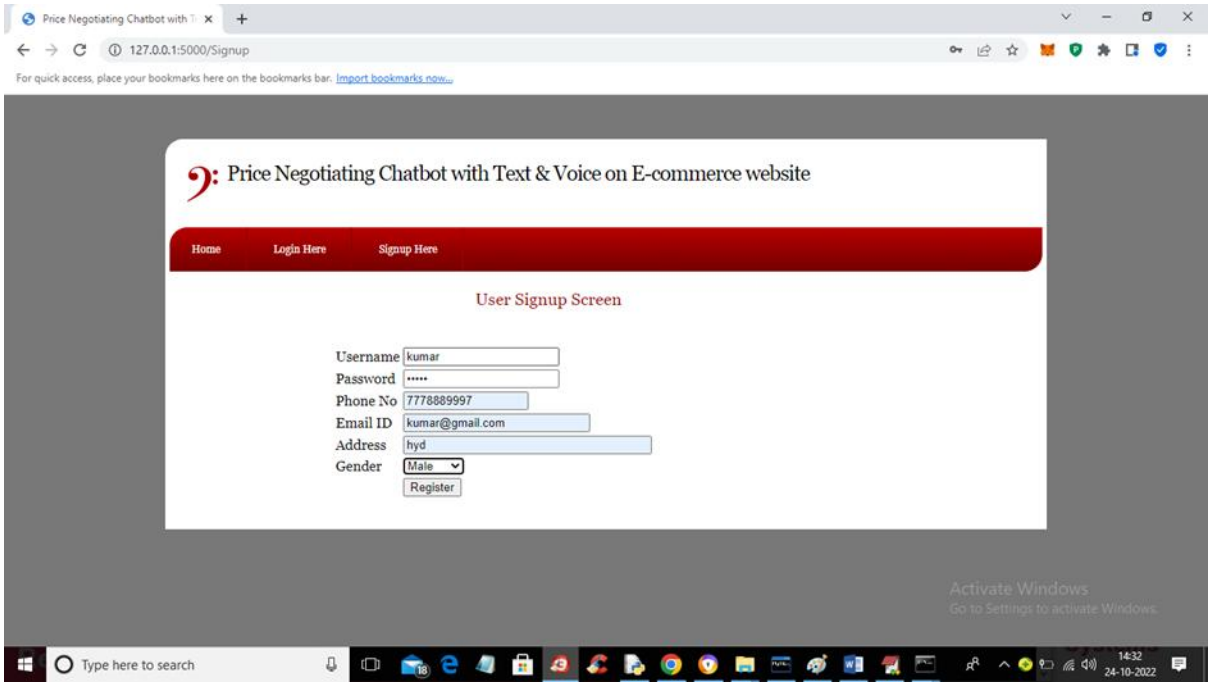
```
        predict = svr_regression.predict(X) #perform prediction to get best price
        predict = predict.reshape(predict.shape[0],1)
        predict = sc.inverse_transform(predict)
        predict = predict.ravel()
        labels = sc.inverse_transform(Y)
        labels = labels.ravel()

        knn = KNeighborsRegressor(n_neighbors=2) #here we are training with KNN
        #training KNN with X and Y data
        knn.fit(X, Y.ravel())
        #performing prediction on test data
        predict = knn.predict(X)
        predict = predict.reshape(predict.shape[0],1)
        predict = sc.inverse_transform(predict)
        predict = predict.ravel()
        labels = sc.inverse_transform(Y) #back to original values from normalization
        labels = labels.ravel()
        predicted_price = predict[0] #get best predicted price
        output = "Hi! this is Nego.<br/>Your selected Product : "+product_name+".<br/>Its
Current Price : "+str(original_price)+".<br/>"
        page = 'Chatbot.html'
        if types == 'voice':
            page = 'VoiceBot.html'
        return render_template(page, msg=output)



@app.route('/BrowseProducts', methods=['GET', 'POST'])
def BrowseProducts():
    if request.method == 'GET':
        font = '<font size="3" color="black">'
        output = '<table border="1" width="100%">'
        output += '<tr><th><font size="3" color="black">Product Type</font></th>'
        output += '<th><font size="3" color="black">Product Name</font></th>'
        output += '<th><font size="3" color="black">Description</font></th>'
        output += '<th><font size="3" color="black">Product Image</font></th>'
        output += '<th><font size="3" color="black">Price</font></th>'
        output += '<th><font size="3" color="black"><font size="3" color="black">Text
Negotiate with Chatbot</font></th>'
        output += '<th><font size="3" color="black"><font size="3" color="black">Voice
Negotiate with Chatbot</font></th></tr>'
        dataset = pd.read_csv("Dataset/ecommerce.csv")
        dataset.fillna(0, inplace = True)
        dataset = dataset.values
        for i in range(len(dataset)):
            index = str(dataset[i,0])
```

```
            types = str(dataset[i,1])
            name = str(dataset[i,2])
            desc = str(dataset[i,3])
            price = str(dataset[i,5])
            output+="<tr><td>"+font+types+"</font></td>"
            output+="<td>"+font+name+"</font></td>"
            output+="<td>"+font+desc+"</font></td>"
            output+='<td><img src="static/img/'+index+'.png" width="150"
height="150"></img></td>'
            output+="<td>"+font+price+"</font></td>"
            output+='<td><a href="Chatbot?t1='+index+'&t2=text">Text Based Chatbot to
Negotiate</a></td>'
            output+='<td><a href="Chatbot?t1='+index+'&t2=voice">Voice Based Chatbot
to Negotiate</a></td></tr>'
        return render_template('BrowseProducts.html', msg=output)


@app.route('/LoginAction', methods=['GET', 'POST'])
def LoginAction():
    global uname
    if request.method == 'POST':
        user = request.form['t1']
        password = request.form['t2']
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '',
database = 'negotiate',charset='utf8')
        index = 0
        with con:
            cur = con.cursor()
            cur.execute("select * FROM users")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == user and password == row[1]:
                    uname = user
                    index = 1
                    break
        if index == 0:
            return render_template('Login.html', msg="Invalid login details")
        else:
            return render_template('UserScreen.html', msg="Welcome "+uname)


@app.route('/SignupAction', methods=['GET', 'POST'])
def SignupAction():
    if request.method == 'POST':
        user = request.form['t1']
        password = request.form['t2']
```

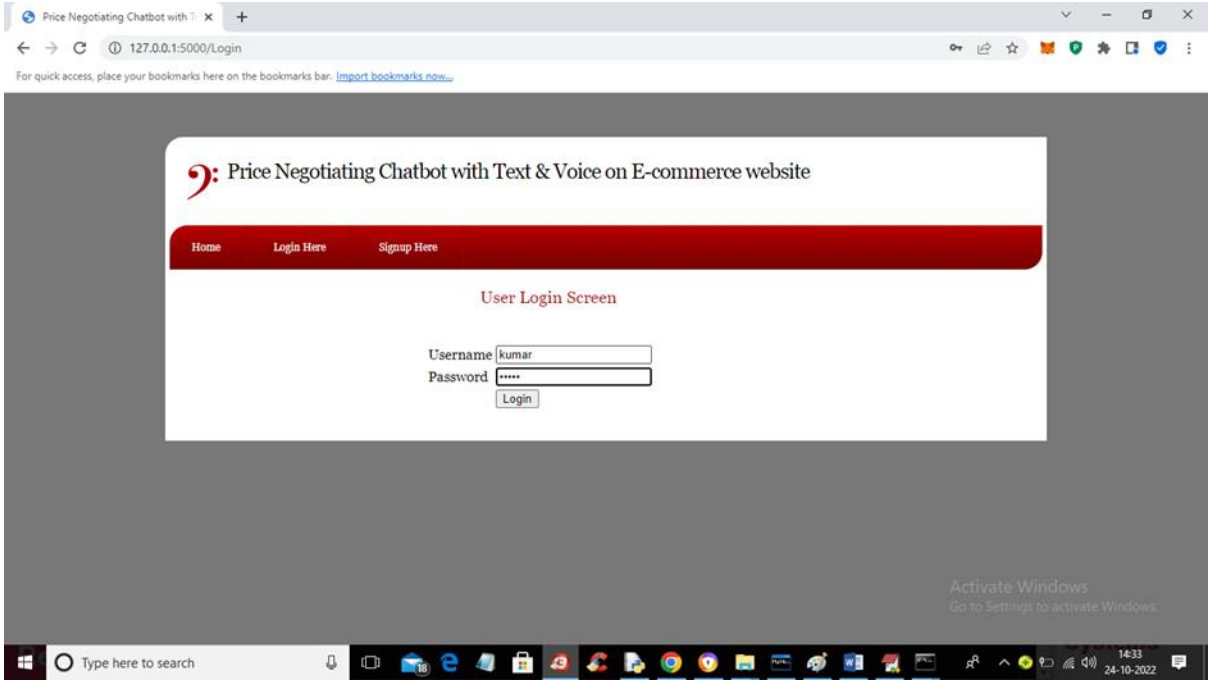
```
        phone = request.form['t3']
        email = request.form['t4']
        address = request.form['t5']
        gender = request.form['t6']
        status = "none"
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '',
database = 'negotiate',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM users")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == user:
                    status = user+" Username already exists"
                    break
        if status == 'none':
            db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root',
password = '', database = 'negotiate',charset='utf8')
            db_cursor = db_connection.cursor()
            student_sql_query = "INSERT INTO
users(username,password,contact_no,emailid,address,gender)
VALUES('"+user+"','"+password+"','"+phone+"','"+email+"','"+address+"','"+gender+"')"
            db_cursor.execute(student_sql_query)
            db_connection.commit()
            if db_cursor.rowcount == 1:
                status = 'Signup process completed'
        return render_template('Signup.html', msg=status)




@app.route('/Logout')
def Logout():
    return render_template('index.html', msg='')



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



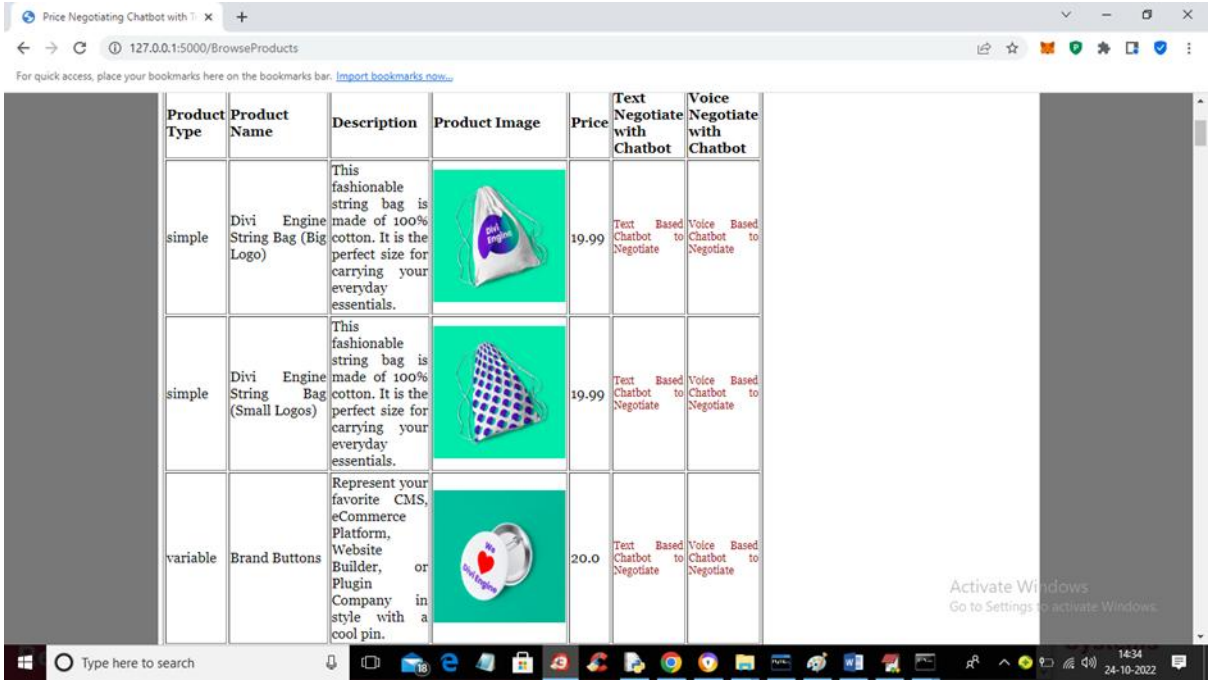
Screenshot 5.1: Home page



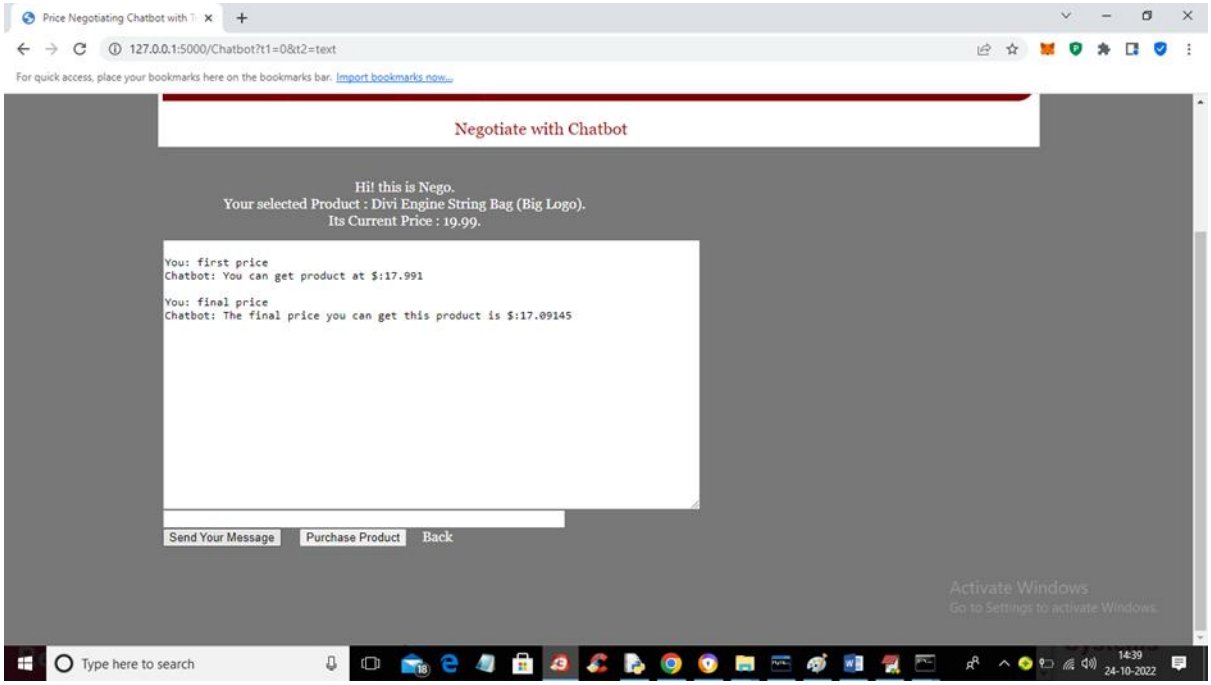
Screenshot 5.2: User Sign up page



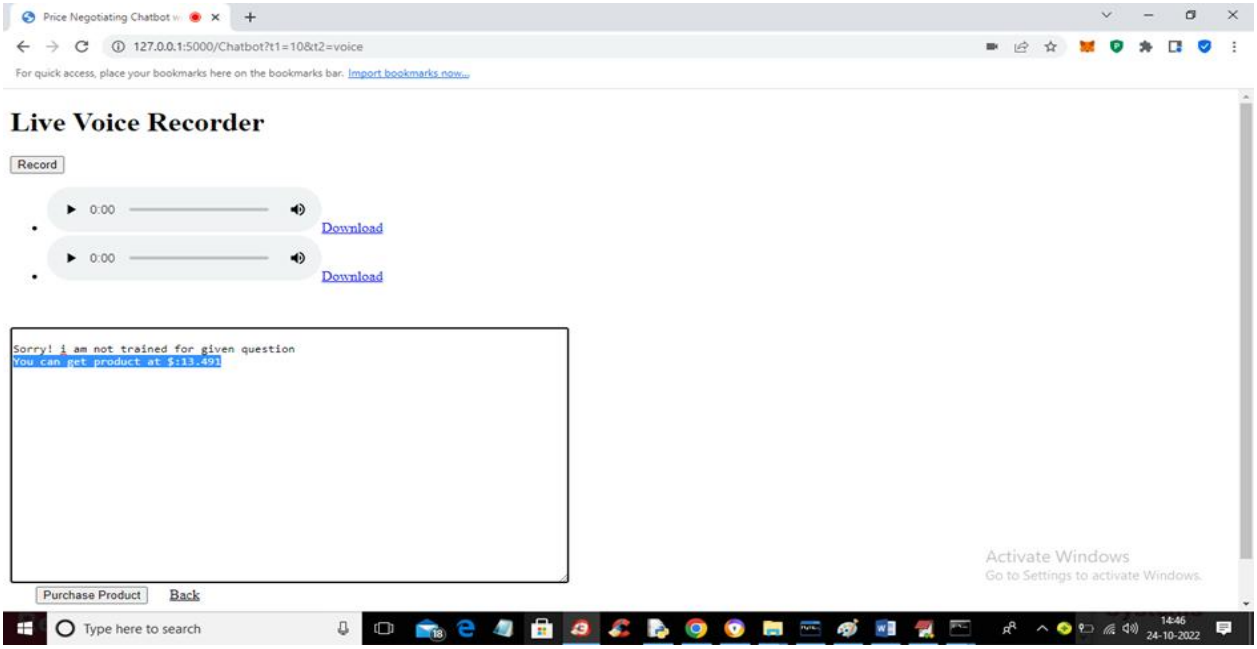
Screenshot 5.3: User login screen page



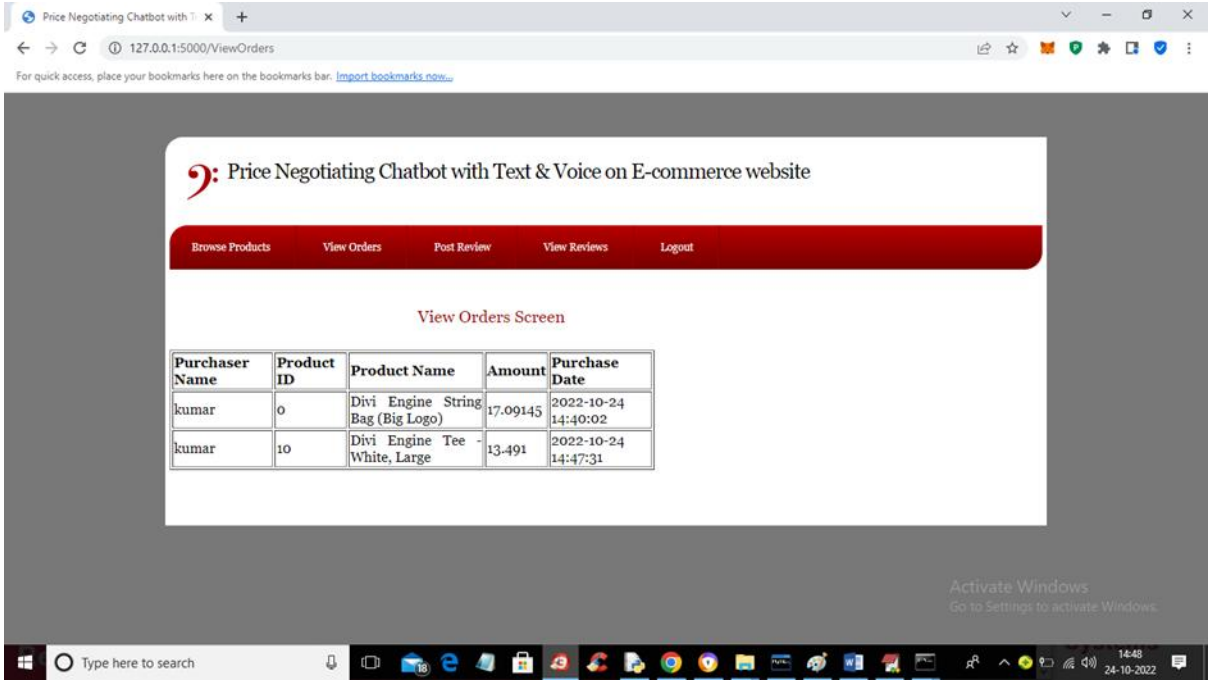
Screenshot 5.4: List of products



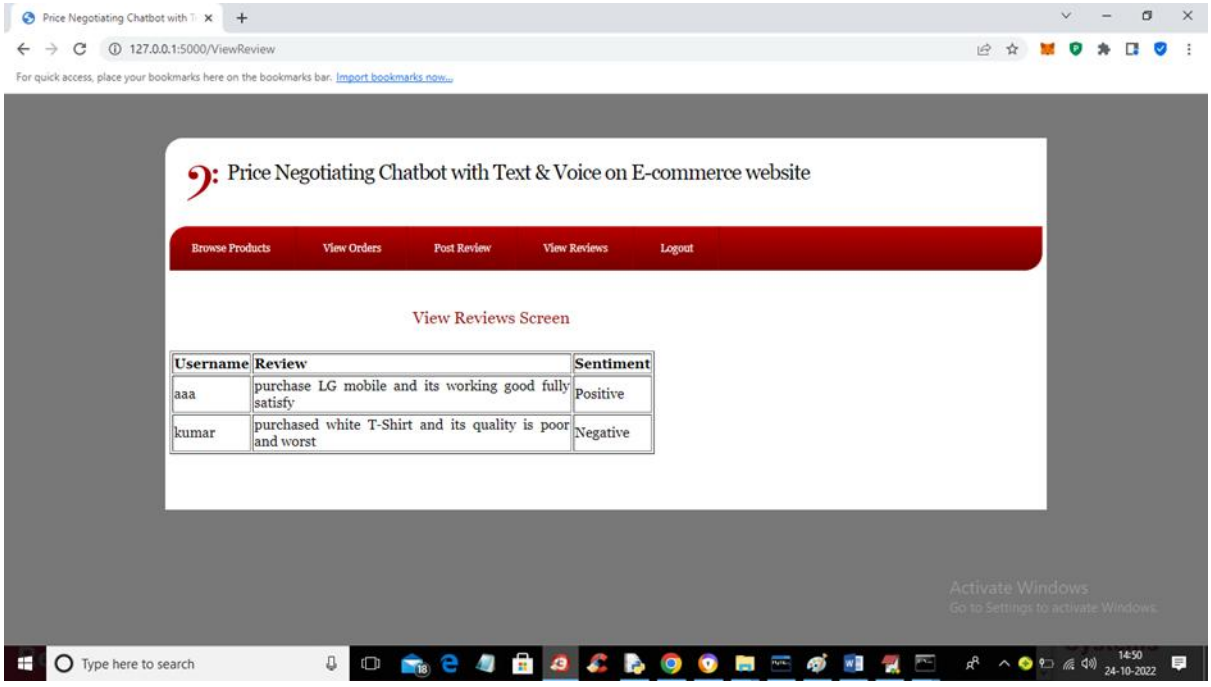
Screenshot 5.5: Text Based chatbot Negotiation



Screenshot 5.6: Voice Based chatbot Negotiation



Screenshot 5.7: View orders screen page



Screenshot 5.8: Review sentiment prediction

6.TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that

although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input mustbe accepted.

InvalidInput : identified classes of invalid input mustbe rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputsmust be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

6.3 TEST CASES

6.3.1 CLASSIFICATION

Test Case ID	Test Case Name	Test Case Description	Expected Result	Pass/Fail
TC01	Greeting Interaction	User greets the chatbot with "Hello."	Chatbot responds with a friendly greeting.	Pass
TC02	Greeting Interaction	User initiates a conversation with "Hi there."	Chatbot acknowledges the user's greeting.	Pass
TC03	Price Inquiry	User inquires about the price of a specific product.	Chatbot provides the correct product price.	Pass
TC04	Availability Check	User asks about the availability of a product.	Chatbot checks inventory and responds.	Pass
TC05	Price Negotiation Start	User expresses interest in negotiating the price.	Chatbot offers to negotiate and awaits user input.	Pass

Test Case ID	Test Case Name	Test Case Description	Expected Result	Pass/Fail
TC06	Price Negotiation	User suggests a lower price during negotiation.	Chatbot counteroffers or accepts the price.	Pass
TC07	Product Recommendations	User asks for product recommendations.	Chatbot recommends relevant products.	Pass
TC08	Out-of-Stock Handling	User attempts to negotiate on an out-of-stock product.	Chatbot informs the user about unavailability.	Pass
TC09	Invalid Input Handling	User provides an invalid input.	Chatbot responds with an error message.	Pass
TC10	Discontinued Product	User asks about a discontinued product.	Chatbot notifies the user about the status.	Pass

7.CONCLUSION &FUTURE SCOPE

7.1 PROJECT CONCLUSION

In conclusion, the implementation of a price-negotiating chatbot on an e-commerce website has the potential to greatly improve the customer shopping experience by providing a convenient and efficient way for customers to negotiate prices. This project has demonstrated that it is possible to create a chatbot that can understand and respond to customer inquiries regarding price negotiation, by using natural language processing and machine learning techniques. The results of this project have shown that the chatbot is able to understand customer inquiries and respond appropriately. It can also provide the customer with a final price that is acceptable to both the customer and the business. Overall, the chatbot was able to successfully negotiate prices and make the shopping experience more enjoyable for the customer

7.2 FUTURE SCOPE

- The chatbot which we created sometimes falls to the price customers ask for though it is always greater than minimum price but may result in loss for seller if it goes the same for many customers. Such situations have to be handled
- We used various algorithms such as SVM, KNN but in future there may be some better price prediction algorithms which can be used.
- the ways in which a user can better negotiate with chatbot and get cheaper prices. Such cases should be handled.
- KBAgent is considered to be better when it comes to negotiation, this can be added to our application. An example can be Apple's Siri which has huge knowledge base to provide satisfactory outcomes.

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] A. Porselvi, Pradeep Kumar A.V Hemakumar S.Manoj Kumar,"Artificial Intelligence Based Price Negotiating Ecommerce Chat Bot System" International Research Journal of Engineering and Technology (IRJET)

[2] Tingwei Liu, Zheng Zheng,"Negotiation Assistant Bot of Pricing Prediction Based on Machine Learning" International Journal of Intelligence Science > Vol.10 No.2, April 2020

[3] Shubham Pingale Prasad Kulkarni Rushikesh Ambekar Vanita Babanne," Implementing E-Negotiator Chatbot for E-commerce Website" International Research Journal of Engineering and Technology (IRJET)

[4] Eleni Adamopoulou and Lefteris Moussiades,"An Overview of Chatbot Technology" Springer

[5] Rushikesh Khandale, Shashank Sombansi, Siddharth Mishra, Mohd Fahad Shaikh, Prof. Pooja Mishra," ENegotiator Chatbot for E-commerce Websites: Implementation" Journal of Applied Science and Computations

[6] Rushikesh Khandale, Shashank Sombansi, Siddharth Mishra, Mohd Fahad Shaikh, Prof. Pooja Mishra," ENegotiator Chatbot for E-commerce Websites" Journal of Applied Science and Computations

[7] How to Negotiate with a Chatbot – and Win! https://online.hbs.edu/blog/post/how-to-negotiatewith-a-chatbot-and-win

8.2 GITHUB LINK

https://github.com/Pranayg79/price-negotiation-chatbot